University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 1: Introduction to expanders

October 5, 2021

Instructor: Thatchaphol Saranurak

Scribe: Aditya Anand

1 Expanders: an introduction

Expanders are central objects in TCS and Math with a lot of applications. In this course, we will see how expanders are so powerful in the area of efficient graph algorithms. Expanders find a lot of applications in graph algorithms - many problems can be solved faster on expanders. We will see *expander decomposition* - the result that allows us to think of any graph as a collection of expanders connected by a small fraction of edges. This will allow us to transfer our study of expanders to general graphs - this nice structure can often be helpful in designing algorithms.

There are other areas where expanders appear a lot: coding theory, pseudo-randomness, PCP (probabilistic checkable proofs). Before we look at a formal definition of expanders, let us try to understand what an expander is "morally" - what are the properties, and what do expander graphs look like.

2 Expanders - sparse but well-connected!

Very simply put, an expander is a **sparse complete graph**! This sounds contradictory - but let us try to understand the context of this statement. What we mean to say is that expanders behave like complete graphs with respect to many properties, but they can be really sparse. Here are some views of expanders: each view captures "connectivity" in its own sense.

- **Cut view**: Robust against deletions
- **Flow view**: Allow all-to-all routing
- **Probabilistic view**: Random walks mix rapidly
- **Algebraic view**: Large spectral gap

Each of the above is useful, but it turns out that they are equivalent! They all characterize the same thing, expanders. In the first half of the course, we will study each of these views and understand how they are all equivalent in charactetrizing expanders. Along the way we will learn a lot of concepts like the flow-cut gap, metric embeddings, Laplacian of graphs, spectral gaps, random

Table 1: Characteristics of a *good* algorithm

Correct					
Fast					
Deterministic					
Parallel					
Distributed					
Dynamic					
Low space					
Simple to implement					
Easy to understand					
What else ???					

walks, etc. Some of the algorithmic applications which we will look at along the way include Multi-commodity flow, multicut, sparsest cut, vertex and edge connectivity. In the second half of the course, we will try to develop *good* algorithms based on expanders. For instance, we will look at graph sparsification from cut, flow and spectral perspectives. We will develop fast near-linear time algorithms for solving many graph problems based on our study of expanders using the key tool of *expander decomposition*, which says that every graph looks like a collection of expanders, with a small fraction of edges between the expanders.

What is a *good* algorithm though? What is the goal of an algorithm designer? Table 1 tries to list some of the properties that a *good* algorithm might hope to achieve. Discuss graph algorithms what you have learned before and try to rate them in terms of the parameters in Table 1.

We will look at some state of the art algorithms for solving fundamental graph problems in the second half of the course. The tentative goal in the second half of the course will include:

- $(1+\epsilon)$-approximate max flow in **undirected** graphs in near-linear time. [All exact max flow algorithms are still much slower.]
- $(1+\epsilon)$-approximate shortest path in **undirected** graphs in near-linear work, polylog depth. [Dijkstra's is not parallel]
- other things along the way... sparsifiers, oblivious routing, dynamic algorithms, etc.

3 Background

I will assume some background knowledge. You can always ask question but I might sometimes tell you where you can look up about these topics instead.

- Basic algorithms and data structures (sorting, binary search trees, graph searching, shortest path, max flow, etc).
- Basic graph theory

- Basic probability and concentration bounds (Markov, Chernoff)
- Linear programming and duality

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 2: High Conductance = Robust Against Deletions

September 2, 2021

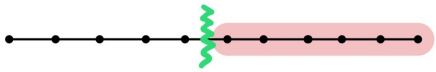
Instructor: Thatchaphol Saranurak

Scribe: Jingyi Gao

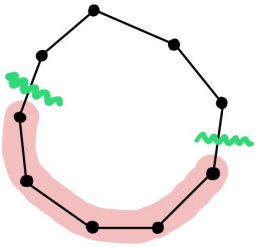
1 Motivate the Definition: Robust Against Deletions

Given a connected graph G, we could describe G is well-connected in the following sense: when we delete d edges in G, the size of the largest component can be reduced by at most $O(d)$ edges. In other words, delete a small set of edges only disconnect small parts of graph.
More formally, let $D \subseteq E$ be a set of d edges. Let $G' = G \setminus D$. Let C be the union of all small components in G'. Then, C contains at most $O(d)$ edges.
Some non-examples are as follows:

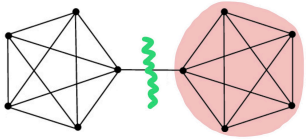
1. paths, where we can delete one edge and disconnect a large chain of vertices,



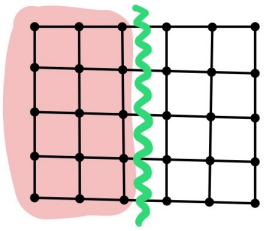
2. cycles, which is similar to a path but we can delete two edges and disconnect a large portion of the cycle,



3. dumbbells, where we can delete one edge and disconnect a large portion of the graph,

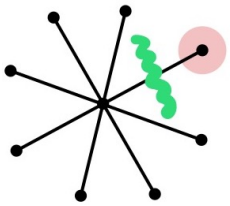


4. grids, where we delete $O(\sqrt{n})$ edges and disconnect $O(n)$ vertices.

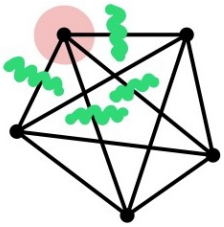


On the other hand, some examples are as follows:

1. stars, where deleting one edge always corresponds to disconnect one vertex from the graph,



2. complete graph, where disconnecting a set A of vertices requires deleting all edges that are connected to vertices outside of A.



An **expander** should satisfies the property above, namely be robust against edge deletions. This motivates the definition of conductance.

2 Formal definition of conductance

We first formalize some notations and the definition of volume

Notation 2.1 (Edges between A and B)**.** Let $G = (V, E)$ be a undirected unweighted graph. $\forall A, B \subseteq V$, $E_G(A, B) = \{(u, v) \in E \mid u \in A, b \in B\}$.

Notation 2.2 (Cut edges)**.** Let $G = (V, E)$ be a undirected unweighted graph. $\forall S \subseteq V$, denote the set of cut edges as $\partial_G S = E_G(S, V \setminus S)$.

Definition 2.3 (Volume)**.** Let $G = (V, E)$ be a undirected unweighted graph. For any $S \subseteq V$, the volume of S is $\mathrm{vol}_G(S) = \sum_{u \in S} \deg(u)$.

Observation 2.4. *Given the definition of volume, some observations are listed below:*

1. $|E(S,V)| \leq \text{vol}(S) \leq 2|E(S,V)|$, *namely the volume counts the number of edges incident to S up to a factor of 2. This is because there are two types of edges that will be counted towards* $\text{vol}(S)$*: (a) edges with two endpoints within S and (b) edges with only one endpoint within S. Edges of type (a) will be counted twice and edges of type (b) will be counted once. Hence,* $\text{vol}(S)$ *is bounded between the number of edges attached to S and twice of that number.*

2. $\text{vol}(S) + \text{vol}(V \setminus S) = \text{vol}(V) = 2|E|$.

Definition 2.5 (Conductance)**.** Let $G = (V, E)$ be a undirected unweighted graph. **Conductance of set/cut** is defined as

$$\Phi_G(S) = \frac{|\partial S|}{\min\{\text{vol}(S), \text{vol}(V \setminus S)\}}$$

Conductance of graph G is defined as

$$\Phi(G) = \min_{S: 0 < vol(S) < vol(V)} \Phi_G(S).$$

Intuitively, conductance describes overall the graph is sparse or not. The smaller the conductance, the sparser the graph.

Remark 2.6. Computing conductance is NP-hard.

Observation 2.7. 1. $\forall S \subset V, \Phi_G(S) \in [0,1]$. *So* $\Phi_G \in [0,1]$.

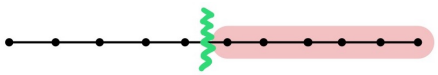
2. $\Phi(G) = 0 \iff$ *G is not connected (with the assumption that each vertex in the graph has a self loop.)*

We say

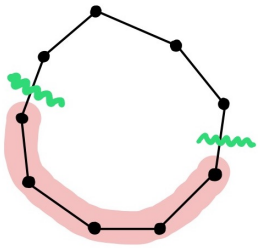
- S is a $\phi-$**sparse cut** $\iff \Phi_G(S) < \phi$ (i.e. its conductance is smaller than ϕ)
- G is a $\phi-$**expander** $\iff \Phi(G) \geq \phi$ (i.e. G has conductance at least ϕ)
- namely, G is $\phi-$expander $\iff$ G has no $\phi-$sparse cut
- larger $\phi \iff$ more well-connected.

Exercise 2.8. To internalize the definition, we look at the following examples:

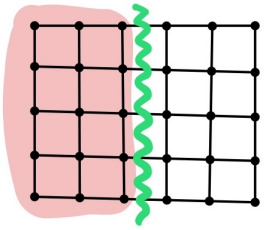
- paths, as mentioned at the beginning, if we pick the right subhalf (the red portion) as S, then $\text{vol}(S) = \text{vol}(V \setminus S) = O(n)$, and $|\partial S| = 1$. Hence $\Phi(G) = O\left(\frac{1}{n}\right)$.



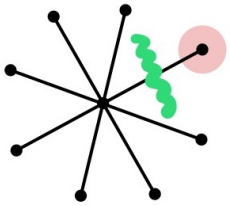
- cycles, which is similar to a path. If we pick the red portion as S, then $\text{vol}(S) = \text{vol}(V \setminus S) = O(n)$, and $|\partial S| = 2$. Therefore $\Phi(G) = O\left(\frac{1}{n}\right)$.



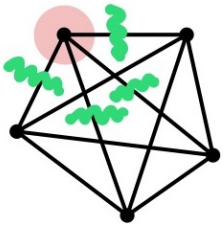
- grids, where we delete $O(\sqrt{n})$ edges and disconnect $O(n)$ vertices. Therefore $\Phi(G) = O\left(\frac{1}{\sqrt{n}}\right)$.



- stars, where disconnect one vertex from the graph always corresponds to delete the edge that attached to that vertex, whence $\Phi(G) = 1$.



- complete graph, where disconnecting a set A of vertices requires deleting all edges that are connected to vertices outside of A, whence $\Phi(G) = \Omega(1)$.



2.1 High Conductance = Robust Against Edge Deletions

We will see how the definition conductance captures the notion of "robustness against edge deletions."

