CSU498 PROJECT
REPORT

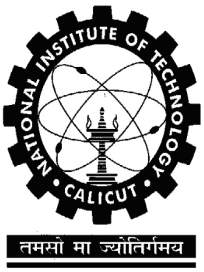
GRAPH EXPANDERS AND THEIR APPLICATIONS

Submitted in partial fulfilment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering
Submitted by

ANAND J B080144CS
KRISHNAPRASAD P B080207CS

Under the guidance of
Dr K Muralikrishnan



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY CALICUT
NIT CAMPUS PO, CALICUT
KERALA, INDIA 673601
February 28, 2012

Certificate

This is to certify that the project work entitled "**Graphs expanders and their applications**",submitted by Anand J (B080144CS) and Krishnaprasad P (B080207CS) to National Institute of Technology Calicut towards partial fulfillment of the requirements of the award of Degree Of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the work carried out by them under my supervision and guidance.

Project Guide
Dr. K Muralikrishnan
Assistant Professor

Place : Calicut
Date : 9-10-2011

Head of Department

Office Seal

Abstract

In graph theory, an expander graph is a sparse graph that has strong connectivity properties. 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. In graph theory, the zig-zag product of regular graphs G and H, takes a large graph (G) and a small graph (H), and produces a graph that approximately inherits the size of the large one but the degree of the small one. An important property of the zig-zag product is that if H is a good expander, then the expansion of the resulting graph is only slightly worse than the expansion of G. Hence it's a very useful method in creating expander graphs. Another important use of zig-zag product was that it was used in proving the $s-t$ connectivity problem in an undirected graph could be solved using a log space algorithm. This is an important result as it is used to prove that symmetric log-space and log-space are equal. The aim of this project is to study zig-zag product and the $s-t$ connectivity problem for the undirected graph and use it to work on other graph problems.

Contents

Chapter 1

Problem Definition

To study and analyze expander graphs, zig-zag product and $s-t$ connectivity problem for undirected graphs and use their properties to work in other graph problems.

Chapter 2

Introduction

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.

2.1 Background and Recent Research

Informally, an expander is an undirected graph that has relatively sparse density, but whose vertices are nevertheless highly connected. Consequently, expanders have the property that any small subset of the vertices has a large set of neighbours outside of the set. This simple graph property has led to highly useful results in a number of branches of computer science. Expander graphs have uses ranging from complexity theory and algorithm derandomization to coding theory and efficient network design. Although it has been proven that a random graph of constant degree at least 3 is an expander with high probability, many of the results in the fields above require an efficiently computable explicit method of construction of infinite families of growing size expanders [**?**].

The original motivation for expanders is to build economical robust networks: an expander with bounded valence is precisely an asymptotic robust graph with number of edges growing linearly with size (number of vertices), for all subsets. Expander graphs have found extensive applications in computer science, in designing algorithms, error correcting codes, extractors, pseudorandom generators, sorting networks and robust computer networks. They have also been used in proofs of many important results in computational complexity theory, such as $SL = L$ [**?**] and the PCP theorem [**?**]. In cryptography, expander graphs are used to construct hash functions.

2.2 Expander graphs

Consider a graph $G = (V, E)$ where V and E represents the vertex set and edge set of G respectively. Let $N(v)$ denote the neighbor set of any vertex $v \in V$. Similarly let $N(S)$ be the set of all neighbors of a vertex set $S \subseteq V$.

A graph is said to have a vertex expansion (k, a) if $\forall\ S \subseteq V$ with $\mid S \mid \leq\ k$,

$$\mid N(S) \mid \geq\ a. \mid S \mid$$

where $k \in I^+, a \in R^+$.

We say that G is a (k, a) expander [**?** **?**].

Let $G(V, E)$ be a $D-$regular graph which can have self loops and parallel edges. Let $|v| = n$. For an $S \subseteq V$, ∂S denote the edge boundary of S, which is defined as the set of edges from S to $V - S$. The expansion parameter $h(G)$ is a constant that measures how well connected a graph G is. Large $h(G)$ means highly connected.

$$h(G) = \min_{\{S||S|\leq n/2\}} \frac{|\partial S|}{|S|}$$

While studying about expanders, the major aspects that are studied are the possibility of existence of the required expanders, eigen vectors and its relation to expanders and also the relationship between random walks and expanders.

A key property of the random walk on an expander graph is that it converges rapidly to its limit distribution [**?**]. This fact has numerous important consequences in various fields.

Let $A(G)$ be the adjacency matrix of G. $A(G)$ is symmetric and sum of every row and every column is D since G is undirected and $D-$regular. Let $A(G)$ has orthonormal base $v_0, ..., v_{n-1}$ with eigenvalues $\mu_0, ..., \mu_{n-1}$ respectively. Without loss of generality we can say that $\mu_0 \geq \mu_1 \geq ... \geq \mu_{n-1}$. The eigen values of $A(G)$ are referred to as the spectrum of G.

Observations:

1. $\mu_0 = D, v_0 = (1, 1, ..., 1)$
2. $\lambda = \max\{|\mu_1|, |\mu_{n-1}|\}$ is a good measure of graph expansion. And $(D - \lambda)$ is called the spectral gap of G.

Theorem 2.1 (Cheeger's Inequality)

$$\frac{D - \lambda}{2} \leq h(G) \leq \sqrt{2D(D - \lambda)}$$

Proof See [**?**] □

2.2.1 Random walk on expander graph

A random walk on a graph is an iterative process through which, starting from an initial random vertex, we move a token from vertex to vertex by choosing an edge based on a probability distribution on the outgoing edge of the current vertex. Since we are dealing with constant degree(D), unweighted graphs(G), from a vertex v the probability we move from v its neighbour v_i is uniformly $\frac{1}{D}$. If A is the adjacency matrix of our D-regular graph G then we will conduct a random walk on the graph with transition probabilities based on the weights of the edges of $\hat{A} = \frac{A}{D}$, where $\hat{A}$ is called the normalized adjacency matrix of A.

