Lecture 1: Introduction

In which we describe what this course is about.

1 Overview

This is class is about applications of linear algebra to graph theory and to graph algorithms. In the finite-dimensional case, linear algebra deals with vectors and matrices, and with a number of useful concepts and algorithms, such as determinants, eigenvalues, eigenvectors, and solutions to systems of linear equations.

The application to graph theory and graph algorithms comes from associating, in a natural way, a matrix to a graph $G = (V, E)$, and then interpreting the above concepts and algorithms in graph-theoretic language. The most natural representation of a graph as a matrix is via the $|V| \times |V|$ *adjacency matrix* of a graph, and certain related matrices, such as the *Laplacian* and *normalized Laplacian* matrix will be our main focus. We can think of $|V|$-dimensional Boolean vectors as a representing a partition of the vertices, that is, a *cut* in the graph, and we can think of arbitrary vectors as *fractional* cuts. From this point of view, eigenvalues are the optima of continuous relaxations of certain cut problems, the corresponding eigenvectors are optimal solutions, and connections between spectrum and cut structures are given by rounding algorithms converting fractional solutions into integral ones. Flow problems are dual to cut problems, so one would expect linear algebraic techniques to be helpful to find flows in networks: this is the case, via the theory of electrical flows, which can be found as solutions to linear systems.

The course can be roughly subdivided into three parts: in the first part of the course we will study *spectral graph algorithms*, that is, graph algorithms that make use of eigenvalues and eigenvectors of the normalized Laplacian of the given graph. In the second part of the course we will look at constructions of expander graphs, and their applications. In the third part of the course, we will look at fast algorithms for solving systems of linear equations of the form $L\mathbf{x} = \mathbf{b}$, where L is Laplacian of a graph, their applications to finding electrical flows, and the applications of electrical flows to solving the max flow problem.

2 Spectral Graph Algorithms

We will study approximation algorithms for the *sparsest cut* problem, in which one wants to find a cut (a partition into two sets) of the vertex set of a given graph so that a minimal number of edges cross the cut compared to the number of pairs of vertices that are disconnected by the removal of such edges.

This problem is related to estimating the edge expansion of a graph and to find *balanced separators*, that is, ways to disconnect a constant fraction of the pairs of vertices in a graph after removing a minimal number of edges.

Finding balanced separators and sparse cuts arises in *clustering* problems, in which the presence of an edge denotes a relation of similarity, and one wants to partition vertices into few clusters so that, for the most part, vertices in the same cluster are similar and vertices in different clusters are not. For example, sparse cut approximation algorithms are used for *image segmentation*, by reducing the image segmentation problem to a graph clustering problem in which the vertices are the pixels of the image and the (weights of the) edges represent similarities between nearby pixels.

Balanced separators are also useful in the design of divide-and-conquer algorithms for graph problems, in which one finds a small set of edges that disconnects the graph, recursively solves the problem on the connected components, and then patches the partial solutions and the edges of the cut, via either exact methods (usually dynamic programming) or approximate heuristic. The sparsity of the cut determines the running time of the exact algorithms and the quality of approximation of the heuristic ones.

We will study a spectral algorithm first proposed by Fiedler in the 1970s, and to put its analysis into a broader context, we will also study the Leighton-Rao algorithm, which is based on linear programming, and the Arora-Rao-Vazirani algorithm, which is based on semidefinite programming. We will see how the three algorithms are based on conceptually similar continuous relaxations.

Before giving the definition of sparsest cut, it is helpful to consider examples of graphs that have very sparse cuts, in order to gain intuition.

Suppose that a communication network is shaped as a path, with the vertices representing the communicating devices and the edges representing the available links. The clearly undesirable feature of such a configuration is that the failure of a single edge can cause the network to be disconnected, and, in particular, the failure of the middle edge will disconnect half of the vertices from the other half.

This is a situation that can occur in reality. Most of Italian highway traffic is along the highway that connect Milan to Naples via Bologna, Florence and Rome. The section between Bologna and Florence goes through relatively high mountain passes, and snow and ice can cause road closures. When this happens, it is almost impossible

to drive between Northern and Southern Italy. Closer to California, I was once driving from Banff, a mountain resort town in Alberta which hosts a mathematical institute, back to the US. Suddenly, traffic on Canada's highway 1 came to a stop. People from the other cars, after a while, got out of the cars and started hanging out and chatting on the side of the road. We asked if there was any other way to go in case whatever accident was ahead of us would cause a long road closure. They said no, this is the only highway here. Thankfully we started moving again in half an hour or so.