Theorem 2.9. *A connected graph $G = (V, E)$ has conductance at least $\Omega(\phi) \iff \forall D \subseteq E$, the total volume of small components $G \setminus D$ is at most $O\left(\frac{|D|}{\phi}\right)$, where we say a connected component C in $G \setminus D$ is small if $\mathrm{vol}_G(C) \leq \frac{\mathrm{vol}_G(V)}{2}$.*

Proof of Theorem. " $\Longleftarrow$ " we prove the contrapositive of the statement, namely: If $\Phi(G) < \phi$, then $\exists D \subseteq E$, s.t. the total volume of small components of $G \setminus D$ is greater than $\frac{|D|}{\phi}$.
According to the definition of conductance, $\exists S \subseteq V$, s.t. $\frac{|\partial S|}{\mathrm{vol}(S)} < \phi$, i.e. $\frac{|\partial S|}{\phi} < \mathrm{vol}(S) \leq \frac{\mathrm{vol}(V)}{2}$ and the last inequality is a consequence of $\mathrm{vol}(S)$ is the minimum between $\mathrm{vol}(S)$ and $\mathrm{vol}(V \setminus S)$.
We simply pick $D = \partial S$. Clearly S is a small component of $G \setminus D$ and $\mathrm{vol}(S) > \frac{|\partial S|}{\phi} = \frac{|D|}{\phi}$.
" $\Longrightarrow$ " we again prove the contrapositive, namely: If $\exists D$ where the total volume of small components of $G \setminus D$ is greater than $|D|/\phi$, then $\Phi(G) < 3\phi$.
We break into cases:

Case 1:

The normal situation is that after we delete a batch of edges, there will be a big component in the graph. Say $\exists C$ which is not a small component of $G \setminus D$, i.e. $\mathrm{vol}(C) > \frac{\mathrm{vol}(V)}{2}$.
Let S be the union of all small components, we have $\frac{|D|}{\phi} < \mathrm{vol}(S) < \mathrm{vol}(V) - \mathrm{vol}(C) < \frac{\mathrm{vol}(V)}{2}$. We know that $\partial S \subseteq D$ since when we delete D from G, S gets disconnected. Then,

$$\Phi(G) \leq \Phi_G(S) = \frac{|\partial S|}{\min\{\mathrm{vol}(S), \mathrm{vol}(V \setminus S)\}} \leq \frac{|D|}{\mathrm{vol}(S)} < \phi.$$

Case 2:

When all connected components in $G \setminus D$ are small components of G, this is harder to tackle.

Claim 2.10. *There exists S a union of small components in $G \setminus D$ s.t. $\frac{\mathrm{vol}(V)}{3} \leq \mathrm{vol}(S) \leq \frac{2\mathrm{vol}(V)}{3}$.*

Proof of Claim. Prove by contradiction. Denote all small components in $G \setminus D$ as $S_1, \cdots, S_p$. Since all components in $G \setminus D$ are small components, $\mathrm{vol}(S_1 \cup \cdots \cup S_p) = \mathrm{vol}(V)$. If for all unions S of small components of $G \setminus D$, either $\mathrm{vol}(S) < \frac{\mathrm{vol}(V)}{3}$ or $\mathrm{vol}(S) > \frac{2\mathrm{vol}(V)}{3}$. Let S be union of some S_is such that $\mathrm{vol}(S) \leq \frac{\mathrm{vol}(V)}{3}$, and $\forall S'$ being union of some S_is such that $\mathrm{vol}(S') \leq \frac{\mathrm{vol}(V)}{3}, \mathrm{vol}(S') \leq \mathrm{vol}(S)$. Then for all j s.t. $S_j \notin S$, $\mathrm{vol}(S_j) > \frac{\mathrm{vol}(V)}{3}$ in order that $\mathrm{vol}(S \cup S_j) > \frac{2\mathrm{vol}(V)}{3}$ as assumption. However, $\frac{\mathrm{vol}(V)}{2} \geq \mathrm{vol}(S_j) > \frac{\mathrm{vol}(V)}{3}$, which contradict with the assumption. □

Since $\partial S \subseteq D, |D| \leq \phi \mathrm{vol}(V)$,

$$\Phi(G) \leq \Phi_G(S) = \frac{|\partial S|}{\min\{\mathrm{vol}(S), \mathrm{vol}(V \setminus S)\}} \leq \frac{\phi \mathrm{vol}(V)}{\mathrm{vol}(V)/3} < 3\phi$$

□

2.2 Application: Checking Connectivity under Failures in Sub-linear time

- Let $G = (V, E)$ be an undirected graph with n vertices and m edges.
- We can preprocess G in $O(m)$. So that, given any $s, t \in V$, return if s and t are connected in $O(1)$ time.
  - How: using BFS to compute all the connected components in linear time.
- What if there are failures? Consider the following dynamic settings
  - Imagine a computer/road network.
  - Some edges might fail because of ... earthquake maybe.
  - We want to know if s and t are still connected, without recomputing from scratch.

Theorem 2.11. *Suppose that $G = (V, E)$ is a ϕ-expander. Without any preprocessing on G, for any $D \subseteq E$, we can check if $s, t \in V$ are connected in $G \setminus D$ in $O\left(\frac{|D|}{\phi}\right)$ time.*

Proof of Theorem. The proof follows from Theorem 2.9. WLOG, assume that $|D| \leq \frac{\phi \text{vol}(V)}{100}$, otherwise it will just trivially take $O(m)$ time which is the same as conducting BFS for the whole graph. Notice that there is a unique large component C_{large} in $G \setminus D$. Then, consider the algorithm below:

Algorithm:

- Start a BFS from s in the graph $G \setminus D$. There are two cases:
  - Once we have explored more than $10|D|/\phi$ volume, then just stop and conclude that s is in C_{large}.
  - Otherwise, we identify that is s is in a component C_s where $\text{vol}_G(C_s) < 10|D|/\phi$.
  - This takes $O(|D|/\phi)$ time, because BFS takes time proportional to the volume explored.
- Do the same for t.

In this way, we can identify which connected components containing s and t respectively in $O(|D|/\phi)$ time. □

2.3 Basic Property: Expanders have Small Diameter

Definition 2.12 (Diameter)**.** The *diameter* of a graph $G = (V, E)$ is $\max_{u,v \in V} \text{dist}(u, v)$.

Proposition 2.13. *The diameter of a ϕ-expander with n vertices and m edges is $O(\log(m)/\phi)$.*

Proof of Proposition. The proof uses ball-growing argument, which is also a useful tool for other proofs in the lecture.
Let $B(s, r) = \{u \mid \text{dist}(s, u) \leq r\}$ be a ball of radius r around s, r_s be the threshold radius such that $\text{vol}(B(s, r_s - 1)) \leq \text{vol}(V)/2$ yet $\text{vol}(B(s, r_s)) > \text{vol}(V)/2$. Let r_t be defined similarly for t. Notice that $B(s, r_s) \cap B(t, r_t) \neq \emptyset$ since both balls occupied more than half of the edges in G. Hence $\text{dist}(s, t) \leq r_s + r_t$.
We will show that $r_s, r_t \leq O(\log(m)/\phi)$.

Since all cut edges in $\partial(B(s,r-1))$ are at the boundry of $B(s,r-1)$, all paths starting from s within $B(s,r-1)$ given access to set $\partial(B(s,r-1))$ can have increment in length at most by 1, and therefore the length of all path in $B(s,r-1)\cup\partial(B(s,r-1))$ have length $\leq r$. Hence,

$$\mathrm{vol}(B(s,r)) \geq \mathrm{vol}(B(s,r-1)) + |\partial B(s,r-1)|.$$

According to the definition of G being a $\phi-$expander, $\forall r \leq r_s$

$$\frac{|\partial B(s,r-1)|}{\min\{\mathrm{vol}(B(s,r-1)), \mathrm{vol}(V \setminus B(s,r-1))\}} \geq \phi.$$

Since $\mathrm{vol}(B(s,r-1)) \leq \mathrm{vol}(B(s,r_s-1)) \leq \frac{\mathrm{vol}(V)}{2}$, $\min\{\mathrm{vol}(B(s,r-1)), \mathrm{vol}(V\setminus B(s,r-1))\} = \mathrm{vol}(B(s,r-1))$,

$$|\partial B(s,r-1)| \geq \phi \mathrm{vol}(B(s,r-1)).$$

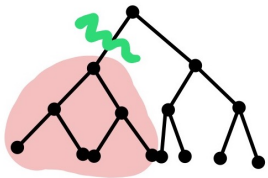
Since $\mathrm{vol}(B(s,0))$ is $\deg s \geq 1$, $\forall r \leq r_s$

$$\mathrm{vol}(B(s,r)) \geq (1+\phi)\mathrm{vol}(B(s,r-1)) \geq (1+\phi)^r.$$

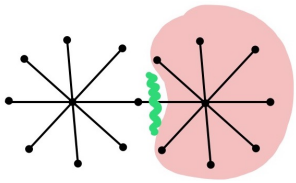
Since $e^{\phi r_s} \approx (1+\phi)^{r_s} \leq \mathrm{vol}(B(s,r_s)) \leq \mathrm{vol}(V) = 2m$, we have $r_s = O(\log(m)/\phi)$. The same holds for r_t. □

Remark 2.14. Conversely, not all low diameter graphs have high conductance. For example,

- Binary trees, where the diameter is $O(\log n)$, but the conductance is $O\left(\frac{1}{n}\right)$,



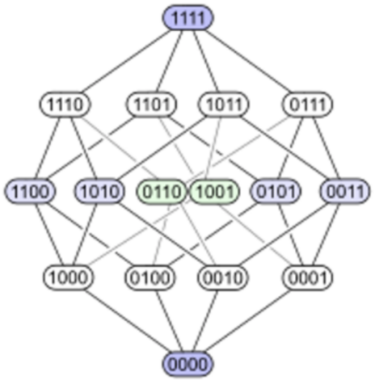
- stars connected by a short path, where the diameter is 4 but the conductance is $O\left(\frac{1}{n}\right)$.



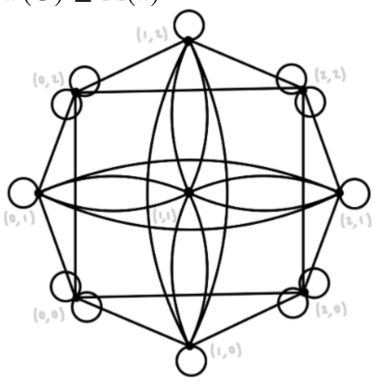
3 Non-trivial Examples: Small Degree Expanders

Intuitively, graphs containing large degree vertices can be more well-connected, whence having higher conductance. For example, the fact that complete graphs and stars have high conductance are not very surprising.
However, the existence of graphs with small maximum degree but still have large conductance is nontrivial. Namely, sparse graphs can still be very well-connected. Some examples are listed as below:

- Deterministic examples:
  - Hypercube:
    - $V = \{0,1\}^{\log n}$ and $E = \{(u,v) \mid u = v \text{ except that one entry}\}$.
    - $\Phi(G) \geq \Theta(1/\log n)$.

