CS369E: Expanders in Computer Science

Cynthia Dwork & Prahladh Harsha

Credits

These are the lecture notes for the course *CS369E: Expanders in Computer Science* taught at the Computer Science Department at Stanford University in the Spring quarter of 2005. We thank the scribes – David Arthur, Adam Barth, Arpita Ghosh, Geir Helleloid, Krishnaram Kenthapadi, and Hovav Shacham for their meticulous note-taking. We also thank Microsoft Research and the Computer Science Department at Stanford.

We had indeed advertised that we will be covering recent results in the course announcement. We had planned to cover Reingold's new result "SL=L", which we presented in lecture as scheduled. Little did we expect that Irit Dinur would come up with a simple proof of the PCP Theorem using expanders and that we would actually present the new proof in the course, when it was hardly a few weeks old.

These notes are by no means polished. Nevertheless, comments are always appreciated.

Cynthia Dwork
Prahladh Harsha

31 Aug, 2005.

Course Announcement

CS369E: Expanders in Computer Science (Stanford)
Expanders, constructions and their applications

Time: Mon 2:15-4:05pm
Location: Gates 159 (Stanford)
Instructors : Cynthia Dwork and Prahladh Harsha
Homepage: http://cs369e.stanford.edu

Expanders in Computer Science

Over the past few decades, expanders have played a pervasive role in diverse fields of computer science - network design, derandomization, distributed computing, random walks, error-correcting codes, metric embeddings etc. Informally, an expander is a sparse graph which is nevertheless highly connected. In this course, we will study these expander graphs and several of their applications.

As part of this course, we will study the relationship between expansion and eigen values. We will then look at various constructions of expanders - Margulis, LPS and zig-zag expanders. A large chunk of the course will be devoted to applications of expanders:

1. Random walks and universal sequences
2. Derandomization
   (including Reingold's "SL=L" result)
3. Error Correcting Codes
4. Distributed Computing
5. Isoperimetric problems

The course will reach the cutting-edge of current research in this area, covering some results from within the last year. At the same time, the concepts we will cover are general and useful enough that hopefully anyone with an interest in the theory of computation or combinatorics could find the material appealing.

Contents

Chapter 1

Expanders – an Introduction

Lecturer: Cynthia Dwork
Scribe: Arpita Ghosh
April 4, 2005

In this lecture, we start with some basic definitions and notation, and then discuss some motivating applications of expanders.

A graph $G = (V, E)$ has vertex set V, with $|V| = N$, and edge set E. G is undirected, unless specified otherwise, and can be a multigraph. A bipartite graph will be denoted as $G = (L \cup R, E)$, *i.e.*, its vertex sets are partitioned as L and R (L and R do not necessarily have the same size).

For each vertex $v \in V$, the neighborhood of v is denoted $\Gamma(v)$, and defined as

$$\Gamma(v) = \{u \in V \mid (u, v) \in E\}.$$

The neighborhood of a subset $S \subseteq V$ is defined as the union of the neighborhoods of the vertices in S, *i.e.*,

$$\Gamma(S) = \cup_{v \in S} \Gamma(v).$$

We also define

$$\Gamma^{'}(S) = S \cup \Gamma(S),$$

i.e., $\Gamma^{'}(S)$ consists of vertices in S as well as their neighbors.

Definition 1. *A graph G is said to have vertex expansion (K, A) if*

$$|\Gamma(S)| \geq A \cdot |S|, \qquad \forall S \subseteq V \text{ with } |S| \leq K.$$

We will then say that G is a (K, A)-expander

Clearly, a complete graph has the best possible expansion. However, we will be interested in constant degree graphs. We will typically want $K = cN$, for some constant c, and A to be of the order of the degree d of the graph. When $K = N/2$ (this will be the most common setting in this course), we will refer to G as an A-expander. Informally, an expander graph is one where all subsets of V (under some constraint on their size) have large neighborhoods. Now we will see some applications and results related to expanders. Surprisingly, such graph do exisit. For the present we will assume the existence of such graphs (and also that we can construct them efficiently). We will defer the actual constructions of such expanders to the second half of this course.

We will now consider applications of such graphs in five different contexts.

- Time-space tradeoffs
- Byzantine Agreement
- Saving Random Bits
- Error Correcting Codes
- Metric Embeddings

Two of these applications (the first and the last) are lower bounds while the other two are upper bounds. These examples illustrate the use of expanders in a wide variety of contexts, both for positive and negative results. We will not furnish full proofs in these examples, however.

1.1 Time-space tradeoffs in computing functions

One of the first uses of expanders in computer science was in studying the computational hardness of certain functions. Given a specific method for computing a function we can create an acyclic computational circuit describing the computation. Valiant was studying the hardness of computing certain linear transforms, and observed that every circuit computing these functions was a *superconcentrator* [Val]. Intuitively, these are graphs with very great flow from inputs to outputs: An n-superconcentrator is a graph with n inputs and n outputs, such that for all subsets S of the input, and all subsets T of the output with $|T| = |S|$, there exist vertex disjoint paths connecting S to T. Valiant hoped to obtain lower bounds for the linear transforms by obtaining high lower bounds for superconcentrators. Valiant conjectured that any superconcentrator must have a superlinear number of edges. However, Valiant himself disproved this conjecture and gave construction of superconcentrators with $O(n)$ edges using expanders. This happens to be the first context when expanders where used in Computer Science. For an interesting account of the details that went into the discovery of expanders, refer the writeup on Reingold's recent "SL=L" result by Sara Robinson [Rob].

However, it turns out that superconcentrators exhibit interesting time/space tradeoffs, leading to time/space tradeoffs for computations (see the work of Paul, Tarjan and Celoni [PTC, PT]). We will show how a pebbling game on such graphs leads to some lower bounds. Consider the following pebbling game played on a DAG (directed acyclic graph). To start with, the DAG has some specified input nodes and output nodes, and some auxiliary nodes connecting inputs to output. All edges are directed and go from left to right (inputs are on the left, and outputs on the right). We are given S pebbles, with the following rules:

- A pebble can be placed on an input at any time.
- A pebble can be placed on a non-input node if all its predecessors (*i.e.*, nodes with edges leading into it) currently hold pebbles.
- A pebble can be removed at any time.

The goal is to evaluate the outputs, *i.e.*, pebble each of the outputs. For example, suppose the DAG represents a computational circuit. Clearly, with a large number of pebbles S, the outputs can be computed in a small number of steps. If every graph to evaluate a function is such that it cannot be pebbled quickly with a small number of pebbles, then the function is hard to compute with a small amount of memory. For further details, refer to the work of Celoni, Tarjan and Paul [PTC, PT].

As stated earlier, superconcentrators with $O(n)$ eedges can be constructed from constant degree expanders (we will not discuss this construction here). This construction uses the Hall's Theorem which states the necessary and sufficient condition for a graph to have a perfect matching. Since this condition is similar in flavor to the definition of expanders, we state and prove Hall's theorem here.

Theorem 2. *Let $G = (L \cup R, E)$ be a bipartite graph. Then, G has a perfect matching if and only if*

$$\Gamma(S) \geq |S|, \qquad \textit{for all } S \subseteq L. \tag{1.1}$$

Proof. The only if direction is obvious (if there is a subset of S with not as many neighbors as vertices, S cannot have a matching). We will show the if direction by induction on the size of L. The base case, where L has at most one vertex is trivial. Suppose now that the claim is true for some $L \leq m$. We will consider two cases:

- **Case 1:** All proper subsets S of L, $S \neq \emptyset$, expand strictly, *i.e.*, $|\Gamma(S)| > |S|$. Consider any vertex x in L, and let (x, y) be an edge in E. Now, consider the bipartite graph G^* with vertex sets $L^* = L - \{x\}$ and $R^* = R - \{y\}$. Since every $S \subset L$ satisfies (1.1) with strict inequality, every subset of L^* satisfies (1.1), since only a single vertex y has been removed from R. Therefore, by the induction hypothesis, the smaller graph G^* has a matching. To this matching add the edge (x, y); this gives a perfect matching in G.

- **Case 2:** There exists a proper subset $T \subset L$, $T \neq \emptyset$, with $|\Gamma(T)| = |T|$. Consider the induced graphs G_1 and G_2 on the vertex sets $T \cup \Gamma(T)$, and $L \setminus T \cup R \setminus \Gamma(T)$ respectively. By the induction hypothesis, G_1 has a perfect matching. (Note that the induction hypothesis cannot be used directly on G_2.) Let $S \subseteq L \setminus T$. Then,

$$\begin{aligned} \Gamma_{G_2}(S) &= \Gamma_G(S \cup T) \setminus \Gamma_G(T) \\ \Rightarrow |\Gamma_{G_2}(S)| &\overset{(a)}{\geq} |S \cup T| - |T| \\ &= |S|, \end{aligned}$$

where (a) is true since $S \cup T$ satisfies $|\Gamma_G(S \cup T)| \geq |S \cup T|$, and by assumption, $|\Gamma(T)| = |T|$. Therefore, the graph G_2 also satisfies (1.1), and by the induction hypothesis, has a matching. The unions of the perfect matchings in G_1 and G_2 is a matching for G.

□

1.2 Almost Everywhere Byzantine Agreement

In the Byzantine agreement problem each of n processors begins with an input value, say, $v_i \in \{0, 1\}$. During the course of the computation each processor must irreversibly *decide* on an output value d_i. The requirements are

- (Unanimty): All non-faulty processors must produce the same decision.
- (Non-triviality): If all non-faulty processors begin with the same value, say v, then every non-faulty processor must output v.

Another application of expanders occurs in the context of solving Byzantine agreement in general networks, where the processors correspond to nodes and a processor can only communicate with its immediate neighbors. Dolev showed that for t-resilient Byzantine agreement, connectivity greater equal $2t + 1$ is necessary [Dol].

When t is linear in the number n of vertices in the network, this requires $O(n^2)$ edges, which is unreasonable for large n. Consider, instead, a constant-degree expander on n vertices. In such a network, if not too many nodes are faulty, then it can be shown that there is a large fraction of non-faulty nodes that can communicate "as if" they were in a completely connected network. Intuitively this is because expanders do not have small cuts.

By relaxing the unanimity requirement, essentially permitting $O(t)$ non-faulty processors to be "lost" (not to join in the majority decision), we obtain almost-everywhere agreement [DPPU]. Using constant-degree expanders (together with other special graphs), the relaxed problem can be solved in networks of bounded degree. For further details, refer [DPPU] and [Upf].

1.3 Saving random bits

Another application of expander graphs occurs in reducing the number of random bits used by a randomized algorithm. We will study several such applications of expanders in the context of derandomization. For today's introductory lecture, we will study a simple and beautiful application due to Karp, Pippenger and Sipser [KPS].

Recall that a language $\mathcal{L}$ belongs to RP if there exists a polynomial time Turing machine M using a sequence of random bits (of length polynomial in the input size), such that

$$x \in \mathcal{L} \Rightarrow \frac{|W_x|}{2^{p(|x|)}} \geq q \geq \frac{3}{4},$$

and

$$x \notin \mathcal{L} \Rightarrow |W_x| = 0,$$

where

$$W_x = \{y | M(x,y) = 1, |y| = poly(|x|)\}.$$

That is, if $x \in \mathcal{L}$, the probability that $M(x,y) = 0$ (which is the probability of error) is less equal $1-q$, a constant (less than 1/4). That is, if the algorithm M returns 1 on input (x,y), then surely $x \in \mathcal{L}$, but if it returns 0, then x may or may not belong to $\mathcal{L}$.

Let r be the number of random bits used to generate the string y, $r = poly(|x|)$. Suppose we want the probability of error to be δ. Since the error decreases by a constant fraction each time, to reduce the probability of error down to δ, we can repeat the algorithm $O(\log(\frac{1}{\delta}))$ times. But for this, the total number of random bits used is $O(r\log(\frac{1}{\delta}))$, *i.e.*, we need to use a large number of random bits to make the probability of error small.

Using expanders, it is possible to obtain, in polynomial time, to reduce the error to as low as $\frac{1}{\text{poly}(r)}$ with no extra random bits (*i.e.*using only the original r random bits).

The expander used is a giant $(N/2, A)$-expander G on $N = 2^r$ vertices where A is the expansion of the expander. Note that since the expander is so large, we cannot even afford to write down the entire expander, forget constructing it. However, we will assume that there exists an implicit construction of the expander in the following sense: given any vertex v and any index i in the range $1 \dots d$ (where d is the degree of the expander), we can in time polynomial in $|v|$ and $|i|$, compute the i^{th}-neighbor of v. The expanders constructions we will discuss later in the course will satisfy such strong properties.

Choose a radius c such that $\frac{1}{4A^c} \leq \delta$, where A is the expansion of G. Choose a vertex v uniformly at random from G. This needs r random bits, since $|V| = 2^r$. The modified RP algorithm is as follows:

1. Run the original RP algorithm M for all strings y lying within a ball of radius c around v.

2. If for all these y, $M(x,y) = 0$, reject x.

3. If $M(x,y) = 1$ for any y, accept x.

We will show that the error of this modified RP algorithm is at most δ.

Suppose $x \in \mathcal{L}$. Define Bad_x to be the set of y for which $M(x,y) = 0$. The output is erroneous only if $\Gamma'_c(v) \subseteq Bad_x$, where the subscript c denotes the set of vertices within a distance c of v. Let

$$B = \{v \mid \Gamma'_c(v) \subseteq Bad_x\}.$$

The algorithm fails only when v is picked from B. Now, by definition of B, for $1 \leq i \leq c-1$,

$$\Gamma'_i(B) \subseteq \Gamma'_{i+1}(B) \subseteq Bad_x.$$

Also, by definition, since we started with an input $x \in \mathcal{L} \in RP$,

$$|Bad_x| \leq N/4.$$

Since G has an expansion of A,

$$|\Gamma'_c(B)| \geq A^c|B|.$$

Combining all of this, we have

$$\frac{N}{4} \geq |Bad_x| \geq |\Gamma_c'(B)| \geq A^c|B|,$$

and therefore,

$$\frac{|B|}{N} \leq \frac{1}{4A^c} \leq \delta.$$

But the algorithm only fails when the vertex v is picked from the set B, which has a probability of $\frac{|B|}{N}$. Observe that the running time of the new algorithm is poly$(1/\delta)$. This limits the minimum error that can be attained by this technique. Therefore, we can get any error $\delta = \frac{1}{\text{poly}(r)}$ by choosing c appropriately, with only r bits of randomness.

1.4 Error correcting codes

The next application comes from error correcting codes. Suppose a sender A is sending a k-bit message to receiver B over a faulty channel, which could flip up to a fraction p of the bits sent over it. The message is encoded by A into an n-bit codeword in $C \subset \{0,1\}^n$, where the size of the C is 2^k (there is a unique codeword for each message). The decoding is simple: given an n-bit vector, find the closest codeword $w \in C$, and return the k-bit message that maps to w. The rate R of the code is defined to be

$$R = \frac{\log |C|}{n} = \frac{k}{n}.$$

Clearly, if the Hamming distance between every two codewords is strictly greater than $2pn$, then the message can be decoded without error (there is a unique closest codeword $w \in C$ for each n-bit string.) Define the distance δ of a codebook to be

$$\delta = \min_{c_1 \neq c_2 \in C} \frac{d_H(c_1, c_2)}{n},$$

where d_H, the Hamming distance, is the number of bit positions in which the vectors differ.

The communication problem is the following:
Is it possible to define a family of codes, $\{C_k\}_{k=1}^{\infty}$, such that $|C_k| = 2^k$, and for each k, $\delta_k > 0$, and $R_k > 0$?

We will show that codes with good distance can be constructed from expanders which have very good expansion (more precisely, when the expansion factor is greater than half the degree). The first such construction of expander based codes was given by Tanner [Tan]. Suppose we have an $(\alpha N, K)$ d-regular expander G, where $K > d/2$. Consider a bipartite graph with $L = n$, with the vertices in L the components of a (n-bit) codeword. Each vertex in R represents a constraint, which are of the form $\oplus x_v = 0$, where x_v is the vth component of the (n-bit) codeword x. Thus if vertex $j \in R$ has neighbors, say $i_1, i_2, i_3 \in L$, then vertex j represents the constraint $x_{i_1} \oplus x_{i_2} \oplus x_{i_3} = 0$. (Thus, the larger the number of constraints, the smaller the size of the code, and vice versa.)

Note that a code which is the set of Boolean vectors satisfying constraints of the form $\oplus x_v = 0$ is a linear code, since $(0, \ldots, 0)$ belongs to the code, and if c_1 and c_2 belong to the code, then so does $c_1 \oplus c_2$. It can easily be checked (using the above definition) that the distance of a linear code is the weight of the minimum weight (non-zero) codeword.

Now, since G has an expansion of $K > d/2$, for every subset S of L of size less equal αN, there is a $v \in \Gamma(S)$ such that v is adjacent to a unique element of S. Suppose not; then, every vertex in $\Gamma(S)$ is connected to at least two vertices in S. Let E be the number of edges between S and $\Gamma(S)$, then

$$\begin{aligned} & d|S| = |E| \geq 2|\Gamma(S)| \\ \Rightarrow \quad & |\Gamma(S)| \leq \frac{d}{2}|S|. \end{aligned}$$

But this contradicts the $K > d/2$ expansion of G.

Now, this means that there cannot be a codeword with only αN ones in it, since then we can choose S to be this subset of αN ones, and have a violated constraint. Therefore, the minimum weight codeword has a weight strictly greater than αN, and thus $\delta \geq \alpha$.

If there are $n(1-c)$ constraints, then the set of codewords simultaneously satisfying these constraints is a linear subspace of $\{0,1\}^n$, with size $|C| \geq 2^{nc}$. Therefore, the rate of such a code would be $nc/n = c$, which is a constant greater than zero.

Thus, the existence of a family of expanders, with an expansion of $K > d/2$, and $|R| = (1-c)|L|$ allows the construction of a family of codes $\{C\}_k$, with rate and minimum Hamming distance strictly greater than 0 for all k. We will return to the applications of expanders to error-correcting codes in one of the later lectures.

1.5 Metric embeddings

Another use of expanders occurs in metric embeddings. Let M on (X, D) (*i.e.*, a metric on X with distance measure D) and M' on (X', D') be two metrics. An embedding $f : M \to M'$ has distortion c if

$$\forall x, y \in X, \quad D(x,y) \leq D'(f(x), f(y)) \leq cD(x,y).$$

Bourgain showed that any n-point metric space can be embedded into l_2-metric [1] with distortion $O(\log n)$ and dimension at most $O(\log n)$ [Bou].

Embeddings are used in several contexts. For instance, it might be the case that a problem (say the Travelling Salesman Problem) is hard in an arbitrary metric, but is slightly more tractable in a well-understood metric such as l_2. In this case, it is plausible that embedding the arbitrary metric into l_2, solving the problem in l_2 and retransforming the problem back to the original metric gives some insight into the solution of the problem in the "hard" metric.

The following is another application of metric embeddings into l_2. Suppose we are given a graph $G = (V, E)$ with weights (or distances) on edges, and the distance between an arbitrary pair of nodes i and j is the shortest path distance between i and j on G. We want to be able to quickly answer (approximately) a query of the form 'what is the shortest path distance between vertices i and j in G', where i and j are arbitrary. To answer this question

[1] l_2 is $\mathbb{R}^n$ equipped with the usual Euclidean metric.

exactly, the storage required is $O(n^2)$ (store all the shortest path distances). However, the l_2 embedding allows us to simply compute the l_2 distance between the queried vertices in the l_2 embedding, which is a $O(\log n)$ approximation to the actual shortest path distance between the same vertices in G. The storage required for the l_2 embedding is $O(n \log n)$; since the l_2 embedding is into a $O(\log n)$ dimensional space (which is, of course, better than the $O(n^2)$ storage for the exact solution).

Expanders happen to be some of the worst case examples for embedding. In this sense, expanders are sometimes used to show the limits of embedding. For instance, London, Linial and Rabinovich showed that this is actually tight, by constructing expanders for which the distortion is $\Omega(\log n)$ [LLR].

Chapter 2

Eigenvalues and Expanders

Lecturer: Cynthia Dwork
Scribe: Geir Helleloid
April 11, 2005

2.1 Lecture Outline

1. A non-constructive proof that expanders exist.

   Our method of proof will be to pick a random graph and show that it is an expander with some non-zero probability. There do exist constructive proofs, but we won't see any today. Given this proof, in order to find an expander in practice, we might want to generate a graph at random and test to see if it is an expander – but testing is co-NP hard.

2. Explore the connection between expanders and the spectrum of the graph (that is, the set of eigenvalues of the graph).

   There is a connection between the expansion of a graph and the eigengap (or spectral gap) of the normalized adjacency matrix (that is, the gap between the first and second largest eigenvalues). Recall that the largest eigenvalue of the normalized adjacency matrix is 1; denote it by λ_1 and denote the second largest eigenvalue by λ_2. We will see that a large gap (that is, small λ_2) implies good expansion and vice versa.

   (a) Large spectral gap implies good expansion.

   (b) Expander Mixing Lemma
   Heuristically, this says that "an expander graph will behave like a random graph." Let S and T be disjoint subsets of a vertex set V. If G is a random d-regular (multi)graph on V, then the expected number of S-T edges (that is, edges with

one endpoint in S and one endpoint in T) is $d|S||T|/N$. The lemma says that in an expander on V, the number of S-T edges will be $d|S||T|/N + \lambda_2$(error term).

3. Alon's proof ([Alo], 1986) that good expansion implies a large spectral gap.

   This can be viewed as a discrete analogue of a result of Cheeger. Though vertex expansion and spectral gap are very closely related, vertex expansion does not seem to be the combinatorial equivalent of spectral expansion. This is because, the connection between vertex expansion and spectral expansion does not seem to be tight. Bitu and Linial ([BL], 2004), via their (partial) converse to the expander mixing lemma, give what might be the combinatorial equivalent notion (at least in the case of constant degree graphs) of the second eigen value.

4. The relationship between d and λ_2.

(Several of the proofs given in this Section are from Lectures 8 and 9 of Salil Vadhan's notes on Pseudo-randomness [Vad]).

2.2 Existence of Expanders

The best probabilistic result on the existence of expanders is:

Theorem 3. *Fix $d \geq 3$. A random d-regular graph is a $(\Omega(N), d - 1.01)$-expander with high probability (as $N \to \infty$, the probability goes to 1).*

We will not prove this result. Instead we will show the existence of bipartite expanders. Let $\mathcal{G}_{d,N}$ denote the set of bipartite graphs with partite sets L and R of cardinality N and left degree d.

Theorem 4. *For all d, there exists $\alpha(d) > 0$ such that for all N,*

$$\Pr[G \textit{ is an } (\alpha N, d-2)\textit{-expander}] \geq 1/2,$$

where G is chosen uniformly at random from $\mathcal{G}_{d,N}$. (In fact, we can take $\alpha(d) = 1/(cd^4)$ for some constant c.)

Proof. To choose G in $\mathcal{G}_{d,N}$ uniformly at random, we choose d (not necessarily distinct) neighbors for each vertex L at random. For $k \leq \alpha N$, let

$$p_k = \Pr[\exists\ S \subseteq L \text{ such that } |S| = k,\ |\Gamma(S)| < (d-2)|S|].$$

Thus p_k is the probability that G is not a $(\alpha N, d-2)$-expander because the neighborhood of a set of size k is not large enough. To prove the theorem, it suffices (by the union bound) to show that $\sum_k p_k \leq 1/2$.