Now, consider a two-dimensional $\sqrt{n} \times \sqrt{n}$ grid. The removal of an edge cannot disconnect the graph, and the removal of a constant number of edges can only disconnected a constant number of vertices from the rest of the graph, but it is possible to remove just $\sqrt{n}$ edges, a $1/O(\sqrt{n})$ fraction of the total, and have half of the vertices be disconnected from the other half.

A k-dimensional hypercube with $n = 2^k$ is considerably better connected than a grid, although it is still possible to remove a vanishingly small fraction of edges (the edges of a dimension cut, which are a $1/k = 1/\log_2 n$ fraction of the total number of edges) and disconnect half of the vertices from the other half.

Clearly, the most reliable network layout is the clique; in a clique, if an adversary wants to disconnect a p fraction of vertices from the rest of the graph, he has to remove at least a $p \cdot (1-p)$ fraction of edges from the graph.

This property of the clique will be our "gold standard" for reliability. The expansion and the sparsest cut parameters of a graph measure how worse a graph is compared with a clique from this point of view.

For simplicity, here we will give definitions that apply only to the case of regular graphs.

Definition 1 (Edge expansion of a set) *Let $G = (V, E)$ be a d-regular graph, and $S \subseteq V$ a subset of vertices. The edge expansion of S is*

$$\phi(S) := \frac{E(S, V-S)}{d|S|}$$

where $E(S, V-S)$ is the number of edges in E that have one endpoint in S and one endpoint in $V - S$.

$d|S|$ is a trivial upper bound to the number of edges that can leave S, and so $\phi(S)$ measures how much smaller the actual number of edges is than this upper bound. We can also think of $\phi(S)$ as the probability that, if we pick a random node v in S and then a random neighbor w of v, the node w happens to be outside of S.

The quantity $1 - \phi(S)$ is the average fraction of neighbors that vertices in S have within S. For example, if G represents a social network, and S is a subset of users

of expansion .3, this means that, on average, the users in S have 70% of their friends within S.

If $(S, V-S)$ is a cut of the graph, and $|S| \leq |V-S|$, then $\phi(S)$ is, within a factor of two, the ratio between the fraction $E(S, V-S)/|E| = 2E(S, V-S)/dn$ of edges that we have to remove to disconnect S from $V-S$, and the fraction $|S| \cdot |V-S| / \binom{n}{2}$ of pairs of vertices that become unreachable if we do so. We define the edge expansion of a cut as

$$\phi(S, V-S) := \max\{\phi(S), \phi(V-S)\}$$

The edge expansion of a graph is the minimum of the edge expansion of all cuts.

Definition 2 (Edge expansion of a graph) *Let $G = (V, E)$ be a d-regular graph, its edge expansion is*

$$\phi(G) := \min_{S: 0<|S|<|V|} \phi(S, V-S) = \min_{S: 0<|S|\leq \frac{|V|}{2}} \phi(S)$$

If A is the adjacency matrix of a d-regular graph $G = (V, E)$, then the *normalized Laplacian* of G is the matrix $L := I - \frac{1}{d}A$. We will prove the Cheeger inequalities: that if $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of L, counted with multiplicities and sorted in nondecreasing order, then

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}$$

The lower bound $\phi(G) \geq \frac{\lambda_2}{2}$ follows by using the variational characterization of eigenvalues to think of λ_2 as the optimum of a continuous optimization problem, and then realizing that, from this point of view, λ_2 is actually the optimum of a *relaxation* of $\phi(G)$.

The upper bound $\phi(G) \leq \sqrt{2\lambda_2}$ has a constructive proof, showing that the set S_F returned by Fiedler's algorithm has size $\leq |V|/2$ and satisfies $\phi(S_F) \leq 2\sqrt{\lambda_2}$. The two inequalities, combined, show that $\phi(S_F) \leq 2\sqrt{\phi(G)}$ and provide a (tight) worst-case analysis of the quality of the cut found by Fiedler-s algorithm, compared with the optimal cut.

To put this result in a broader context, we will see the Leighton-Rao approximation algorithm, based on linear programming, which finds a cut of expansion $\leq \phi(G) \cdot O(\log |V|)$, and the Arora-Rao-Vazirani algorithm, based on semidefinite programming, which finds a cut of expansion $\leq \phi(G) \cdot O(\sqrt{\log |V|})$. The spectral, linear programming, and semidefinite programming relaxation can all be seen as very related.

We will then consider combinatorial characterizations, and algorithms for other laplacian eigenvalues.