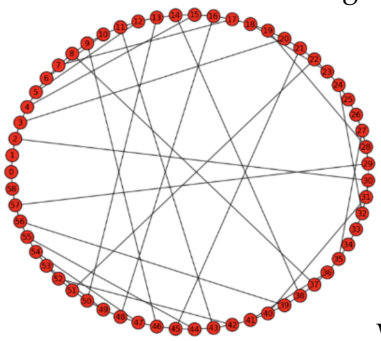


  - Margulis–Gabber–Galil
    - $V = (\mathbb{Z}/n\mathbb{Z}) \times (\mathbb{Z}/n\mathbb{Z})$
    - Each $(x,y) \in V$ has 8 neighbors: $(x \pm 2y, y), (x \pm (2y+1), y), (x, y \pm 2x), (x, y \pm (2x+1))$
    - $\Phi(G) \geq \Omega(1)$



when $n = 3$ [1]

  - Prime expander
    - $V = \mathbb{F}_p = \{0, \ldots, p-1\}$
    - $x \in V$ is connected to 3 neighbors: $x+1$, $x-1$ and x^{-1} (for $x \neq 0$)



when $p = 59$ [2]

[1] https://web.stanford.edu/~styopa/pdfs/expanders.pdf
[2] https://lucatrevisan.github.io/teaching/expanders2016/lecture16.pdf

- Randomized examples:
  - Erdos-Renyi graph $G_{n,p}$ when $p = \Omega(\log(n)/n)$:
    * With high probability, G has maximum degree $O(\log n)$ and$\Phi(G) \geq \Omega(1)$.
    * This means that almost all graphs are actually expanders.
  - Union of random $O(1)$ matchings
    * G has maximum degree $O(1)$
    * $\Phi(G) \geq \Omega(1)$ with high probability[3]

Note 3.1. "Canonical expanders" can be depicted as: well-connected, yet with low degree. (But of course, in general graphs with high conductance can also have various other appearances.)

4 Vertex Expansion

Conductance describes the ratio between #(cut edges) and #(edges disconnected by the cut). We can also talk about the analogue of it for vertices.

Definition 4.1 (Vertex-expansion)**.** Let (L, S, R) be a vertex cut, i.e. L, S, R partition V and $E(L, R) = \emptyset$. **Vertex-expansion of a cut** (L, S, R) is

$$h_G(S) = \frac{|S|}{\min\{|L \cup S|, |R \cup S|\}} \in [0, 1]$$

Vertex-expansion of a graph G is

$$h(G) = \min_{(L,S,R)} h(L, S, R) \in [0, 1]$$

- We say that
  - S is a ϕ**-vertex-sparse cut** iff $h_G(S) < \phi$
  - G is a ϕ**-vertex-expander** iff $h(G) \geq \phi$

Exercise 4.2. (vertex expansion and conductance are not necessarily the same, but could be under certain conditions) Show that

1. If a graph G has maximum degree $O(1)$, then vertex expansion and conductance has within a constant factor $h(G) = \Theta(\Phi(G))$.

   Proof. First of all, notice that all edge cuts corresponds to a vertex cut by simply letting the endpoints of cut edges to be the vertex cut. Since $\deg G = O(1)$, there is some constant $M, s.t. \deg G \leq M$. Let the vertex cut (L, S, R) be such that $h(G) = h(L, S, R)$. If $\Phi(G) = \frac{|\partial A|}{\text{vol}(A)}$, then $\frac{|\partial A|}{M} \leq |S| \leq 2|\partial A|$, which shows that $|S| = \Theta(|\partial A|)$. Since the degree of G is bounded by a constant, this means that for any subgraphs of G, the total number of edges is also bounded by the number of vertices up to a constant factor, namely $\min\{|L \cup S|, |R \cup S|\} = \Theta(\text{vol}(A))$. Hence, $h(G) = \Theta(\Phi(G))$. □

[3] https://lucatrevisan.github.io/teaching/expanders2016/lecture19.pdf

2. There is a graph G where $\Phi(G) \geq \Omega(1)$ but $h(G) \leq O(1/n)$.

   Proof. A star is an example where we need to delete the same amount of edges in order to disconnect that amount of vertices, whence $\Phi(star) = 1$. On the other hand we only need to delete the central vertex of the star to disconnect all vertices in the graph, therefore $h(star) = \frac{1}{n}$. □

3. There is a graph G where $h(G) \geq \Omega(1)$ but $\Phi(G) \leq O(1/n)$.

   Proof. Consider a graph contains 2 same cliques with a perfect matching between them. To disconnect one vertex from the graph, we need to delete the whole clique that this vertex stays in plus the vertex that matches with it. Therefore, $h(G) \approx \frac{1}{2} = \Omega(1)$. On the other hand, deleting the perfect matching let the whole graph break into two parts with same volume, i.e. $\Phi(G) = \frac{n}{n^2} = \Theta(\frac{1}{n})$. □

Exercise 4.3 (Robustness of vertex-expander against vertex deletion)**.** Suppose that $G = (V, E)$ is a ϕ-vertex-expander. Without any preprocessing on G, for any $D \subseteq V$, we can check if $s, t \in V$ are connected in $G \setminus D$ in $O((|D|/\phi)^2)$ time.

Proof. The proof should be an analogue of the proof of Theorem 2.11.

Claim 4.4. *$h(G) \geq \Omega(\phi) \iff \forall D \subseteq V$, the total number of vertices of small components of $G \setminus D$ is at most $O\left(\frac{|D|}{\phi}\right)$, where we say a connected components C is small in $G \setminus D$ is the number of vertices in C is less than or equal to $\frac{|V|}{2}$.*

Proof of Claim. Suffices to show the contrapositive. We first show that if $h(G) < \phi \implies \exists D \subseteq V$, s.t. the total number of vertices of small components of $G \setminus D > O\left(\frac{|D|}{\phi}\right)$. Since $h(G) < \phi, \exists (L, S, R)$ a vertex cut of V s.t. $\frac{|S|}{|L \cup S|} < \phi \implies \frac{|S|}{\phi} < |L \cup S| = |L| + |S| \implies O\left(\frac{|S|}{\phi}\right) = \frac{(1-\phi)|S|}{\phi} < |L|$. Let $S = D$, then since $\min\{|L \cup S|, |R \cup S|\} = |L \cup S|, |L| \leq \frac{|V|}{2}$, and hence L is a small component of $G \setminus S$. We next show that if $\exists D \subseteq V$, s.t. the total number of vertices of small components of $G \setminus D > O\left(\frac{|D|}{\phi}\right) \implies h(G) < 3\phi$. We break into cases:

- Case 1. $\exists C \subseteq G \setminus D$ s.t. C is not a small component in $G \setminus D$. Let A denote the union of all small components of $G \setminus D$, then $\frac{|D|}{\phi} < |A| \leq \frac{|V|}{2}$. Then

  $$h_G(D) = \frac{|D|}{\min\{|L \cup D|, |R \cup D|\}} < \frac{|D|}{|A|} < \phi$$

  Hence $h(G) \leq h_G(D) < \phi$.

- Case 2. when all elements in $G \setminus D$ are small components. Make a similar argument as in the claim in the proof of Theorem 2.11, we know that there exists A a union of small components in $G \setminus D$, s.t. $\frac{|V|}{3} \leq |A| \leq \frac{2|V|}{3}$. We also know that $|D| < \phi|V|$. Then

  $$h_G(D) = \frac{|D|}{\min\{|L \cup D|, |R \cup D|\}} \leq \frac{|D|}{|A|} < \frac{\phi|V|}{\frac{|V|}{3}} = 3\phi$$

□

Knowing that for a $\phi-$vertex expander, all small components in $G \setminus D$ will have total size at most $O\left(\frac{|D|}{\phi}\right)$, we did the similar algorithm as before: we start BFS from s in $G \setminus D$, and once we have explored more than $10\frac{|D|}{\phi}$ vertices, stop and conclude that s is in C_{large}. Otherwise, we identify that s is in a small component C_s. This takes $O\left(\left(\frac{|D|}{\phi}\right)^2\right)$ time since the time complexity of BFS is proportional to the number of edges, which is bounded by the square of number of vertices. We do the same for t, whence the total time is $O\left(\left(\frac{|D|}{\phi}\right)^2\right)$. □

Exercise 4.5. (vertex-expander has small diameter) Show that a ϕ-vertex-expander has diameter $O(\log(n)/\phi)$.

Proof. We will use the ball-growing argument again. Let $B(s,r) = \{u | dist(s,u) \leq r\}$. Let r_s be a threshold s.t. $|B(s, r_s - 1)| \leq \frac{|V|}{2}$, yet $|B(s, r_s)| > \frac{|V|}{2}$. Define the analogue for t.
We know that $B(s, r_s) \cap B(t, r_t) \neq \emptyset$ since they all contains more than $\frac{|V|}{2}$ vertices and by PHP there must be at least one vertex falls into both balls. Hence, $dist(s,t) \leq r_s + r_t$.
We next show that $r_s, r_t < O\left(\frac{\log(n)}{\phi}\right)$. Indeed, since $|B(s, r_s)| \geq |B(s, r_s - 1)| + |S_r|$ where S_r is the vertex cut that separate $B(s,r)$ from G. Since G is a $\phi-$vertex expander, and $\forall r \leq r_s - 1, |B(s,r)| \leq \frac{|V|}{2}$, $h_G(S_r) = \frac{|S_r|}{\min\{|L \cup S_r|, |R \cup S_r|\}} = \frac{|S_r|}{|B(s,r)|} \geq \phi$ i.e. $|S_r| \geq \phi |B(s,r)|$. Hence, $|B(s, r_s)| \geq (1+\phi)|B(s, r_s - 1)| \implies n = |V| \geq |B(s, r_s)| \geq (1+\phi)^{r_s} \approx e^{\phi r_s} \implies r_s \leq \frac{\log n}{\phi}$. Similar for r_t. □

5 Edge Expansion with General Demand

- The following notion generalizes conductance.
- Let $\boldsymbol{d} : V \to \mathbb{R}_{\geq 0}$ be a *demand function* of vertices.
- A **$\boldsymbol{d}$-expansion of a cut** S where $0 < \boldsymbol{d}(S) < \boldsymbol{d}(V)$ is

$$\Phi_{G,\boldsymbol{d}}(S) = \frac{\partial S}{\min\{\boldsymbol{d}(S), \boldsymbol{d}(V \setminus S)\}}$$

  where $\boldsymbol{d}(S) = \sum_{u \in S} \boldsymbol{d}(u)$.
- A **$\boldsymbol{d}$-expansion of a graph** G is

$$\Phi(G, \boldsymbol{d}) = \min_{S : 0 < \boldsymbol{d}(S) < \boldsymbol{d}(V)} \Phi_{G,\boldsymbol{d}}(S)$$

- Conductance is the same as $\boldsymbol{d}$-expansion when $\boldsymbol{d}(u) = \deg(u)$ for all u.

Exercise 5.1. (a graph with big expansion and large diameter: expansion and conductance can be different) In the literature, people often consider $\boldsymbol{d}^{unit}$-expansion where $\boldsymbol{d}^{unit}(u) = 1$ for all $u \in V$ and call this "*expansion*". This measures the ratio between #(cut edges) and #(vertices disconnected by the cut). Show that even if $\Phi(G, \boldsymbol{d}^{unit}) \geq \Omega(1)$, then diameter of G can be large.

Remark 5.2. This is unrelated to the solution, but notice that this value $\Phi(G, \boldsymbol{d}^{unit})$ is not bounded by 1 but can be as big as the max degree. For example, in a complete graph, $\Phi(G, \boldsymbol{d}^{unit}) = \frac{|S| \times |V \setminus S|}{\min\{|S|, |V \setminus S|\}} = \Omega(n)$.