2.2.2 Properties of expander graphs

Property 2.1 (Expander mixing lemma) *Let $G = (V, E)$ be a d-regular graph with (un-)normalized second-largest eigenvalue λ. Then for any two subsets $S, T \subseteq V$, let $E(S, T)$ denote the number of edges between S and T. We have*

$$\mid E(S,T) - \frac{d \mid S \mid . \mid T \mid}{n} \mid \leq \lambda \sqrt{\mid S \mid . \mid T \mid}$$

Proof Denote by χ_S and χ_T the characteristic vectors of S and T(χ_S is a vector with ones $\forall\ v \in S$ and zeros in all other places).Let $\chi_S = \sum_i \alpha_i v_i$ and $\chi_T = \sum_j \beta_j v_j$ be their representation as linear combinations of the orthonormal base $v_0, ..., v_{n-1}$, where $v_0 = \frac{\bar{1}}{\sqrt{n}}$.Let μ_i be the eigenvalue of v_i. We have:

$$\begin{aligned} |E(S,T)| &= \chi_S A \chi_T \\ &= (\sum_i \alpha_i v_i)\ A\ (\sum_j \beta_j v_j) \end{aligned}$$

and since the v_i's are eigenvectors and orthonormal:

$$\begin{aligned} |E(S,T)| &= (\sum_i \alpha_i v_i)\ (\sum_j \beta_j A v_j) \\ &= (\sum_i \alpha_i v_i)\ (\sum_j \beta_j \mu_j v_j) \\ &= \sum_i \mu_i \alpha_i \beta_i \end{aligned}$$

Since $\alpha_0 = \langle \chi_S, \frac{\bar{1}}{\sqrt{n}} \rangle = \frac{|S|}{\sqrt{n}}$ and $\beta_0 = \frac{|T|}{\sqrt{n}}$

$$|E(S,T)| = \mu_0 \frac{|S||T|}{n} + \sum_{i=1}^{n-1} \mu_i \alpha_i \beta_i$$
$$= d\frac{|S||T|}{n} + \sum_{i=1}^{n-1} \mu_i \alpha_i \beta_i$$

Due to the triangle inequality and the definition of λ:

$$||E(S,T)| - d\frac{|S|.|T|}{n}| = |\sum_{i=1}^{n-1} \mu_i \alpha_i \beta_i|$$
$$\leq \sum_{i=1}^{n-1} |\mu_i \alpha_i \beta_i|$$
$$\leq \lambda \sum_{i=1}^{n-1} |\alpha_i \beta_i|$$

And by the Cauchy-Schwartz inequality:

$$||E(S,T)| - d\frac{|S|.|T|}{n}| \leq \lambda \parallel \alpha \parallel \parallel \beta \parallel$$
$$= \lambda \parallel \chi_S \parallel \parallel \chi_T \parallel$$
$$= \lambda \sqrt{|S||T|}$$

□

Property 2.2 *Given an intial probability distribution $\pi_0 = (p_0^0, p_1^0, ..., p_n^0)$ over the n vertices of a constant degree graph G with normalized adjacency matrix A. The probability distribution after doing a random walk in the graph π_1 is given by $A\pi_0$ [**?**].*

Proof Let vertex X be selected initially at random over this distribution. Now, let Y be the uniformly chosen neighbour of X. Since Y represents the probability distribution that a given vertex will be chosen after 1 step, $Y = \pi_1$. Enough to prove that the random variable Y is given by the matrix product $A\pi_0$.

We are looking for the probability $Pr[Y = i]$ that Y is vertex i. This is of course conditioned on the probabilistic choice of X. Thus we have

$$
\begin{aligned}
Pr[Y = i] &= \sum_j Pr[Y = i|X = j]Pr[X = j] \\
&= \sum_j A_{ij} p_j^0 \\
&= (A\pi_0)_i \\
\therefore \pi_1 &= A\pi_0
\end{aligned}
$$

Corollary 2.1

$$\pi_{i+1} = A\pi_i = A^{i+1}\pi_0$$

,where π_i is the probability distribution at step i in the random walk.

where the second to last step follows since A gives the transition probability of each edge, thus if we are at node j, the probability of selecting node i is A_{ij}. Thus the full vector $A\pi_0$ gives the value of the random variable Y, where the i^{th} component is the probability of being at node i after one step. Thus $A\pi_0 = \pi_1$. □

Property 2.3 *Let A be the normalized adjacency matrix of G with eigen vectors $v_0, v_1, ..., v_{n-1}$ with corresponding ordered eigen values $\mu_0 \geq \mu_1 \geq .... \geq \mu_{n-1}$. Also let π be a probability distribution over the vertices of G such that $\pi \perp u, u$ is uniform distribution. Then*

$$\| A\pi \| \leq \lambda \| \pi \|$$

*,where λ is the second largest eigen value in absolute value [**?**].*

Proof Assume that the eigen vectors are all normalized. Then, it is clear that $v_0 = u$, since v_0 has the corresponding eigen value $\mu_0 = 1$. Then we can decompose π such that $\pi = c_1 v_1 + c_2 v_2 + ... + c_{n-1} v_{n-1}$. Thus

$$
\begin{aligned}
\pi &= c_1 v_1 + c_2 v_2 + ... + c_{n-1} v_{n-1} \\
A\pi &= c_1 A v_1 + c_2 A v_2 + ... + c_{n-1} A v_{n-1} \\
A\pi &= c_1 \mu_1 v_1 + c_2 \mu_2 v_2 + ... + c_{n-1} \mu_{n-1} v_{n-1} \\
\| A\pi \|^2 &= (c_1\mu_1)^2 + (c_2\mu_2)^2 + ... + (c_{n-1}\mu_{n-1})^2 \\
\| A\pi \|^2 &\leq \lambda^2 (c_1^2 + c_2^2 + ... + c_{n-1}^2) \\
&= \lambda^2 \| \pi \|^2 \\
\| A\pi \| &\leq \lambda \| \pi \|
\end{aligned}
$$