If $S \subseteq L$ has cardinality k, then the total number of neighbors of vertices in S, counted with multiplicity, is dk. So if $|\Gamma(S)| < (d-2)k$, then there must be $2k$ repeats among the neighbors of vertices in S. We can compute this probability:

$$\Pr[\text{at least } 2k \text{ repeats among the } kd \text{ neighbors of vertices in } S] \leq \binom{kd}{2k}\left(\frac{kd}{N}\right)^{2k}.$$

Here, the binomial coefficient represents the number of ways to choose $2k$ neighbors to be repeats, and the fraction kd/N represents an upper bound on the probability that any given choice of a neighbor is a repeat. That this is an upper bound follows from the union bound. Since there are $\binom{N}{k}$ possibilities for S, we have

$$\begin{aligned} p_k &\le \binom{N}{k}\binom{kd}{2k}\left(\frac{kd}{N}\right)^{2k} \\ &\le \left(\frac{Ne}{k}\right)^{k}\left(\frac{kde}{2k}\right)^{2k}\left(\frac{kd}{N}\right)^{2k} \\ &= \left(\frac{cd^4k}{N}\right)^{k}, \end{aligned}$$

where $c = e^3/4$. When $\alpha = 1/(cd^4)$ and $k \le \alpha N$, we see that $p_k \le 4^{-k}$. Then

$$\Pr[G \text{ is not an } (\alpha N, d-2)\text{-expander}] \le \sum_{k=1}^{\alpha N} p_k \le \sum_{k=1}^{\alpha N} 4^{-k} < 1/2.$$

This completes the proof. □

2.3 Exploring the spectral connection

Let G be a d-regular multigraph with normed adjacency matrix A. The largest eigenvalue of A is $\lambda_1 = 1$ with eigenvector $u = (1/N, \ldots, 1/N)$. Then the second largest eigenvalue is given by

$$\lambda_2 = \max_{\|x\|=1, x\perp u} \|Ax\|.$$

If π is a probability distribution on the vertices of G (represented as a vector), we can write $\pi = u + \pi^\perp$, where $\pi^\perp \perp u$. View A as the transition matrix for a Markov chain and use the initial distribution π. Then

$$A\pi - u = A(u + \pi^\perp) - u = Au - u + A\pi^\perp = A\pi^\perp.$$

Thus

$$\|A\pi - u\|^2 = \|A\pi^\perp\|^2 \le \lambda_2^2\|\pi^\perp\|^2 = \lambda_2^2\|\pi - u\|^2.$$

Definition 5. *G has* spectral expansion *λ if $\lambda_2(G) \le \lambda$.*

So if G has spectral expansion λ, at each step of the Markov chain, distance to uniformity shrinks by at least λ. Note that the term spectral expansion suggests that large λ is good for expansion, but the opposite is true.

Definition 6. *Given a probability distribution π, the* collision probability *of π is* $\mathrm{Coll}(\pi) = \|\pi\|^2 = \sum_x \pi_x^2$.

Lemma 7. $\mathrm{Coll}(\pi) = \|\pi - u\|^2 + 1/N$.

Proof. Write $\pi = u + \pi^{\perp}$. Then

$$\|\pi\|^2 = \|u\|^2 + \|\pi^{\perp}\|^2 = 1/N + \|\pi - u\|^2.$$

$\square$

Note that $A\pi$ is also a probability distribution and using the lemma, we can compute the associated collision probability:

$$\mathrm{Coll}(A\pi) - 1/N = \|A\pi - u\|^2 \le \lambda^2 \|\pi - u\|^2 = \lambda^2(\mathrm{Coll}(\pi) - 1/N).$$

Given a probability distribution π, let the *support* of π be $\mathrm{support}(\pi) = \{x \,:\, \pi_x \neq 0\}$.

Lemma 8. *Let π be a probability distribution. Then* $\mathrm{Coll}(\pi) \ge 1/|\mathrm{support}(\pi)|$.

Proof. Let $m = |\mathrm{support}(\pi)|$. We claim that if $x_1 + \cdots + x_m = x$, then $x_1^2 + \cdots + x_m^2$ is minimized (with value x/m) when $x_1 = \cdots = x_m = x/m$. This easily follows from the fact that $x^2 + y^2 \ge ((x+y)/2)^2 + ((x+y)/2)^2$. Thus $\mathrm{Coll}(\pi) \ge 1/m$ and we are done. $\square$

Theorem 9. *If G has spectral expansion λ, then for all $\alpha > 1$, G has vertex expansion $\left(\alpha N, \frac{1}{(1-\alpha)\lambda^2 + \alpha}\right)$.*

Proof. Let $|S| \le \alpha N$. Choose π a probability distribution that is uniform on S and 0 on the complement of S. Then

$$Coll(\pi) = 1/|S| \qquad and \qquad Coll(A\pi) \ge 1/|support(A\pi)| = 1/|\Gamma(S)|.$$

Then

$$1/|\Gamma(S)| - 1/N \le \lambda^2(1/|S| - 1/N).$$

But $N \ge |S|/\alpha$, so solving the above inequality gives

$$|\Gamma(S)| \ge \frac{|S|}{(1-\alpha)\lambda^2 + \alpha}.$$

Thus G is an $(\alpha N, 1/((1-\alpha)\lambda^2 + \alpha))$-expander. $\square$

Now we turn to a theorem on the spectral expansion of random graphs.

Theorem 10 (Alon's Conjecture, Friedman ([Fri], 2003))**.** *For any d and any constant $\varepsilon > 0$, a random d-regular graph has spectral expansion at most $2\sqrt{d-1}/d + \varepsilon$ with probability $1 - 1/N^{\Omega(d)}$.*

This theorem says that with high probability, the spectral expansion of a random d-regular graph is approximately bounded by $2/\sqrt{d}$. The previous theorem implies that such a graph has expansion at least $d/4$. In fact, there do exist graphs with $\lambda_2 \le 2/\sqrt{d}$ and expansion greater than $d/2$.

There is a theorem of Alon and Boppana that gives a lower bound for spectral expansion, showing that Alon's Conjecture is essentially sharp.

Theorem 11 (Alon-Boppana (stated in [Alo]))**.** *Any infinite family of d-regular graphs has spectral expansion (as $N \to \infty$) at least $2\sqrt{d-1}/d - o(1)$.*

2.3.1 Expander Mixing Lemma

Heuristically, the following lemma, due to Alon and Chung [AC], says that an expander graph behaves like a random graph.

Theorem 12 (Expander Mixing Lemma, [AC] 1988). *For any subsets S and T of $V(G)$, let $e(S,T)$ denote the set of $S-T$ edges in G (edges with one endpoint in S and one endpoint in T). Let G be d-regular with $\lambda_2 = \lambda$. Then*

$$|\#e(S,T) - d|S||T|/N| \leq \lambda d\sqrt{|S||T|}.$$

Proof. Let χ_S and χ_T be the characteristic vectors of S and T respectively. First note that

$$\#e(S,T) = \sum_{u\in S, v\in T} (dA)_{uv} = \sum_{u,v} \chi_S(u)(dA)_{uv}\chi_T(v) = \chi_S^t(dA)\chi_T.$$

Write χ_S in terms of something parallel to u and $\chi_S^\perp$. Then the coefficient of u is the projection

$$\frac{\chi_S \cdot u}{\|u\|^2} = \frac{(1/N)\sum_i \chi_S(i)}{(1/N)} = |S|.$$

So

$$\chi_S = |S|u + \chi_S^\perp \qquad \text{and} \qquad \chi_T = |T|u + \chi_T^\perp.$$

(The intuition should be that the term $|S|u$ "spreads the weight evenly" and $\chi_S^\perp$ is an error term.)

Now:

$$\begin{aligned}
\#e(S,T) &= (|S|u + \chi_S^\perp)^t(dA)(|T|u + \chi_T^\perp) \\
&= d|S||T|(u\cdot u) + d|S|u^tA\chi_T^\perp + d|T|(\chi_S^\perp)^tAu + d(\chi_S^\perp)^tA\chi_T^\perp
\end{aligned}$$

Since $\chi_T^\perp \cdot u = 0$, we see that $u^tA\chi_T^\perp = 0$, and similarly $\chi_S^\perp Au = 0$. Then

$$\begin{aligned}
\#e(S,T) &= d|S||T|/N + d(\chi_S^\perp)^tA\chi_T^\perp \\
&\leq d|S||T|/N + \|\chi_S^\perp\|\|A\chi_T^\perp\| \\
&\leq d|S||T|/N + d\lambda\|\chi_S\|\|\chi_T\| \\
&= d|S||T|/N + d\lambda\sqrt{|S||T|}.
\end{aligned}$$

From the first line, it is evident that $\#e(S,T) \geq d|S||T|/N$. Thus

$$|\#e(S,T) - d|S||T|/N| \leq d\lambda\sqrt{|S||T|}.$$

□

There is a partial converse to this theorem.

Theorem 13 (Bilu-Linial, ([BL], 2004)). *Let G be a d-regular graph and fix θ. If for all $S, T \subset V$, the inequality*

$$|\#e(S,T) - d|S||T|/N| \leq \theta d\sqrt{|S||T|}$$

holds, then G has spectral expansion $\lambda = O(\theta(1 + \log(d/\theta)))$.

In particular, this means that for a d-regular graph, λ is essentially (up to $\log d$ factor), the best constant that can occur in the expander mixing lemma.

2.4 Vertex Expansion Implies Spectral Expansion

The following theorem (due to Alon) is a discrete version of Cheeger's result. We first prove for the special case when the normalized adjacency matrix A has only non-negative eigenvalues.

Theorem 14 (Alon, ([Alo], 1986))**.** *Let G be a d-regular $(N/2, 1+\alpha)$-expander and let $\lambda_2(G)$ be the second largest eigenvalue of the normalized adjacency matrix $A(G)$ of G in absolute value. If the matrix $A = A(G)$ has all non-negative eigen-values, then G is a λ-spectral expander for $\lambda = 1 - \alpha^2/(d(8+4\alpha^2))$.*

Proof. Let x be an eigenvector with eigenvalue $\lambda_2(A)$. Since $x \perp u$, the vector x has both positive and negative entries. Let $V_+ = \{i : x_i > 0\}$ and $V_- = \{i : x_i \leq 0\}$. Without loss of generality $|V_+| \leq N/2$. Let $\overline{x}$ be the vector that agrees with x on V_+ and is 0 elsewhere.

Note that $\langle \overline{x}, \overline{x} \rangle = \langle x, \overline{x} \rangle$, so it can be shown that

$$\lambda_2(A) = \frac{\lambda_s \langle x, \overline{x} \rangle}{\langle x, \overline{x} \rangle} = \frac{\lambda_2 \langle x, \overline{x} \rangle}{\langle \overline{x}, \overline{x} \rangle} = \frac{\langle Ax, \overline{x} \rangle}{\langle \overline{x}, \overline{x} \rangle}.$$

Also,

$$\begin{aligned}
\lambda_2(A) \langle \overline{x}, \overline{x} \rangle &= \langle Ax, \overline{x} \rangle \\
&= \sum_{i,j} A_{ij} x_j \overline{x}_i \\
&= \|x\|^2 - \frac{1}{d}\left(d\|x\|^2 - \sum_{i \in V_+, \{i,j\} \in E} \overline{x}_i x_j \right) \\
&= \|x\|^2 - \frac{1}{d}\left(d\|x\|^2 - 2\sum_{i,j \in V_+, \{i,j\} \in E} \overline{x}_i \overline{x}_j - \sum_{i \in V_+, j \in V_-, \{i,j\} \in E} \overline{x}_i \overline{x}_j \right) \\
&\leq \|x\|^2 - \frac{1}{d}\left(d\|x\|^2 - 2\sum_{\{i,j\} \in E} \overline{x}_i \overline{x}_j \right) \\
&= \|x\|^2 - \frac{1}{d} \sum_{\{i,j\} \in E} (\overline{x}_i - \overline{x}_j)^2 \\
\lambda_2 &\leq 1 - \frac{\sum_{\{i,j\} \in E} (\overline{x}_i - \overline{x}_j)^2}{d \sum_{i \in V} \overline{x}_i^2}. \qquad (2.1)
\end{aligned}$$

Build a new (directed) graph H as follows. Let

$$V(H) = \{s\} \cup \{v_i : i \in V_+\} \cup \{w_j : j \in V\} \cup \{t\}.$$

For all $i \in V_+$, put the arcs (s, v_i) in H with capacity $1 + \alpha$. For each $i \in V_+$ and $j \in V$ where j is a neighbor of i in G, put the arcs (v_i, w_j) in H with capacity 1. Finally, for each $j \in V$, put the arcs (w_j, t) in H with capacity 1.

We claim that the minimum cut in this graph is $(1+\alpha)|V_+|$. A cut of this size is given by the set of arcs $\{(s, v_i) \ : \ i \in V_+\}$. Given any other cut C, let $W = \{i \in V_+ \ : \ (s, v_i) \notin C\}$. For each $j \in N(W)$, there must be an arc in C adjacent to w_j. But $|N(W)| \geq (1+\alpha)|W|$, so the capacity of C must be at least $(1+\alpha)|V_+ - W| + |N(W)| \geq (1+\alpha)|V_+|$ (since $|W| \leq N/2$). So the minimum cut has capacity $(1+\alpha)|V_+|$.

By the min-cut max-flow theorem, there exists a flow on H of size $(1+\alpha)|V_+|$. In particular, note that the flow through each vertex v_i must be $1+\alpha$. Reading off the flow along arcs (v_i, w_j), it follows that there is a function $F : V \times V \to \mathbb{R}$ satisfying the following conditions (here $\tilde{E}$ denotes the set of ordered pairs (i,j) where $\{i,j\} \in E$, so that each edge in E is counted twice in $\tilde{E}$):

1. $0 \leq F(i,j) \leq 1$ for all $i, j \in V$.
2. $F(i,j) = 0$ if $i \notin V_+$ or $(i,j) \notin \tilde{E}$.
3. $\sum_{j \,:\, (i,j) \in \tilde{E}} F(i,j) = 1 + \alpha$ for each $i \in V_+$.
4. $\sum_{i \,:\, (i,j) \in \tilde{E}} F(i,j) \leq 1$ for each $j \in V$.

We need to calculate two bounds involving F in order to bound $\lambda_2(G)$. Keeping in mind that $2(a^2+b^2) \geq (a+b)^2$ for all real a and b, we find:

$$\begin{aligned}
\sum_{(i,j) \in \tilde{E}} F^2(i,j)(\overline{x}_i + \overline{x}_j)^2 &\leq 2 \sum_{(i,j) \in \tilde{E}} F^2(i,j)(\overline{x}_i^2 + \overline{x}_j^2) \\
&= 2 \sum_{i \in V} \overline{x}_i^2 \left(\sum_{(i,j) \in \tilde{E}} F^2(i,j) + \sum_{(i,j) \in \tilde{E}} F^2(j,i) \right) \\
&\leq (4 + 2\alpha^2) \sum_{i \in V} \overline{x}_i^2. \\
\sum_{(i,j) \in \tilde{E}} F(i,j)(\overline{x}_i^2 - \overline{x}_j^2) &= \sum_{i \in V} \overline{x}_i^2 \left(\sum_{(i,j) \in \tilde{E}} F(i,j) - \sum_{(i,j) \in \tilde{E}} F(j,i) \right) \\
&\geq \alpha \sum_{i \in V} \overline{x}_i^2
\end{aligned}$$

Note that in the third line, we used the fact that if $x_1 + \cdots + x_n = 1 + \alpha$ and $0 \leq x_i \leq 1$ for all i, then $x_1^2 + \cdots + x_n^2 \leq 1 + \alpha^2$. Multiplying equation (2.1) by

$$1 = \frac{\sum_{(i,j) \in \tilde{E}} F^2(i,j)(\overline{x}_i + \overline{x}_j)^2}{\sum_{(i,j) \in \tilde{E}} F^2(i,j)(\overline{x}_i + \overline{x}_j)^2}$$

and using Cauchy-Schwarz, we get

$$
\begin{aligned}
\lambda_2(G^2) &\leq 1 - \frac{\sum_{\{i,j\}\in E} (\overline{x}_i - \overline{x}_j)^2}{d \sum_{i \in V} \overline{x}_i^2} \\
&= 1 - \frac{\sum_{\{i,j\}\in E} (\overline{x}_i - \overline{x}_j)^2 \cdot \sum_{(i,j)\in \tilde{E}} F^2(i,j)(\overline{x}_i + \overline{x}_j)^2}{d \sum_{i \in V} \overline{x}_i^2 \cdot \sum_{(i,j)\in \tilde{E}} F^2(i,j)(\overline{x}_i + \overline{x}_j)^2} \\
&\leq 1 - \frac{\left(\sum_{(i,j)\in \tilde{E}} F(i,j)(\overline{x}_i^2 - \overline{x}_j^2)\right)^2}{2d(4+2\alpha^2)\left(\sum_{i\in V} \overline{x}_i^2\right)^2} \\
&\leq 1 - \frac{\alpha^2}{d(8+4\alpha^2)}.
\end{aligned}
$$

This completes the proof. □

We now move to the general case, when the eigen-values of $A(G)$ need not all be non-negative.

Corollary 15. *If G is a d-regular $(N/2, 1+\alpha)$-expander, then G is also a λ-spectral expander for $\lambda = \sqrt{1 - \alpha^2/(d^2(8+4\alpha^2))}$.*

Proof. Consider the graph G^2. If the normalized adjacency matrix of G is A, then the normalized adjacency matrix of G^2 is A^2. Also G^2 is d^2-regular and has all non-negative eigenvalues. Finally, G^2 is a $(N/2, 1+\alpha)$-expander, as follows: If S is a subset of the vertices of size at most $N/2$, then $|N(S)| \geq (1+\alpha)|S|$. Choose a subset S' of $N(S)$ with $|S| \leq |S'| \leq N/2$. Then $|N(N(S))| \geq |N(S')| \geq (1+\alpha)|S'| \geq (1+\alpha)|S|$. But $N(N(S))$ is the neighborhood of S in G^2, and S was an arbitrary subset of vertices of size at most $N/2$, so G^2 is a $(N/2, 1+\alpha)$-expander.

By Theorem 14, $\lambda_2(G^2) \leq 1 - \alpha^2/(d^2(8+4\alpha^2))$. Since the eigenvalues of G^2 are the squares of the eigenvalues of G, taking the square root of the right-hand side proves the corollary. □

Chapter 3

Random Walks

Lecturer: Prahladh Harsha
Scribe: David Arthur
April 18, 2005

3.1 Introduction

Consider an undirected graph G. A random walk of length l starting at the vertex u is a sequence of vertices $u = v_0, v_1, v_2, \cdots, v_l$, where each v_i is chosen to be a random neighbor of v_{i-1} for all $i > 0$. One considers the distribution of v_i for $i \leq l$.

Intuitively, a random walk can be thought of as choosing a globally random vertex on a graph using only local choices. This is something that people actually do in practice. For example, one might shuffle a deck of cards by repeatedly moving the top card to a random position in the deck. We can model all orderings of the deck as the vertices of a graph with edges corresponding to the operation described above. This process of repeatedly moving the top card can then be thought of as a random walk that provides a more convenient way of shuffling a deck than explicitly choosing 1 of 52! possible orderings.

Traditionally, random walks were considered on infinite graphs, and the following result is typical of what was studied.

Theorem 16 (Polya, 1921)**.** *Consider a random walk on an infinite D-dimensional grid. If $D = 2$, then with probability 1, the walk returns to the starting point an infinite number of times. If $D > 2$, then with probability 1, the walk returns to the starting point only a finite number of times.*

For the purpose of this lecture, we will consider random walks on finite undirected graphs and even more specifically, d-regular undirected graphs. Refer [Lov] for an excellent survey on Random Walks on Graphs.

Several questions will motivate this lecture. Let $\pi_0 = \pi$ be the starting distribution on the graph G (mostly, we will consider cases when π is concentrated on a single vertex). Let π_i denote the probability distribution of v_i for a random walk beginning at the starting distribution π_0. Since we are interested in the ability of random walks to generate a globally random vertex, it is natural to consider π_i as i gets large.

Question 17. *For which π_0, does π_i converge to some stationary distribution as i approaches infinity? What is the stationary distribution that it converge to?*

If we let $A = A(G)$ denote the normalized adjacency matrix of G, then it is easy to check that $\pi_{i+1} = A\pi_i$ for all i. Thus, a given distribution x is a stationary distribution for some starting distribution π_0 only if $x = Ax$. This is equivalent to stating x is an eigenvector of A with corresponding eigenvalue 1. As noted in previously lectures, the uniform distribution $u = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})$ has this property, but there could be other possible stationary distributions as well. If the graph is disconnected, then there exist multiple (independent) eigenvectors with eigenvalue 1. In fact, one can show the following.

Lemma 18. *The multiplicity of the largest eigenvalue (i.e.,. 1) in $A(G)$ is equal to the number of connected components in G.*

In particular, if G is connected, the only possible stationary distribution is u. Thus, the stationary distribution (if it exists) is independent of the starting distribution π_0. This largely answers the second part of Question 1.

Before answering the first part of Question 1, we consider a related question.

Question 19. *If π_i converges to a stationary distribution, how fast does it converge?*

We can also recast this in terms of mixing time, described below.

Definition 20 (Mixing Time)**.** *The "mixing time" of a graph G with n vertices is the minimum l such that for all starting distributions π*

$$\|A^l\pi - u\|_\infty < \frac{1}{2n}. \tag{3.1}$$

We will define the $\|\cdot\|_\infty$-norm shortly.

The $\frac{1}{2\pi}$ is largely arbitrary, but this value will prove convenient. If we take the mixing time to be infinity for graphs where no l satisfies (3.1), answering our remaining questions is equivalent to understanding mixing time.

Finally, we present two concepts related to mixing time, which are interesting in their own right.

Definition 21. *(Hitting Time) For a graph G, let $H(u, v)$ denote the expected number of steps a random walk beginning at u must take before reaching v. Define the "hitting time" of G by $H(G) = \max_{u,v} H(u, v)$.*

Definition 22. *(Cover Time) For a graph G, let C_u denote the expected number of steps a random walk beginning at u must take before reaching every other vertex at least once. Define the "cover time" of G by $C(G) = \max_u C_u$.*

It easily follows from the definitions that $H(G) \leq C(G) \leq n{\cdot}H(G)$. The latter inequality can be tightened (using the coupon-collectors' problem) to show that $C(G) \leq O(\log n) \cdot H(G)$.

3.2 Bounding the Mixing Time

As discussed above, the distribution of random walks on disconnected graphs need never converge to u. Bipartite graphs are similarly problematic. Specifically, if a random walk begins at a vertex in one part, it will always be in that part after an even number of steps, and it will always be in the other part after an odd number of steps. Thus, π_l can never converge to u on a bipartite graph.

As with Lemma 18, we can characterize this failure case in terms of eigenvalues of A.

Lemma 23. *G is bipartite iff -1 is an eigenvalue of $A(G)$.*

For example, if G is bipartite, consider the vector v with a value of 1 at all vertices in one part and a value of -1 at all other vertices. One can check this is an eigenvector of $A(G)$ with eigenvalue -1.

Now, fix a graph G with n vertices and consider the eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_n$ of $A(G)$. Without loss of generality we may assume $\lambda_1 = 1$ and $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$. Furthermore, let $\lambda = |\lambda_2|$. By Lemmas 18 and 23, we know $\lambda = 1$ iff G is either bipartite or disconnected. Therefore, G has infinite mixing time if $\lambda = 1$. We now show that conversely, if $\lambda < 1$ then G has finite mixing time.

Theorem 24. *If G is a connected, d-regular, non-bipartite graph on n vertices, then $\lambda < 1$ and G has mixing time $O\left(\frac{\log n}{1-\lambda}\right)$.*

We review the l^1, l^2 and l^∞ norms before proceeding with the proof.

Definition 25. *If $v = (v_1, v_2, \cdots, v_m)$ is an arbitrary vector, define*

$$\begin{aligned} \|v\|_\infty &= \max_i |v_i|, \\ \|v\| = \|v\|_2 &= \sqrt{\sum_i v_i^2}, \text{ and} \\ \|v\|_1 &= \sum_i |v_i|. \end{aligned}$$

Fact 26. $\|v\|_\infty \leq \|v\| \leq \|v\|_1 \leq \sqrt{n}\ \|v\|$.

The first two inequalities here can easily be verified and the third follows from the Cauchy-Schwarz inequality. Furthermore, these are all norms, which implies that they satisfy the triangle inequality.