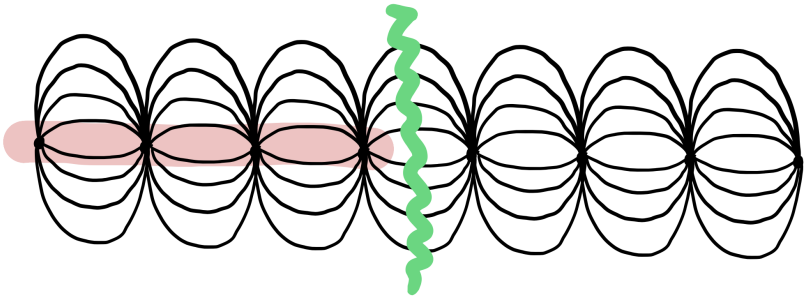


Figure 1: G_1

Solution. Consider the graph above, where $\Phi(G_1, \boldsymbol{d}^{unit}) = 2 \geq \Omega(1)$, yet the diameter is 7 which is pretty large.
Generalize to simple graph, consider changing one vertex into layer of n vertices, and connecting each layers with the layer next to it using a complete bipartite graph as below

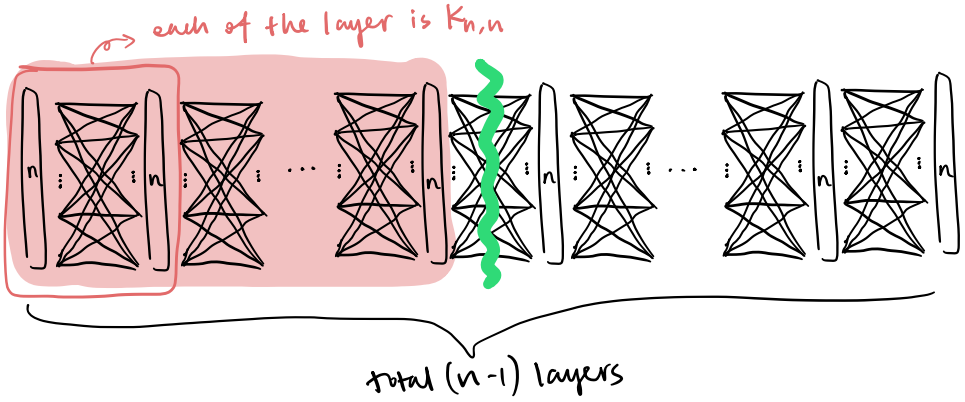


Figure 2: G_2

In this case, $\Phi(G_2, \boldsymbol{d}^{unit}) = \frac{n^2}{n^2/2} = 2 \geq \Omega(1)$, but the diameter is $O(n)$. □

Exercise 5.3. (a graph with big d-expansion but small conductance) Let $T \subseteq V$ be a subset called terminal. Let $\boldsymbol{d}(u) = \deg(u)$ if $u \in T$ otherwise $\boldsymbol{d}(u) = 0$. Give an example of a connected graph G where $\Phi(G, \boldsymbol{d}) = \Omega(1)$ but $\Phi(G)$ is very small.
Hint: Start with a graph $G_0 = (T, E_0)$ where $\Phi(G_0) = \Omega(1)$. Let $G = (V, E)$ be obtained from G_0 by subdividing each edge into a path of length n. Show that $\Phi(G, \boldsymbol{d}) = \Omega(1)$.

Solution. 1. Consider the connected graph G consist of a clique connected with a path, where the clique is T. Then, the conductance of the graph is very small since it's basically the con-

ductance of the path. However, $\Phi(G, \boldsymbol{d})$ is large since it evaluates how well-connected the subset T is, in this case the clique.

2. A more nontrivial example is as suggested in the hint. We first show the conductance of G is small. Indeed, $\Phi(G) \leq O\left(\frac{1}{n}\right)$ since we could pick any random edge in G_0, which turns into a path of length n in G, and for that path in G, we delete two edges that are attached to the original vertices in G_0, and this will make the whole chunk of path get disconnected from the graph.
   We next show that $\Phi(G, d) \geq \Omega(\Phi(G_0))$. Indeed, given a cut $S \subseteq V$ in G, we can construct a corresponding cut S_0 of G_0 by contracting G back to G_0 and take the contraction S_0 of S in G_0. Hence, $|\partial_G(S)| \geq |\partial_{G_0}(S_0)|$. Since d is defined to be $\deg(u)$ on T and 0 otherwise, $\min\{\mathrm{vol}_G(S), \mathrm{vol}_G(V \setminus S)\} = \min\{\mathrm{vol}_G(T \cap S), \mathrm{vol}_G(V \setminus S \cap T)\} = \min\{\mathrm{vol}_{G0}(S_0), \mathrm{vol}_{G_0}(T \setminus S_0)\}$. Then

$$\Phi_{(G,\boldsymbol{d})}(S) = \frac{|\partial_G(S)|}{\min\{\mathrm{vol}_G(S), \mathrm{vol}_G(V \setminus S)\}} \geq \frac{|\partial_{G_0}(S_0)|}{\min\{\mathrm{vol}_{G0}(S_0), \mathrm{vol}_{G_0}(T \setminus S_0)\}} = \Phi_{G_0}(S_0) \geq \Omega(1)$$

   Hence $\Phi(G, \boldsymbol{d}) = \Omega(1)$.

□

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 3: Approximating Conductance

September 7, 2021

Instructor: Thatchaphol Saranurak

Scribe: Gary Hoppenworth

1. Overview

Conductance is an important property of graphs that measures how well-connected they are. Expanders are graphs with "large" conductance. (Typically, graphs on n vertices are considered good expanders if they have conductance greater than $1/\text{polylog}\, n$.) Unfortunately, computing conductance exactly is NP-hard in general graphs.

However, there is an efficient $O(\log n)$-approximation of conductance which allows us to check the expansion of a graph (up to some extent). This approximation relies on two ingredients: 1) Bourgain's metric embedding into ℓ_1-metrics and 2) connections between ℓ_1-metrics and cut-metrics. These two tools are used to obtain an approximation via linear programming relaxation. In this lecture, we introduce the necessary background for the above two ingredients and then prove the LP relaxation approximation.

2. Metrics

A metric space (or metric) is a mathematical notion that captures distance. Metric spaces are composed of a set X of points and a distance function d that lets us measure the distance between points.

2.1. Definition. Let X be a set and let d be a function $d : X \times X \mapsto \mathbb{R}_{\geq 0}$. We say that (X, d) is a **metric** if the following hold:

1. $d(x, x) = 0$ for all $x \in X$ [1]
2. $d(x, y) = d(y, x)$ for all $x, y \in X$
3. $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in X$

The function d is called the **distance function**.

[1] In mathematics, this condition is stated as $d(x, y) = 0$ if and only if $x = y$. What we call metric in computer science is usually referred to as semi-metric in math.

2.1. Examples of Metrics

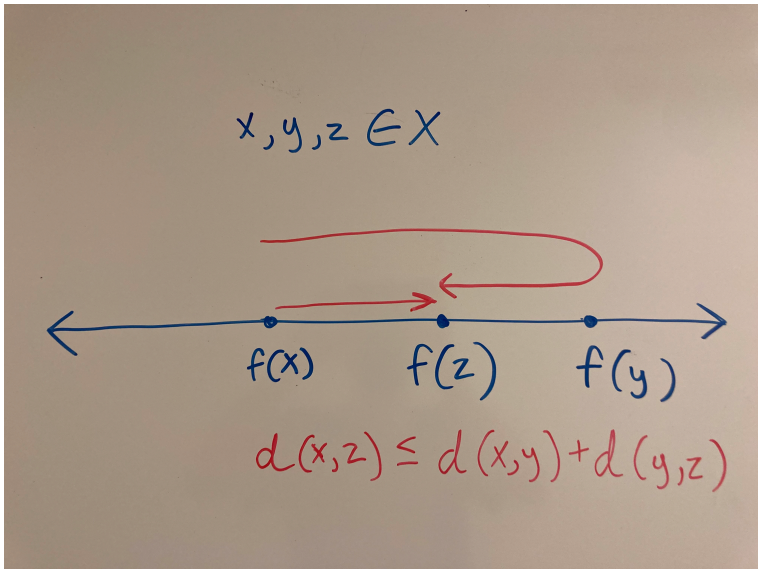
Line Metric: Given a set X and a function $f : X \mapsto \mathbb{R}$, define the distance function as $d(x, y) = |f(x) - f(y)|$.

Proof that this is a metric:

1. $d(x, x) = |f(x) - f(x)| = 0$ for all $x \in X$
2. $d(x, y) = |f(x) - f(y)| = |f(y) - f(x)| = d(y, x)$ for all $x, y \in X$.
3. $d(x, z) = |f(x)-f(z)| = |f(x)-f(y)+f(y)-f(z)| \leq |f(x)-f(y)|+|f(y)-f(z)| = d(x, y)+d(y, z)$ for all $x, y, z \in X$

□

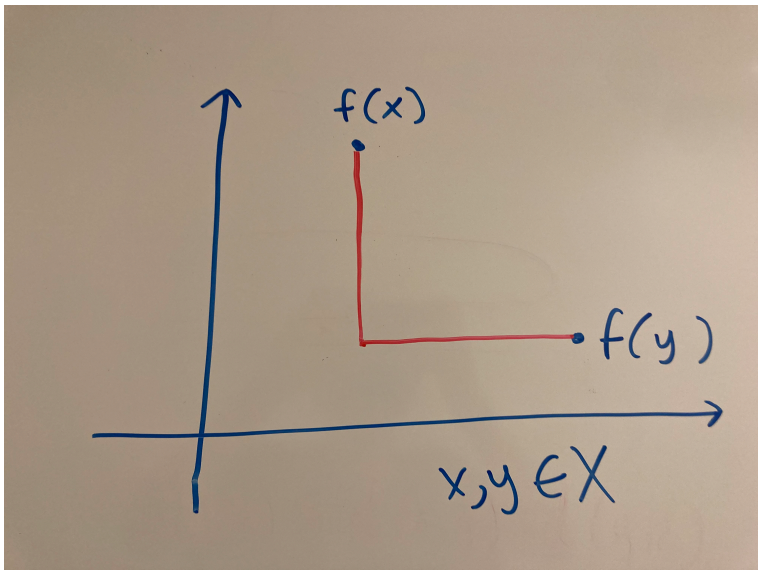
Below is a visualization of a line metric and the proof of statement 3.



ℓ_1-Metric: Given a set X and a function $f : X \mapsto \mathbb{R}^m$, define the distance function as $d(x, y) = \|f(x) - f(y)\|_1 = \sum_i |f_i(x) - f_i(y)|$. (Note when $m = 1$ this is the line metric.)

Proof that this is a metric: The ℓ_1-metric is really a sum of line metrics. This proof is essentially the same as the previous proof. □

Below is a visualization of distances in an ℓ_1-metric.