□

Corollary 2.2 *The second largest eigen value is bounded by* $\lambda \geq \frac{\|A\pi\|}{\|\pi\|}$, *or equivalently,*

$$\lambda = \max_{\alpha \perp 1_N} \frac{< \alpha, A\alpha >}{< \alpha, \alpha >} = \max_{\alpha \perp 1_N} \frac{\| A\alpha \|}{\| \alpha \|}$$

Property 2.4 *Random walk on expanders converges to uniform probability distribution* (u).

Proof Suppose we start with an arbitrary distribution $\pi = (\pi_0, \pi_1, ..., \pi_{n-1})$ on vertices of graph G with eigen vectors $v_0, v_1, ..., v_{n-1}$ and corresponding ordered eigen values $\mu_0 \geq \mu_1 \geq .... \geq \mu_{n-1}$. We know that $v_0 = \frac{1}{\sqrt{n}}(1, 1, ..., 1)$ and $\mu_0 = 1$, also $\pi_0 + ... + \pi_{n-1} = 1$. Let A be the normalized adjacency matrix of G. We can write,

$$\pi = c_0 v_0 + c_1 v_1 + ... + c_{n-1} v_{n-1}$$

$$c_0 =< \pi, v_0 >= \pi^T v_0 = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} \pi_i = \frac{1}{\sqrt{n}}$$

$$\therefore \quad c_0 v_0 = \frac{1}{\sqrt{n}} . \frac{1}{\sqrt{n}} (1, 1, ..., 1) = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}) = u$$

So,

$$\begin{aligned} \pi &= u + c_1 v_1 + ... + c_{n-1} v_{n-1} \\ \| A\pi - u \|^2 &= \| Au + c_1 Av_1 + ... + c_{n-1} Av_{n-1} - u \|^2 \\ &= \| c_1 Av_1 + ... + c_{n-1} Av_{n-1} \|^2 \\ &\leq \lambda^2 (c_1^2 + ... + c_{n-1}^2) = \lambda^2 \| \pi - u \|^2 \\ \| A\pi - u \| &\leq \| \pi - u \| \end{aligned}$$

After a random walk in the graph G the distribution vector moves closer to uniform distribution. □

2.3 Graph Products

For a D-regular graph G on $[N]$ vertices, where $[N] = \{1, 2...N\}$, the rotation map $Rot_G : [N] \times [D] \to [N] \times [D]$ is defined as follows:

$$Rot_G(v, i) = (w, j)$$

if the i^{th} edge leaving v leads to w, and the j^{th} edge leaving w leads to v [**?**]. Here $i, j \in [D]$.

We refer to a graph G is an (N, D, λ)-graph if it has N vertices, degree D and second largest eigenvalue of $A[G]$ is at most λ, where $A[G]$ is the normalized adjacency matrix of G.

2.3.1 Powering

Let G be a (N, D, λ) -graph given by rotation map Rot_G. The t^{th} power of G is the D^t-regular graph G^t whose rotation map is given by,

$$Rot_{G^t}(v_0, (k_1, k_2, ..., k_t)) = (v_t, (l_t, l_{t-1}, ..., l_1))$$

where $(v_i, l_i) = Rot_G(v_{i-1}, k_i)$. Then G^t will be a (N, D^t, λ^t) -graph.

2.3.2 Tensor product

Let G_1 be (N_1, D_1, λ_1) -graph and G_2 be (N_2, D_2, λ_2) -graph. Tensor product of G_1 and G_2, denoted by $G_1 \otimes G_2$, is $(N_1N_2, D_1D_2, \max\{\lambda_1, \lambda_2\})$ -graph.

$$Rot_{G_1 \otimes G_2}((v, w), (i, j)) = ((v', w'), (i', j'))$$

where $(v', i') = Rot_{G_1}(v, i)$ and $(w', j') = Rot_{G_2}(w, j)$. In tensor product eigenvalues get multiplied pairwise. So largest eigenvalue is 1.1 and second largest is $\max\{1.\lambda_2, \lambda_1.1\}$.

2.3.3 Zig-zag product

Let G be a D-regular graph on $[N]$ with rotation map Rot_G and let H be a d-regular graph on $[D]$ with rotation map Rot_H. The zig-zag product G z H is defined to be the d^2-regular graph on $[N] \times [D]$ whose rotation map Rot_{GzH} is as follows:

1. Let $(a', i') = Rot_H(a, i)$
2. Let $(w, b') = Rot_G(v, a')$
3. Let $(b, j') = Rot_H(b', j)$
4. $Rot_{GzH}((v, a), (i, j)) = ((w, b), (j', i'))$

The zig-zag product of regular graphs G and H, takes a large graph (G) and a small graph (H), and produces a graph that approximately inherits the size of the large one but the degree of the small one. An important property of the zig-zag product is that if H is a good expander, then the expansion of the resulting graph is only slightly worse than the expansion of G.

The zig-zag product replaces each vertex of G with a copy (cloud) of H, and connects the vertices by moving a small step (zig) inside a cloud, followed by a big step (zag) between two clouds, and finally performs another small step inside the destination cloud. The zig-zag product was introduced by Reingold, Vadhan and Wigderson (2002).