We will prove a "higher order" Cheeger inequality that characterizes λ_k for $k \geq 2$ similarly to how the standard Cheeger inequality characterizes λ_2, and the proof will provide a worst-case analysis of spectral partitioning algorithms similarly to how the proof of the standard Cheeger inequality provides a worst-case analysis of Fiedler's algorithm.

The outcome of these results is that small Laplacian eigenvalues characterize the presence of sparse cuts in the graph. Analogously, we will show that the value of λ_n characterizes large cuts, and the proof a Cheeger-type inequality for λ_n will lead to the worst-case analysis of a spectral algorithm for max cut.

3 Constructions and Applications of Expander Graphs

A family of constant-degree expanders is a collection of arbitrarily large graphs, all of degree $O(1)$ and edge expansion $\Omega(1)$. Expanders are useful in several applications, and a common theme in such applications is that even though they are sparse, they have some of the "connectivity" properties of a complete graph.

For example, if one removes a $o(1)$ fraction of edges from an expander, one is left with a connected component that contains a $1 - o(1)$ fraction of vertices.

Lemma 3 *Let $G = (V, E)$ be a regular graph of expansion ϕ. Then, after an $\epsilon < \phi$ fraction of the edges are adversarially removed, the graph has a connected component that spans at least $1 - \epsilon/2\phi$ fraction of the vertices.*

Proof: Let d be the degree of G, and let $E' \subseteq E$ be an arbitrary subset of $\leq \epsilon|E| = \epsilon \cdot d \cdot |V|/2$ edges. Let $C_1, \ldots, C_m$ be the connected components of the graph $(V, E - E')$, ordered so that $|C_1| \geq |C_2| \geq \cdots \geq |C_m|$. We want to prove that $|C_1| \geq |V| \cdot (1 - 2\epsilon/\phi)$. We have

$$|E'| \geq \frac{1}{2} \sum_{i \neq j} E(C_i, C_j) = \frac{1}{2} \sum_i E(C_i, V - C_i)$$

If $|C_1| \leq |V|/2$, then we have

$$|E'| \geq \frac{1}{2} \sum_i d \cdot \phi \cdot |C_i| = \frac{1}{2} \cdot d \cdot \phi \cdot |V|$$

but this is impossible if $\phi > \epsilon$.

If $|C_1| \geq |V|/2$, then define $S := C_2 \cup \cdots \cup C_m$. We have

$$|E'| \geq E(C_1, S) \geq d \cdot \phi \cdot |S|$$

which implies that $|S| \leq \frac{\epsilon}{2\phi} \cdot |V|$ and so $C_1 \geq \left(1 - \frac{\epsilon}{2\phi}\right) \cdot |V|$. $\square$

In a d-regular expander, the removal of k edges can cause at most $O(k/d)$ vertices to be disconnected from the remaining "giant component." Clearly, it is always possible to disconnect k/d vertices after removing k edges, so the reliability of an expander is essentially best possible.

Another way in which expander graphs act similarly to a complete graph is the following. Suppose that, given a graph $G = (V, E)$, we generate a sequence $v_1, \ldots, v_k$ by choosing $v_1 \in V$ uniformly at random and then performing a $(k-1)$-step random walk. If G is a complete graph (in which every vertex has a self-loop), this process uses $k \log |V|$ random bits and generates k uniform and independent random vertices. In an expander of constant degree, the process uses only $\log |V| + O(k)$ random bits, and the resulting sequence has several of the useful statistical properties of a sequence generated uniformly at random. Especially in the case in which k is of the order of $\log |V|$, using $O(\log |V|)$ instead of $O(\log^2 |V|)$ random bits can be a significant saving in certain application. (Note, in particular, that the sample space has polynomial size instead of quasi-polynomial size.)

Constructions of constant-degree expanders are useful in a variety of applications, from the design of data structures, to the derandomization of algorithms, from efficient cryptographic constructions to being building blocks of more complex quasirandom objects.

There are two families of approaches to the explicit (efficient) construction of bounded-degree expanders. One is via algebraic constructions, typically ones in which the expander is constructed as a Cayley graph of a finite group. Usually these constructions are easy to describe but rather difficult to analyze. The study of such expanders, and of the related group properties, has become a very active research program. There are also combinatorial constructions, which are somewhat more complicated to describe but considerably simpler to analyze.

4 Mixing time of random walks

If one takes a random walk in a regular graph that is connected and not bipartite, then, regardless of the starting vertex, the distribution of the t-th step of the walk is close to the uniform distribution over the vertices, provided that t is large enough. It is always sufficient for t to be quadratic in the number of vertices; in some graphs, however, the distribution is near-uniform even when t is just poly-logarithmic, and, indeed, the time is at most $O\left(\frac{1}{\lambda_2} \log |V|\right)$, and thus it is at most logarithmic in expander graphs.