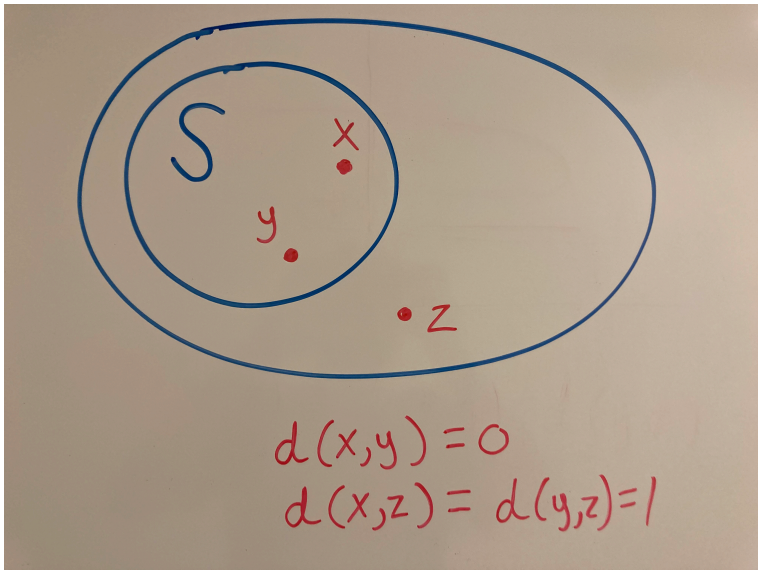
Cut Metric: Given a set X and a set $S \subseteq X$, let $d(x, y) = 0$ if $x, y \in S$ or $x, y \in X \setminus S$. Else, $d(x, y) = 1$.

Proof that this is a metric:

1. $d(x, x) = 0$, since x cannot belong to both S and $X \setminus S$.
2. $d(x, y) = d(y, x)$ since the definition of d does not depend on the order of x and y.
3. If $d(x, z) = 0$, then the inequality immediately holds. Else $d(x, z) = 1$. Assume without loss of generality that $x \in S$ and $z \in X \setminus S$. If $y \in S$, then $d(y, z) = 1$. Otherwise, $y \in X \setminus S$ and $d(x, y) = 1$. Either way, $d(x, y) + d(y, z) \geq 1 = d(x, z)$.

□

Below is a visualization of distances in the cut metric for a set S.



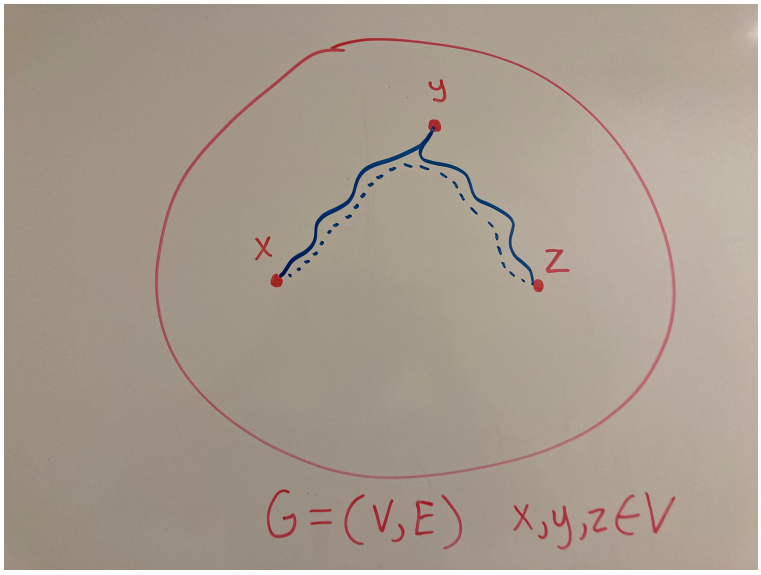
Shortest Path Metric: Given a set X and an undirected graph $G = (X, E)$, define the distance function as $d(x, y) = d_G(x, y)$, where $d_G(x, y)$ is the length of the shortest (x, y)-path in G.

Proof that this is a metric:

1. $d(x, x) = \text{dist}_G(x, x) = 0$. (The shortest path from a vertex to itself has length zero.)
2. $d(x, y) = \text{dist}_G(x, y) = \text{dist}_G(y, x) = d(y, x)$. (A shortest (x, y)-path has the same length as a shortest (y, x)-path in any undirected graph.)
3. $d(x, z) = \text{dist}_G(x, z) \leq \text{dist}_G(x, y) + \text{dist}_G(y, z) = d(x, y) + d(y, z)$. The inequality follows from the fact that a (x, y)-shortest path can be combined with a (y, z)-shortest path to obtain a (x, z)-path.

□

Below is a visualization of the proof of the triangle inequality (statement 3) for undirected graphs.



3. Metric Embeddings

3.1. Motivation

Computing the distance in some metrics is more expensive than in other metrics. This can be seen by comparing the costs of computing distances in the shortest path metric and the ℓ_1-metric.

Consider a shortest path metric of a graph G on n vertices and m edges. Computing $d(x, y) = \text{dist}_G(x, y)$ where $x, y \in V(G)$ takes $\Omega(m)$ time using Dijkstra's algorithm. By precomputing all distances you can answer queries in $O(1)$ time, but this requires $\Omega(n^2)$ space. These computations can be expensive for large graphs.

On the other hand, for an ℓ_1-metric with dimension k, computing $d(x, y) = \|f(x) - f(y)\|$, where $x, y \in \mathbb{R}^k$, takes $O(k)$ time by simply reading x and y and performing $O(k)$ operations. This computation is cheap for small k.

Because we can compute distances in a low dimension ℓ_1-metric quickly, it would be advantageous to represent any metric (X, d) as an ℓ_1-metric with low dimension. This would allow us to measure distance in the (X, d) metric (perhaps within some distortion factor) using cheaper computations. We call this representation an embedding and define it formally in the next section.

3.2. Formal Definition

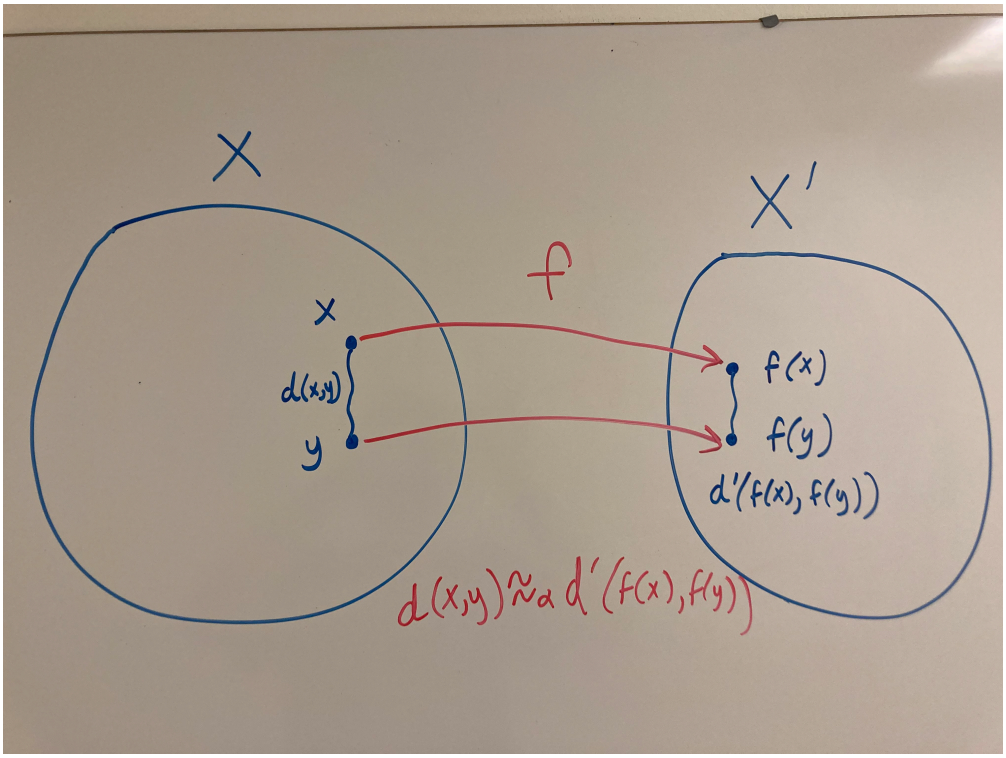
3.1. Definition (Metric Embeddings)**.** Let (X, d) and (X', d') be two metrics. Let $f : X \mapsto X'$ be a mapping such that for any $x, y \in X$

$$d'(f(x), f(y)) \leq d(x, y) \leq \alpha \cdot d'(f(x), f(y))$$

Then f is a **metric embedding**. We say that f **embeds** (X, d) into (X', d') with **distortion** α. This can also be written as

$$(X, d) \overset{\alpha}{\hookrightarrow} (X', d')$$

Below is a visualization of the notion of a metric embedding.



3.2. *Remark.* When $\alpha = 1$, we say that f has **no distortion**, or is **distance-preserving**, or is **isometric**.

3.3. Definition (ℓ_p^k-metrics)**.** We say that a metric (X, d) is a ℓ_p^k-metric ($p \geq 1$) if and only if there is an embedding f that embeds (X, d) into (X', d') with no distortion, and (X', d') is an ℓ_p-metric with dimension k.

3.4. Example. A cut metric (X, d_S) is a line metric (and therefore it is an ℓ_1^1-metric).

Proof: Define the function $1_S : X \mapsto \{0, 1\}$ so that $1_S(x) = 1$ if and only if $x \in S$. Let $d'(x, y) = |1_S(x) - 1_S(y)|$ for all $x, y \in X$. Note that (X, d') is a line metric. It follows from the definition of the cut metric that $d_S(x, y) = |1_S(x) - 1_S(y)| = d'(x, y)$ for all $x, y \in X$. Thus we have an embedding of (X, d_S) to (X, d') with distortion $\alpha = 1$. □

We now describe the two main ingredients that we use to approximate conductance. The first ingredient will show that any metric can be embedded in a ℓ_1^p metric for "small" p with low distortion. The second ingredient will show that any ℓ_1^p metric can be expressed as a conic combination of cut-metrics. Thus cut-metrics are in some sense universal: they capture every metric in some (approximate) sense.

3.3. Ingredient 1: Bourgain's Metric Embedding

A surprising fact is that every n-point metric can be embedded into an ℓ_1-metric of dimension $O(\log^2 n)$ with distortion $O(\log n)$. This is powerful because the ℓ_1-metric we are embedding into has both small dimension and small distortion. The formal statement of this fact is as follows.

3.5. Theorem (Bourgain)**.** *Given an n-point metric (X, d), there is an embedding $f : X \to \mathbb{R}^{O(\log^2 n)}$ such that for any $x, y \in X$,*

$$\|f(x) - f(y)\|_1 \leq d(x, y) \leq O(\log n) \cdot \|f(x) - f(y)\|_1$$

Moreover, given d, in $\tilde{O}(n^2)$ time, the mapping f can be computed correctly with probability at least $1 - 1/n$. If (X, d) is a shortest path metric of an m-edge graph, then the running time is $\tilde{O}(m)$ time. (Here $\tilde{O}(\cdot)$ is used to hide polylog *factors.)*

Note that we may ensure that for all $x \in X$, the i-th coordinate $f_i(x) = d(x, y)$ for some $y \in X$, so the coordinate $f_i(x)$ is not *too* big. The original embedding to an ℓ_1-metric was by Bourgain (1985)[2], but the ℓ_1-metric had more dimensions. The bound of $O(\log^2 n)$ dimensions is due to Linial, London, and Rabinovich (1995)[3]. There are many other examples of embedding metrics into more friendly metrics, such as the Johnson-Lindenstrauss dimension reduction and Bartal's tree embedding. In fact, there are entire courses on metric embeddings[4].