Theorem 2.2 (The zig-zag theorem) *Let G be a (N_1, D_1, λ_1)-expander and H be a (D_1, D_2, λ_2)-expander then G z H is a $(N_1 D_1, D_2^2, f(\lambda_1, \lambda_2))$-expander where $f(\lambda_1, \lambda_2) \leq \lambda_1 + \lambda_2 + \lambda_2^2$. [?]*

Proof It is obvious from the definition of the zig-zag product that the product graph will be D_2^2-regular with $N_1 D_1$ vertices. So enough to prove the bound for the second largest eigen value of the zig-zag product.

Consider a random walk on G z H with normalized adjacency matrix M. Let $\pi \in \mathbb{R}^{N_1 D_1}$ be any initial probability distribution on vertices of G z H, which is non-uniform.

Before proving the theorem lets define some notations. For every $v \in [N_1]$, define $\pi_v \in \mathbb{R}^{D_1}$ by $(\pi_v)_i = (\pi)_{vD_1+i}$, where $(\pi)_i$ is the i^{th} element of π. Also define a linear map $C : \mathbb{R}^{N_1 D_1} \to \mathbb{R}^{N_1}$ by $(C(\alpha))_i = \sum_{k=1}^{D_1} (\alpha_v)_k$, where $\alpha \in \mathbb{R}^{N_1 D_1}$. Thus for a probability distribution π on the vertices of G z H, (π_v) is a multiple of the conditional distribution on "cloud v" and $C(\pi)$ gives the marginal distribution on set of clouds. Also define "tensor product" denoted by $\otimes$ as,

1. For $\alpha \in \mathbb{R}^{N_1}$ and $\beta \in \mathbb{R}^{N_2}$, $\alpha \otimes \beta \in \mathbb{R}^{N_1 N_2}$ whose $(i,j)^{th}$ entry is $(\alpha)_i (\beta)_j$.

2. For $A_{N_1 \times N_1}, B_{N_2 \times N_2}$ $\quad \exists$ a unique $(A \otimes B)_{N_1 N_1 \times N_2 N_2}$:

$$(A \otimes B)(\alpha \otimes \beta) = (A\alpha) \otimes (B\beta) \qquad \forall \alpha, \beta$$

By definition, $\pi = \sum_v e_v \otimes \pi_v$ where e_v denotes the v^{th} standard basis vector in $\mathbb{R}^{N_1}$.

By basic linear algebra π_v can be decomposed like, $\pi_v = \pi_v^{\|} + \pi_v^{\perp}$ uniquely, where $\pi_v^{\|}$ is parallel to 1_{D_1} and $\pi_v^{\perp}$ is orthogonal to 1_{D_1}.

$$\begin{aligned} \pi &= \sum_v e_v \otimes \pi_v \\ &= \sum_v e_v \otimes \pi_v^{\|} + \sum_v e_v \otimes \pi_v^{\perp} \\ &= \pi^{\|} + \pi^{\perp} \end{aligned}$$

where $\pi^{\|}$ is constant over vertices within each of the N_1 clouds of H, and $\pi^{\perp}$ is orthogonal to the uniform distribution over vertices within each H cloud. In other words, $\pi^{\|}$ represents the distribution of vertices in G and $\pi^{\perp}$ that of H.

To analyze M, we relate M to the normalized adjacency matrices of G and H which we denote by A and B. First we decompose M into product of 3 matrices corresponding to the 3 steps in definition of zig-zag product.

Let $\hat{B}$ be the normalized adjacency matrix of the graph on $[N_1] \times [D_1]$ where we connect the vertices within each cloud according to the edges in H. $\hat{B}$ is related to B by,

$$\hat{B} = I_{N_1} \otimes B$$

Let $\hat{A}$ be the permutation matrix in G. ie, in the graph defined by $\hat{A}$ a vertex (a, b) will be connected only to $(a[b], b)$, where $a[b]$ means b^{th} neighbour of a and a, b are vertices in the $[N_1] \times [D_1]$. $\hat{A}$ is related to A as,

$$C[\hat{A}(e_v \otimes \frac{1_{D_1}}{D_1})] = Ae_v$$

where $v \in [N_1]$ and $e_v \in \mathbb{R}^{N_1}$.

Now, $M = \hat{B}\hat{A}\hat{B}$

$$< M\pi, \pi > = < \hat{B}\hat{A}\hat{B}\pi, \pi > = < \hat{A}\hat{B}\pi, \hat{B}\pi >$$

$\because \hat{B}$ is symmetric.

$$\begin{aligned} < M\pi, \pi > &= < \hat{A}(\hat{B}\pi), (\hat{B}\pi) > \\ &= < \hat{A}(\pi^{\|} + \hat{B}\pi^{\perp}), (\pi^{\|} + \hat{B}\pi^{\perp}) > \end{aligned}$$

Lemma 2.1 $\hat{B}\pi^{\|} = \pi^{\|}$

Proof Since $\pi^{\|}$ is uniform, it is clear that $\pi^{\|}$ is an eigen vector for $\hat{B}$, with eigen value 1. □

Lemma 2.2 $\| \hat{B}\pi^{\perp} \| \leq \lambda_2 \| \pi^{\perp} \|$

Proof

$$\begin{aligned} \hat{B}\pi^{\perp} &= \hat{B}(\sum_v e_v \otimes \pi_v^{\perp}) \\ &= \sum_v e_v \otimes (B\pi_v^{\perp}) \end{aligned}$$

From property 2.3, $\| B\pi_v^{\perp} \| \leq \lambda_2 \| \pi_v^{\perp} \|$

$$\therefore \ \| \hat{B}\pi^{\perp} \| \leq \lambda_2 \| \pi^{\perp} \|$$

□

Lemma 2.3 $| < \hat{A}\pi^{\|}, \pi^{\|} > | \leq \lambda_1 . < \pi^{\|}, \pi^{\|} >$