Proof of Theorem 24: Note that $A(G)$ is a real, symmetric matrix, which implies that it has n orthonormal eigenvectors $u = v_1, v_2, \cdots, v_n$. Let π denote any (starting) probability distribution on the vertices of G. Then, we can decompose π uniquely as $\sum_{i=1}^n \pi_i$ where π_i is a constant multiple of v_i. Furthermore, as discussed in the previous lecture, the fact that

π is a probability distribution guarantees $\pi_1 = u$. Now,

$$\begin{aligned}
\|A\pi - u\|^2 &= \|Au + A\pi_2 + A\pi_3 + \cdots + A\pi_n - u\|^2 \\
&= \|\lambda_2\pi_2 + \lambda_3\pi_3 + \cdots + \lambda_n\pi_n\|^2 \text{ since } Au = u \\
&= \lambda_2^2\|\pi_2\|^2 + \lambda_3^2\|\pi_3\|^2 + \cdots + \lambda_n^2\|\pi_n\|^2 \text{ by the Pythagorean theorem} \\
&\leq \lambda^2\left(\|\pi_2\|^2 + \|\pi_3\|^2 + \ldots + \|\pi_n\|^2\right) \\
&= \lambda^2\|\pi_2 + \pi_3 + \cdots + \pi_n\|^2 \text{ again by the Pythagorean theorem} \\
&= \lambda^2\|\pi - u\|^2.
\end{aligned}$$

Thus, each step of the random walk decreases the l^2-distance of the distribution on the vertices to the uniform distance by a factor of at least λ. Therefore, $\|A^l\pi - u\| \leq \lambda^l\|\pi - u\|$ for all $l \geq 0$. It follows that

$$\begin{aligned}
\|A^l\pi - u\|_\infty &\leq \|A^l\pi - u\| \\
&\leq \lambda^l\|\pi - u\| \\
&< \lambda^l\|\pi\| \text{ since } \pi - u \text{ and } u \text{ are orthogonal} \\
&\leq \lambda^l\|\pi\|_1 \\
&= \lambda^l.
\end{aligned}$$

It follows that $\|A^l\pi - u\|_\infty < \frac{1}{2n}$ when $l = O\left(\frac{\log n}{\log\frac{1}{\lambda}}\right) \approx O\left(\frac{\log n}{1-\lambda}\right)$. To see this last step, note that $\log(1+x) = 1 - \frac{1}{1+x} + O\left(\frac{1}{(1+x)^2}\right)$ by taking the Taylor expansion of both sides. □

3.2.1 Bounding the Spectral Gap $1 - \lambda$

Since Theorem 24 depends so heavily on $1-\lambda$, it is natural to try to bound this quantity for various graphs G. We have already seen that $1 - \lambda = \Omega(1)$ for expanders. We now consider its value for other graphs.

Theorem 27. *If G is a connected, d-regular, non-bipartite graph on n vertices, then $1-\lambda \geq \frac{1}{dn^2}$.*

We will prove the result this theorem only for the case where G has only non-negative eigenvalues.

Proof. As discussed in the previous lecture, we can obtain the following characterization of the spectral gap.

$$\begin{aligned}
\lambda &= \max_{x\perp u,\|x\|=1} \langle Ax, x\rangle \text{ since } G \text{ has only non-negative eigen values} \\
&= \max_{x\perp u,\|x\|=1} \frac{1}{d}\sum_{(u,v)\in E} 2x_u x_v \\
&= \max_{x\perp u,\|x\|=1} \frac{1}{d}\sum_{(u,v)\in E} \left(x_u^2 + x_v^2 - (x_u - x_v)^2\right) \\
&= \max_{x\perp u,\|x\|=1} \frac{1}{d}\sum_{(u,v)\in E} (x_u^2 + x_v^2 - (x_u - x_v)^2 \\
&= \max_{x\perp u,\|x\|=1} \frac{1}{d}\left(d\cdot x_u^2 - \sum_{(u,v)\in E} (x_u - x_v)^2\right)
\end{aligned}$$

Hence, the spectral gap $1-\lambda$ is given by

$$1-\lambda = \min_{x\perp u,\|x\|=1} \frac{1}{d}\sum_{(u,v)\in E} (x_u - x_v)^2.$$

Then there exists x with $x \perp u$ and $\|x\| = 1$ so that $1-\lambda = \frac{1}{d}\sum_{(u,v)\in E}(x_u - x_v)^2$. Since $\|x\| = 1$, there exists v' for which $|x_{v'}| \geq \frac{1}{\sqrt{n}}$. However, since $x \perp u$, we know $\sum x_v = 0$, and hence there exists v'' for which $x_{v'}$ and $x_{v''}$ have different signs. It follows that $|x_{v'} - x_{v''}| \geq \frac{1}{\sqrt{n}}$.

Now, G is connected so there exists some shortest path $v_0(= v'), v_1, \cdots, v_k(= v'')$ from v' to v''. The triangle inequality now implies that

$$\sum_{i=0}^{k-1} |x_{v_i} - x_{v_{i+1}}| \geq |x_{v_0} - x_{v_k}| \geq \frac{1}{\sqrt{n}}.$$

Therefore,

$$\begin{aligned}
1-\lambda &= \frac{1}{d}\sum_{(u,v)\in E} (x_u - x_v)^2 \\
&\geq \frac{1}{d}\sum_{i=0}^{k-1}(x_{v_i} - x_{v_{i+1}})^2 \\
&\geq \frac{1}{dk}\left(\sum_{i=0}^{k-1}|x_{v_i} - x_{v_{i+1}}|\right)^2 \quad \text{by Fact 26} \\
&\geq \frac{1}{dkn} \geq \frac{1}{dn^2}.
\end{aligned}$$

As mentioned earlier, our proof only applies if G has no negative eigenvalues. In the general case, one can apply similar analysis to G^2 to bound $1-\lambda^2$. This gives us a weaker bound $1-\lambda^2 \geq \frac{1}{d^2n^2}$ and hence $1-\lambda \geq \frac{1}{\mathrm{poly}(n,d)}$. In fact the same bound of $\frac{1}{dn^2}$ can be obtained for the general case using a tighter analysis. □

Note that Theorems 24 and 27 imply the mixing time of any d-regular, connected, non-bipartite graph on n vertices is $O(dn^2 \log n)$.

3.2.2 A Combinatorial Notion of Mixing

So far we have related the mixing time to the spectral gap $1 - \lambda$. There is a combinatorial parameter of the graph that relates more directly to the mixing time. Suppose G is made up of two cliques joined by just a few edges. This creates a bottleneck that should intuitively limit the mixing time of G. To characterize this, one defines the following. For any set of vertices S, let $\overline{S}$ denote $V(G) - S$, and let $|E(S, \overline{S})|$ denote the number of edges between S and $\overline{S}$. Then, define

$$\begin{aligned} \Phi(S) &= \frac{|E(S,\overline{S})|}{|S|}, \text{ and} \\ \Phi(G) &= \min_{S:|S|\leq\frac{n}{2}} \Phi(S). \end{aligned}$$

$\Phi(G)$ is called the edge-expansion of the graph G. $\Phi(G)$ has a direct relation to the mixing time. The edge expansion $\Phi(G)$ is related to the spectral gap as follows:

Theorem 28. *Let G be a d-regular graph. Then,*

$$\frac{d(1-\lambda)}{2} \leq \Phi(G) \leq d\sqrt{2(1-\lambda)}.$$

3.3 Applications

3.3.1 Undirected s-t Connectivity

Let G be a d-regular, connected, non-bipartite graph with n vertices and mixing time l. Consider a random walk beginning at some vertex s. Then for any vertex t and any integer $l' \geq l$, we know that

$$\text{Prob}\left[\text{Random walk is at vertex } t \text{ after } l' \text{ steps}\right] = \left(A^{l'}\pi\right)_t \geq \frac{1}{n} - \frac{1}{2n} = \frac{1}{2n}.$$

Therefore, a random walk of length $2nl'$ will reach t with constant probability. We know that the mixing time for a d-regular connected bipartite graph is $O(dn^2 \log n)$. This suggests an algorithm for s-t connectivity. Take a random walk of length $\Theta(dn^3 \log n)$ starting at s. If the walk reaches t, then s and t are connected. Otherwise, s and t are disconnected with high probability. Note this runs in polynomial time and uses only $\log n$ space to track the current vertex.

Now, as stated, our argument relies on G being d-regular, connected and and non-bipartite to find a path from s to t. We can remove these assumptions as follows.

1. *(Connected)* Restrict to the connected component of G containing s.
2. *(Bipartite)* Add a self loop at each vertex. This does not affect whether s and t are connected and it causes the graph to be no longer bipartite.

3. *(Regular)* Each vertex of degree $D > 3$ can be replaced by D vertices of degree 3 in a cycle to make the graph 3-regular. Also, non-regular graphs do in fact mix already and the same algorithm works. We have not shown this, however.

We summarize all this as follows.

Theorem 29 (Undirected Connectivity is in RL, [AKLLR])**.** *There is a polynomial time, log space Monte Carlo algorithm for s-t connectivity in undirected graphs.*

Recently, Reingold obtained a deterministic algorithm for undirected s-t connectivity (also using expanders)[Rei]. We will cover it in future lectures.

3.3.2 Hitting Time and Cover Time

As in the previous section, let l be the mixing time of a graph G on n vertices. Consider a random walk beginning at an arbitrary vertex s. Then, recall that for $l' \geq l$, it is true for any t that

$$\text{Prob}\left[\text{Random walk is at vertex } t \text{ after } l' \text{ steps}\right] \geq \frac{1}{2n}.$$

It follows that the expected time for a random walk to reach t is at most $2n \cdot l$, so the hitting time of G is at most $2n \cdot l$, which is polynomial in n. Similarly, after $2n^2 \cdot l$ steps, the walk will have reached each vertex with high probability. Thus, the cover time of G is also polynomial in n.

Tighter results are known for both the hitting time and cover time, as summarized in the following theorems.

Theorem 30. *Let G be an arbitrary undirected graph on n vertices. Then,*

1. [BW] $H(G) \leq \frac{4}{27}n^3 - \frac{1}{9}n^2 + O(n)$, *and*
2. [Fei1] $C(G) \leq \left(\frac{4}{27} + o(1)\right) n^3$.

Theorem 31 ([Fei2])**.** *Let G be a d-regular undirected graph on n vertices. Then, $C(G) \leq 2n^2$.*

The bound given in Theorem 31 is also known to be tight.

3.3.3 Universal Traversal Sequences

Universal traversal sequences (UTS) were originally defined by Cook, and later suggested by [AKLLR] as a possible means to derandomize Theorem 29.

Consider a sequence $\mathcal{S} \in \{1, 2, \cdots, d\}^{l(n)}$ for some function l. Now, consider a d-regular undirected graph G where all the edges adjacent to each vertex have been labeled with distinct integers from 1 to d. These labellings need not be consistent in the sense that one edge might have two different labels assigned to it by two different vertices. For each vertex s, we can now use $\mathcal{S}$ to define a walk on G starting at s. Specifically, if we are at vertex v after i steps, we go to vertex v' where $\overline{vv'}$ is the edge labeled $\mathcal{S}_i$ by v. We say $\mathcal{S}$ is a "universal traversal sequence" if this walk traverses every vertex of the graph for every possible beginning vertex on every labeling of every n vertex d-regular graph. We are interested in constructing UTS of short length (i.e., $l(n) = \text{poly}(n)$).

It is not obvious how to construct a UTS or even whether one exists. However, a sufficiently lengthy random string will be a UTS with high probability.

Theorem 32. *Suppose every d-regular, n vertex graph has cover time at most C (note $C \leq 2n^2$ by Theorem 31). Then, there exists a UTS for d-regular, n vertex graphs of length at most $4ndC \log n$.*

Proof. Choose $\mathcal{S}$ uniformly at random from $\{1, 2, \cdots, d\}^{4ndC \log n}$.

Let G be a random labeled d-regular graph on n vertices and u be a random starting vertex in G. The expected time for the random walk $\mathcal{S}$ to cover every vertex of G is at most C, so by Markov's inequality, a random walk will cover every vertex within $2C$ steps with probability at least $\frac{1}{2}$. Since $\mathcal{S}$ can be decomposed into $2nd \log n$ disjoint, and hence independent, random sequences of length $2C$, it follows that $\mathcal{S}$ will cover all the vertices of G with probability at least $1 - \frac{1}{2^{2nd \log n}} = 1 - \frac{1}{n^{2nd}}$. Hence,

$$\operatorname*{Prob}_{G,u,\mathcal{S}} [\mathcal{S} \text{ covers all vertices of } G \text{ starting at } u] \geq 1 - \frac{1}{n^{2nd}}$$

.

Let N denote the number of ways of choosing a labeled d-regular graph G and a starting vertex u. For each vertex and each label, we can choose an adjacent vertex, and we can also choose one distinguished starting vertex. Thus, $N \leq n \cdot n^{nd}$.

Now, the probability that there exists one configuration (i.e., a labeled graph G and a starting vertex u) where $\mathcal{S}$ does not cover every vertex is at most $N \cdot \frac{1}{n^{2nd}} \leq \frac{n}{n^{nd}} = o(1)$. Thus, with high probability, $\mathcal{S}$ is a UTS. The result follows. ∎

To use a UTS to derandomize Theorem 29, this result is insufficient. For that purpose, we would need to construct a UTS deterministically using logarithmic space. In general, such a construction has not been found. However, we know how to construct UTS if we relax either the restriction that the length of the UTS must be polynomial or the restriction that the UTS holds good for all d-regular graphs on n vertices. The following is a flavor of such results under such relaxations.

Theorem 33.

- [Ist] *A UTS (of polynomial length) can be constructed deterministically in $O(\log n)$ space for cycles.*
- [HW] *A UTS (of polynomial length) can be constructed deterministically in $O(\log n)$ space for d-regular expanders which are "consistently labelled"*[1]

3. [Rei] *A UTS (of polynomial length) can be constructed deterministically in $O(\log n)$ space for d-regular undirected graphs which are consistently labelled.*
4. [Nis] *A UTS, of length $O(n^{\log n})$, can be constructed deterministically in $O(\log^2 n)$ space for general d-regular graphs with general labellings.*

[1]A labeling is said to be be consistent if for any two edges $e_1 = (u_1, v)$ and $e_2 = (u_2, v)$ incident on the same vertex v, it is the case that the label of e_1 wrt u_1 is different from that of e_2 wrt u_2.

3.3.4 Random Walks on Expanders

Finally, we apply Theorem 24 to the case of expanders. We know the mixing time is $O(\frac{\log n}{1-\lambda})$ but λ is bounded by a constant for expanders. Thus, the mixing time on an expander is just $O(\log n)$, which is the best possible (up to constant factors) since the diameter of an expander is $O(\log n)$. It follows that expanders are rapidly mixing, which will allow for some applications to derandomization in the next lecture.

Chapter 4

Derandomization

Lecturer: Cynthia Dwork
Scribe: Adam Barth & Prahladh Harsha
April 25, 2005

In today's (and the next) lecture(s), we will discuss applications of expanders in the context of derandomization. The three applications we will consider are the following:

- Use of random walks on expanders as an error reduction technique for randomized algorithms.
- a pseudo-random generator to fool space bounded machines.
- Derandomized linearity testing

We will discuss the first two applications in this lecture and postpone the linearity testing to the next lecture.

4.1 RP error reduction

Consider an RP algorithm with constant error probability that uses r random bits. We will improve the error probability to 2^{-k} with $r + O(k)$ random bits. Compare this with (1) the brute force k-independent trials, which would require $O(kr)$ random bits to achieve the same error probability and (2) the technique due to Karp, Pippenger and Sipser [KPS] (discussed in Lecture 1) which uses r random bits (i.e., no extra random bits) and achieves an error probability of $1/\text{poly}(r)$. We will use random walks on expanders to reduce the error of RP algorithms. Ajtai, Komlos and Szemeredi first used random walks on expanders in the context of small-space derandomization [AKS]. The proof we present in lecture is due to Impagliazzo and Zuckerman [IZ].

The KPS technique, though great in terms of the number of extra random bits being used is limited by the fact that the running time of the improved algorithm is at least $\mathrm{poly}(1/\delta)$ where δ is the (new reduced) error of the algorithm. Hence, we can reduce the error to at most 1/poly. The technique, discussed today, will further reduce the error to 2^{-k} at the cost of only $O(k)$ extra random bits as opposed $O(rk)$ random bits in the k independent trails, while the algorithm still runs in (randomized) polynomial time.

As in KPS, we will use a d-regular expander with $V = \{0,1\}^r$, thus $|V| = 2^r$ and d is a constant. As in KPS, we will assume that there exists an implicit construction of such expanders in the following sense: given any vertex v and any index i in the range $1 \ldots d$ (where d is the degree of the expander), we can in time polynomial in $|v|$ and $|i|$, compute the i^{th}-neighbor of v. The expanders constructions we will discuss later in the course will satisfy such strong properties.

Recall that to find witnesses, KPS began at a random vertex and completely explored all vertices within a ball of radius $O(k)$. Here, we also start at a random vertex but instead of exploring all vertices in a ball, we will walk randomly for k steps and run the original RP algorithm along all vertices along this random walk. Thus the total randomness uses is at most $r + k \log d$ since $\log d$ bits are required to choose a random neighbor.

Clearly, if the input is a NO instance, then this new algorithm will also reject. Our concern is that there might exist YES instances, for which the random walk fails to arrive at even one membership witness. The following theorem shows that this is highly unlikely.

Theorem 34 (Hitting Property of Expander Random Walks)**.** *Given a graph $G = (V, E)$ with spectral expansion λ and $B \subset V$, the probability a random walk of length k, starting from a random vertex $r_0 \in_R V$, starts and remains in B is $\leq \left(\mu^2 + \lambda^2\right)^{k/2}$, where $\mu = |B|/|V|$ is the density of B.*

For every RP language L and every $x \in L$, $|W_x|/2^r \geq 3/4$. For our application to derandomization, we set $B = V \setminus W_x$ and obtain the required error-reduction for RP.

Proof. We wish to bound $\Pr_{r_0,\ldots,r_k}\left[r_0, \ldots, r_k \in B\right]$, where r_i is the ith vertex encountered in the random walk on G.

Let A be the normalized adjacency matrix for G, where the vertices of G are ordered such that the first $|B|$ vertices are the elements of B. Fix P to be the projection matrix onto B, that is

$$P = \left(\begin{array}{c|c} I_{|B|\times|B|} & 0_{|B|\times|V\setminus B|} \\ \hline 0_{|V\setminus B|\times|B|} & 0_{|V\setminus B|\times|V\setminus B|} \end{array}\right).$$

In other words, $P_{i,j} = 1$ if $i = j$ and $i, j \in 1, \ldots, |B|$ and is 0 otherwise. For any distribution π on the set of vertices V, note that $\|P\pi\|_1$ is the probability that a vertex chosen according to π is in the set B.

Fix u to be the uniform distribution on V. As mentioned above, $\|Pu\|_1$ is the probability a uniformly randomly selected vertex lies in B, *i.e.* $\|Pu\|_1 = \mu$. Similarly, $\|P(AP)u\|_1$ is the probability r_1 is also in B, *i.e.* the probability both $r_0, r_1 \in B$. By an inductive argument, we seek to bound $\|P(AP)^k u\|_1$. Observe $\|(PAP)^k u\|_1 = \|P(AP)^k u\|_1$, as P is idempotent. It will be more convenient to work with $\|(PAP)^k u\|_1$ than $\|P(AP)^k u\|_1$. We now switch to the L_2 norm and will later return to the L_1 norm.

We first show that a single application of PAP to any vector x reduces its L_2-norm by a factor of $\sqrt{\mu^2+\lambda^2}$.

Claim 35. *$\forall x \in \mathbb{R}^n$, $\|(PAP)x\|_2 = \sqrt{\mu^2+\lambda^2} \cdot \|x\|_2$.*

Assuming this claim, we complete the proof of the theorem. Applying the claim k times, we obtain

$$\|(PAP)^k x\|_2 \leq \left(\mu^2+\lambda^2\right)^{k/2} \cdot \|x\|_2.$$

We now return to the L_1 norm.

$$\begin{aligned}
\|(PAP)^k u\|_1 &\leq \sqrt{N}\|(PAP)^k u\|_2 && \text{By Cauchy-Schwarz Inequality} \\
&\leq \sqrt{N}\left(\mu^2+\lambda^2\right)^{k/2}\|u\|_2 \\
&= \left(\mu^2+\lambda^2\right)^{k/2}.
\end{aligned}$$

□

We now prove Claim 35.

Proof of Claim 35: The main intuition behind the proof is that A reduces the length of component of the vector x that is orthogonal to u while P reduces the length of the component of x along u. Together, they reduce the length of x.

Fix $y = Px$ and split $y = y^{\|} + y^{\perp}$, where $y^{\|}$ is parallel to u and $y^{\perp}$ is perpendicular to u. By the triangle inequality and $Ay^{\|} = y^{\|}$, $\|PAy\|_2 \leq \|Py^{\|}\|_2 + \|PAy^{\perp}\|_2$. Because of the second eigenvalue, $\|Ay^{\perp}\|_2 \leq \lambda\|y^{\perp}\|_2 \leq \lambda\|y\|_2 \leq \lambda\|x\|_2$ (since the length of y is at most that of x, recall that y is the projection of x). Therefore,

$$\|PAy^{\perp}\|_2 \leq \|Ay^{\perp}\|_2 \leq \lambda\|x\|_2 \tag{4.1}$$

As for the $y^{\|}$ term,

$$y^{\|} = \left(\frac{y \cdot u}{\|u\|_2^2}\right) u, \text{ which implies } y^{\|} = \left(\sum_i y_i\right) u.$$

By observing $y = Px$ and so y has support $|B| = \mu N$.

$$\begin{aligned}
\|y^{\|}\|_2^2 &= \sum_{i=1}^{N} \frac{(\sum_{i=1}^{\mu N} y_i)^2}{N^2} \\
&= \frac{(\sum_{i=1}^{\mu N} y_i)^2}{N} \\
&\leq \frac{\mu N \left(\sum_i y_i^2\right)}{N} && \text{(By Cauchy-Schwarz inequality)} \\
&= \mu\|y\|_2^2
\end{aligned}$$

On the other hand, since $y^{\|} = (\sum_i y_i)\, u$, we have that $(Py^{\|})_j = (\sum_i y_i)/N$ for $j = 1, \ldots, \mu N$ and 0 otherwise. Hence,

$$\begin{aligned} \|Py^{\|}\|_2^2 &= \sum_{j=1}^{\mu N} \frac{\left(\sum_i y_i\right)^2}{N^2} \\ &= \mu N \frac{\left(\sum_i y_i\right)^2}{N^2} \\ &= \mu \|y^{\|}\|_2^2 \end{aligned}$$

Combining the two we have, $\|Py^{\|}\|_2^2 \le \mu\|y^{\|}\|_2^2 \le \mu^2\|y\|_2^2 \le \mu^2\|x\|_2^2$. Combining equation (4.1) we have $\|(PAP)x\|_2^2 \le (\mu^2 + \lambda^2)\|x\|_2^2$. Hence, $\|(PAP)x\|_2 \le \sqrt{\mu^2 + \lambda^2}\|x\|_2$ $\square$

In this result about RP, we worry about not hitting a witness. For a similar result about BPP, however, we need to show we encounter approximately the correct fraction of appropriate witnesses. For random walks on the complete graph (*i.e.* independent trials), Chernoff bounds tell us the witness fraction will be close to the appropriate ratio with high probability. It is possible to obtain a similar Chernoff bound for random walks on expanders, but we omit the details.

4.2 PRGs for Space Bounded Computation

The general idea of pseudo-random generators is to output a long string from a short, truly random seed such that some restricted class of adversaries (typically time-bounded adversaries) cannot distinguish (with greater than some probability) the long string from a long string of truly random bits. We think of a generator as taking a seed of length $s(n)$ and producing a string of length $f(n)$.

Definition 36. *Let M be a randomized Turing machine using, on input w, $f(|w|)$ random bits. The family $\{g_n\}_{n=1}^{\infty}$ of functions $g_n : \{0,1\}^{s(n)} \to \{0,1\}^{f(n)}$ is an ϵ-generator for M if, for all w,*

$$\left| \Pr_{r \in \{0,1\}^{f(|w|)}} \Big[M(w,r) \text{ accepts} \Big] - \Pr_{z \in \{0,1\}^{s(|w|)}} \Big[M(w, g_{|w|}(z)) \text{ accepts} \Big] \right| \le \epsilon$$

Typically, pseudo-random generators are constructed to fool time-bounded adversaries. Here, we consider constructing a generator to fool (randomized) space-bounded adversaries (specifically logspace machines).

4.2.1 Randomized Logspace TM

Before proceeding any further, we have to clarify a point regarding how the random bits are accessed in a randomized space bounded computation (logspace in our case). There are 2 varying definitions of randomized space bounded TMs based on the manner the random bits are accessed

- The TM obtains the random bits as and when required by it.

- The string of random bits is fed as an auxiliary off-line input (on a separate tape) in addition to the regular input to the TM. In this case, the head accessing the random bits on this tape can move back and forth on the tape.

It is to be noted that in the case of randomized time bounded computation it is immaterial which convention we observe. For randomized space bounded computation, we shall consider only TMs of the first kind or equivalently consider TMs of the second kind in which the head on the random tape is restricted in the sense that it can only move right along the tape (i.e., the random tape is one-way read-only tape) Recall that a space S-bounded machine is also effectively time-bounded, for some large time bound 2^S.

4.2.2 Nisan's Generator

In this lecture, we will construct a pseudo-random generator for space bounded machines using expanders. The first such PRG construction was given by Nisan [Nis]. Nisan's construction used hash functions instead of expanders. In this lecture, we give the construction due to Impagliazzo, Nisan and Wigderson [INW], that uses expanders.