We can use Bourgain's embedding to approximate the shortest path distances in a graph G efficiently. For instance, given a graph G on n vertices, Bourgain's theorem allows us to embed the shortest path metric of G into a $\ell_1^{O(\log^2 n)}$-metric with $O(\log n)$ distortion. Then we can label each vertex of G with $O(\log^2 n)$ coordinates so that given vertices $u, v \in V(G)$, we can obtain a $O(\log n)$-approximation of $\text{dist}_G(u, v)$ in $O(\log^2 n)$ time by reading the labels of u and v and

[2] J. Bourgain. On Lipschitz embeddings of finite metric spaces in Hilbert space. Israel J. of Math. 1985.

[3] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. Combinatorica, 15(2):215–245, 1995. (Preliminary version in 35th FOCS, 1994).

[4] https://home.ttic.edu/ harry/teaching/pdf/lecture1.pdf

performing $O(\log^2 n)$ operations compute the corresponding distance in the $\ell_1^{O(\log^2 n)}$-metric. The total space used is $O(n \log^2 n)$, which is sublinear in the number of edges. However, we will be using Bourgain's embedding in a very different way.

4. Ingredient 2: ℓ_1-Metrics are in the Cut Cone

We saw earlier that a cut metric is a very simple ℓ_1-metric. Now we will see the other direction: every ℓ_1-metric is a combination of cut metrics. To formalize this, we will need some additional notation.

4.1. Observation. *For any n-point metric (X, d), we may think of d as a vector $d \in \mathbb{R}^{\binom{n}{2}}$. Each coordinate in this vector corresponds to a pair of points $x, y \in X$ and has value $d(x, y)$.*

We will refer to the vector $d \in \mathbb{R}^{\binom{n}{2}}$ as an n-point metric, since it encodes everything about (X, d).

4.2. Fact (Closure under addition)**.** *If d_1 and d_2 are n-point metrics, then $d_1 + d_2$ is an n-point metric.*

Proof. We must verify that the three requirements of a metric space hold for the $d_1 + d_2$ metric space, which is defined as $(d_1 + d_2)(x, y) = d_1(x, y) + d_2(x, y)$.

1. $(d_1 + d_2)(x, x) = d_1(x, x) + d_2(x, x) = 0$, where the final equality follows since d_1 and d_2 are metric spaces.
2. $(d_1 + d_2)(x, y) = d_1(x, y) + d_2(x, y) = d_1(y, x) + d_2(y, x) = (d_1 + d_2)(y, x)$, where the final equality follows since d_1 and d_2 are metric spaces.
3. $(d_1 + d_2)(x, z) = d_1(x, z) + d_2(x, z) \leq d_1(x, y) + d_1(y, z) + d_2(x, y) + d_2(y, z) = (d_1 + d_2)(x, y) + (d_1 + d_2)(x, z)$. The inequality follows from the fact that d_1 and d_2 are metric spaces.

□

4.3. Fact (Closure under scalar multiplication)**.** *If d is an n-point metric, then $\alpha \cdot d$ is a metric.*

Proof. We must verify that the three requirements of a metric space hold for the $\alpha \cdot d$ metric space, which is defined as $(\alpha \cdot d)(x, y) = \alpha \cdot d(x, y)$.

$(\alpha \cdot d)(x, y) = \alpha \cdot d(x, y) = 0$, since d is a metric.

$(\alpha \cdot d)(x, y) = \alpha \cdot d(x, y) = \alpha \cdot d(y, x) = (\alpha \cdot d)(y, x)$, since d is a metric.

$(\alpha \cdot d)(x, z) = \alpha \cdot d(x, z) \leq \alpha(d(x, y) + d(y, z)) = (\alpha \cdot d)(x, y) + (\alpha \cdot d)(y, z)$. (The inequality follows from the fact that d is a metric.) □

To summarize, non-negative linear combinations of metrics are metrics.

4.1. The Cone of Cut Metrics

4.4. Notation. Let $\text{CUTCONE}_n = \{d \in \mathbb{R}^{\binom{n}{2}} \mid d = \sum_{S \subseteq V} \alpha_S \cdot d_S$ where $\alpha_S \geq 0$ and d_S is a cut metric$\}$

CUTCONE_n contains all nonnegative combinations of all cut metrics (i.e. it is a *convex cone* of cut metrics). By Fact 4.2 and Fact 4.3, any $d \in \text{CUTCONE}_n$ is a metric.

4.2. Key Insight: $[0,1]$-Line Metrics are in the Cut Cone

4.5. Definition. We define a $[0,1]$-line metric to be a line metric with distance function $d(x,y) = |f(x) - f(y)|$, where f is a function $f : X \mapsto [0,1]$. That is, f maps only to points in the interval $[0,1]$ on the line.

4.6. Lemma. *Let (X,d) be a $[0,1]$-line metric where $d(x,y) = |f(x) - f(y)|$ for some $f : X \mapsto [0,1]$. Pick a threshold $t \in [0,1]$ uniformly at random. Then define*

$$S_t = \{x \mid f(x) \leq t\}$$

Then for every $u, v \in X$,

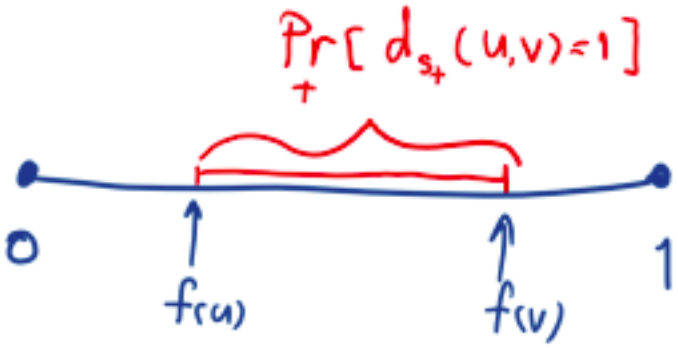
$$\mathbb{E}_t[d_{S_t}(u,v)] = d(u,v).$$

That is, $d = \mathbb{E}_t[d_{S_t}]$ is simply a distribution of cut metrics.

Proof. Note that d_{S_t} is the distance function for the cut metric defined by cut $S_t \subseteq X$. Consider $u, v \in X$, and assume without loss of generality that $f(u) \leq f(v)$.

$$\begin{aligned}
\mathbb{E}_t[d_{S_t}(u,v)] &= \Pr_t[d_{S_t}(u,v) = 1] && \text{by the definition of expectation} \\
&= \Pr_t[|S_t \cap \{u,v\}| = 1] && \text{by the definition of the cut metric} \\
&= \Pr_t[f(u) \leq t \leq f(v)] && \text{by the definition of } S_t \\
&= |f(v) - f(u)| && \text{since } t \text{ is sampled uniformly from } [0,1] \text{ (see figure)} \\
&= d(u,v)
\end{aligned}$$

□



If d is an $n-$point metric, we can rephrase this result in an equivalent way. Write $X = \{x_1, x_2, \ldots, x_n\}$, where $f(x_i) \leq f(x_{i+1})$ (i.e. so the points are ordered lowest to highest). Then for $1 \leq i < n$, define $S'_i = \{x_1, \ldots, x_i\}$, and let $\alpha_i = f(x_{i+1}) - f(x_i)$.

4.7. Observation. *Let d' be the metric*

$$d' = \sum_{i=1}^{n-1} \alpha_i d_{S'_i}$$

then $d' \in \text{CUTCONE}_n$.

Proof. Observe that for $1 \leq i < n$, we have that $d_{S'_i}$ is a cut metric and $\alpha_i > 0$. Then d' is a nonnegative linear combination of cut metrics and therefore is in CUTCONE_n. □

Example: Suppose our set X was defined as $X = \{x_1, x_2, x_3, x_4\}$ as below.

Then we would define $S'_1 = \{x_1\}$, $S'_2 = \{x_1, x_2\}$, and $S'_3 = \{x_1, x_2, x_3\}$. Additionally, $\alpha_1 = 0.7$, $\alpha_2 = 0.1$, and $\alpha_3 = 0.2$. Then the metric d' would equal $d' = 0.7d_{S'_1} + 0.1d_{S'_2} + 0.2d_{S'_3}$. We will prove that metric d' as defined in Observation 4.7 is identical to the original $[0, 1]$-line metric d.

4.8. Claim. $d = d' = \sum_{i=1}^{n-1} \alpha_i d_{S'_i}$

Proof. Observe that for any choice of $t \in [0, 1]$, we have that $S_t = S'_i$ for some i. Specifically, $S_t = S'_i$ if and only if $f(x_i) \leq t < f(x_{i+1})$. Then we have that:

$$\begin{aligned} d &= \mathbb{E}_t[d_{S_t}] \\ &= \sum_{i=1}^{n-1} \Pr_t[S_t = S'_i] d_{S'_i} \\ &= \sum_{i=1}^{n-1} \Pr_t[f(x_i) \leq t < f(x_{i+1})] d_{S'_i} \\ &= \sum_{i=1}^{n-1} \alpha_i d_{S'_i} \\ &= d' \end{aligned}$$

□

Observe that $S'_i \subset S'_{i+1}$ for $1 \leq i < n$. Then following corollary is immediate.

4.9. Corollary. *Any $[0, 1]$-line metric d of n points is a nonnegative combination of $n - 1$ **nested** cut metrics. In particular, $d \in \text{CUTCONE}_n$.*

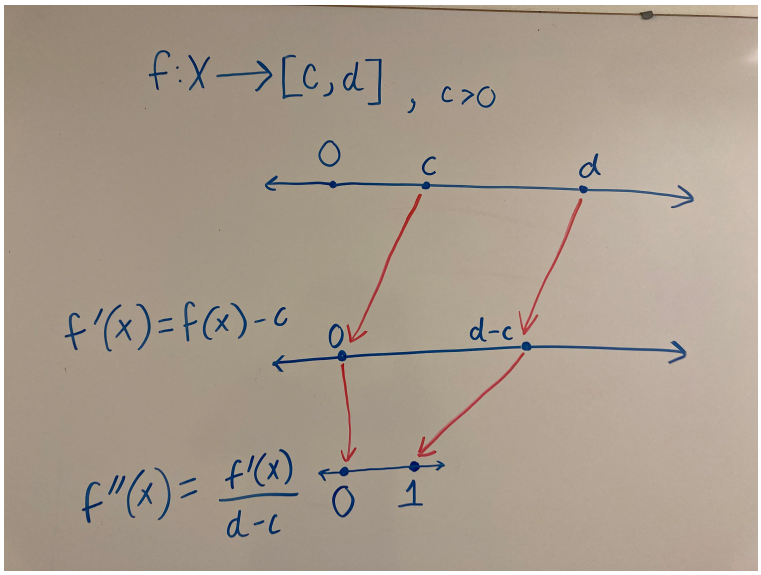
4.3. Generalizing to any Line Metric

Now we extend this result to arbitrary $n-$point line metrics. Consider an $n-$point line metric (X, d) where $d(x, y) = |f(x) - f(y)|$ for some $f : X \mapsto \mathbb{R}$. Without loss of generality, we can assume that $\min_x f(x) = 0$. This is because if $\min_x f(x) = c > 0$, then we may let $f'(x) = f(x) - c$ for all x (translating f), so that $d(x, y) = |f(x) - f(y)| = |f'(x) - f'(y)|$.