Proof

$$\pi = \pi^{\|} + \pi^{\perp} \implies \pi^{\|} = \pi - \pi^{\perp}$$

$\pi, \pi^{\perp} \perp u \implies \pi^{\|} \perp u$.
Since,

$$\frac{\| A\pi \|}{\| \pi \|} = \frac{| < A\pi, \pi > |}{< \pi, \pi >} \leq \lambda \qquad \forall \pi \perp u$$
$$| < \hat{A}\pi^{\|}, \pi^{\|} > | \leq \lambda_1 < \pi^{\|}, \pi^{\|} >$$

□

Substituting all the results proved before in the equation we get,

$$| < M\pi, \pi > | \leq \lambda_1 \| \pi^{\|} \|^2 + 2\lambda_2 \| \pi^{\|} \| \| \pi^{\perp} \| + \lambda_2^2 \| \pi^{\perp} \|^2$$

Put $P = \frac{\pi^{\|}}{\pi}$ and $Q = \frac{\pi^{\perp}}{\pi}$.

Then $P^2 + Q^2 = 1$, $PQ \leq \frac{1}{2}$, $P \leq 1$ and $Q \leq 1$

$$So, \qquad f(\lambda_1, \lambda_2) = \frac{| < M\pi, \pi > |}{< \pi, \pi >} \leq \lambda_1 P^2 + 2\lambda_2 PQ + \lambda_2^2 Q^2$$
$$\leq \lambda_1 + \lambda_2 + \lambda_2^2$$

□

Theorem 2.3 *There is an improved result for the bound of* $f(\lambda_1, \lambda_2)$,

$$f(\lambda_1, \lambda_2) \leq \frac{1}{2}\left\{(1 - \lambda_2^2)\lambda_1 + \sqrt{(1 - \lambda_2^2)^2\lambda_1^2 + 4\lambda_2^2}\right\}$$

Proof See [**?**]. □

The two major applications of zig-zag products are given in [**? ?**]:

Construction of constant degree expanders

From the properties of the zig-zag product, we see that the product of a large graph with a small graph, inherits a size similar to the large graph, and degree similar to the small graph, while preserving its expansion properties from both, thus enabling to increase the size of the expander without deleterious effects.

Solving the undirected $s - t$ connectivity problem in logarithmic space

In 2005, Omer Reingold introduced an algorithm that solves the undirected $s - t$ connectivity problem, the problem of testing whether there is a path between two given vertices in an undirected graph, using only logarithmic space. The algorithm relies heavily on the zig-zag product. Roughly speaking, in order to solve the undirected $s-t$ connectivity problem in logarithmic space, the input graph is transformed, using a combination of powering and the zig-zag product, into a constant-degree regular graph with a logarithmic diameter. The power product increases the expansion (hence reduces the diameter) in the price of increasing the degree, and the zig-zag product is used to reduce the degree while preserving the expansion.

Chapter 3

$s - t$ connectivity problem

3.1 Introduction

For an undirected graph G and two vertices s and t, the undirected $s - t$ connectivity ($USTCON$) problem is to decide whether there is a path between s and t. It is complete for the class of SL problems solvable by symmetric, non-deterministic log-space computations. Using the idea of expander graphs Reingold showed that $USTCON \in L$, which inplies $SL \subseteq L$ [**?**].

3.2 History

$USTCON$ can be solved in linear time with algorithms like BFS and DFS but these take linear space also. Savitch's theorem showed that the problem can be solved in $O(\log^2 n)$ space as it can be solved by a non-deterministic algorithm in $O(\log n)$ space by simply checking all possible paths starting from s. Later a randomized log-space algorithm was given by Aleliunas [**?**] who showed that a random walk from any vertex in an undirected connected graph will visit all other vertices of the graph in polynomial number of steps.

Consider a language L and a non-deterministic logarithmic space bounded turing machine which decides L as follows: If $x \in L$, atleast half of the computations returns 'YES'. If $x \notin L$, all the computations returns 'NO'. Such a machine is known as an RP machine and all languages decidable by an RP machine belongs to class RL.

Theorem 3.1 $USTCON \in RL$

Proof Let $G(V, E)$ be an undirected graph and $1, 2, ..., n$ belongs to V. In the algorithm, we do a random walk starting from $s \in V$ consisting of a polynomial number of steps in n. Here we assume each vertex has a self-loop.

Let v_t denote the node visited at time t. $v_0 = 1$ and if $v_t = i$ and $(i,j) \in E$, then $Pr[v_{t+1} = j] = \frac{1}{d_i}$, where d_i is the degree of the vertex i. Let $P_t[i] = Pr[v_t = i]$.

Lemma 3.1 *Let $G(V,E)$ be a connected graph. Then $\lim_{t\to\infty} P_t[i] = \frac{d_i}{2|E|}$ $\forall i \in V$.*

Proof At time t, $P_t[i]$ will deviate from the asymptotic value $\frac{d_i}{2|E|}$. Let $\delta_t[i] = P_t[i] - \frac{d_i}{2|E|}$ be the deviation at node i and let $\Delta_t = \sum_{i\in v} |\delta_t[i]|$ be the total absolute deviation at time t.

Each node i splits its $P_t[i]$ into d_i equal portions and passes it on to each of its neighbors. Each node i adds up the portions obtained from its neighbors and the result is $P_{t+1}[i]$.