Typically, we would like the generator to also run within the space-bound S. If this were the case, we would be able to completely derandomize the randomized space S-bounded machine. Unfortunately, we won't be able to achieve something as strong as that. Instead, we let the generator use more space (specifically S^2 space) than the space S-bounded TM it fools.

We will prove the following theorem in today's lecture.

Theorem 37. *There exists a n-space-bounded Turing machine $G : \{0,1\}^{O(\log^2 m)} \to \{0,1\}^m$ such that for all randomized S-space-bounded Turing machines M, G is a $(1/2^S)$-generator for M, where $m = 2^S$ and $n = O(\log^2 m)$.*

4.2.3 Proof of Theorem 37

Before going into proving the existence of a PRG as mentioned in Theorem 37, we shall first study the structure of the computation tableau of a randomized space S TM and find how this structure can be exploited to reduce the randomness. The computation tableau for a randomized space S TM is as shown in Figure 1, i.e., it is a very thin (width at most S), but possibly very long (length can be as long as 2^S) tableau.

In the original definition of the randomized TM, the computation requires at most 2^S random bits. We will break the tableau into several components (see Figure 4.2), each of which require exactly R random bits. Thus, there are at most $2^S/R$ such components. For simplicity, we will assume that there are actually 2^S components. R will be typically $\Theta(S)$ for our purposes, but our proof will work even for larger R.

Consider the first two components A_1 and A_2 both of which require random strings r_1 and r_2 respectively each of length R. If r_1 and r_2 are chosen independently, then by definition the 2 components work to give the right results. We would like to choose r_1 and r_2 in such a manner that their behavior is not significantly different from the case when r_1 and r_2 are chosen independently. We would now use the fact that the computation tableau is very thin (more specifically, at most S bits are communicated between the two components A_1 and A_2) to let us choose r_1 and r_2 in a manner better than independently.

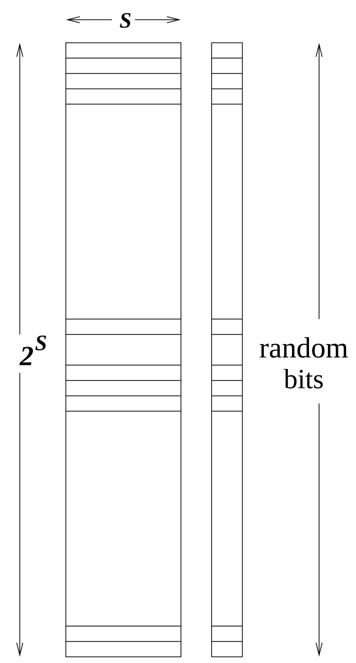

Figure 4.1: The Computation Tableau of a space S TM

To put things more formally, we have 2 algorithms A_1 and A_2 such that

- A_1 takes an input x_1 of length r and outputs a string b_1 of length c.
- A_2 takes as input the output b_1 of A_1 and another string x_2 of length r and outputs a string b_2 of length c (see Figure 4.3).

What we are in search of is a generator that supplies strings x_1 and x_2 in a fashion better than choosing them independently. For notational brevity, given a function g, define functions g^l and g^r such that $g^l(z)$ and $g^r(z)$ denote the left half and the right half of the string $g(z)$ (i.e., $g(z) = g^l(z) \circ g^r(z)$ and $|g^l(z)| = |g^r(z)|$)[1]. See Figure 4.4.

Definition 38. *A function g ($g : \{0,1\}^t \to \{0,1\}^r \times \{0,1\}^r$) is defined to be a $\epsilon-$ generator for communication c if for all functions A_1 and A_2 such that $A_1 : \{0,1\}^r \to \{0,1\}^c$ and $A_2 : \{0,1\}^c \times \{0,1\}^r \to \{0,1\}^c$, we have that*

$$\forall b \qquad \left| \operatorname*{Prob}_{x_1,x_2 \in \{0,1\}^r} [A_2(A_1(x_1), x_2) = b] - \operatorname*{Prob}_{z \in \{0,1\}^t} \left[A_2(A_1(g^l(z)), g^r(z)) = b \right] \right| < \epsilon$$

For notational convenience, we shall call a $2^{-2c}-$generator for communication c a $c-$generator.

For the present we shall assume the following lemma and present its proof later (in Section 4.3).

Lemma 39. *There exists a constant $k > 0$ such that for all r, c, there exists a polynomial time and linear space computable $c-$generator g where g is such that $g : \{0,1\}^{r+kc} \to \{0,1\}^r \times \{0,1\}^r$.*

[1] $\circ$ denotes the concatenation operator

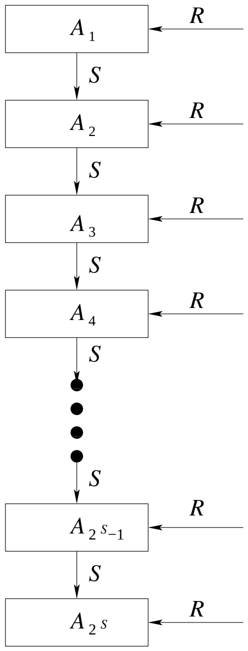

Figure 4.2: Breaking the tableau into components each requiring R random bits

$$\begin{tikzcd}
A_1 \arrow[d, "b_1"] & x_1 \arrow[l] \\
A_2 \arrow[d] & x_2 \arrow[l] \\
b_2 &
\end{tikzcd}$$

Figure 4.3: A_1, A_2 with random inputs

Given such a c-generator, we can generate pseudo-random strings for every pair of successive components (A_{2i-1} and A_{2i}), such that their behavior is almost similar to the case when pure random strings are fed to all the components. More formally, we let $g_1 : \{0,1\}^{R+kS} \to \{0,1\}^R \times \{0,1\}^R$ be the S-generator guaranteed by lemma 39 (i.e., by setting $r = R$ and $c = S$ in the lemma). For every pair of components (A_{2i-1} and A_{2i}), instead of feeding them each with pure random strings of length R, we now take one random string of length $R + kS$, run the generator g_1 on this string and feed the output of the generator to the two components A_{2i-1} and A_{2i} (See Figure 4.5). By doing so, we require only $2^{S-1} \cdot (R + KS)$ random bits as opposed to $2^S \cdot R$ random bits. Furthermore, each application of the generator g_1 causes an error of at most $1/2^{2S}$ (since g_1 is a S-generator). Hence, the total error incurred is at most $2^{S-1}/2^{2S}$ since we run the generator g_1 at most 2^{S-1} times.

We now have 2^{S-1} components each of which require $R+kS$ random bits (see Figure 4.5). Furthermore, as before each pair of successive components communicate at most S bits of

$$\begin{tikzcd} A_1 \arrow[d, "b_1"] & g^l(z) \arrow[l] \\ A_2 \arrow[d] & g^r(z) \arrow[l] \\ b_2 & \end{tikzcd}$$

Figure 4.4: A_1, A_2 with inputs from generator

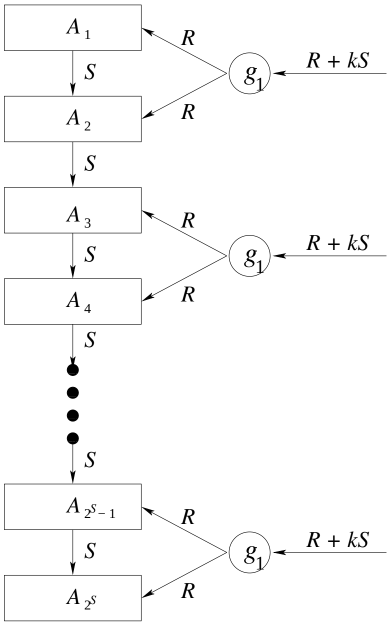

Figure 4.5: Using S-generator to save randomness

information. Hence, we can once again apply the generator to reduce the number of random bits required by every pair of components from $R + kS$ each to $R + 2kS$ total. Moreover, we can perform this operation repeatedly till we finally have just one component left. More formally we do the following (also see Figure 4.6).

Let $g_i : \{0,1\}^{R+ikS} \to \{0,1\}^{R+(i-1)kS} \times \{0,1\}^{R+(i-1)kS}$ be a $S-$generator for $i = 1, 2, \ldots, S$ as guaranteed by lemma 39 (i.e., by setting $r = R + (i-1)kS$ and $c = S$ in the lemma). Define functions $G_i : \{0,1\}^{R+ikS} \to \{0,1\}^{2^i \cdot R}$ for $i = 0, 1, \ldots S$ inductively as follows

$$\begin{aligned} G_0(z) &= z \\ G_i(z) &= G_{i-1}(g_i^l(z)) \circ G_{i-1}(g_i^r(z)) \end{aligned}$$

We shall show that the existence of G_S implies Theorem 37. Clearly, by definition of G_S, G_S is a Space(n) TM (i.e., it runs in space $O(R + kS^2)$). We only have to show that G_S fools all randomized space S TMs, which is implied by the following lemma.

Lemma 40. *For all* TMs *A that run in space S and in time 2^S, G_S is an $2^{-S}-$generator for A*

Proof. Each application of the generator g_i, for any i, incurs an error of at most $1/2^{2S}$ (as guaranteed by Lemma 39). There are at most $2^{S-1}+2^{S-2}+\cdots+2+1 = 2^S-1$ applications of the generator g_i (over all i) (see Figure 4.6). Hence, the maximum error incurred is at most $(2^S-1)/2^{2S} < 1/2^S$. Thus, proved.

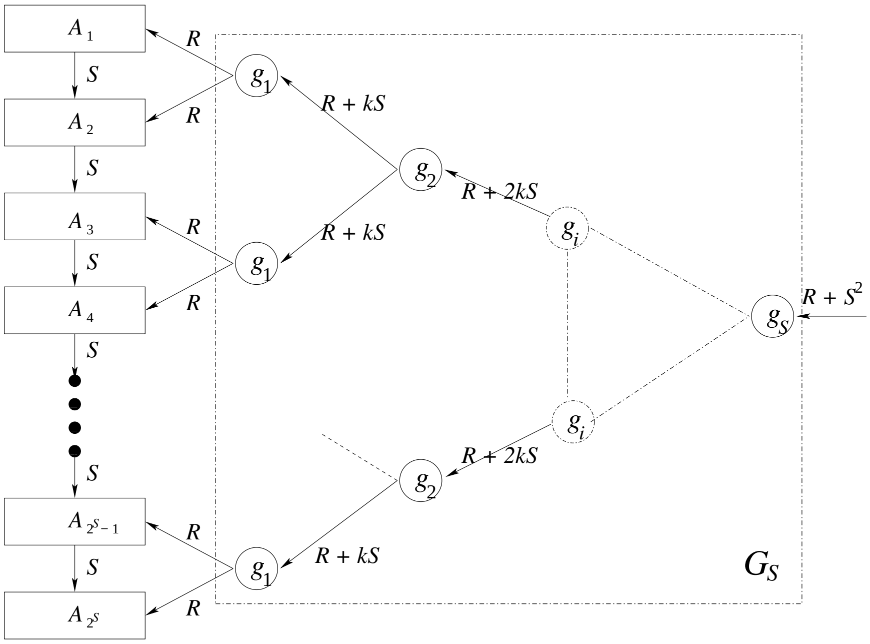

Figure 4.6: Pseudo-random generator G_S for Space S machines

□

4.3 Proof of Lemma 39 using Expanders

We prove Lemma 39 using the expander mixing lemma

Lemma 41 (Expander Mixing Lemma)**.** *If $G = (V, E)$ is a D-regular graph with spectral expansion λ, then for all sets S and $T \subseteq V$, we have*

$$\left|\frac{e(S,T)}{|E|} - \frac{|S|}{|V|}\cdot\frac{|T|}{|V|}\right| \le \lambda\sqrt{\frac{|S|}{|V|}\cdot\frac{|T|}{|V|}} \le \lambda,$$

where $e(S,T)$ denotes the number of edges between the sets S and T.

Lemma 39 basically tells us that Figure 4.3 can be replaced by Figure 4.4. In other words, the $\epsilon-$generator for communication $r+kc$, g, should construct strings $g^l(z)$ and $g^r(z)$ such that functions A_1 and A_2 are fooled into believing that these strings were random ones. The main idea is to view the set of strings $\{0,1\}^r$ as the vertices of an expander $G = (V, E)$ and choose the strings $g^l(z)$ and $g^r(z)$ to be the endpoints of a random edge of the expander.

The actual construction of the c-generator is as follows: Let $G = (V, E)$ be a $D = 2^{6c}$-regular Ramanujan expander graph on $|V| = 2^r$ vertices. (Recall that a D-regular Ramanujan graph is a D-regular graph with the best possible spectral expansion, namely $\lambda \approx 1/\sqrt{D}$. Such graphs are constructed by Lubotsky, Philips and Sarnark [LPS]). Note that for super-constant c, the above expander has super-constant degree. We can either construct such a Ramanujan expander explicitly or start with a constant degree Ramanujan expander and then take a suitable power of it to increase the degree). The generator $g : \{0,1\}^{r+6c} \to \{0,1\}^r \times \{0,1\}^r$ works as follows: On input $z = (x, i) \in \{0,1\}^r \times \{0,1\}^{d(=6c)}$, output $(g^l(z), g^r(z)) = (x, y)$ where y is the vertex reached by taking the i^{th} edge out of x.

Proof of Lemma 39: Let b be any string that is a possible output of the pair of algorithms (A_1, A_2). For every $b' \in \{0,1\}^c$, define the following:

$$\begin{aligned} S_{b'} &= \left\{x \in \{0,1\}^r | A_1(x) = b'\right\} \\ T_{b'} &= \left\{x \in \{0,1\}^r | A_2(b', x) = b\right\} \end{aligned}$$

i.e., if $x_1 \in S_{b'}$ and $x_2 \in T_{b'}$, $A_1(x_1) = b'$ and $A_2(b', x_2) = b$. Hence,

$$\begin{aligned} \operatorname*{Prob}_{x_1,x_2}[A_2(A_1(x_1), x_2) = b] &= \sum_{b' \in \{0,1\}^c} \operatorname*{Prob}_{x_1,x_2}[x_1 \in S_{b'} \wedge x_2 \in T_{b'}] \\ &= \sum_{b' \in \{0,1\}^c} \frac{|S_{b'}|}{|V|} \cdot \frac{|T_{b'}|}{|V|} \end{aligned}$$

Similarly,

$$\begin{aligned} \operatorname*{Prob}_{z \in \{0,1\}^{r+d}}\left[A_2(A_1(g^l(z)), g^r(z)) = b\right] &= \sum_{b' \in \{0,1\}^c} \operatorname*{Prob}_{(x,i)}[x \in S_{b'} \wedge (\ i\text{th edge out of } x \text{ leads to } T_{b'})] \\ &= \sum_{b' \in \{0,1\}^c} \frac{e(S_{b'}, T_{b'})}{|E|} \end{aligned}$$

Hence,

$$\begin{aligned}
&\left|\operatorname*{Prob}_{x_1,x_2}[A_2(A_1(x_1),x_2)=b] - \operatorname*{Prob}_{z\in\{0,1\}^{r+d}}\left[A_2(A_1(g^l(z)),g^r(z))=b\right]\right| \\
= \quad & \left|\sum_{b'\in 0,1^c} \frac{|S_{b'}|}{|V|}\cdot\frac{|T_{b'}|}{|V|} - \frac{e(S_{b'},T_{b'})}{|E|}\right| \\
\leq \quad & \sum_{b'\in 0,1^c} \left|\frac{|S_{b'}|}{|V|}\cdot\frac{|T_{b'}|}{|V|} - \frac{e(S_{b'},T_{b'})}{|E|}\right| \\
\leq \quad & \sum_{b'\in 0,1^c} \lambda \qquad \text{(By Expander Mixing Lemma)} \\
\leq \quad & 2^c\lambda \\
\leq \quad & 2^c\cdot\frac{1}{\sqrt{D}} \\
= \quad & 2^c\cdot\frac{1}{2^{3c}} \\
= \quad & \frac{1}{2^{2c}}
\end{aligned}$$

Thus, proved. □

Chapter 5

Derandomization (Part II)

Lecturer: Prahladh Harsha
Scribe: Adam Barth
May 1, 2005

Today we will use expanders to derandomize the algorithm for linearity test.

Before presenting the linearity testing algorithm and its derandomization, we review some expander preliminaries.

5.1 Expander Preliminaries

So far we have considered vertex expansion. For the derandomized linearity testing, we will need a notion of edge-expansion. Informally, edge-expansion implies that every small set of vertices has a large number of edges leaving the set. We show below that any graph that is an expander (i.e., has spectral expansion λ) is also an edge-expander.

Lemma 42 (Edge expansion)**.** *If G has spectral expansion λ, then for all $A \subseteq V$ with $|A| \leq n/2$,*

$$e(S, \overline{S}) \geq \frac{d(1-\lambda)}{2}|S|$$

where $e(S, \overline{S})$ denotes the number of edges between S and $\overline{S}$.

Proof. Given a vertex set G on n vertices with adjacency matrix A, recall that $\lambda = \max_{x \perp u}\langle Ax, x\rangle / \langle x, x\rangle$. For a subset $S \subseteq V$ of vertices, fix

$$x = \frac{\chi_S}{|S|} - \frac{\chi_{\overline{S}}}{|\overline{S}|}.$$

In other words, x is the n-dimension vector such that

$$x_v = \begin{cases} \frac{1}{|S|} & \text{if } v \in S \\ -\frac{1}{|\overline{S}|} & \text{if } v \notin S \end{cases}$$

Observe the following: $x \perp u$, and $\langle x, x\rangle = \frac{1}{|S|} + \frac{1}{|\overline{S}|}$, and

$$\begin{aligned}
\langle Ax, x\rangle &= \sum a_{ij}x_ix_j \\
&= \sum_{i,j\in S} \frac{2}{d}x_ix_j + \sum_{i,j\in \overline{S}} \frac{2}{d}x_ix_j + \sum_{i\in S, j\in \overline{S}} \frac{2}{d}x_ix_j \\
&= \frac{2}{d|S|^2}\left(\frac{d|S| - e(S,\overline{S})}{2}\right) + \frac{2}{d|\overline{S}|^2}\left(\frac{d|\overline{S}| - e(S,\overline{S})}{2}\right) - \frac{2e(S,\overline{S})}{d|S||\overline{S}|} \\
&= \left(\frac{1}{|S|} + \frac{1}{|\overline{S}|}\right)\left[1 - \frac{e(S,\overline{S})}{d}\left(\frac{1}{|S|} + \frac{1}{|\overline{S}|}\right)\right],
\end{aligned}$$

Since $\langle Ax, x\rangle \leq \lambda\langle x, x\rangle$, we have that

$$1 - \frac{e(S,\overline{S})}{d}\left[\frac{1}{|S|} + \frac{1}{|\overline{S}|}\right] \leq \lambda.$$

which implies

$$e(S,\overline{S}) \geq d(1-\lambda)\frac{|S||\overline{S}|}{|S| + |\overline{S}|} \geq \frac{d(1-\lambda)}{2}|S|$$

since $|S| \leq n/2$. □

Suppose you remove a few edges from a graph. It is possible we might have partitioned the graph into several small (disconnected) pieces. However, if the graph is an expander, there must exist a huge connected component. This is captured in the following lemma.

Lemma 43. *For all $\delta \leq (1-\lambda)/12$, after removing any $2\delta dn$ edges from a graph G with spectral expansion at most λ, there exists a connected component of size at least*

$$\left(1 - \frac{4\delta}{1-\lambda}\right)n.$$

Proof. We prove this lemma in two steps. If removing the edges partitions the graph into two halves, these two halves must be unbalanced with the smaller side containing less than $n/3$ vertices. More precisely, fix a partition $S, \overline{S}$ of G such that the edges in $e(S,\overline{S})$ is contained in the set of removed edges and S is the smaller half (i.e., $|S| \leq n/2$). Then (by Lemma 42),

$$\frac{d(1-\lambda)|S|}{2} < e(S,\overline{S}) \leq 2\delta dn \text{ implies } |S| < \frac{4\delta}{1-\lambda}n \leq \frac{n}{3}.$$

Therefore, if there is a component of size at least $n/2$, then there is one of size $1 - \frac{4\delta}{1-\lambda} \geq 2n/3$. On the other hand, the graph (on removal of the edges) could consist of several small components (of size less than $n/3$) and not have any large component. The following claim shows that this cannot be the case

Claim 44. *The union of all components of size less than $n/3$ is itself of size less than $n/3$.*

Proof. Consider two components C_1 and C_2, each of size less than $n/3$. Their union is of size at most $|C_1 \cup C_2| < 2n/3$. We know from above that any component is of size greater than $2n/3$ or less than $n/3$. Hence, $|C_1 \cup C_2| < n/3$. We could repeatedly do this for all components of size less than $n/3$ to show that their union is of size at most $n/3$. $\square$

Thus, there must exist a large sized component and we are done in this case. $\square$

5.2 Linearity testing

Linearity testing is an instance of the more general problem of property testing [RS, GGR]. In general property testing, the goal is to check whether a huge string has a specific property. The string is so huge that one can not afford to read it in its entirety.

For example, given the adjacency matrix of a huge graph G, suppose we wish to design an algorithm A (also called a (property) tester) to determine whether the graph is bipartite without reading the entire matrix. Clearly, A can not determine exactly whether the graph is bipartite without looking at the entire graph because a single edge may destroy the property of being bipartite. Therefore, we relax the requirements and require A to only distinguish between the cases when G is bipartite and when G is "far" from being bipartite[1] rather than the cases when G is bipartite and when it is non bipartite. Note that A must be randomized because otherwise an adversary could fool A by placing "bad" edges in parts of the matrix A does not inspect. We thus, require the following of the tester A:

- If the graph is bipartite, A must accept with probability at least $2/3$.
- If the graph is "far" from bipartite, A must reject with probability at least $2/3$.

We know to design testers A which satisfy the above properties and needs to probe at most a constant number of locations of the matrix (the precise constant depends on how "far" we want the graph to be from bipartite) [GGR].

We now consider the linearity testing of Blum, Luby and Rubinfeld [BLR]. In linearity testing, the string we wish to test is a function from $\mathbb{Z}_2^n$ to $\mathbb{Z}_2$, presented as a table. Our proofs will work for the more general case when $f : G \to H$ where G and H are arbitrary groups (not even abelian). For simplicity, we will assume that the groups G and H are abelian. Also it is a good idea to consider the case $G = \mathbb{Z}_2^n$ and $H = \mathbb{Z}_2$. The table lists the value of the function for each input value $x \in G$.

Definition 45.
- *A function $f : G \to H$ is said to be* linear *if for all $x, y \in G$, we have $f(x) + f(y) = f(x + y)$.*
- *A function $h : G \to H$ is said to be an* affine function*, if there exists a linear function $f : G \to H$ and a constant $a \in H$ such that for all $x \in G$, we have $h(x) = f(x) + a$.*

[1] We say G is "far" from being bipartite if a "lot of edges" need to be removed in order to make it bipartite

We say that a function is δ-far (δ-close) from linear, if the value of the function for at least (at most) δ-fraction of the points in G needs to be changed in order to make it linear.

The goal in linearity testing is to test whether the given function (specified as a table of values) is linear or far from linear, without reading the entire table. Linearity testing has applications to locally testable codes (Hadamard codes) and to probabilistically checkable proof constructions.

Blum, Luby and Rubinfeld proposed the following simple linearity testing algorithm (see Figure 5.1) [BLR].

> $\mathsf{LT}(G, H)$
>
> Input: function $f : G \to H$, specified as a table of values.
>
> 1. Choose $x, y \in_R G$ uniformly at random.
> 2. Query the table for $f(x), f(y)$ and $f(x+y)$.
> 3. Check whether $f(x+y) = f(x) + f(y)$. If the check succeeds, LT accepts f, otherwise, LT rejects f.

Figure 5.1: Linearity Test of Blum, Luby and Rubinfeld [BLR]

We state, without proof, two properties of LT.

Proposition 46.

- Completeness: *If f is linear, then* $\Pr\left[\mathsf{LT} \textit{ accepts } f\right] = 1$.
- Soundness: *If f is δ-far from linear, then* $\Pr\left[\mathsf{LT} \textit{ accepts } f\right] < 1 - O(\delta)$.

The number of random bits used by LT is $2\log|G|$ ($= 2n$ in the case when $G = \mathbb{Z}_2^n$) because LT selects two element of G uniformly at random. The main question we will address is whether this randomness can be further reduced. Goldreich and Sudan showed that at least $log|G| - O(1)$ random bits is required [GS]. They also suggested that the random bits can be reduced by selecting the second point y from a smaller set S instead of the entire group G. Ben-Sasson et.al. showed that a set with the following properties suffices [BSVW].