It follows that $d(x, y) = |f(x) - f(y)|$ for some function $f : X \mapsto [0, \alpha]$, where $\alpha > 0$. Now if we define $f'(x) = 1/\alpha \cdot f(x)$ for all x (rescaling f), then observe that f' is a function from X to $[0, 1]$.

Then the line metric d' defined as $d'(x,y) = |f'(x) - f'(y)|$ is a $[0,1]$-line metric. Furthermore, $d = \alpha \cdot d'$, so d is a nonnegative combination of a $[0,1]$-line metrics.

This translation and rescaling can be visualized in the figure below.



Note that since metric d' is a nonnegative combination of $n-1$ nested cut metrics by Corollary 4.9, it follows that metric d is a nonnegative combination of $n-1$ cut metrics as well.

4.10. Corollary. *Any line metric d on n points is a nonnegative combination of $n-1$ nested cut metrics. In particular, $d \in \text{CUTCONE}_n$.*

4.4. Generalizing to any ℓ_1-Metric

Consider any ℓ_1^k-metric (X,d) where $d(x,y) = \|f(x) - f(y)\|_1$ for some $f : X \mapsto \mathbb{R}^k$.

4.11. Observation. *d is a sum of k line metrics.*

Proof. Let $d_i(x,y) = |f_i(x) - f_i(y)|$, where $f_i(x)$ is the ith coordinate of $f(x)$. Note that (X,d_i) is a line metric. Furthermore, by the the definition of an ℓ_1^k-metric, we have that

$$d(x,y) = \|f(x) - f(y)\|_1 = \sum_{i=1}^{k} |f_i(x) - f_i(y)| = \sum_{i=1}^{k} d_i(x,y)$$

Then d is a nonnegative combination of the k line metrics $d_1, \ldots, d_k$. □

Since d is a nonnegative combination of k line metrics, which are in turn each a nonnegative combination of $n-1$ cut metrics by Corollary 4.10, it follows that d is a nonnegative combination of $k(n-1)$ cut metrics.

4.12. Corollary. *Any ℓ_1^k-metric d of n points is a nonnegative combination of $k(n-1)$ cut metrics. In particular, $d \in \text{CUTCONE}_n$.*

Given an ℓ_1-metric and its mapping f, we can efficiently compute its equivalent nonnegative combination of cut metrics specified in Corollary 4.12.

4.13. Notation. Given a mapping f where $d(x,y) = \left\|f(x) - f(y)\right\|_1$, define $\mathcal{S}_f$ as a collection of cuts $S \subseteq X$ such that $d = \sum_{S \in \mathcal{S}_f} \alpha_S \cdot d_S$ where $\alpha_S > 0$ and d_S is a cut metric.

4.14. Corollary. *Given a mapping f of an n-point ℓ_1^k-metric $d(x,y) = \left\|f(x) - f(y)\right\|_1$, we can compute S_f and $\{\alpha_S\}_{S \in \mathcal{S}}$ in polynomial time.*

Proof. In $O(kn)$ time we can obtain the k line metrics $d_1, \ldots, d_k$ of Observation 4.11. Now in $O(kn)$ time we can rescale and translate each of these k line metrics $d_1, \ldots, d_k$ to obtain k $[0,1]$-line metrics $d_1^{01}, \ldots, d_k^{01}$ as in the proof of Corollary 4.10. Finally, for each $[0,1]$-line metric d_i^{01}, we can obtain the corresponding collection of cuts by replicating the procedure in Corollary 4.9. This requires us to sort the n points according to function f_i and obtain $O(n)$ cuts of size $O(n)$ each in $O(n^2)$ time. To compute the coefficients for the collection of cuts it suffices to compute $O(n)$ distances between points, which can be done in $O(n)$ time. Then overall this requires $O(kn^2)$ time. To obtain our final collection $\mathcal{S}_f$ we simply need to undo the scaling we performed to obtain $[0,1]$-line metrics earlier by changing the cut coefficients appropriately for each $[0,1]$-line metric. This can be done in $O(kn)$ time. Then we can compute $\mathcal{S}_f$ and $\{\alpha_S\}_{S \in \mathcal{S}_f}$ in $O(kn^2)$ time. □

4.15. *Remark.* Notice that the bottleneck is actually the size of the output - we can actually compute $\mathcal{S}_f$ in $O(k \cdot n \log n)$ time if the nested cuts in $\mathcal{S}_f$ are represented implicitly.

5. Approximating Conductance via LP Relaxation

5.1. Overview

Our goal is to approximate the conductance $\Phi(G)$ of a graph G. To accomplish this we will first define another notion of expansion $\Phi(G,H)$, which generalizes the notion of conductance (this will make notation easier). Then we will define a linear program (LP) relaxation of $\Phi(G,H)$ denoted by $\mathrm{LR}(G,H)$ such that $\mathrm{LR}(G,H) \leq \Phi(G,H)$. Then $\mathrm{LR}(G,H)$ can be computed in polynomial time since it can be modeled by a LP. We will show that $\mathrm{LR}(G,H)$ is not too small using the ingredients we have seen above.

5.2. A General Notion of Edge Expansion: H-expansion $\Phi(G,H)$

Let $G = (V, E_G, w_G)$ be a weighted undirected graph, and let $H = (V, E_H, w_H)$ be a weighted undirected graph with the same set of vertices as G.

5.1. Definition. The H-expansion of a cut S is defined as

$$\Phi_{G,H}(S) = \frac{w_G(\partial_G S)}{w_H(\partial_H S)}$$

where $w_G(E') = \sum_{e \in E'} w_G(e)$ and $w_H(E') = \sum_{e \in E'} w_H(e)$.

5.2. Definition. The H-expansion of a graph G is defined as [5]

$$\Phi(G,H) = \min_{S: w_H(\partial_H S)\neq 0} \Phi_{G,H}(S)$$

Observe that the H-expansion of a graph G minimizes the H-expansion over all cuts S where this value is well-defined. The lemma below shows that $\Phi(G,H)$ generalizes the $\mathbf{d}$-expansion $\Phi(G,\mathbf{d})$ of a graph.

5.3. Lemma. *Given $G=(V,E)$, let H be a complete weighted graph with $w_H(u,v) = \frac{\boldsymbol{d}(u)\boldsymbol{d}(v)}{\boldsymbol{d}(V)}$ for each u,v. We have $\Phi_{G,H}(S) = \Theta(\Phi_{G,\boldsymbol{d}}(S))$ for all $S \subseteq V$ and so $\Phi(G,H) = \Theta(\Phi(G,\boldsymbol{d}))$.*

Proof. This follows because for any S where $\mathbf{d}(S) \leq \mathbf{d}(V)/2$, we have

$$w_H(\partial_H S) = \sum_{u\in S}\sum_{v\notin S} \frac{\mathbf{d}(u)\mathbf{d}(v)}{\mathbf{d}(V)} = \sum_{u\in S} \mathbf{d}(u)\frac{\mathbf{d}(V\setminus S)}{\mathbf{d}(V)} = \Theta(\mathbf{d}(S))$$

since $\mathbf{d}(V\setminus S) \geq \frac{1}{2}\mathbf{d}(V)$.

It follows that

$$\Phi_{G,H}(S) = \frac{w_G(\partial_G S)}{w_H(\partial_H S)} = \frac{w_G(\partial_G S)}{\Theta(\mathbf{d}(S))} = \Theta(\Phi_{G,\mathbf{d}}(S)).$$

□

Recall that conductance $\Phi(G) = \Phi(G,\mathbf{d})$, where $\mathbf{d}(u) = \deg(u)$. It immediately follows from Lemma 5.3 that we can choose a graph H so that $\Phi(G,H) = \Theta(\Phi(G))$.

5.4. Corollary. *Given $G=(V,E)$, let H be a complete weighted graph where $w_H(u,v) = \frac{\deg(u)\deg(v)}{\mathrm{vol}(V)}$ for each u,v. Then $\Phi(G,H) = \Theta(\Phi(G))$.*

5.3. LP Relaxation of $\Phi(G,H)$

We start with the following observation:

5.5. Observation.

$$w_G(\partial_G S) = \sum_{u,v:|\{u,v\}\cap S|=1} w_G(u,v) = \sum_{u,v} w_G(u,v)\cdot d_S(u,v)$$

where $d_S : V\times V \mapsto \{0,1\}$ is a cut metric defined by $S\subset V$, and the sum $\sum_{u,v}$ is summing over unordered *pairs of vertices.*

Using this observation, we may rewrite $\Phi(G,H)$ as

$$\Phi(G,H) = \min_{S: w_H(\partial_H S)\neq 0} \frac{w_G(\partial_G S)}{w_H(\partial_H S)} = \min_{d_S:\text{cut metric over } V} \frac{\sum_{u,v} w_G(u,v)\cdot d_S(u,v)}{\sum_{u,v} w_H(u,v)\cdot d_S(u,v)}$$

Then it follows that we may consider computing $\Phi(G,H)$ as an optimization problem minimizing over all cut metrics. Now the crucial step by Leighton and Rao that allows us to obtain a LP approximation is to instead minimize over *all metrics* instead of all cut metrics. We now define this new measure, which we denote $\mathrm{LR}(G,H)$.

[5] $\Phi(G,H)$ is also called the "non-uniform sparsity of G with respect to H".

5.6. Notation.

$$\mathrm{LR}(G,H) = \min_{d:\text{metric over } V} \frac{\sum_{u,v} w_G(u,v)\cdot d(u,v)}{\sum_{u,v} w_H(u,v)\cdot d(u,v)}$$

Now we claim that $\mathrm{LR}(G,H)$ can be modeled as a linear program (LP).