Since $P_t[i] = \frac{d_i}{2|E|} + \delta_t[i]$, this splitting and passing can be thought of as keeping the $\frac{d_i}{2|E|}$ part, and splitting and passing only the $\delta_t[i]$'s splitting and passing the $\frac{d_i}{2|E|}$ part results in an amount equal to $\frac{1}{2|E|}$ being exchanged between any two neighbors, with canceling effects. Since the $\delta_t[i]$'s are exchanged between adjacent nodes, the sum of the absolute values cannot increase. However, it can decrease if two $\delta_t[i]$'s of opposite sign ever meet at a node.

Since at time t the total absolute deviation is Δ_t, there is a node i^+ with $\delta_t[i] > \frac{\Delta_t}{2|V|}$ and a node i^- with negative deviation $\delta_t[i] < -\frac{\Delta_t}{2|V|}$. There is a path $[i^+ = i_0, i_1, ..., i_m, ..., i_{2m} = i^-]$ with an even number of edges between i^+ and i^- (if the shortest path between i^+ and i^- has an odd number of edges, add a self-loop to the path). The positive deviation from i^+ will travel along this path for m steps, always subdivided by the degree of the current node; similarly for the negative deviation. At least a positive deviation equal to $\frac{1}{|V|^m}$ of the original amount will arrive at the middle node i_m. The same happens for a negative deviation from the opposite direction. We conclude that after $m < n$ steps, a positive deviation of at least $\frac{\Delta}{|V|^n}$ will cancel an equal amount of negative deviation, and thus in n steps the total absolute deviation has been decreased from Δ_t to at most $\Delta_t(1 - \frac{1}{|V|^n})$. Continuing like this, in the limit $\Delta_t \to 0$ and $P_t[i]$ converges to $\frac{d_i}{2|E|}$. □

From the above lemma, it is clear that the random walk returns to i in every $\frac{2|E|}{d_i}$ steps.

Suppose now that the input graph G of our randomized algorithm for $USTCON$ does have a path from s to t, say $[s = i_1, i_2, ..., i_m = t]$ (if no such path exists, the random walk can never return a false positive).

Since we start from i_1, we know that every $\frac{2|E|}{d_1}$ steps we will be returning to 1. So, after an expected number of d_1 of such returns, and therefore an expected number of $2|E|$ steps totally, the walk will head in the right direction to i_2.

Now that we are in i_2, we will be returning there on the average every $\frac{2|E|}{d_2}$ steps, and after an expected number d_2 of such returns, or $2|E|$ steps, we will arrive at i_3. And so on. It follows that after an expected number of fewer than $2n|E|$ steps we will have arrived at t. *ie*, the expected number of steps before the random walk arrives at t is at most $2n|E|$. The full randomized algorithm is this:

Run the random walk from node s for $4n|E|$ steps.

- If node t is ever visited, reply "there is a path from s to t".
- Otherwise, reply "there is probably no path from s to t".

Obviously there are no false positives, and the probability of a false negative is at most $\frac{1}{2}$ from Markov's inequality (because we run the algorithm for twice the expected time of convergence). Finally, it is obvious each computation of the algorithm can be implemented in logarithmic space. □

Hence we get

$$L \subseteq SL \subseteq RL \subseteq NL \subseteq L^2$$

Nisan et al [**?**] later showed that $USTCON \in L^{3/2}$. This bound was further improved by Armoni [**?**] et al who managed to prove that $USTCON \in L^{4/3}$.

3.3 The approach

The essence of the algorithm is to improve the connectivity of the graph and then check for $s-t$ connectivity. This can be done by transforming the graph into a constant degree expander. A good method to do so is powering. But it has a disadvantage - it increases the degree of the graph and the final degree need not be a constant. To solve this problem, we use zig-zag product. The algorithm can be described in a nutshell in the following steps. First convert input graph into a constant degree non-bipartite regular graph. Then each connected component of the graph is converted into a constant degree expander in logarithmic number of phases. The $USTCON$ problem is then solved in the resulting graph.

3.4 Some results

Alon [**?**] proved that for every (N, D, λ) -graph with $\lambda < 1$, $\exists \epsilon > 0$ such that any vertex subset S of N with $|S| \leq |\frac{N}{2}|$ has at least $(1+\epsilon)|S|$ neighbors. The following proposition comes as a result:

Lemma 3.2 *Let $\lambda < 1$ be some constant. Then for every (N, D, λ) -graph G and any two vertices s and t in G, there exists a path of length $O(\log N)$ that connects s to t i.e. G has logarithmic diameter.*

Proof By the vertex expansion of G, for some $l = O(\log N)$, both s and t have more than $\frac{N}{2}$ vertices of distance at most l from them in G. Therefore, there exists a vertex v that is of distance at most l from both s and t. We can therefore conclude that $s - t$ connectivity in constant-degree expanders can be solved in log-space. □

Lemma 3.3 *Let $\lambda < 1$ be some constant. Then there exists a $O(\log D. \log N)$ space algorithm A_{exp} such that when a D-regular undirected graph G on N vertices is given to A_{exp} as input, the following hold:*

1. *If s and t are in the same connected component and this component is an (N, D, λ)-graph, then A_{exp} outputs connected.*
2. *If A_{exp} outputs connected, then s and t are indeed in the same connected component.*

Proof The algorithm A_{exp} simply enumerates all D^l paths of length $l = O(\log N)$ from s. The algorithm A_{exp} outputs connected $\iff$ at least one of these paths encounters t.Following any particular path from s of length l requires space $O(\log N)$ (when given as input the sequence of l edge labels in $[D] = 1, 2, ..., D$ traversed by this path). Enumerating all these D^l paths requires space $O(\log D.logN)$. By Lemma 3.2, in case (1), s and t are of distance at most l of each other and A_{exp} will indeed find a path from s to t and will output connected. On the other hand, A_{exp} never outputs connected unless it finds a path from s to t, implying (2).