1. $s \in S$ implies $-s \in S$.
2. The Cayley graph $G_S = (V_S, E_S)$, where $V_S = G$ and $E_S = \{(x, x+s) \mid x \in G, s \in S\}$, must have spectral expansion λ.

With such a set S in hand, we modify LT to choose $x \in_R G$ and $y \in_R S$ uniformly at random. We call the modified algorithm $\mathsf{derand\text{-}LT}$. The modified derandomized linearity testing due to Ben-Sasson et. al. [BSVW] is shown in Figure 5.2.

When $G = \mathbb{Z}_2^n$ there exist deterministic constructions for such sets S of size $\operatorname{poly}\log|G|$. For general groups, deterministic constructions exist for sets S of size G^ϵ, for every $\epsilon > 0$. In terms of number of random bits, $G = \mathbb{Z}_2^n$ requires $\log|G| + \log\log|G|$ bits and generic groups G require $(1+\epsilon)\log|G|$ bits. This might not seem as a great savings in randomness – a mere constant factor of 2; however, in PCP constructions and Locally testable codes, one derand-LT(G, H, S)

Input: function $f : G \to H$, specified as a table of values.

1. Choose $x \in_R G$ and $y \in_R S$ uniformly at random.
2. Query the table for $f(x), f(y)$ and $f(x+y)$.
3. Check whether $f(x+y) = f(x) + f(y)$. If the check succeeds, derand-LT accepts f, otherwise, derand-LT rejects f.

Figure 5.2: Derandomized Linearity Test of Ben-Sasson, Sudan, Vadhan and Wigderson [BSVW]

is actually interested in reducing the constant before the leading term (which is typically $O(\log n)$).

Clearly, every linear function is accepted by derand-LT with probability 1. To prove the soundness of derand-LT, we follow the approach of Shpilka and Wigderson [SW].

Theorem 47 (Derandomized LT)**.** *Let $\delta < (1-\lambda)/12$ where λ is the spectral expansion of the Cayley graph G_S. Then, if* derand-LT *accepts f with probability at least $1-\delta$, then f is $4\delta/(1-\lambda)$-close to an affine function.*

Note that we do not show that f is close to a linear function (this is in fact not true), but only the weaker statement that f is close to some affine function.

Proof. Suppose derand-LT rejects with probability $p \le \delta$. That is,

$$\Pr_{x \in G, s \in S}\Big[f(x+s) \neq f(x) + f(s)\Big] \le \delta.$$

Given $y \in G$, we define the "opinion of y about $f(x)$" as $f(x+y) - f(y)$. We define a function $\varphi : G \to G$ such that for all $x \in G$, $\varphi(x)$ is the plurality over $y \in G$ of the opinion of y about $f(x)$, that is $f(x+y) - f(y)$ i.e.,.

$$\varphi(x) \quad = \quad \text{plurality}_{y \in G}\left(f(x+y) - f(y)\right).$$

The following three claims prove the theorem.

Claim 48 (Popularity is majority)**.** *For all x,*

$$\Pr_{y}\Big[\varphi(x) = f(x+y) - f(y)\Big] > 1 - \left(\frac{4\delta}{1-\lambda}\right).$$

Proof. Given $x \in G$, we remove the following edges from G_S. If $f(y+s) \neq f(y) + f(s)$, then remove edge $(y, y+s)$ from G_S. If $f(x+y+s) \neq f(x+y) + f(s)$, then remove edge $(y, y+s)$ from G_S. Notice we have removed at most $2\delta dn$ edges from G_S and hence by Lemma 43, there exists a huge connected component of size at least $1 - \frac{4\delta}{1-\lambda}$.

If an edge remains in the graph, then $f(x+y+s) - f(y+s) = f(x+y) - f(y)$ and therefore y and $y+s$ share the same opinion about $f(x)$. Hence, all vertices in the huge connected component share the same opinion about $f(x)$ which must agree with the plurality. ☐

Claim 49 (φ is linear)**.** *For all $x, y \in G$, $\varphi(x+y) = \varphi(x) + \varphi(y)$.*

Proof. Let $x, y \in G$. We first show $\Pr_{z \in G}\left[\varphi(x+y) = \varphi(x)+\varphi(y)\right] > 0$. This will prove that $\varphi)x + y) = \varphi(x) + \varphi(y)$ since this event is independent of z. Hence, φ is linear. Consider the following events for a random z:

E_1: $\varphi(x+y) = f(x+y+z) - f(z)$.

E_2: $\varphi(x) = f(x+y+z) - f(y+z)$.

E_3: $\varphi(y) = f(y+z) - f(z)$.

Each of these events occurs with probability $1 - 4\delta/(1-\lambda)$ (by Claim 48). By the union bound, the probability that at least one of them fails to occur is at most $12\delta/(1-\lambda) < 1$. Hence, the events E_1, E_2 and E_3 occur simultaneously with non-zero probability. However, if E_1, E_2 and E_3 all occur, we then have that $\varphi(x+y) = \varphi(x) + \varphi(y)$. Thus, proved. □

Claim 50 (f is close to being affine)**.** *f is $\frac{4\delta}{1-\lambda}$-close to an affine shift of φ.*

Proof. By Claim 48, for every $x \in G$, we have the following

$$\Pr_{y \in G}\left[\varphi(x) = f(x+y) - f(y)\right] > 1 - \frac{4\delta}{1-\lambda}.$$

Hence, by an averaging argument, there exists a $y \in G$ such that

$$\Pr_{x \in G}\left[\varphi(x) = f(x+y) - f(y)\right] > 1 - \left(\frac{4\delta}{1-\lambda}\right).$$

Therefore,

$$\Pr_{z \in G}\left[f(z) = \varphi(z-y) + f(y) = \varphi(z) + (f(y) - \varphi(y))\right] > 1 - \left(\frac{4\delta}{1-\lambda}\right),$$

and so $f(z)$ is $\frac{4\delta}{1-\lambda}$-close is φ with an affine shift of $f(y) - \varphi(y)$. □

□

Chapter 6

Expander Codes

Lecturer: Prahladh Harsha
Scribe: Hovav Shacham
May 2 & 9, 2005

In today's lecture, we will discuss the application of expander graphs to error-correcting codes. More specifically, we will describe the construction of linear-time decodable expander codes due to Sipser and Spielman. We begin with some preliminaries on error-correcting codes.

(Several of the proofs presented in this lecture are adapted from the lecture notes of Venkatesan Guruswami's course on Codes and Pseudo-random objects [Gur1]).

6.1 Error Correcting Codes – Preliminaries

We first recall the definition of error-correcting codes. For more information, see, e.g., the excellent survey by Guruswami [Gur2]. Suppose Alice wants to send a k-bit message to Bob over a noisy channel (i.e., the channel flips some bits of the message). In order for Bob to recover (decode) the correct message even after the channel corrupts the transmitted word, Alice instead of sending the k-bit message, encodes the message by adding several redundancy bits and instead sends an n-bit encoding of it across the channel. The encoding is chosen in such a way that a decoding algorithm exists to recover the message from a codeword that has not been corrupted too badly by the channel. (What this means depends on the specific application.)

More formally, a code $\mathcal{C}$ is specified by a injective map $E : \Sigma^k \to \Sigma^n$ that maps k-symbol messages to n-symbol codewords where Σ is the underlying set of symbols called the alphabet. For the most most of today's lecture, we will only consider the binary alphabet (i.e., $\Sigma = \{0, 1\}$). The map E is called the *encoding*. The image of E is the set of codewords of the code $\mathcal{C}$. Some times, we abuse notation and refer to the set of codewords $\{E(x)|x \in$

$\{0,1\}^k\}$ as the code. k is refered to as the *message-length* of the code $\mathcal{C}$ while n is called the *block-length*.

The *rate* of the code, (denoted by r), is the ratio of the logarithm of number of codewords to the block-length n, i.e, $r(C) = \log(\#\text{codewords})/n = k/n \leq 1$. Informally, a rate is the amount of information (about the message) contained in each bit of the codeword.

The (Hamming) distance $\Delta(x, y)$ between any two strings $x, y \in \{0,1\}^n$ is the number of bits in which they differ. The *distance* of the code, denoted by d, is the minimum Hamming distance of any two of its codewords, i.e., $d(C) = \min_{x,y\in\mathcal{C}} \Delta(x, y)$. The *relative distance*, denoted by δ, is the ratio of the distance to the block-length, i.e. $\delta = d/n$.

We will refer to a code $\mathcal{C}$ that maps k message-bits to n codewords with distance d as a (n, k, d)-code.

If the distance of codeword is large and if not too many codeword bits are corrupted by the channel (more precisely if not more than $d/2$ bits are flipped), then we can uniquely decode the corrupted codeword by picking the codeword with the smallest Hamming distance from it. Note that for this unique decoding to work, it must be the case that there are no more than $d/2$ errors caused by the channel. Furthermore, clearly the above algorithm is not efficient as we need to search over the entire space $\{0,1\}^n$ to find the nearest codeword. In today's lecture, we will describe a code (based on expanders) for which this decoding can be done efficiently (more precisely in time linear in the length of the codeword).

Linear Code A code $\mathcal{C}$ is *linear* if 0^n is (a codeword) in $\mathcal{C}$ and if, whenever x and y are in $\mathcal{C}$, so is $x \oplus y$[1]. Linear codes are usually defined in the more general setting when the underlying alphabet for the codeword is some finite field, however for the purpose of this lecture we will restrict ourselves to the binary alphabet $\{0,1\}$. We refer to a linear code that maps k message bits to n codeword bits with distance d as a $[n, k, d]$-code.

It is an easy observation that , in a linear code, d equals the smallest Hamming weight of a non-zero codeword. A linear code can be described as an $n \times k$ generator matrix C (such that $Cm \in \{0,1\}^n$ is the codeword corresponding to a message $m \in \{0,1\}^k$), or by an $(n-k) \times n$ parity-check matrix H (such that $x \in \{0,1\}^n$ is a codeword whenever Hx equals 0^{n-k}). The existence of a generator matrix C immediately implies that the encoding time for linear codes is at most quadratic. The parity check-matrix H implies that a $[n, k, d]$-code can be described by the $n - k = n(1 - r)$ linear constraints (i.e., the columns of H) imposed on the codewords bits. Conversely, if a linear code is described by a set of t (consistent) linear equations, then the rate of the code is at least $r \geq 1 - t/n$.

A random $[n, k, d]$-linear code is formed by choosing a random $n \times k$ generator matrix C of zeros and ones (where each entry is chosen to be either 0 or 1 with probability 1/2) and defining the code $\mathcal{C}$ accordingly (i.e, setting $E(x) = Cx, \forall x \in \{0,1\}^k$). The Gilbert-Varshamov bound states that such a random linear code has (with high probability) distance δ if the block-length is at least $n \geq k/(1 - H(\delta))$ where $H(\cdot)$ is the binary entropy function $H(p) = p \log\left(\frac{1}{p}\right) + (1-p)\log\left(\frac{1}{1-p}\right)$.

Theorem 51 (Gilbert-Varshamov)**.** *Let $\delta < \frac{1}{2}$. If $\mathcal{C}$ is a random linear code with rate at*

[1] where $\oplus$ refers to the bit-wise xor operation

most $1 - H(\delta)$, *then*

$$\text{Prob}\,[d(\mathcal{C}) \geq \delta] = 1 - o(1).$$

6.2 Expander based codes

We now present the expander codes of Sipser and Spielman [SS]. These expander codes have the advantage that the decoding process is very efficient, can be performed in linear time on a single processor or in $\log n$ time by a parallel processor with n-machines. We describe the construction due to Zémor [Zem] which is a slight modification of the original construction of Sipser and Spielman [SS]

6.2.1 Zémor Codes – construction

The family of expander codes is parametrized by a fixed-size code $\mathcal{C}$ with some small block-length d and a family expander graph $\mathcal{G}_n$ on n vertices with constant degree d. The construction due to Zémor is specified by a process that converts a fixed-size code $\mathcal{C}$ of block-length d and an expander graph $\mathcal{G}$ on n vertices and degree d into a new code $\mathcal{Z} = \mathcal{Z}(\mathcal{C}, \mathcal{G})$ with block-length nd. The rate and distance of the new code $\mathcal{Z}$ depend on the rate and distance r and δ of $\mathcal{C}$, and on the spectral expansion λ of $\mathcal{G}$.

The construction proceeds as follows. Take the graph $\mathcal{G} = (V, E)$, and duplicate its vertex set V into left and right vertex sets L and R. For an edge $e \in E$ with endpoints u and v in V, connect both u_{L} in L to v_{R} in R and v_{L} in L to u_{R} in R. This creates a d-regular $2n$-vertex bipartite graph $\mathcal{G}'$. Since the graph G' is constructed from an expander G with spectral expansion λ, the expander mixing lemma can be applied to this graph. In other words, for all sets $S \subset L$ and $T \subset R$, we have that

$$\left| e(S,t) - d\frac{|S||T|}{n} \right| \leq \lambda d \sqrt{|S||T|},$$

where $e(S,T)$ represents the number of edges between the sets S and T.

Now we will use the new graph $\mathcal{G}'$ to describe codewords in $\mathcal{Z}$. These codewords are dn bits long, and $\mathcal{G}'$ has dn edges. We will associate each bit position in the codeword with an edge in $\mathcal{G}'$. It is thus possible to consider a codeword x as an assignment of ones and zeroes to the edges in $\mathcal{G}'$. Moreover, for each vertex v (on either side) we can consider the d-bit restriction x_v of x to edges incident on v. For this purpose, we assume some canonical ordering among the edges incident on any vertex. If $x \in \{0,1\}^{dn}$ and $e_1, \ldots, e_d$ are the edges incident on vertex v, then $x_v = (x_{e_1}, \ldots, x_{e_d}) \in \{0,1\}^d$. Observe that this association also works if x is not a proper codeword: for example, if it has been corrupted by the channel.

Given these mappings, the code itself is actually quite simple. A bit string $x \in \{0,1\}^{dn}$ is a codeword in $\mathcal{Z}$ if, for each vertex $v \in L \cup R$, x_v is a codeword in $\mathcal{C}$. Note that each edge label must satisfy constraints imposed by both its left and right endpoint.

If $\mathcal{C}$ is a linear code, then so is $\mathcal{Z}$. The following theorem characterizes $\mathcal{Z}$ in terms of $\mathcal{C}$ and $\mathcal{G}$.

Theorem 52. *Suppose* $\mathcal{C}$ *is a* $[d, rd, \delta d]$*-code with rate* $r < 1/2$ *and* $\mathcal{G}$ *is a* d*-regular expander on* n *vertices with spectral expansion* $\lambda < \delta$*. Then* $\mathcal{Z}(\mathcal{C}, \mathcal{G})$ *is a* $[dn, (2r-1)dn, \delta(\delta - \lambda)dn]$*-code.*

Proof. It is clear that block length of $\mathcal{Z}$ is dn.

The codes $\mathcal{C}$ and $\mathcal{Z}$ are linear, and so we can consider their rates in terms of constraints in their parity-check matrices. Since $\mathcal{C}$ has block-length d and rate r, its parity-check matrix imposes at most $d - rd = (1-r)d$ constraints on codewords. These constraints are imposed in $\mathcal{Z}$ at each of the $2n$ vertices, so the total number of constraints in $\mathcal{Z}$ is at most $2n(1-r)d = \big(1-(2r-1)\big)nd$, and $\mathcal{Z}$'s rate is at least $(2r-1)$.

Since $\mathcal{Z}$ is linear, its distance equals the minimum Hamming weight of a non-zero codeword. Consider such a codeword x. Let X be the edges labeled 1 in x: $X = \{e \mid x_e = 1\}$, and let S and T be the sets of left- and right-hand vertices, respectively, on which edges in X are incident.

The degree of X with respect to vertices in S and T must be at least δd, since otherwise x would not locally be a $\mathcal{C}$-codeword at these vertices. With a factor of 2 to allow for the double-counting of edges, we have that

$$|X| \geq \frac{\delta d}{2}(|S|+|T|) \ . \tag{6.1}$$

However, by the Expander Mixing Lemma, we have

$$|X| \leq e(S,T) \leq \frac{d}{n}|S||T| + \lambda d\sqrt{|S||T|} \ .$$

Combining the inequalities and dividing out d yields

$$\frac{\delta}{2}(|S|+|T|) < \frac{1}{n}|S||T| + \lambda\sqrt{|S||T|} \ .$$

We can simplify this inequality by means of the following AM-GM inequality

$$|S||T| \leq \frac{(|S|+|T|)^2}{4} \ , \tag{6.2}$$

obtaining

$$\frac{\delta}{2}(|S|+|T|) < \frac{1}{4n}(|S|+|T|)^2 + \frac{\lambda}{2}(|S|+|T|) \ ,$$

or, after some algebra,

$$|S|+|T| > 2n(\delta-\lambda) \ ; \tag{6.3}$$

Substituting 6.3 into (6.1) shows that $\mathcal{Z}$'s distance is at least $\delta(\delta-\lambda)dn$, as required. □

6.2.2 Decoding Algorithm

We now consider how one might decode corrupted codewords in the code $\mathcal{Z}(C,G)$. The algorithm we give has the considerable advantage of being *local*: at each round, algorithm choices at a node depend only values local to that node. In a distributed or multi-processing environment, these local choices can all be made in parallel. It will be evident from the algorithm below, how the decoding can be implemented in $O(\log n)$ time on a parallel machine with n machines. To obtain a linear-time decoding algorithm (on a single machine), a little more book-keeping is necessary. However, for today's lecture, we will ignore this book-keeping (the details of which can be found in [BZ]).

Algorithm 6.1 (Local Decoding for $\mathcal{Z}(\mathcal{C},\mathcal{G})$). *The algorithm is given as input a corrupted codeword x, interpreted as consisting of corrupted local codewords x_v at vertices v in the left or right vertex sets L and R. It corrects these codewords locally as follows.*

1. *Set $V_0 \leftarrow L$*
2. *Set $i \leftarrow 0$*
3. *while there exists a vertex v such that $x_v \notin \mathcal{C}$ do:*
   (a) *For each $v \in V_i$ such that $x_v \notin \mathcal{C}$, decode x_v to nearest codeword in $\mathcal{C}$*
   (b) *If $V_i = L$, set $V_{i+1} \leftarrow R$, otherwise set $V_{i+1} \leftarrow L$.*
   (c) *set $i \leftarrow i+1$*
4. *Return x*

Since all the choices in any single round of the algorithm can be carried out in parallel, we will analyze the algorithm in terms of the number of rounds before it converges on a codeword. In particular, we obtain the following theorem, parametrized on α, which can take values between 0 (faster convergence, but corrects less noise) and 1 (slower, but corrects more).

Theorem 53. *Suppose $\mathcal{C}$ is a $[d, rd, \delta d]$-code and $\mathcal{G}$ is a d-regular expander on n vertices with spectral expansion $\lambda < \delta/3$. Then for all α, $0 \le \alpha \le 1$, the decoding algorithm given in Algorithm 6.1 corrects $\alpha\frac{\delta}{2}(\frac{\delta}{2}-\lambda)dn$ errors in $O\left(\frac{\lg n}{\lg(2-\alpha)}\right)$ rounds.*

Proof. Because $\mathcal{Z}$ is linear, we can, wlog, assume that the closet codeword to the corrupted codeword x is the all-zeros codeword. Denote the working word after i rounds by $x^{(i)}$. Thus, $x^{(0)}$ represents the initial corrupted codeword. Let $E^{(i)}$ be the edges labeled 1 after i rounds:

$$E^{(i)} = \left\{ e \mid x_e^{(i)} = 1 \right\} \quad ,$$

Thus, $E^{(i)}$ represents the codeword bits that haven't been decoded correctly after i rounds. Let $S^{(i)}$ be the vertices in the set V_{i-1} (i.e., left or right side depending on i) with edges in $E^{(i)}$, i.e., the vertices that have not yet correctly decoded at the end of the i rounds.

$$S^{(i)} = \left\{ v \in V_{i-1} \mid E_v \cap E^{(i)} \neq \emptyset \right\} \quad .$$

where E_v represents the set of edges incident on v.

By a series of claims, we will show a geometric decline in the size of $S^{(i)}$,

$$|S^{(i+1)}| < \frac{|S^{(i)}|}{2-\alpha} \quad ,$$

proving the theorem.

We first make some observations:

- Every edge in $E^{(i)}$ has at least one endpoint in $S^{(i)}$.

- With respect to $E^{(i)}$, every vertex in $S^{(i+1)}$ has degree at least $\delta d/2$, since otherwise the vertex would have been locally decoded to the zero codeword in the $(i+1)^{\text{th}}$ round.

Claim 54. $|S^{(1)}| \leq \alpha(\frac{\delta}{2} - \lambda)n$.

Proof. By assumption, $|E^{(0)}| \leq \alpha\frac{\delta}{2}(\frac{\delta}{2} - \lambda)dn$. By the observations above, $|E^{(0)}| \geq |S^{(1)}|\frac{\delta d}{2}$; adjoining the two inequalities proves the claim. ☐

Claim 55. *If $\delta > 3\lambda$ and $|S^{(i)}| < \alpha(\frac{\delta}{2} - \lambda)n$, then $|S^{(i+1)}| < \frac{|S^{(i)}|}{2-\alpha}$.*

Proof. Consider the edge set $E(S^{(i)}, S^{(i+1)})$. This set certainly includes all edges in $E^{(i)}$ between $S^{(i)}$ and $S^{(i+1)}$, and these number (by the observation) at least $\frac{\delta d}{2}|S^{(i+1)}|$. Thus we have

$$e(S^{(i)}, S^{(i+1)}) \geq \frac{\delta d}{2}|S^{(i+1)}| \ .$$

By the Expander Mixing Lemma, however, we have

$$e(S^{(i)}, S^{(i+1)}) \leq \frac{d|S^{(i)}||S^{(i+1)}|}{n} + \lambda d\sqrt{|S^{(i)}||S^{(i+1)}|} \ .$$

Combining these inequalities, we obtain

$$\frac{\delta d}{2}|S^{(i+1)}| \leq \frac{d|S^{(i)}||S^{(i+1)}|}{n} + \lambda d\sqrt{|S^{(i)}||S^{(i+1)}|} \ ,$$

and, using the AM-GM inequality (6.2) with $S = S^{(i)}$ and $T = S^{(i+1)}$,

$$\frac{\delta d}{2}|S^{(i+1)}| < \frac{d|S^{(i)}||S^{(i+1)}|}{n} + \frac{\lambda d}{2}(|S^{(i)}| + |S^{(i+1)}|) \ .$$

Applying the claim precondition $|S^{(i)}| < \alpha(\frac{\delta}{2} - \lambda)n$ to the first summand gives

$$\frac{\delta d}{2}|S^{(i+1)}| < \alpha d(\frac{\delta}{2} - \lambda)|S^{(i+1)}| + \frac{\lambda d}{2}(|S^{(i)}| + |S^{(i+1)}|) \ ,$$

which we can rearrange as

$$\big(\delta - \alpha(\delta - 2\lambda) - \lambda\big)|S^{(i+1)}| < \lambda|S^{(i)}| \ . \tag{6.4}$$

Using the other claim precondition, $\delta > 3\lambda$, we see that

$$\delta - \alpha(\delta - 2\lambda) - \lambda = (1-\alpha)(\delta - 2\lambda) + \lambda > (1-\alpha)\lambda + \lambda = (2-\alpha)\lambda \ ,$$

which, applied to (6.4), gives $(2-\alpha)|S^{(i+1)}| < |S^{(i)}|$, which proves the claim. ☐

We have thus shown a geometric decline in the size of $S^{(i)}$;. Expressed in terms of $|S^{(1)}|$, $|S^{(i)}|$ drops exponentially as $1/(2-\alpha)^i$, and the algorithm will complete in $O\left(\frac{\lg n}{\lg(2-\alpha)}\right)$ rounds. ☐

The above code and decoding algorithm, together with explicit constructions for $\mathcal{C}$ and $\mathcal{G}$, give the following general theorem. We choose $\mathcal{C}$ to be a fixed-size linear code satisfying the Gilbert-Varshamov bound and $\mathcal{G}$ to be a family of Ramanujan graphs with $\lambda \approx 1/\sqrt{d}$ (in fact, any family of expander graphs with $\lambda = 1/o(d)$ will suffice.