Among other applications, the study of the "mixing time" (the time that it takes to

reach the uniform distribution) of random walks has applications to analyzing the convergence time of certain randomized algorithms.

The design of approximation algorithms for *combinatorial counting* problems, in which one wants to count the number of solutions to a given NP-type problem, can be reduced to the design of *approximately uniform sampling* in which one wants to approximately sample from the set of such solutions. For example, the task of approximately counting the number of perfect matchings can be reduced to the task of sampling almost uniformly from the set of perfect matchings of a given graph. One can design approximate sampling algorithms by starting from an arbitrary solution and then making a series of random local changes. The behavior of the algorithm then corresponds to performing a random walk in the graph that has a vertex for every possible solution and an edge for each local change that the algorithm can choose to make. Although the graph can have an exponential number of vertices in the size of the problem that we want to solve, it is possible for the approximate sampling algorithm to run in polynomial time, provided that a random walk in the graph converges to uniform in time poly-logarithmic in its size.

The study of the mixing time of random walks in graphs is thus a main analysis tool to bound the running time of approximate sampling algorithms (and, via reductions, of approximate counting algorithms).

As a way of showing applications of results proved so far, we will show that, because of Cheeger's inequality, the mixing time is upper-bounded by $O\left(\frac{1}{\phi^2}\log|V|\right)$, and then we will use the dual of the Leighton-Rao relaxation to show that $1/\phi$ can be upper-bounded by the congestion of a certain flow problem. We will apply this theory to the analysis of an algorithm that approximates the number of perfect matchings in a given dense bipartite graph.

5 Linear Systems, Electrical Flows, and Applications

In the last part of the course, we will turn to connections between graph theory and a different aspect of linear algebra, namely the solution of systems of linear equations. If we have a system of linear equations of the form

$$A\mathbf{x} = \mathbf{b}$$

we can solve it (or determine that it has no solution) in polynomial time using Gaussian elimination. Sometimes, it is possible to develop faster and more numerically stable algorithms by thinking of the problem has an *optimization*, such as, for example,

$$\min_{\mathbf{x}} ||A\mathbf{x} - \mathbf{b}||$$

for an appropriate choice of norm.

If A is positive definite (that is, all the eigenvalues are strictly positive), then another way of turning a linear system into an optimization problem is to consider the problem

$$\min \ \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{1}$$

The problem is strictly convex, because the Hessian of the function $f(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}$, that is, the matrix of partial second derivatives of $f(\cdot)$, is, at every point, the matrix A itself, which we assumed to be positive definite.

The strongly convex optimization problem (1) has a unique minimum, achieved at a point $\mathbf{x}^*$. The gradient of $f(\cdot)$ at a point $\mathbf{x}$, that is, the vector of partial derivates at $\mathbf{x}$, is $\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$. The gradient has to be equal to the $\mathbf{0}$ vector at the optimum $\mathbf{x}^*$, and so we have $A\mathbf{x}^* = \mathbf{b}$.

If we want to solve the linear system $A\mathbf{x} = \mathbf{b}$, and A is positive definite, then a possible strategy is to solve the convex optimization problem (1) using gradient descent, or similar local-search algorithms for convex optimization. The running time of such algorithms will be determined by the smallest eigenvalue A. In order to deal with matrix having small eigenvalues, one resorts to *preconditioning*, which is a technique that reduces the $A\mathbf{x} = \mathbf{b}$ system to a $B\mathbf{y} = \mathbf{b}'$ system in which B has a larger smallest eigenvalue. In the interesting special case in which A is the Laplacian matrix of an undirected graph, the running time is determined by the expansion of the graph, and preconditioning can be understood in graph-theoretic terms.

(Technically, the Laplacian is not positive definite. What we mean above is that we are interested in solving an equation of the form $L\mathbf{x} = \mathbf{b}$ where L is a Laplacian matrix, and $\mathbf{x}$ is further constrained to be orthogonal to the eigenspace of zero.)

Efficiently solving "Laplacian systems" of the form $L\mathbf{x} = \mathbf{b}$ is closely related to the problem of finding *sparsifiers* of graphs, and we will see nearly linear time algorithms for both problems.

One application of finding solutions to systems of the form $L\mathbf{x} = \mathbf{b}$ is to find *electrical flows* in networks. We will then see how to use fast algorithms for finding electrical flows and turn them into algorithm for the Max Flow problem.