We need a constant sized expander to be used in the algorithm and this can be obtained by an exhaustive search. □

Lemma 3.4 *There exists some constant D_e and a $((D_e)^{16}, D_e, \frac{1}{2})$ -graph.*

Proof See [**?**]. □

The following lemma says that spectral gap of the graph G decreases with increase in N and D.

Lemma 3.5 *For every D-regular, connected, non-bipartite graph G on $[N]$, we have $\lambda(G) \leq 1 - \frac{1}{DN^2}$.*

Proof See [**?**]. □

The next lemma uses the bound for the eigen value obtained from the zig-zag product.

Lemma 3.6 *If G is an (N, D, λ) -graph and H is a (D, d, α) -graph, then $1 - \lambda(GzH) \geq \frac{1}{2}(1 - \alpha^2).(1 - \lambda)$.*

Proof We know that $f(\lambda,\alpha) \leq \frac{1}{2}\left\{(1-\alpha^2)\lambda_1 + \sqrt{(1-\alpha^2)^2\lambda_1^2 + 4\alpha^2}\right\}$. Since $\lambda \leq 1$, we have

$$\begin{aligned}
\frac{1}{2}\sqrt{(1-\alpha^2)^2\lambda^2 + 4\alpha^2} &\leq \frac{1}{2}\sqrt{(1-\alpha^2)^2 + 4\alpha^2} \\
&= \frac{1}{2}\sqrt{1+\alpha^4-2\alpha^2+4\alpha^2} = \frac{1}{2}\sqrt{1+\alpha^4+2\alpha^2} \\
&= \frac{1}{2}(1+\alpha^2) = 1 - \frac{1}{2}(1-\alpha^2) \\
\therefore f(\lambda,\alpha) &\leq 1 - \frac{1}{2}(1-\alpha^2).(1-\lambda)
\end{aligned}$$

□

3.5 The main transformation

Now we turn to the main transformation of the input graph which converts each of its connected components into expanders.

Definition On input G and H, where G is a D^{16}-regular graph on $[N]$ and H is a D-regular graph on $[D^{16}]$, both given by their rotation maps, the transformation τ outputs the rotation map of a graph G_l defined as follows:

1. Set $l = 2\lceil logDN^2 \rceil$
2. Set G_0 to equal G, and for $i > 0$ define G_i recursively by the rule:

$$G_i = (G_{i-1} z H)^8$$

Denote by $\tau_i(G,H)$ the graph G_i, and $\tau_i(G,H) = G_i$

From the above transformation we get that G_i is a D^{16} regular graph on $[N] \times [D^{16}]^i$. From the definition, it is also obvious that if D is kept as a constant, $l = O(\log N)$ and G_l has polynomial number of vertices with respect to N.

Theorem 3.2 *Let G and H be the inputs of T as given above. If $\lambda(H) \leq \frac{1}{2}$ and G is connected and non-bipartite, then $\lambda(\tau(G,H)) \leq \frac{1}{2}$.*

Proof We have $\lambda(G_0) = \lambda(G) \leq 1 - \frac{1}{DN^2}$ as G is connected and non-bipartite(from Lemma 3.5). Since $l = 2\lceil logDN^2 \rceil$, we can show that $(1 - \frac{1}{DN^2})^{2^l} \leq \frac{1}{2}$. Let $\lambda = \lambda(G_{i-1})$. Then $\lambda(G_{i-1}zH) \leq 1 - \frac{1}{3}(1-\lambda)$ (from *Lemma* 5) and $\lambda(G_i) \leq (1 - \frac{1}{3}(1-\lambda))^8$. Here if $\lambda < \frac{1}{2}$, $\lambda(G_i) \leq \frac{1}{2}$. Else, we have $(1 - \frac{1}{3}(1-\lambda))^4 \leq \lambda$ and therefore $\lambda(G_i) < \lambda^2$. So we have $\lambda(G_i) \leq \max\{\lambda(G_{i-1})^2, \frac{1}{2}\}$. Hence we get the result $\lambda(\tau(G,H)) \leq \frac{1}{2}$.

From the above definition of the main transformation, its obvious that each connected component of the input graph is transformed into an expander as zig-zag product and powering works on each connected component separately. Now we prove that the above main transformation can be done in log-space. □

Theorem 3.3 *For every constant D the transformation τ can be computed in space $O(\log N)$ on inputs G and H, where G is a D^{16}-regular graph on $[N]$ and H is a D-regular graph on $[D^{16}]$.*

Proof We define an algorithm A_τ that on inputs G and H computes the rotation map Rot_{G_l} of $G_l = \tau(G, H)$. Given G and H (written on the read only input tape), it enumerates all values $(\overline{v}, \overline{a})$ in the domain of Rot_{G_l} and outputs $((\overline{v}, \overline{a}), Rot_{G_l}(\overline{v}, \overline{a}))$. $\overline{v}$ is a vertex of G_l and $\overline{a}$ is an edge label in that graph. Hence $\overline{v} \in [N] \times ([D^{16}])^l$ and $\overline{a} \in [D^{16}]$.

Since $l = O(\log N)$ and D is a constant, the length of each value $(\overline{v}, \overline{a})$ is $O(\log N)$ and therefore enumerating all of these values can be done in space $O(\log N)$.

v will take values in $[N]$ as it represents a vertex of G , and $l+1$ variables $a_0, a_1 ... a_l$ each takes value in $[D^{16}]$ (and each specifying a vertex name of H; also, a_0 may specify an edge label of G). Since $\overline{a_i} \in [D^{16}]$, we denote $a_i = k_{i,1} ... k_{i,16}$. $\overline{v}$ is copied into $v, a_0, a_1 ... a_{l-1}$ and $\overline{a}$ into a_l. $(\overline{v}, \overline{a})$ will contain the output after the algorithm is done.