Theorem 56 (Sipser and Spielman; Zémor)**.** *For all $0 < \delta, \epsilon < 1$ such that $1-2H(\sqrt{\delta}) < 1$, there exists an explicit family of codes with rate $1 - 2H(\sqrt{\delta})$ and relative distance $\delta - \epsilon$, with a decoding algorithm that corrects $\delta/4 - \epsilon$ errors in $O(\lg n)$ rounds, where n is the block-length of the code and $H(\cdot)$ is the binary entropy function.*

6.2.3 Expander codes with nearly optimal rate

The condition $1 - 2H(\sqrt{\delta}) < 1$ in Theorem 56 allows for only codes with relative distance $\delta < 0.0121$ and thus allows error-recovery from a small fraction $\approx 1\%$ of errors. Expanders and the expander mixing lemma can be used again (!!) to improve the relative distance to $1/2 - \epsilon$ and linear-time decoding upto $1/4 - \epsilon$ fraction of errors at the cost of increasing the alphabet size from binary to a large constant depending on ϵ. This construction is due to Guruswami and Indyk [GI].

Theorem 57 (Guruswami Indyk [GI])**.** *For every $0 < r, \epsilon < 1$, there is an explicit infinite family of linear codes with rate r and relative distance at least $(1 - r - \epsilon)$ such that the codes in the family can be decoded from a fraction $(1 - r - \epsilon)/2$ of errors in time linear in the block-length of the code.*

The details of this construction can be found in [Gur2, GI].

Chapter 7

Expander Constructions (Zig-Zag Expanders)

Lecturer: Cynthia Dwork
Scribe: Geir Helleloid
May 9 & 16, 2005

7.1 Lecture Outline

In this lecture we will see three explicit constructions of expanders. By an "explicit construction", we mean a construction with the following three properties:

1. We can build the entire N-vertex graph in $\text{poly}(N)$ time.
2. From a vertex v, we can find the i-th neighbor in $\text{poly}(\log N, \log D)$ time where D is the degree of the graph.
3. Given vertices u and v, we can determine if they are adjacent in $\text{poly}(\log N)$ time.

The first two constructions will be presented without proof, but we will see the proof in the case of the zig-zag construction.

1. The first construction is due to Margulis and Gaber-Galil.
2. The second construction is due to Lubotsky, Phillips, and Sarnak, and achieves optimal spectral expansion $\lambda \approx 2/\sqrt{d}$.
3. The third construction is due to Reingold, Vadhan, and Wigderson. These so-called zig-zag expanders are built via repeated applications of two basic operations that jointly increase the number of nodes but keep the degree and expansion λ small.

These operations are graph squaring and the zig-zag product. The proof that these graphs are expanders will use the tensor product of two vectors.

7.2 The First Two Constructions

Construction 7.1 (Margulis [Mar]) Fix a positive integer M and let $[M] = \{1, 2, \ldots, M\}$. Define the bipartite graph $G = (V, E)$ as follows. Let $V = [M]^2 \cup [M]^2$, where vertices in the first partite set are denoted $(x, y)_1$ and vertices in the second partite set are denoted $(x, y)_2$. From each vertex $(x, y)_1$, put in edges to $(x, y)_2$, $(x, x + y)_2$, $(x, x + y + 1)_2$, $(x + y, y)_2$, and $(x + y + 1, y)_2$, where all arithmetic is done modulo M. Then G is an expander. The proof uses Fourier analysis.

Construction 7.2 (Lubotsky-Phillips-Sarnak [LPS]) Fix primes q and p such that $q \equiv 1 \pmod 4$ and $p \equiv 1 \pmod q$. Let i be an integer such that $i^2 \equiv -1 \pmod q$. Define the graph $G = (V, E)$ as follows. Let $V = GF(q) \cup \{\infty\}$. Put an edge between (z, z') if

$$z' = \frac{(a_0 + ia_1)z + (a_2 + ia_3)}{(-a_2 + ia_3)z + (a_0 - ia_1)}$$

for some $a_0, a_1, a_2, a_3 \in \mathbb{N}$ such that $a_0^2 + a_1^2 + a_2^2 + a_3^2 = p$. It can be shown that the number of integral solutions to $a_0^2 + a_1^2 + a_2^2 + a_3^2 = p$ is $p + 1$. Hence, G has degree $d = p + 1$. It can further be shown that the spectral expansion of G is at most $\lambda(G) \leq 2\sqrt{d-1}/d$, which is optimal. Families of graphs with such optimal spectral expansion are called Ramanujan graphs.

7.3 The Zig-Zag Product

In this section, we define the zig-zag product of two graphs. We will use this product in Sections 7.4 and 7.5 to construct expander graphs and prove their spectral properties. This construction is due to Reingold, Vadhan and Wigderson [RVW]. For convenience, we say that G is an (N, d, λ)-expander if G has N vertices, degree d, and spectral expansion λ.

To construct the zig-zag product, we begin with an (N_1, d_1, λ_1)-expander G and a (d_1, d_2, λ_2)-expander H. Assume that $V(H) = [d_1] = \{1, 2, \ldots, d_1\}$. Each vertex in G has d_1 neighbors, and we can label them as the 1st, 2nd, …, and d_1-th neighbors of v. Define a matching Rot_G on $V(G) \times V(H)$ by $\mathrm{Rot}_G(u, i) = (v, j)$ where v is the i-th neighbor of u and u is the i-th neighbor of v. This is the *rotation map* associated to G.

One intuitive way to approach the zig-zag product is to suppose that we want to construct a random walk on $V(G) \times V(H)$. Starting at (u, i), what can we do? We can choose a random neighbor i' of i in H, and use that to pick a random neighbor v of u in G. Since this isn't quite reversible, we need to end by choosing another random neighbor of our current vertex in H. This attempts to motivate the following definition.

The *zig-zag product* of G and H is denoted $G \textcircled{z} H$. The vertex set of $G \textcircled{z} H$ is $V(G) \times V(H)$, so the vertices of $G \textcircled{z} H$ are pairs (v, i) with $v \in V(G)$ and $i \in V(H)$. Put an edge between (u, i) and (v, j) if and only if there exist $i', j' \in V(H)$ such that (i, i') and (j, j') are edges of H and $\mathrm{Rot}_G(u, i') = (v, j')$. (See Figure 7.1)

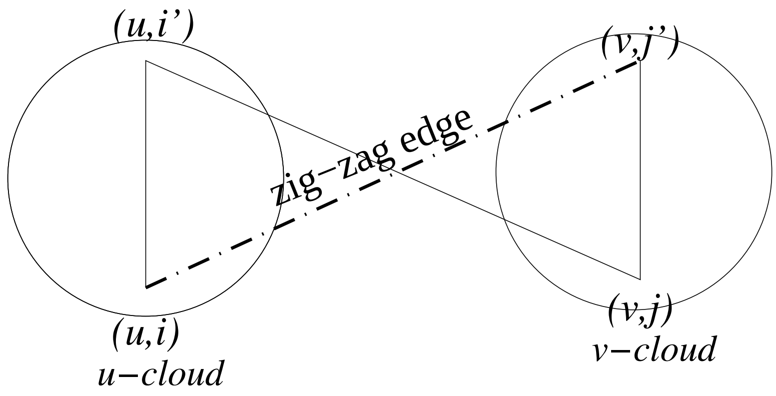

Figure 7.1: Zig-Zag Product

More formally,

Definition 58. *The zig-zag product between rotation map representations of two graphs G, a (N, D_1, λ_1)-graph and H, a (D_1, D_2, λ_2)-graph, is a rotation map representation of a graph, denoted by $G ⓩ H$. The graph $G ⓩ H$ and its rotation map are defined as below.*

1. *$G ⓩ H$ has ND_1 vertices.*
2. *$G ⓩ H$ is a D_2^2-regular graph.*
3. $\mathrm{Rot}_{G ⓩ H}((u, i), (a_1, a_2)) = ((v, j), (b_1, b_2))$ *if the following is satisfied: There exist $i', j' \in [D_2]$ such that*
    - $\mathrm{Rot}_H(i, a_1) = (i', b_2)$
    - $\mathrm{Rot}_G(u, i') = (v, j')$
    - $\mathrm{Rot}_H(j', a_2) = (j, b_1)$

Less formally, let's explore what this really looks like. First, form an intermediate graph K by replacing each vertex v of G by a copy H_v of H. For each neighbor w of v in G, choose a vertex in H_w. Then construct a matching between these d_1 vertices and the d_1 vertices of H_v. Of course, the matchings constructed for the vertices in G must be compatible in the obvious way. We refer to H_v as the *cloud* corresponding to v.

Now, if there is an edge between (u, i') and (v, j') in K (with $u \neq v$), then in $G ⓩ H$, all the neighbors of (u, i') in H_u are connected to all the neighbors of (v, j') in H_v. So $G ⓩ H$ is the edge union of many complete bipartite graphs K_{d_2, d_2}. We can easily calculate the degree of $G ⓩ H$: from (u, i), there are d_2 choices for (u, i'), then one choice for $(v, j') = \mathrm{Rot}_G(u, i')$, and finally d_2 choices for (v, j). Thus the degree of $G ⓩ H$ is d^2.

Intuitively, why should $G ⓩ H$ have good expansion when G and H do? If we are given a distribution that is mixed on the G component, then the rapid mixing on H suggests that the distribution will rapidly mix on $G ⓩ H$. Similarly, given a distribution that is mixed on the H component, then the rapid mixing on G suggests that the distribution will rapidly mix on $G ⓩ H$. So we might hope that every distribution mixes rapidly on $G ⓩ H$.

7.4 The Zig-Zag Expander Construction

Using both the zig-zag product and graph squaring, we can demonstrate the zig-zag construction of expanders. Recall that the square of G is denoted G^2; it has the same vertex set as G, and $(x, y) \in E(G^2)$ if and only if there exists a path of length two from x to y in G. If G is an (N, d, λ)-expander, then G^2 is an (N, d^2, λ^2)-expander. In the next section we will show that $\lambda(G \textcircled{z} H) \leq \lambda(G) + \lambda(H) + \lambda(H^2)$. Assuming this result, we can state and prove the zig-zag construction of expanders.

Theorem 59. *Let H be a (d^4, d, λ_0)-expander for some $\lambda_0 \leq 1/5$[1]. Define $G_1 = H^2$ and $G_{t+1} = G_t^2 \textcircled{z} H$ for $t \geq 1$. Then for all t, G_t is a (d^{4t}, d^2, λ)-expander with $\lambda \leq 2/5$.*

Proof. The proof is by induction on t. When $t = 1$, based on what we know about the square of a graph, we see that G_1 is a (d^4, d^2, λ_0^2)-expander where $\lambda_0^2 \leq 1/25$.

Now assume that G_{t-1} is a $(d^{4(t-1)}, d^2, \lambda)$-expander with $\lambda \leq 2/5$. It is clear that G_t has d^{4t} nodes since the number of nodes in the zig-zag product of two graphs is the product of the number of nodes in each of the two graphs. Also G_t has degree d^2, since the degree of a zig-zag product is the degree of the second factor.

Finally,

$$\begin{aligned} \lambda(G_t) &\leq \lambda(G_{t-1}^2) + \lambda(H) + \lambda(H^2) \\ &\leq \left(\frac{2}{5}\right)^2 + \frac{1}{5} + \frac{1}{25} \\ &= \frac{2}{5}. \end{aligned}$$

□

7.5 Spectral Property of the Zig-Zag Product

It remains to give an upper bound for the spectral expansion of a zig-zag product.

Theorem 60. *Suppose G is an (N_1, d_1, λ_1)-expander and H is a (d_1, d_2, λ_2)-expander. Then $G \textcircled{z} H$ is an $(N_1 d_1, d_2^2, f(\lambda_1, \lambda_2))$-expander, where $f(\lambda_1, \lambda_2) \leq \lambda_1 + \lambda_2 + \lambda_2^2$.*

Proof. Recall that given vectors $x \in \mathbb{R}^{N_1}$ and $y \in \mathbb{R}^{N_2}$, their tensor product is given by $x \otimes y = (x_i \cdot y_j) \in \mathbb{R}^{N_1 N_2}$. Let M be the normalized adjacency matrix of $G \textcircled{z} H$. Then

$$\lambda(G \textcircled{z} H) = \max_{\alpha \perp 1_{N_1 d_1}} \frac{|\langle M\alpha, \alpha \rangle|}{|\langle \alpha, \alpha \rangle|}.$$

So we need to show that for all $\alpha \in \mathbb{R}^{N_1 d_1}$, if $\alpha \perp 1_{N_1 d_1}$, then $|\langle M\alpha, \alpha\rangle| \leq f(\lambda_1, \lambda_2) |\langle \alpha, \alpha \rangle|$.

Let $\alpha \in \mathbb{R}^{N_1 d_1}$ such that $\alpha \perp 1_{N_1 d_1}$. For all $v \in [N_1]$, define $\alpha_v \in \mathbb{R}^{d_1}$ by $(\alpha_v)_k = \alpha_{vk}$. Also define a linear map $C : \mathbb{R}^{N_1 d_1} \to \mathbb{R}^{N_1}$ by $(C\alpha)_v = \sum_{k=1}^{d_1} \alpha_{vk}$. Then $\alpha = \sum_v (e_v \otimes \alpha_v)$.

[1]Since H is a fixed-size graph, such an expander graph can be found by brute-force search.

Furthermore, we can decompose α_v into $\alpha_v = \alpha_v^\perp + \alpha_v^\|$, where $\alpha_v^\perp \perp 1_{d_1}$. Thus we can decompose α into $\alpha^\|$ and $\alpha^\perp$ as follows:

$$\begin{aligned} \alpha &= \sum_v (e_v \otimes \alpha_v^\|) + \sum_v (e_v \otimes \alpha_v^\perp) \\ &=: \alpha^\| + \alpha^\perp. \end{aligned}$$

Note that $\alpha^\|$, if viewed as a distribution on $G \textcircled{z} H$, is uniform within any given cloud. In fact,

$$\alpha^\| = \frac{C\alpha \otimes 1_{d_1}}{d_1} = \left(\frac{\text{total on } v_1}{d_1}, \ldots, \frac{\text{total on } v_i}{d_1}, \ldots, \frac{\text{total on } v_N}{d_1} \right).$$

Evidently, since the sum of the entries in $\alpha^\|$ equals the sum of the entries in α, namely 0, we have $\alpha^\| \perp 1_{N_1 d_1}$ and $C\alpha^\| \perp 1_{N_1}$.

Define $\widetilde{B} = I_{N_1} \otimes B$, where B is the normalized adjacency matrix for H. Thus $\widetilde{B}$ is a block diagonal square matrix of size $N_1 d_1$ with blocks B. Furthermore, let $\widetilde{A}$ be the permutation matrix corresponding to the Rot_G mapping. Then $M = \widetilde{B}\widetilde{A}\widetilde{B}$.

Note that

$$\langle M\alpha, \alpha \rangle = \left\langle \widetilde{B}\widetilde{A}\widetilde{B}\alpha, \alpha \right\rangle = \left\langle \widetilde{A}\widetilde{B}\alpha, \widetilde{B}\alpha \right\rangle,$$

since $\widetilde{B}$ is a real symmetric matrix and hence self-adjoint. Also $\widetilde{B}\alpha^\| = \alpha^\|$, since the uniform distribution on H is invariant under B. Then

$$\widetilde{B}\alpha = \widetilde{B}(\alpha^\perp + \alpha^\|) = \alpha^\| + \widetilde{B}\alpha^\perp.$$

Computing, we find

$$\begin{aligned} \langle M\alpha, \alpha \rangle &= \left\langle \widetilde{A}(\alpha^\| + \widetilde{B}\alpha^\perp), (\alpha^\| + \widetilde{B}\alpha^\perp) \right\rangle \\ &= \left\langle \widetilde{A}\alpha^\|, \alpha^\| \right\rangle + \left\langle \widetilde{A}\alpha^\|, \widetilde{B}\alpha^\perp \right\rangle + \left\langle \widetilde{A}\widetilde{B}\alpha^\perp, \alpha^\| \right\rangle + \left\langle \widetilde{A}\widetilde{B}\alpha^\perp, \widetilde{B}\alpha^\perp \right\rangle \\ |\langle M\alpha, \alpha \rangle| &\leq \left| \left\langle \widetilde{A}\alpha^\|, \alpha^\| \right\rangle \right| + \|\widetilde{A}\alpha^\|\| \cdot \|\widetilde{B}\alpha^\perp\| + \|\widetilde{A}\widetilde{B}\alpha^\perp\| \cdot \|\alpha^\|\| + \|\widetilde{A}\widetilde{B}\alpha^\perp\| \cdot \|\widetilde{B}\alpha^\perp\| \\ &= \left| \left\langle \widetilde{A}\alpha^\|, \alpha^\| \right\rangle \right| + 2\|\alpha^\|\| \cdot \|\widetilde{B}\alpha^\perp\| + \|\widetilde{B}\alpha^\perp\|^2, \end{aligned}$$

where the last line uses the fact that $\widetilde{A}$ is a permutation and hence $\|\widetilde{A}x\| = \|x\|$, for all $x \in \mathbb{R}^{N_1 d_1}$.

To simplify this expression, we first see that

$$\begin{aligned} \|\widetilde{B}\alpha^\perp\|^2 &= \|\widetilde{B}(\sum_v e_v \otimes \alpha_v^\perp)\|^2 \\ &= \|\sum_v e_v \otimes B\alpha_v^\perp\|^2 \\ &= \sum_v \|B\alpha_v^\perp\|^2 \\ &\leq \sum_v \lambda_2^2 \|\alpha_v^\perp\|^2 \\ &\leq \lambda_2^2 \|\alpha^\perp\|^2. \end{aligned}$$

Secondly, we need to bound $\left|\left\langle \widetilde{A}\alpha^{\|}, \alpha^{\|}\right\rangle\right|$. Let A be the normalized adjacency matrix for G; we want to relate $\widetilde{A}$ and A. Fix $e_v \in \mathbb{R}^{N_1}$. Then Ae_v gives a uniform distribution on the neighbors of v in G. This means that

$$Ae_v = C\widetilde{A} \cdot \frac{e_v \otimes 1_{d_1}}{d_1}.$$

The tensor product gives a uniform distribution on the cloud corresponding to v, multiplying by $\widetilde{A}$ moves the distribution to the neighbors of v, and multiplying by C adds up the distribution in each cloud.

By linearity, it follows that for all $\beta \in \mathbb{R}^{N_1}$,

$$A\beta = C\widetilde{A} \cdot \frac{\beta \otimes 1_{d_1}}{d_1}.$$

Take $\beta = C\alpha$. Then $\alpha^{\|} = (\beta \otimes 1_{d_1})/d_1$, so we get that $C\widetilde{A}\alpha^{\|} = AC\alpha$. Thus

$$\begin{aligned}
\left\langle \widetilde{A}\alpha^{\|}, \alpha^{\|}\right\rangle &= \left\langle \widetilde{A}\alpha^{\|}, C\alpha \otimes 1_{d_1}\right\rangle /d_1 \\
&= \left\langle C\widetilde{A}\alpha^{\|}, C\alpha\right\rangle /d_1 \\
&= \langle AC\alpha, C\alpha\rangle /d_1 \\
\left|\left\langle \widetilde{A}\alpha^{\|}, \alpha^{\|}\right\rangle\right| &\leq \lambda_1 \langle C\alpha, C\alpha\rangle /d_1 \\
&= \lambda_1 \langle C\alpha \otimes 1_{d_1}, C\alpha \otimes 1_{d_1}\rangle /d_1^2 \\
&= \lambda_1 \left\langle \alpha^{\|}, \alpha^{\|}\right\rangle.
\end{aligned}$$

Combining the two inequalities, we find

$$|\langle M\alpha, \alpha\rangle| \leq \lambda_1\|\alpha^{\|}\|^2 + 2\lambda_2\|\alpha^{\|}\| \cdot \|\alpha^{\perp}\| + \lambda_2^2\|\alpha^{\perp}\|^2.$$

Take $p = \|\alpha^{\|}\|/\|\alpha\|$ and $q = \|\alpha^{\perp}\|/\|\alpha\|$, so that $p^2 + q^2 = 1$. Then

$$\begin{aligned}
\frac{|\langle M\alpha, \alpha\rangle|}{|\langle \alpha, \alpha\rangle|} &\leq \lambda_1 p^2 + 2\lambda_2 pq + \lambda_2^2 q^2 \\
&= \lambda_1 + \lambda_2 + \lambda_2^2.
\end{aligned}$$

$\square$

Chapter 8

Undirected Connectivity is in logspace

Lecturer: Prahladh Harsha
Scribe: Cynthia Dwork & Prahladh Harsha
May 16, 2005

In the second half of today's lecture, we will discuss a deterministic logspace algorithm for undirected connectivity, a recent and beautiful result due to Omer Reingold [Rei]. In fact, Reingold's algorithm is one of the reasons this course is being offered this quarter.

8.1 Undirected S-T Connectivity

The undirected s-t connectivity problem is the problem of finding if there exists a path between two specified vertices in a given undirected graph. More formally, the problem is as follows:
Input: A undirected graph $G = (V, E)$ and two vertices $s, t \in V$ (s denotes source and t target).
Problem: Are s and t connected? I.e., does there exist a path in G from the source s to the target t?

$$\text{USTCONN} = \{\langle G, s, t\rangle \mid G\text{– undirected graph}, s, t \in V(G); s \text{ and } t \text{ are connected in } G\}.$$

Clearly, any of the standard search algorithms (depth-first-search, breadth-first-search etc.) solve USTCONN in linear time. Thus, the time complexity of USTCONN is well-understood. What we would be interested in today's lecture is the same complexity of USTCONN. It is to be noted that the standard search algorithms perform poorly with respect to space (this is because their implementation requires a stack or queue which in the worst case could be as large as the graph). The question we will concern ourselves is the following: Is USTCONN in Logspace. In other words, does there exist a deterministic logspace algorithm that can decide connectivity in an undirected graph. Reingold resolved this question positively and gave a logspace algorithm for USTCONN using the zig-zag product. Before that, we will briefly look at the history of USTCONN.

8.1.1 History of Space Complexity of USTCONN

USTCONN is in NL. In fact, the directed counterpart of USTCONNis the complete problem for non-deterministic logspace. In 1970, Savitch demonstrated [Sav] a simulation of a non-deterministic space S machine by a deterministic space S^2 machine. Thus, USTCONN $\in$ SPACE($\log^2 n$). In one of the initial lectures of this course on random walks (Lecture 3), we saw a randomized logspace algorithm for USTCONN due to Aleliunas et. al. [AKLLR]. Thus, USTCONN $\in$ RL (RL denotes randomized logspace). Saks and Zhou, in 1995, then showed that any randomized space S machine can be simulated by a deterministic space $S^{3/2}$ machine [SZ]. Putting both these results together, we have USTCONN $\in$ SPACE($\log^{3/2} n$). Later, in 1997, Armoni, Ta-Shma, Wigderson and Zhou improved this deterministic simulation to give a $O(\log^{4/3} n)$-space algorithm for USTCONN. The status of this problem has been open since then till it was resolved recently due to Reingold. Note that $\log n$ space is required to even index a vertex in the graph.

8.2 Savitch's Deterministic Simulation

To begin with, we will look at Savitch's algorithm for USTCONN. The main idea in Savitch's algorithm is that squaring improves connectivity. More formally, for any graph G, define G^{sq} to be the graph on the same set of vertices as G, but has an edge between any two vertices u, v in $V(G)$ if there exists a path of length at most two between u and v in the original graph G. Note this is not the same as the more natural G^2 which corresponds to the graph where there are as many edges between u and v as the number of walks of length exactly two between u and v in G. A simple observation reveals that if u and v are connected in G, then (u, v) is an edge in $G^{\text{sq}^{\log n}}$. Savitch gave a $O(\log^2 n)$ algorithm that computes the graph $G^{\text{sq}^{\log n}}$ from G, thus proving USTCONN $\in$ SPACE($\log^2 n$).

We will now perform Savitch's algorithm with the more natural G^2 instead of G^{sq}. Though, this will also solve USTCONN, this will not give us another proof of Savitch's Theorem. In fact, the space complexity of computing G^{2^n} is huge. However, this exercise will be illuminating and will lead us towards Reingold's algorithm.

For the purpose of this discussion, we will assume all graphs are regular. Before discussing the algorithm for G^{2^n}, we need to indicate which representation of the graph we use. For reasons that will become clear later, we will use the rotation map representation introduced while talking about the zig-zag product. If the graph G is d-regular, then the rotation map Rot_G is the permutation that maps $(u, i) \in V \times [d]$ to $(v, j) \in V \times [d]$ if the i^{th} edge from u leads to v and the label of this edge with respect to v is j. Note, $\mathrm{Rot}_G(\mathrm{Rot}_G(u, i)) = (u, i)$ for all (u, i). We can compute this representation from the adjacency matrix and list in $O(\log n)$ space.