A_τ is defined recursively. At each level of the recursion, A_τ will evaluate Rot_{G_i} for some i on the appropriate prefix $v, a_0, ..., a_i$. For the base case $i = 0$, $Rot_{G_0} = Rot_G$ is written on the input tape, and can therefore be evaluated in space $O(\log N)$ by simply searching the input tape. For larger i , the evaluation of Rot_{G_i} is as follows:

For $j = 1$ to 16

- Set $(a_{i-1}, k_{i,j}) \leftarrow Rot_H(a_{i-1}, k_{i,j})$.
- If j is odd, recursively set $(v, a_0, ..., a_{i-1}) \leftarrow Rot_{G_i}((v, a_0, ..., a_{i-2}), a_{i-1})$.
- If $j = 16$, reverse the order of the individual labels in a_i : Set $k_{i,1} ... k_{i,16} \leftarrow k_{i,16} ... k_{i,1}$.

Each node of the recursion tree performs a constant number of operations and recursive calls and the depth of the recursion is $l+1 = O(\log N)$.Therefore, maintaining the recursion can be done in space $O(\log N)$. Furthermore, each one of the basic operations (evaluating Rot_G, evaluating Rot_H, and reversing the order of labels in the last step) can be performed in space O(log N). Finally, the only memory that needs to be kept after a basic operation is performed, is the memory holding the variables $v, a_0, ..., a_l$ (that are shared by all of these operations), and the memory for maintaining the recursion.

This can also be done space $O(\log N)$. We therefore conclude that the space complexity of A_τ is $O(\log N)$ which completes the proof. $\square$

3.5.1 The algorithm

Theorem 3.4 $USTCON \in L$

Proof We give an algorithm A_{con} that gets as input a graph G over the set of vertices $[N]$, and two vertices s and t in $[N]$. We assume that the graph is given via the adjacency matrix representation. A_{con} will answer connected $\iff$ there exists a path in G between s and t (i.e., s and t are in the same connected component). Furthermore, G will use space which is logarithmic in its input size.

The algorithm A_{con} will need to evaluate the rotation map of a $(D_e^{16}, D_e, \frac{1}{2})$ -graph H, where D_e is some constant. There exists such a graph and therefore A_{con} can obtain it by exhaustive search using constant amount of memory.

Let τ be the main transformation. However, we will first need to preprocess G in order to get a new graph G_{reg} such that (G_{reg}, H) is a correct input to T. In particular, we need G_{reg} to be a D_e^{16} -regular graph given by its rotation map. There are various ways of transforming G to G_{reg}. We replace every vertex of G with a cycle of lenfgth N and each of the vertices (v, w), where there is an edge between v and w in G, is also connected to (w, v) (the rest of the edges are self loops). The rotation map $Rot_{G_{reg}} : ([N] \times [N]) \times [D_e^{16}] \to ([N] \times [N]) \times [D_e^{16}]$ of G_{reg} is formally defined as follows:

1. $Rot_G((v, w), 1) = ((v, w), 2)$, where
$$w = \begin{cases} w + 1 & w < N \\ 1 & otherwise \end{cases}$$

2. $Rot_G((v, w), 2) = ((v, w), 1)$, where
$$w = \begin{cases} w - 1 & w < N \\ N & otherwise \end{cases}$$

3.
$$Rot_G((v, w), 3) = \begin{cases} ((w, v), 3) & if \ \ (v, w) \in E(G) \\ ((v, w), 3) & otherwise \end{cases}$$

4. For $i > 3$, $Rot_G((v, w), i) = ((w, v), i)$

The transformation from G to G_{reg} is clearly computable in logarithmic space. Furthermore, G_{reg} is D_e^{16} regular by definition and all its connected

components are non-bipartite (as every vertex in G_{reg} has self loops). Finally, for every connected component $S \subseteq [N]$ of G we have that $S \times [N]$ is a connected component in G_{reg}. Now define $G_{reg} = \tau(G_{reg}, H)$, and $l = O(\log N)$. Let S be the connected component of G, such that $s \in S$. $S \times [N]$ is a connected component of G_{reg}, and $G_{reg}\,|_{S\times[N]}$ is non-bipartite which is clear from its definition.

Let A_{exp} be the algorithm which decides undirected $s - t$ connectivity correctly in graphs where the connected component of the starting vertex is an expander. The algorithm A_{con} will now invoke A_{exp}, on the graph G_{exp} and the vertices $s' = (s, 1^{l+1})$ and $t' = (t, 1^{l+1})$. If A_{exp} outputs that s' and t' are connected in G_{exp}, then A_{con} will output that s and t are connected in G. Otherwise A_{con} will output that s and t are not connected. The algorithm A_{con} is log-space since it is composed of a constant number of log-space procedures:

1. The transformation from G to G_{reg}.
2. The transformation from G_{reg} to G_{exp}, which is computable by a log-space algorithm A_τ by Theorem 3.3.
3. The algorithm A_{exp} which is log-space by Lemma 3.3.

□

Each of the above operations can be done in log-space and hence by the transitivity of log-space transformations we can conclude that the entire algorithm can be completed in log-space.

Chapter 4

Conclusion

Expander graph is a very important class while considering graph theory. They have applications in various fields like communication, randomized algorithms, Super-concentrators etc. Hence an understanding in this area can be utilized to good effect in many practical applications. Zig-zag product is useful because it gives a very useful method in constructing expanders from smaller graphs. It has already been used by Reingold to prove a very important result that the $s-t$ connectivity problem in an undirected graph can be solved in log space. In complexity theory, this problem used to be complete for the class SL(symmetric log space) [**?**] and by proving it belongs to class L, it was obtained that $SL = L$.