Let us first calculate the space-complexity of computing the rotation map of H^2 given the rotation map of H. Let H be a d-regular graph, then H^2 is a d^2 regular graph. The edge labels of H^2 can be assumed to be from the set $[d]\times[d]$. How do we compute $\mathrm{Rot}_{H^2}(u,(i_1,i_2))$ efficiently in space? To start with we assume that the tape contains $(u,(i_1,i_2))$ and at the end of the computation, we would like this to be replaced by $\mathrm{Rot}_{H^2}(u,(i_1,i_2))$. For a graph G, let SPACE(G) denote the additional space required to replace the input (u,i) on the tape with $\mathrm{Rot}_G(u,i)$. We first compute $\mathrm{Rot}_H(u,i_1)=(w,j_2)$ and replace the tape contents $(u,(i_1,i_2))$ with $(w,(j_2,i_2))$. This requires an additional space for computing Rot_H, i.e., SPACE(H). In the second step, we then reuse this extra space to compute $\mathrm{Rot}_H(w,i_2)=(v,j_1)$ and replace the tape contents $(w,(j_2,i_2))$ with $(v,(j_2,j_1))$. Finally, we swap the indices j_1 and j_2 in the tape to obtain $(v,(j_1,j_2))$ which is in fact $\mathrm{Rot}_{H^2}(u,(i_1,i_2))$. A analysis of the above shows that

$$\mathrm{SPACE}(H^2) = \mathrm{SPACE}(H) + O(\log \deg(H)).$$

Performing the above procedure $O(\log n)$ times, we can compute the rotation map of $G^n = G^{2^{\log n}}$, thus solving USTCONN. The space complexity of this algorithm is given by the following:

$$\begin{aligned}
\mathrm{SPACE}(G^n) &= \mathrm{SPACE}(G^{2^{\log n}}) \\
&= \mathrm{SPACE}(G^{2^{\log n-1}}) + O(\log\deg(G^{2^{\log n-1}})) \\
&\vdots \\
&= \mathrm{SPACE}(G) + O(\log\deg G) + O(\log\deg G^2) + \cdots + O(\log\deg G^{2^{\log n-1}}) \\
&= \sum_{i=1}^{\log n-1} O(\log\deg G^{2^i})
\end{aligned}$$

The reason this is a bad algorithm for solving USTCONN is because the degree of G^{2^i} grows prohibitively large with i, in fact $\deg(G^{2^i}) = (\deg(G))^{2^i}$. If somehow we could keep the degree constant through out the process, then in fact the above procedure would give a $O(\log n)$ space algorithm for USTCONN. But this is not possible, squaring a graph will also square the degree. Is it possible to obtain the same effect as squaring without actually increasing the degree of the graph? The main advantage of squaring is that it improves the expansion of the graph (and thus the connectivity of the graph). The crucial idea in Reingold's algorithm is that the zig-zag product (discussed in the first half of today's lecture) can be used to decrease the degree without altering the expansion of the graph by too much. Reingold's algorithm thus alternates between squaring and zig-zag to improve the expansion of the graph (via squaring) while not increasing the degree.

8.3 Zig-Zag Product

We say that a graph G is a (N,d,λ)-graph if G is a d-regular graph on N vertices and the spectral expansion of G is at most λ. Let us quickly recall the definition of the zig-zag product.

Definition 61. *The zig-zag product between rotation map representations of two graphs G, a (N, D_1, λ_1)-graph and H, a (D_1, D_2, λ_2)-graph, is a rotation map representation of a graph, denoted by $G \textcircled{z} H$. The graph $G \textcircled{z} H$ and its rotation map are defined as below.*

1. *$G \textcircled{z} H$ has ND_1 vertices.*
2. *$G \textcircled{z} H$ is a D_2^2-regular graph.*
3. $\mathrm{Rot}_{G \textcircled{z} H}((u, i), (a_1, a_2)) = ((v, j), (b_1, b_2))$ *if the following is satisfied: There exist $i', j' \in [D_2]$ such that*
   - $\mathrm{Rot}_H(i, a_1) = (i', b_2)$
   - $\mathrm{Rot}_G(u, i') = (v, j')$
   - $\mathrm{Rot}_H(j', a_2) = (j, b_1)$

We proved the following result in the earlier lecture on zig-zag products.

Theorem 62. *Suppose G is an (N_1, d_1, λ_1)-expander and H is a (d_1, d_2, λ_2)-expander. Then $G \textcircled{z} H$ is an $(N_1 d_1, d_2^2, f(\lambda_1, \lambda_2))$-expander, where $f(\lambda_1, \lambda_2) \leq \lambda_1 + \lambda_2 + \lambda_2^2$.*

Note that the above result is useful only when both the graphs G and H have fairly good expansion to start with. For our case, all we know is that the original graph, if connected and non-bipartite, has spectral expansion bounded away from 1 by at least a inverse polynomial. In fact, we proved the following result in Lecture 3.

Lemma 63. *If G is a connected, d-regular, non-bipartite graph on n vertices, then*

$$1 - \lambda \geq \frac{1}{dn^2}.$$

The zig-zag product is useful even in this case as long as the other graph H has good spectral expansion. We will use the following result (which we will not prove in class) on zig-zag products for this purpose.

Lemma 64. *If $\lambda(H) \leq 1/2$, then*

$$1 - \lambda(G \textcircled{z} H) \leq \frac{1}{3}\left(1 - \lambda(G)\right).$$

We mention that the zig-zag product is not the only graph product that satisfies such properties. The replacement product would have sufficed for our purposes (see [MR1, MR2]). However, we use the zig-zag product since we are already familiar with it from the previous lecture. A proof of Lemma 64 can be found in [RVW] while a proof of a similar statement for the replacement product can be found in Martin and Randall [MR2].

8.4 Reingold's Algorithm

As mentioned in the previous sections, the main idea in Reingold's algorithm is to alternate squaring with zig-zag product (with a constant sized expander). Lemma 64 tells us the following: as long as H is a good expander (i.e., $\lambda(H) \leq 1/2$), zig-zagging G with H only reduces the spectral gap (i.e., $1 - \lambda$) by a factor of three (3). This is good for us, since squaring improves the spectral gap $1 - \lambda$ from to $1 - \lambda^2$ while zig-zag product deteriorates the spectral gap from $1 - \lambda$ to $1 - \lambda/3$. Thus, we could alternate a couple of squarings with a zig-zag product to reduce the spectral gap by a factor of two while keeping the degree constant.

We are now ready to describe Reingold's algorithm.

let H be a $(d^{16}, d, 1/2)$-graph for some constant G. Such a graph H can be found either by exhaustive search or by using one of the expander constructions (described earlier in the course). For this section, we will assume that the input graph G for which we need to check (s, t) connectivity is a d^{16}-regular non-bipartite graph. We will later remove these restrictions on G.

Furthermore, we will assume G is a connected graph. Actually, this is a stupid assumption since if G were indeed connected, then there is nothing to prove. What we actually mean is the following: Reingold's algorithm works independently for each connected component of the graph and checks if t exists in the connected component that contains s. Since every component of G is d^{16}-regular, connected and non-bipartite, we have from Lemma 63 that $1 - \lambda(C) \geq 1/d^{16}n^2$, for all components C of G.

Checking connectivity on an expander We first argue that checking connectivity on a graph, each of whose connected components is an expander (i.e., $\lambda \leq 1/2$) can be done in logspace. This follows from the simple observation that in an expander, the distance between any two vertices in $O(\log n)$. Thus, it suffices to enumerate all $O(\log n)$ paths in the graph originating at s and check if any of them lead to t. This can be done in logspace. Thus, it suffices for us to convert G into another G' in logspace such that each connected component of G' is an expander (i.e., $\lambda \leq 1/2$) and furthermore, two vertices are connected in G iff they are connected in G' (i.e., the transformation does not alter the connectivity of the graph).

Reingold's Algorithm

Input: G – d^{16}-regular graph and two vertices $s, t \in V(G)$.

1. Set l to be the smallest integer such that $\left(1 - \frac{1}{d^{16}n^2}\right)^{2^l} \leq \frac{1}{2}$.

   Comment: l is $O(\log n)$

2. Set $G_0 \leftarrow G$.
3. For $i = 1, \ldots, l$ do, set $G_i \leftarrow (G_{i-1} ⓩ H)^8$.

   Comment: (1) Each G_i is a d^{16}-regular graph.
   (2) Each connected component of G_l is an expander with spectral expansion at most $1/2$ (see Theorem 66)

4. Check if s and t are connected in G_l by enumerating over all $O(\log n)$ paths originating at s.

We first prove the comment in Step 3, which will suffice to prove the correctness of Reingold's algorithm. For this we need the following proposition.

Proposition 65. *For $i = 1, \ldots, l$, $\lambda(G_i) \leq \min\{\lambda^2(G_{i-1}), 1/2\}$.*

Proof. Since $G_i = (G \textcircled{z} H)^8$, we have from Lemma 64 that $\lambda(G_i) = \lambda^8(G_{i-1} \textcircled{z} H) \leq [1 - (1 - \lambda(G_{i-1}))/3]^8$. Now, consider the following two cases.

Case (i): $\lambda(G_{i-1}) \leq 1/2$. Then,

$$\lambda(G_i) = (\lambda(G_{i-1} \textcircled{z} H))^8 \leq \left(1 - \frac{1}{3} \cdot \frac{1}{2}\right)^8 = \left(\frac{5}{6}\right)^8 < \frac{1}{2}.$$

Case (ii): $\lambda(G_{i-1}) > 1/2$. In this case, we can by expansion check that

$$\left(1 - \frac{1}{3}(1 - x)\right)^4 \leq x, \qquad \text{for all } \frac{1}{2} \leq x \leq 1$$

Hence,

$$\lambda(G_i) = (\lambda(G_{i-1} \textcircled{z} H))^8 \leq \left(1 - \frac{1}{3}(1 - \lambda(G_{i-1}))\right)^8 \leq \lambda^2(G_{i-1}).$$

□

By our choice of l, we have the following theorem on the expansion of each connected component of G_l.

Theorem 66. *The spectral expansion of each connected component of G_l is at most $1/2$.*

Space Complexity of Reingold's Algorithm We noted before that each squaring operation requires an additional space of $O(\log \deg G)$. Similarly, it can be shown that each zig-zag product with H (of constant size) also requires additional space at most $O(\log \deg G)$. Since there are at most $O(\log n)$ squaring and zig-zag products (since $l = O(\log n)$) and the degree of all the graphs is at most d^{16}, a constant, the total space complexity of the algorithm is at most $O(\log n)$.

Handling non-regular bipartite graph To start with, we will convert the graph into a 3-regular graph by replacing each vertex of degree d greater than 3 by a cycle of size d and connecting each of the d neighbors of the vertex to the d distinct points on the circle. To convert the graph into a d^{16}-regular graph, we then add $d^{16} - 3$ self loops to each vertex. Note, the addition of self loops also makes the graph non-bipartite. Both these conversions can be effected in log space.

Chapter 9

Dinur's Proof of the PCP Theorem

Lecturer: Prahladh Harsha
Scribe: Krishnaram Kenthapadi
May 31, 2005

In this lecture, we will describe a recent and remarkable proof of the PCP theorem, due to Irit Dinur [Din]. This proof is a beautiful application of expanders for "soundness amplification" of randomized algorithms without using too many random bits.

Before describing the proof, we will first look at the PCP theorem, by relating it with the theory of NP-completeness.

9.1 Hardness of Optimization Problems

The theory of NP-completeness, as developed by Cook, Levin, and Karp, states that any language, L in NP is reducible to the Boolean satisfiability problem, 3SAT. By this, we mean that for every instance, x of the language L, we can obtain a satisfiability instance, ϕ such that $x \in L$ if and only if ϕ is satisfiable. Thus, 3SAT is at least as hard as any other problem in NP. Karp further showed that 3SAT can be reduced to other problems such as CLIQUE and 3-COLORABILITY and hence that these problems are at least as hard as any problem in NP. In other words, solving these problems *optimally* is as hard as solving any other problem in NP optimally.

However the question of the hardness of approximation was left open. For instance, can the following be true – finding a satisfying assignment for 3SAT is NP-hard, however it is easy to find an assignment that satisfies 99% of the clauses. Questions such as Other examples: can we approximate the clique size in a graph? Or, can we obtain a 3-coloring that satisfies 99% of the edge constraints? In other words, is the approximation version of some of NP-hard problems easier than the optimization versions. The PCP Theorem [FGLSS, AS, ALMSS] states that this is not the case – for several of the NP-hard problems, the approximation version is just as hard as the optimization version. The PCP theorem can be viewed as a strengthening of Karp reductions. It provides a reduction from a 3SAT instance ϕ to another 3SAT instance ψ such that if $\phi \in 3SAT$, then $\psi \in 3SAT$ and if $\phi \notin 3SAT$, then any assignment to ψ violates at least α fraction of the clauses. This provides a hardness of approximation result for MAX-3SAT (i.e., the problem of finding the assignment that satisfies the most number of clauses in a given 3CNF formula).

Theorem 67. *There exists a constant* $0 < \alpha < 1$ *such that MAX-3SAT is* $(1-\alpha)$*-hard to approximate unless* $P = NP$.

Observe that the theory of NP-completeness provides $\alpha = 1/n$ instead of a constant. The PCP Theorem amplifies this sub-constant soundness $1/n$ to some constant α. Starting from the PCP Theorem, it is possible to derive hardness of approximation results for several optimization problems.

9.2 Dinur's Proof of the PCP Theorem

We define the NP-complete problem called CONSTRAINT-GRAPH (CG). An instance of CG is of the form $G = ((V,E), \Sigma, \mathcal{C})$ where (V,E) is an undirected graph, Σ a constant-sized set of colors and $\mathcal{C}$ is a set of constraint functions, one corresponding to each graph edge, i.e., $\mathcal{C} = \{c_E : \Sigma^2 \to \{0,1\} | e \in E\}$. A coloring that assigns color c_1 to vertex v_1 and c_2 to vertex v_2 is said to satisfy the coloring constraint $c_{(v_1,v_2)}$ on edge (v_1,v_2) is $c_{(v_1,v_2)}(c_1,c_2) = 1$. An instance $((V,E),\Sigma,\mathcal{C})$ is an YES instance of CG, if there exists a coloring $\sigma : V \to \Sigma$ that satisfies all of $\mathcal{C}$. Since 3-COLORING is NP-complete, it easily follows that CG is also NP-complete.

The α-approximate version of CG is whether there exists a coloring that satisfies at least $(1-\alpha)$-constraints. In today's lecture, we will prove the following equivalent version of the PCP Theorem. It is an easy exercise to show that these two versions are equivalent. Now the PCP theorem can be equivalently stated as follows.

Theorem 68 (Dinur [Din])**.** *There exists a constant* α *such that the* α*-approximation version of CONSTRAINT-GRAPH is NP-hard.*

We will restate this theorem in a more convenient form in terms of a reduction. For this we need some notation. For any instance $G = ((V,E),\Sigma,\mathcal{C})$ of CG, let $n = |\mathcal{C}|$ and $size(G) = |V| + |E|$. Let σ_G be the best colorings for G (i.e., it is the coloring that violates the least number of edge constraints.) If there is more than one, set σ_G to be one of them arbitrarily. Let $UNSAT(G)$, called the *unsatisfiability factor*, be the fraction of edge constraints violated by σ_G. Note that if there exists a coloring that satisfies all the constraints of G, then $UNSAT(G) = 0$. Since CG is itself a NP-complete problem, the above theorem can be restated as follows in terms of a reduction from the decision version of CG to its approximation version.

Theorem 69 (PCP Theorem as a reduction)**.** *There exists a constant* α *and a polynomial time reduction from CG to the* α*-approximation version of CG that maps the instance,* $G = ((V,E),\Sigma,\mathcal{C})$ *to the instance,* $G' = ((V',E'),\Sigma',\mathcal{C}')$ *such that*

- $size(G') = poly(size(G))$
- *Completeness:* $UNSAT(G) = 0 \Rightarrow UNSAT(G') = 0$
- *Soundness:* $UNSAT(G) \geq 1/n \Rightarrow UNSAT(G') \geq \alpha$

We will prove the above theorem by applying the following *Gap Amplification* Lemma for $O(\log n)$ steps.

Lemma 70 (Gap Amplification Lemma)**.** *There exists a constant $0 < \alpha < 1$, a color set Σ and a polynomial time reduction from CG to itself mapping the instance, $G = ((V, E), \Sigma, \mathcal{C})$ to the instance, $G' = ((V', E'), \Sigma', \mathcal{C}')$ such that*

- $size(G') = O(size(G))$ *and* $\Sigma' = \Sigma$
- $UNSAT(G) = 0 \Rightarrow UNSAT(G') = 0$
- $UNSAT(G') \geq 2 \cdot \min(UNSAT(G), \alpha)$

We can now prove the PCP Theorem 69 starting from the Gap Amplification Lemma.
Proof of Theorem 69: We first observe that the gap amplification increases the unsatisfiability factor of the instance G by a factor of 2 (if it is not already a constant) and in doing so it blows up the size of the instance by at most a constant factor. We can hence apply this lemma $O(\log n)$ times to improve the gap from $1/n$ to α with at most a polynomial blowup in size, thus proving the PCP Theorem 69. □

Thus it suffices for us to prove the Gap Amplification Lemma 70 and this will be our goal for the rest of the lecture. But first we need some preliminaries regarding edge expansion of expanders.

9.3 Expanders – Edge Expansion

For a graph $G = (V, E)$, the edge expansion $\phi(G)$ is defined as follows:

$$\phi(G) = \min_{|S| \leq n/2} \frac{E(S, \overline{S})}{|S|}.$$

The following lemma provides an inverse relationship between the edge expansion, $\phi(G)$ and the spectral expansion, $\lambda(G)$.

Lemma 71. *For a d-regular graph G,* $\frac{\phi^2(G)}{2d} \leq 1 - \lambda(G) \leq \frac{2\phi(G)}{d}$

Thus, if we a graph with good spectral expansion, it also has good edge expansion. Since we know how to construct good spectral expanders, we can also assume the following. There exists a family of constant degree edge-expanders that can be explicitly constructed, i.e., there exists a constant ϕ_0, such that for all n, there exists a d-regular graph G_n on n vertices such that $\phi(G_n) \geq \phi_0$ and furthermore there exists an explicit construction of such graphs.

We also require the following estimate on the random-like behavior of random walk on an expander.

Lemma 72. *Let $G = (V, E)$ be a d-regular graph with spectral expansion λ. Let $B \subset E$ be a set of edges. The probability p that a random walk that starts at a random edge in B takes its $(i+1)^{\text{st}}$ step in B as well, is bounded above by $\frac{|B|}{|E|} + \lambda^i$.*

Note that if the edges were chosen randomly and independently (instead of choosing them along a random walk) then the above probability p is exactly $\frac{|B|}{|E|}$. The above lemma states that choosing the edges according to a random walk worsens this probability by at most λ^i.

The proof of this lemma is similar to that of the Expander Mixing Lemma. This proof is reproduced verbatim from Dinur's paper [Din].

Proof. Let K be the distribution on vertices of G induced by selecting a random edge in B and then a random vertex on which the edge is incident on. Let W be the support of the distribution K. As always, let A be the normalized adjacency matrix of G.

Let π be the vector corresponding to the distribution K. Hence, π_v is the fraction of edges incident on v that are in B, divided by 2. For any vertex v, let B_v denote the set of edges incident on v that are in B. Hence, $\pi_v = |B_v|/2|B| \le d/2|B|$ since G is d-regular. Let y_v be the probability that a random step from v is in B, so $y_v = |B_v|/d = 2|B|\pi_v/d$. The probability p equals the probability of landing in W after i steps and then taking a step in B. Hence

$$p = \sum_{v \in W} y_v (A^i \pi)_v = \sum_{v \in V} y_v (A^i \pi)_v = \langle y, A^i x \rangle.$$

let u be all ones vector. Decomposing π along u and its orthogonal component we have $\pi = \pi^{\|} + \pi^{\perp}$. Observe that

$$\|\pi\|_2^2 \le \left(\sum_v \pi_v \right) \cdot \left(\max_v \pi_v \right) \le 1 \cdot \frac{d}{2|B|} = \frac{d}{2|B|}.$$

Since G has spectral expansion λ,

$$\begin{aligned} \|A^i \pi^{\perp}\|_2 &\le \lambda^i \|\pi^{\perp}\|_2 \\ &\le \lambda^i \|\pi\|_2 \\ &\le \lambda^i \sqrt{\frac{d}{2|B|}} \end{aligned}$$

By Cauchy-Schwarz,

$$\langle y, A^i \pi^{\perp} \rangle >\le \|y\|_2 \|A^i \pi^{\perp}\|_2 \le \frac{2|B|}{d} \lambda^i \|\pi\|_2^2 \le \lambda^i$$

Combining we have,

$$p = \langle y, A^i \pi \rangle = \langle y, A^i \pi^{\|} \rangle + \langle y, A^i \pi^{\perp} \rangle \le \frac{2|B|}{dn} + \lambda^i = \frac{|B|}{|E|} + \lambda^i.$$

□

9.4 Proof of Gap Amplification Lemma

The reduction in the gap amplification lemma is achieved by a two-step process:

Preprocessing step This preprocessing step converts an arbitrary graph into a constant degree expander and worsens the unsatisfiability by at most a constant factor. This step blows up the size by at most a constant factor.

Powering step Assuming that the graph is a constant-degree graph, this step performs a "powering operation" that amplifies the unsatisfiability factor of the graph while blowing up the size by at most a constant factor. This step increase the size of the color-set Σ. There exist standard techniques, namely proof composition in PCP technology, that are precisely designed for reducing the size of the color-set. Proof Composition attains color-set reduction while just mildly worsening other parameters. For want of time, we will not consider this issue in lecture but for stating the required result.

9.4.1 Graph Preprocessing

The preprocessing step involves converting the graph into a constant degree expander graph. This is performed in two steps: (a) converting the graph into a constant degree graph and (b) "expanderizing" the constant degree graph.

Conversion into a constant degree graph Let G_n be a family of expander graph with degree d and edge expansion at least ϕ_0.

The given graph $G = (V, E)$ is transformed as follows: A vertex, v with degree d_v is replaced by an expander G_{d_v} on d_v vertices and the edges incident on v are now assigned to the vertices of G_{d_v}, one edge per vertex. All the vertices in the transformed graph, $G' = (V', E')$ thus have degree $d + 1$, where d is the degree of any graph in the expander family. All the edges inside each expander graph have equality constraints while the external edges retain the constraint they had earlier.

$$|V'| = \sum d_v = 2|E|$$

$$|E'| = \frac{d+1}{2}|V'| = (d+1)|E|$$

Thus, the size of the new graph $G' = (V', E')$ is at most a constant factor that of G.

Clearly, if $UNSAT(G) = 0$, then so is $UNSAT(G')$.

We now need to show that if $UNSAT(G)$ is non-zero, then $UNSAT(G')$ is worsened (i.e., reduced) at most by a constant factor. The intuition is that we can try to cheat by giving different colors to the d_v vertices. However, due to the property of the expander, this will result in violating several of the equality constraints within each expander.

Let $\sigma' = \sigma'_{G'} : V' \to \Sigma$ be the best coloring for G'. From this, we can obtain a coloring $\sigma : V \to \Sigma$ for G, in which the color of a vertex v is the most popular of the colors assigned to the corresponding "cloud" of d_v vertices in G'.

Let $\mu = UNSAT(G)$. Let B be the set of edges violated by σ in G and B' be the set of edges violated by σ' in G'. Define S to be the set of vertices in G' whose color is not the popular one (in the corresponding cloud). Since every edge in B should either be in B' or contribute to S, we have $\mu|E| \leq |B| \leq |B'| + |S|$.

- *Case 1*: $|B'| \geq \mu|E|/2$

$$UNSAT(G') = \frac{|B'|}{|E'|} \geq \frac{\mu|E|}{2|E'|} = \frac{\mu}{2(d+1)} = \frac{UNSAT(G)}{2(d+1)}$$

- *Case 2*: $|S| \geq \mu|E|/2$

  Consider any vertex v in G and its corresponding cloud of vertices in G'. Let S^v be the set of vertices in the cloud which did not get the popular color. For each color a, define $S_a^v = \{u \in S^v | \sigma'(u) = a\}$. By the definition of popularity, $|S_a^v| < d_v/2$. Now, from the expansion property within each cloud, we get that $|E(S_a^v, \overline{S_a^v})| \geq \phi_0 |S_a^v|$. Note that the constraints for all the edges in $E(S_a^v, \overline{S_a^v})$ are violated. Summing over the colors and clouds,

$$|B'| \geq \frac{\sum |E(S_a^v, \overline{S_a^v})|}{2} \geq \frac{\phi_0 |S|}{2} \geq \frac{\mu\phi_0}{4}|E| \geq \frac{\mu\pi_0}{4(d+1)}|E'|$$

  Thus, $UNSAT(G') \geq UNSAT(G)\frac{\mu\phi_0}{4(d+1)}$

In either case, the transformation results in at most a constant factor drop in the fraction of violated edge constraints.

The graph G is thus converted into a constant degree $(d+1)$ graph G'.

Expanderizing the graph The transformed graph G' is $(d+1)$-regular. We superimpose with a $\tilde{d}$-regular expander on $|V'|$ nodes (i.e, th new superimposed graph has the same vertex set as the original constraint edges, its edges are however the union of the two graphs – the original constraint graph and the expander). Furthermore, we add self-loops for each vertex to get G''. We then impose dummy constraints on the new edges (i.e., constraint that are always satisfied). G'' is still an expander (with constant degree, $(d+2+\tilde{d})$), but with slightly weaker spectral expansion given as follows: (this calculation uses Lemma 71)

$$\lambda(G'') \leq 1 - \frac{\phi^2(G'')}{2(d+\tilde{d}+2)} \leq 1 - \frac{\phi^2(G_{|V'|})}{2(d+\tilde{d}+2)} \leq 1 - \frac{\phi_0^2}{2(d+\tilde{d}+2)} = \lambda(\text{say}).$$

Observe that if G' is satisfiable, so is G''.

$$UNSAT(G') = \mu \Rightarrow UNSAT(G'') = \mu\left(\frac{d+1}{d+2+\tilde{d}}\right)$$

Thus, $G = (V, E)$ is converted into a constant-degree $\Delta = (d+\tilde{d}+2)$ expander graph $G'' = (V'', E'')$ with spectral expansion λ. This completes the preprocessing step.

Hence, we will assume without loss of generality that the given constrain graph $G = (V, E)$ is Δ-regular and has spectral expansion λ.

9.4.2 Graph Powering

Due to the preprocessing step, we can assume without loss of generality that the given constrain graph $G = (V, E)$ is Δ-regular expander graph with self loops and has spectral expansion λ. Suppose the unsatisfiability factor of this graph is α. In other words, every coloring of the graph violates at least α-fraction of the edge-constraints. We need to double this factor to 2α without blowing up the size of the graph too much. A natural way to do this is the powering operation. i.e., we build a new *powered graph* G^t on the same set of vertices, each edge of which corresponds to a walk of length t in the original graph.

More formally, the power graph $G^t = ((V', E'), \Sigma', \mathcal{C}')$ for some parameter $t \in \mathbb{Z}^{\geq 0}$ is defined as follows.

- $V' = V$ i.e., the set of vertices is the same.
- $(u, v) \in E'$ iff there exists a walk of length t between the vertices u and v in the original graph G. I.e., there exist vertices $v_0, v_1, \dots, v_t \in V$ such that $v_0 = u$, $v_t = v$ and $(v_{i-1}, v_i) \in E$ for $i = 1, \dots, t$. Thus, every edge in E' is a t-walk in G. To distinguish between edges of G and G^t, we will refer to the edges of G as edges and the edges of G^t as t-walks or (just) walks.
- $\Sigma' = \Sigma^{\Delta^{\lceil t/2 \rceil}}$. More precisely, the color $\sigma' \in \Sigma'$ of any vertex v gives not only the color of the vertex v, but also v's opinion of the colors of all vertices which are reachable from v by a walk of length at most $t/2$. Note, such a coloring allows for the case that two (different) vertices u and v could have different opinions about the color of some other vertex which is within distance $t/2$ of both vertices u and v.
- The set of constraints $\mathcal{C}' = \{c_w : \Sigma' \times \Sigma' \to \{0,1\} | w \in E'\}$ is defined as follows: For any t-walk $w = (u, v) \in E'$, the constraint c_w checks that the opinion of the vertices v and u agree on their intersection and that they satisfy all the edge constraints along the walk (u, v). We call each such constraint a walk-constraint.

Below we give a very informal (and in fact wrong) argument as to why powering should amplify the unsatisfiability factor. Let us make two egregious assumptions. Suppose the coloring σ' to the vertices of $G' = (V', E')$ satisfies the property, that if two vertices have an opinion about a third vertex, then their opinions are consistent. Furthermore, let us assume that the edges along a random t-walk appear like t random edges. Then, the following calculations show that the fraction of violated constraints in the powered graph increases by a factor of t. Since by assumption one, the coloring $\sigma : V \to \Sigma'$ is consistent across the vertices, we have that σ' is actually derived from a coloring $\sigma : V \to \Sigma$ of the original constraint graph. However, we know that σ violates at least α-fraction of the edge-constraints in the original graph G. Consider any such edge. This edge occurs in exactly $t\Delta^{t-1}$ walks of the powered graph and the constraints corresponding to each of these walks is violated. Hence, the fraction of violated walk-constraints in G' is $\approx \frac{t\Delta^{t-1}\alpha|E|}{|E'|} \approx \frac{t\Delta^{t-1}\alpha|E|}{\Delta^{t-1}|E|} = t \cdot \alpha$. Thus the unsatisfiability factor increases t-fold. This argument is wrong as both the assumptions are wrong. We won't be able to completely get over the first assumption. However, we will be able to show that if a violated edge occurs in the middle of a walk (ie., between $t/2 - \sqrt{t}$ and $t/2 + \sqrt{t}$), then it is very likely that the constraint on the walk is also violated. Regarding the second assumption, we will use the fact that the underlying graph is an expander and hence the set of edges along a walk do in "some sense" look random (See Lemma 72). Using both these we will prove the following lemma which shows that the unsatisfiability factor is amplified $\sqrt{t}$-fold instead of t-fold as in fallacious argument above.

Lemma 73. *There exists $\beta > 0$ such that if $UNSAT(G) \leq 1/\sqrt{t}$, then $UNSAT(G') \geq \beta\sqrt{t}\ UNSAT(G)$.*

Proof. Let $\sigma' : V \to \Sigma'$ be the best possible coloring satisfying the most number of walk-constraints on G'. Hence, $\alpha = UNSAT(G')$ is exactly the fraction of walk-constraints violated by σ'. From the coloring σ', we build a coloring $\sigma : V \to \Sigma$ for the original constraint graph G as follows: Define the random variable $X_{v,i}$ to be the opinion that a vertex which is i random steps away from v has about v. Let $\sigma(v) = a$ such that $\Pr[X_{v,t/2} = a]$ is maximized. In other words, this is the most popular color for v assigned by vertices which are at a distance $t/2$ far from v. By definition of popularity, we have that if $\sigma(v) = a$, then $\Pr[X_{v,t/2} = a] \geq \frac{1}{|\Sigma|}$.

Let B be the set of edges violated by σ in G. Since σ can color no better than the best coloring for G, we have $\frac{|B|}{|E|} \geq UNSAT(G) = \alpha$.

Consider any edge $e = (u, v) \in B$. Let $i \in I = [t/2 - \sqrt{t}, t/2 + \sqrt{t}]$. Consider a random t-walk $w = (v_0, v_1, \ldots, v_t)$ in G' conditioned on the fact that e is the i^{th} edge of the walk (i.e, $u = v_{i-1}$ and $v = v_i$). We will now analyze the probability that the coloring assigned by σ' to the end-vertices of the walk v_0 and v_t violates the walk-constraint c_w.

Let $\sigma(u) = a$ and $\sigma(v) = b$. We know that the coloring a and b to vertices u and v respectively violates the edge constraint c_e. Now, if the opinion of the color of $u = v_{i-1}$ held by the σ'-color of v_0 is a and that of $v = v_i$ held by the σ'-color of v_t is b, then the walk constraint c_w is violated. These events are $X_{u,i-1} = a$ and $X_{v,t-i} = b$. Hence,

$$\Pr_{w\text{: } (u,v) \text{ is } i\text{th edge of } w}\big[c_w \text{ is violated}\big] \;\geq\; \Pr[X_{u,i-1} = a] \cdot \Pr[X_{v,t-i} = b] \tag{9.1}$$

For simplicity, let us consider the case that $i - 1 = t/2$ and $t - i = t/2$. In other words, e is exactly the middle edge. This case actually does not arise since it assumes the walk is of length $t + 1$ as opposed to t. In this case, we have that $\Pr[X_{u,i-1} = a] = \Pr[X_{u,t/2} = a]$ which is at least $1/|\Sigma|$ since $\sigma(u) = a$ which implies a is the most popular color assigned to u by σ'. Similarly, $\Pr[X_{v,t-i} = b] \geq 1/|\Sigma|$. Thus, in this case we have that

$$\Pr_{w\text{: } (u,v) \text{ is } i\text{th edge of } w}\big[c_w \text{ is violated}\big] \;\geq\; \frac{1}{|\Sigma|^2}$$

We now have to consider the more general case when e is the i^{th} edge for some $i \in I$. Note that this i satisfies the property that $|i - t/2| \leq \sqrt{t}$, i.e., it is within one standard deviation of the mean. The general idea is that since i is at most one standard deviation of the mean, the behavior at i is similar up to certain constant factors to that at the mean. More formally, it can be shown that

$$\text{There exists } \tau > 0 \text{ such that if } |l - t/2| \leq \sqrt{t}, \text{ then } Pr[X_{u,l} = a] \geq \tau \cdot \Pr[X_{u,t/w} = a] \tag{9.2}$$

This is proved by considering the random walk at u using properties of the binomial distribution. The main intuition is that self loops of G make the distribution of vertices reached by a random $t/2$-step walk from u roughly the same as the distribution on vertices reached by an l-step from u, for $l \in I$. This argument can be formalized; for want of time, we do not present the proof in lecture.

Thus, it follows from (9.1) and (9.2) that for all $i \in I$,

$$\Pr_{w:\,(u,v) \text{ is } i\text{th edge of } w}\left[c_w \text{ is violated}\right] \geq \left(\frac{\tau}{|\Sigma|}\right)^2 = \mu \text{ (say)} \tag{9.3}$$

So far we have shown that an edge is bad, then a constant fraction of the walks in which it occurs nearly in the middle are also bad. We will now show that these bad walks do not overlap too much and hence there is not too much of over-counting. For this purpose, we define the random variable N. Let w be a random t-walk in the powered graph G' (ie., chosen by starting at a random vertex and walking t random steps),

$$N = \begin{cases} \text{Number of bad edges in } I \text{ if } w \text{ is a rejecting } t\text{-walk according to } \sigma' \\ 0 \text{ otherwise} \end{cases}$$

i.e., N is the number of bad edges encountered along the walk w around the middle of the walk if any bad edges are encountered at all and is 0 otherwise.

Clearly this definition of N satisfies,

$$UNSAT(G') \geq \Pr[N > 0].$$

So it suffices to lower bound $\Pr[N > 0]$. We do so using the following two claims.

Claim 74. *There exists $\mu > 0$ such that $E[N] \geq 2\mu\sqrt{t}\frac{|B|}{|E|}$.*

Claim 75. *There exists $C > 0$ such that $E[N^2] \leq C\sqrt{t}\frac{|B|}{|E|}$.*

We can now bound $\Pr[N > 0]$ using the second moments inequality as follows:

$$\begin{aligned} \Pr[N > 0] &\geq \frac{(E[N])^2}{E[N^2]} \\ &\geq \frac{4\mu^2}{C} \cdot \sqrt{t} \cdot \frac{|B|}{|E|} \end{aligned}$$

Choosing $\beta = \frac{4\mu^2}{C}$ completes the proof of Lemma 73. □

We now need to prove Claims 74 and 75. For this purpose, we define the following two random variables. For a random walk w, let

$$\begin{aligned} Z_i &= \begin{cases} 1 & \text{if } i^{\text{th}} \text{ edge of } w \text{ is in } B \\ 0 & \text{otherwise} \end{cases} \\ Y_i &= \begin{cases} 1 & \text{if } w \text{ is a rejecting } t\text{-walk and } i^{\text{th}} \text{ edge } \in B \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Observe that $Y_i \leq Z_i$ always.

Proof of Claim 74: We observe that $N = \sum_{i \in I} Y_i$. Hence,

$$\begin{aligned} E[N] &= \sum_{i \in I} E[Y_i] \\ &= \sum_{i \in I} \Pr[Y_i = 1] \\ &= \sum_{i \in I} \Pr[Y_i = 1 | Z_i = 1] \cdot \Pr[Z_i = 1] \end{aligned}$$

$\Pr[Z_i = 1]$ is just the probability that the i^{th} edge of a random walk is in B and is thus $|B|/|E|$. $\Pr[Y_i = 1|Z_i = 1]$ is the probability that the random walk w violates the constraint c_w conditioned on the fact that the i^{th} edge is in B. This is precisely the probability calculated in (9.3). Hence,

$$\begin{aligned} E[N] &\geq \sum_{i \in I} \mu \cdot \frac{|B|}{|E|} \\ &= 2\mu\sqrt{t} \cdot \frac{|B|}{|E|} \end{aligned}$$

$\square$

Proof of Claim 75: Using $Y_i \leq Z_i$, we have $N \leq \sum_{i \in I} Z_i$. Hence,

$$\begin{aligned} E[N^2] &\leq E\left[\left(\sum_{i \in I} Z_i\right)^2\right] \\ &= 2 \sum_{i \in I} \sum_{j \in I, j \geq i} E\left[Z_i Z_j\right] \\ &= 2 \sum_{i \in I} \Pr[Z_i = 1] \left(\sum_{j \in I, j \geq i} \Pr[Z_j = 1 | Z_i = 1]\right) \\ &= \frac{2|B|}{|E|} \sum_{i \in I} \sum_{j \in I, j \geq i} \Pr[Z_j = 1 | Z_i = 1] \end{aligned}$$

The probability $\Pr[Z_j = 1|Z_i = 1]$ is precisely the probability that a random walk has its $(j - i + 1)$th edge in B conditioned on the fact that the first edge of the walk is in B. Here, we use the fact that G is an expander and use Lemma 72. We thus, have

$$\begin{aligned} E[N^2] &\leq \frac{2|B|}{|E|} \sum_{i \in I} \sum_{j \in I, j \geq i} \left(\frac{|B|}{|E|} + \lambda^{j-i-1}\right) \\ &\leq C\sqrt{t}\frac{|B|}{|E|} \qquad \text{since } \frac{|B|}{|E|} \leq \frac{1}{\sqrt{t}} \end{aligned}$$

where C is some constant. $\square$

This almost completes the proof of the Gap Amplification Lemma, modulo one fact – the size of the color set has expanded fro $|\Sigma|$ to $\left|\Sigma^{\lceil \Delta^{t/2} \rceil}\right|$. We now, perform what is known as Proof Composition, to reduce the size of the color set. Proof Composition is a standard procedure (introduced by Arora and Safra [AS]), to reduce the size of the alphabet while only mildly worsening other parameters. After proof composition, we have a similar statement to Lemma 73 only with a smaller β, but the color set being the same Σ.

Thus, by choosing an appropriate constant t, we obtain the Gap Amplification Lemma, from which the proof of the PCP theorem follows.

Bibliography

[**Main References**]

[LW] Nathan Linial and Avi Wigderson, Lecture notes for a course on expanders, Hebrew University, Israel.
`http://www.math.ias.edu/~boaz/ExpanderCourse/`

[Gur1] Venkatesan Guruswami, Lecture notes for a course on Codes and Pseudorandom objects, University of Washington, Seattle.
`http://www.cs.washington.edu/education/courses/590vg/03wi/notes.html`

[Vad] Salil Vadhan, Lecture notes for a course on pseudorandomness, Harvard University, Spring 2004.
`http://www.courses.fas.harvard.edu/~cs225/Lectures/`

Bibliography

[**Other References**]

[AKS] Miklos Ajtai, Janos Komlos, Endre Szemeredi: "Deterministic Simulation in LOGSPACE", STOC 1987: 132-140.

[AKLLR] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, Charles Rackoff: "Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems". FOCS 1979: 218–223

[Alo] Noga Alon, "Eigenvalues and expanders", Combinatorica 6(2): 83–96 (1986).

[AC] Noga Alon, and Fan R. K. Chung, "Explicit Constructions of linear sized tolerant networks", Discr. Math. 2, 15–19, 1988.

[ALMSS] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, Mario Szegedy. "Proof verification and the hardness of approximation problems". Journal of the ACM 45(3):501–555, 1998.

[AS] Sanjeev Arora, Shmuel Safra. "Probabilistic Checking of Proofs: A New Characterization of NP". Journal of ACM, 45(1):70–122, 1998.

[ATWZ] Roy Armoni, Amnon Ta-Shma, Avi Wigderson, Shiyu Zhou: "An $O\left(\log^{4/3} n\right)$ space algorithm for (s,t) connectivity in undirected graphs". J. ACM 47(2): 294–311 (2000).

[BZ] Alexander Barg and Gilles Zémor, "Error Exponents of Expander Codes", IEEE Transactions on Information Theory, 48(6):1725–1729 (2002).

[BGHSV] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, Salil Vadhan, "Robust PCPs of Proximity, Shorter PCPs and applications to coding", STOC 2004: 1–10.

[BSVW] Eli Ben-Sasson, Madhu Sudan, Salil P. Vadhan, Avi Wigderson: "Randomness-efficient low degree tests and short PCPs via epsilon-biased sets". STOC 2003: 612–621

[BL] Yonatan Bilu, and Nati Linial, "Lifts, discrepancy and nearly optimal spectral gaps", To appear in Combinatorica, (A preliminary version appeared in FOCS 2004).

[BLR] Manuel Blum, Michael Luby, Ronitt Rubinfeld: "Self-Testing/Correcting with Applications to Numerical Problems". J. Comput. Syst. Sci. 47(3): 549–595 (1993)

[Bou] Jean Bourgain: "On Lipschitz embedding of finite metric spaces in Hilbert space", Israel J. Math 52, 46–52 (1985).

[BW] Graham Brightwell, Peter Winkler: "Maximum Hitting Time for Random Walks on Graphs". Random Struct. Algorithms 1(3): 263–276 (1990)

[Din] Irit Dinur. "The PCP Theorem by Gap Amplification". ECCC Technical Report TR05–046, 2005.

[DR] Irit Dinur, Omer Reingold "Assignment Testers: Towards a Combinatorial Proof of the PCP-Theorem". FOCS 2004: 155–164

[Dol] Danny Dolev: "The Byzantine Generals Strike Again". J. Algorithms 3(1): 14–30 (1982)

[DPPU] Cynthia Dwork, David Peleg, Nicholas Pippenger, Eli Upfal: "Fault Tolerance in Networks of Bounded Degree". SIAM J. Comput. 17(5): 975–988 (1988)

[Fei1] Uriel Feige: "A Tight Upper Bound on the Cover Time for Random Walks on Graphs". Random Struct. Algorithms 6(1): 51–54 (1995)

[Fei2] Uriel Feige: "Collecting Coupons on Trees, and the Cover Time of Random Walks". Computational Complexity 6(4): 341–356 (1997)

[FGLSS] Uriel Feige, Shafi Goldwasser, Lázló Lovász, Shmuel Safra, Mario Szegedy. "Interactive proofs and the hardness of approximating cliques". Journal of ACM, 43(2):268–292, 1996.

[Fri] Joel Friedman, "A proof of Alon's second eigenvalue conjecture and related problems", CoRR cs.DM/0405020: (2004) (A preliminary version appeared in STOC 2003).

[GG] Ofer Gabber and Zvi Galil, "Explicit construction of linear size superconcentrators", JCSS 22(1981), 407–420.

[GGR] Oded Goldreich, Shafi Goldwasser, Dana Ron: "Property Testing and its Connection to Learning and Approximation". J. ACM 45(4): 653–750 (1998)

[GS] Oded Goldreich, Madhu Sudan: "Locally Testable Codes and PCPs of Almost-Linear Length". FOCS 2002: 13–22

[Gur2] Venkatesan Guruswami, "Error-correcting codes and Expander graphs", SIGACT News, 35(3): 25–41, September 2004.

[GI] Venkatesan Guruswami, Piotr Indyk, "Linear time encodable/decodable codes with near-optimal rate", To appear in IEEE Transactions on Information Theory. (Preliminary Version in STOC'02).

[HW] Shlomo Hoory, Avi Wigderson: "Universal Traversal Sequences for Expander Graphs". Inf. Process. Lett. 46(2): 67–69 (1993)

[Ist] Sorin Istrail: "Polynomial Universal Traversing Sequences for Cycles Are Constructible (Extended Abstract)". STOC 1988: 491–503

[INW] Russell Impagliazzo, Noam Nisan, Avi Wigderson: "Pseudorandomness for network algorithms". STOC 1994: 356–364

[IZ] Russell Impagliazzo, David Zuckerman: "How to Recycle Random Bits", FOCS 1989: 248–253.

[KPS] Richard Karp, Nicholas Pippenger and Michael Sipser: "A time-randomness tradeoff", AMS Conference on Probabilistic Computational Complexity, 1985.

[LLR] Nathan Linial, Eran London, Yuri Rabinovich: "The Geometry of Graphs and Some of its Algorithmic Applications". Combinatorica 15(2): 215–245 (1995)

[Lov] Lázló Lovász: "Random Walks on Graphs: A Survey", Combinatorics, Paul Erdös is Eighty, Vol. 2 Jänos Bolyai Mathematical Society, Budapest, 1996, 353–398

[LPS] Alexander Lubotzky, R. Phillips, P. Sarnak: "Ramanujan graphs". Combinatorica 8(3): 261–277 (1988)

[MR1] Neal Madras, Dana Randall: "Factoring Graphs to Bound Mixing Rates". FOCS 1996: 194–203

[Mar] Gregori A. Margulis. "Explicit constructions of expanders". Problemy Peredaci Informacii, 9(4):71–80, 1973.

[MR2] Russell A. Martin, Dana Randall: "Sampling Adsorbing Staircase Walks Using a New Markov Chain Decomposition Method". FOCS 2000: 492–502

[Nis] Noam Nisan: "Pseudorandom generators for space-bounded computation". Combinatorica 12(4): 449–461 (1992)

[PT] Wolfgang J. Paul, Robert Endre Tarjan: "Time-Space Trade-Offs in a Pebble Game". Acta Inf. 10: 111–115 (1978)

[PTC] Wolfgang J. Paul, Robert Endre Tarjan, James R. Celoni: "Space Bounds for a Game on Graphs". Mathematical Systems Theory 10: 239–251 (1977)

[Rei] Omer Reingold: "Undirected ST-connectivity in log-space". STOC 2005: 376–385 (See also ECCC Tech Report TR04–094, 2004).

[RVW] Omer Reingold, Salil Vadhan, and Avi Wigderson: "Entropy Waves, The Zig-Zag Graph Product, and New Constant-Degree Expanders and Extractors", Annals of Math 155: 157–187, 2002.

[SZ] Michael E. Saks, Shiyu Zhou: "$\mathrm{BP}_H\mathrm{Space}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$". J. Comput. Syst. Sci. 58(2): 376–403 (1999)

[Rob] Sara Robinson: "Computer Scientist Finds Small-Memory Algorithm for Fundamental Graph Problem", SIAM News, Volume 38, Number 1, January/February 2005.

[RS] Ronitt Rubinfeld, Madhu Sudan: "Robust Characterizations of Polynomials with Applications to Program Testing". SIAM J. Comput. 25(2): 252–271 (1996)

[Sav] Walter J. Savitch: "Relationships Between Nondeterministic and Deterministic Tape Complexities". J. Comput. Syst. Sci. 4(2): 177–192 (1970).

[SW] Amir Shpilka, Avi Wigderson: "Derandomizing homomorphism testing in general groups". STOC 2004: 427–435

[SS] Michael Sipser, Daniel A. Spielman: "Expander codes". IEEE Transactions on Information Theory 42(6): 1710–1722 (1996)

[Tan] Robert M. Tanner: "A recursive approach to low complexity codes", IEEE Trans. Information Theory, 27(5): 533–547, 1981.

[Upf] Eli Upfal: "Tolerating a Linear Number of Faults in Networks of Bounded Degree" Inf. Comput. 115(2): 312–320 (1994)

[Val] Leslie G. Valiant: "Graph-Theoretic Properties in computational Complexity." J. Comput. Syst. Sci. 13(3): 278–285 (1976)

[Zem] Gilles Zemor: "On Expander Codes", IEEE Transactions on Information Theory 47(2):835–837 (2001).