5.7. Claim. $\mathrm{LR}(G,H)$ *can be modeled as a linear program (LP).*

Proof. First, note that by normalizing, we can assume that $\sum_{u,v} w_H(u,v)\cdot d(u,v) = 1$. More concretely, if there exists a metric d' that minimizes $LR(G,H)$, then there exists a metric d'' that can be obtained by scaling d' that minimizes $LR(G,H)$ and further $\sum_{u,v} w_H(u,v)\cdot d''(u,v) = 1$. Minimizing $\sum_{u,v} w_G(u,v)\cdot d(u,v)$ over all metrics d where $\sum_{u,v} w_H(u,v)\cdot d(u,v) = 1$ is captured exactly by the following LP:

$$\begin{aligned}
\text{minimize} \quad & \sum_{u,v} w_G(u,v)\cdot d(u,v) \\
\text{subject to} \quad & \sum_{u,v} w_H(u,v)\cdot d(u,v) = 1 \\
& d(u,v) \le d(u,w) + d(w,v) \qquad \forall u,v,w \\
& d(u,v) \ge 0 \qquad \forall \{u,v\} \in \tbinom{V}{2}
\end{aligned} \tag{1}$$

This follows from the fact that any collection of $\binom{V}{2}$ values $d(u,v)$ that satisfy the above conditions correspond to a $|V|$-point metric (by definition), and so the value of the solution to this LP is exactly $\mathrm{LR}(G,H)$. □

The following is immediate.

5.8. Lemma. *Let λ^* be the optimal value of the above LP. We have $\lambda^* = \mathrm{LR}(G,H)$ and we can compute it in polynomial time.*

Now we need to ensure that $\mathrm{LR}(G,H)$ is a good approximation of $\Phi(G,H)$. It is immediate that $\mathrm{LR}(G,H) \le \Phi(G,H)$ because we consider all metrics including all cut metrics. We will now show that $\mathrm{LR}(G,H)$ cannot be too much smaller than $\Phi(G,H)$.

5.4. Showing that LR(G,H) is a Good Relaxation

We will prove that $\mathrm{LR}(G,H) \ge \frac{1}{\Theta(\log n)}\Phi(G,H)$. In other words, we will show that relaxing cut metrics to all metrics may reduce our desired value by at most a $O(\log n)$ factor. Why might we expect this to be true? At a high level, our ingredients say the following:

1. **Bourgain's embedding:** Any metric (X,d) can be "captured" by an ℓ_1-metric with just $O(\log n)$ distortion.
2. **ℓ_1-metrics are in the cut cone:** any ℓ_1-metric is just a combination of cut metrics.

Taken together, these two facts basically suggest that any metric is captured by a combination of cut metrics within a $O(\log n)$ factor. We will now formally prove our desired inequality.

5.9. Lemma. $\mathrm{LR}(G,H) \ge \frac{1}{\Theta(\log n)}\Phi(G,H)$

Proof. Let d^{OPT} be the optimal metric from the linear program 1. Then we may write $\mathrm{LR}(G,H)$ as

$$\mathrm{LR}(G,H) = \frac{\sum_{u,v} w_G(u,v)\cdot d^{\mathrm{OPT}}(u,v)}{\sum_{u,v} w_H(u,v)\cdot d^{\mathrm{OPT}}(u,v)}$$

Then by applying Bourgain's to d^{OPT}, we get an embedding $f : V \mapsto \mathbb{R}^{O(\log^2 n)}$ where for all $u,v \in V$,

$$\left\|f(u)-f(v)\right\|_1 \le d^{\mathrm{OPT}}(u,v) \le O(\log n)\cdot\left\|f(u)-f(v)\right\|_1$$

Then by applying Corollary 4.14 to f, we have

$$\left\|f(u)-f(v)\right\|_1 = \sum_{S\in\mathcal{S}_f} \alpha_S d_S(u,v)$$

where $\mathcal{S}_f$ is a collection of $O(n\log^2 n)$ cuts. So we have

$$\begin{aligned}
\mathrm{LR}(G,H) &\ge \frac{1}{\Theta(\log n)}\cdot\frac{\sum_{u,v} w_G(u,v)\sum_{S\in\mathcal{S}_f}\alpha_S d_S(u,v)}{\sum_{u,v} w_H(u,v)\sum_{S\in\mathcal{S}_f}\alpha_S d_S(u,v)}\\
&= \frac{1}{\Theta(\log n)}\cdot\frac{\sum_{S\in\mathcal{S}_f}\alpha_S(\sum_{u,v} w_G(u,v)d_S(u,v))}{\sum_{S\in\mathcal{S}_f}\alpha_S(\sum_{u,v} w_H(u,v)d_S(u,v))}\\
&\ge \frac{1}{\Theta(\log n)}\cdot\min_{S\in\mathcal{S}_f}\frac{\sum_{u,v} w_G(u,v)d_S(u,v)}{\sum_{u,v} w_H(u,v)d_S(u,v)} \qquad \text{as } \frac{\sum_{i=1}^k a_i}{\sum_{i=1}^k b_i} \ge \min_i \frac{a_i}{b_i}\\
&= \frac{1}{\Theta(\log n)}\cdot\min_{S\in\mathcal{S}_f}\frac{w_G(\partial_G S)}{w_H(\partial_H S)}\\
&= \frac{1}{\Theta(\log n)}\cdot\min_{S\in\mathcal{S}_f}\Phi_{G,H}(S)\\
&\ge \frac{1}{\Theta(\log n)}\cdot\Phi(G,H)
\end{aligned}$$

Therefore we can conclude

$$\mathrm{LR}(G,H) \le \Phi(G,H) \le O(\log n)\cdot\mathrm{LR}(G,H)$$

□

The following theorem is immediate.

5.10. Theorem. *There is a randomized algorithm that $O(\log n)$-approximates $\Phi(G,H)$ in polynomial time. The same holds for $\Phi(G)$.*

5.11. *Remark.* In fact, the only slow step in the above algorithm is to obtain d^{OPT}. All other steps take near linear time.

5.12. Exercise. Given an embedding f of the optimal metric d^{OPT} of the LP, show how to compute a cut S such that $\Phi_{G,H}(S) \le O(\log n)\cdot\Phi(G,H)$ in $\tilde{O}(|E(G)|+|E(H)|)$ time. In the case where we want to compute conductance, the running time is just $\tilde{O}(|E(G)|)$.

6. Limitation and Outlook

This approach gives a $O(\log n)$ approximation to $\Phi(G, H)$, which is tight (you will prove this in the homework). We can obtain better approximations via smaller distortion metric embeddings. Bourgain's embedding is very general in that it can embed any metric into an ℓ_1-metric. However, we can obtain better embeddings by considering a more specific set of metrics, like the set of *negative-type* metrics.

6.1. Definition. A *negative-type* metric (X, d) is such that there exists $f : X \mapsto \mathbb{R}^k$ such that $d(x, y) = \|f(x) - f(y)\|_2^2$.

Any n-point negative-type metric can be embedded into an ℓ_1-metric with distortion $O(\sqrt{\log n} \log \log n)$ [6]. This embedding is similar to Bourgain's, but with better distortion on a smaller class of metrics. A relaxation of $\Phi(G, H)$ to all negative-type metrics can be computed using semidefinite programming (SDP). Following the same argument as the one in this lecture (but more complicated), we get an $O(\sqrt{\log n} \log \log n)$-approximation algorithm for $\Phi(G, H)$. This topic is further exmplored in a survey by Naor on connections between expansion and functional analysis[7].

This approach works for a general notion of edge expansion in undirected graphs. Metric embeddings have also been studied in the context of vertex expansion[8]. Later, we will see a more robust framework based on the "cut-matching game" which works for both **vertex expansion** and **directed graphs**.

7. Exercises

The shortest path metric is universal in the following sense:

7.1. Exercise. Show that any (finite) metric (X, d) is a shortest path metric of a weighted graph $G = (X, E, w)$ where $w(u, v) = d(u, v)$.

7.2. Exercise (More examples of metrics)**.** Convince yourself that the following are metrics:

1. **(ℓ_p metric):** $X = \mathbb{R}^m$ and $d(x, y) = \|x - y\|_p = (\sum_i (x_i - y_i)^p)^{1/p}$.
   - For $p = 2$, $\|x - y\|_2 = \sqrt{\sum_i (x_i - y_i)^2}$ is just a Euclidean distance.
   - For $p = \infty$, $\|x - y\|_\infty = \max_i \{|x_i - y_i|\}$
2. **(uniform metric):** Let X be any set and $d(x, y) = 1$ for all $x \neq y$.
3. **(1-2 metric):** Let X be any set and $d(x, y) \in \{1, 2\}$ for all $x \neq y$.
4. **(edit distance):** Let $X = \{0, 1\}^*$ be a set of strings. Let $d(x, y)$ be the edit distance between x and y.

The earlier discussion about ℓ_1-metrics and cut metrics gives the following characterization:

7.3. Exercise. Let $L1_n$ contain all n-point ℓ_1-metrics. Prove that $L1_n = \text{CUTCONE}_n$.

[6] Sanjeev Arora, James Lee, and Assaf Naor. Euclidean distortion and the sparsest cut

[7] Assaf Naor. L1 embeddings of the heisenberg group and fast estimation of graph isoperimetry, 2010.

[8] https://homes.cs.washington.edu/ jrl/papers/pdf/fhl-sicomp.pdf

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 4: Bourgain's Theorem

September 9, 2021

Instructor: Thatchaphol Saranurak

Scribe: Cheng-Hao Fu

1. Briefly Motivate and State the Theorem

In general, an arbitrary distance metric (X, d) can be difficult to work with directly. As seen in the algorithm for computing conductance, it is far easier to work with ℓ_1 metric. In this pursuit, finding embeddings of (X, d) into adequately small-dimension $\mathbb{R}^n$ quickly is a gold-standard of computation.

Bourgain's Theorem gives us a guarantee that we can find a low distortion embedding into low-dimension Euclidean space:

1.1. Theorem (Bourgain)**.** *Given an n-point metric (X, d), there is an embedding $f : X \to \mathbb{R}^{O(\log^2 n)}$ such that, for any $x, y \in X$,*

$$\|f(x) - f(y)\|_1 \leq d(x, y) \leq O(\log n) \cdot \|f(x) - f(y)\|_1.$$

Moreover, given d, in $\tilde{O}(n^2)$ time, the mapping f can be computed correctly with probability at least $1-1/n$. If (X, d) is a shortest path metric of an m-edge graph, then the running time is $\tilde{O}(m)$ time.

Note that our proof will show an embedding f where

$$\Omega(\log n) \cdot d(x, y) \leq \|f(x) - f(y)\|_1 \leq O(\log^2 n) \cdot d(x, y).$$

And after scaling $\bar{f} = \frac{f}{(\log n)^2}$, $\bar{f}$ satisfies the desired inequality in the theorem.

The usefulness of this theorem is clear. For example, in the case of graphs with n vertices, the theorem implies the existence of an assignment of $O(\log^2 n)$ numbers (representing the dimension of Euclidean space) to each node that allows distance calculation on the graph to be done by comparing the labels element-wise with a small amount of distortion. This is much better than the $O(n^2)$ guarantee that Dijkstra's gives on calculating Single Source Shortest Path.

2. The Algorithm

The algorithm itself is quite simple. We first define a notion of the distance of any point to a particular set of points $A \subseteq X$ as follows:

2.1. Definition. The distance between a point x and a set of points A with respect to the distance metric is:

$$d(x, A) := \min_{a \in A} d(x, a).$$

We then define A_{ij} for $i = 1, \ldots, i_{\max} = 10000 \log n$ and for $j = 1, \ldots, j_{\max} = \lceil \log n \rceil$ by letting $A_{ij} \subseteq X$ include each element of X with probability $1/2^j$. We sample all A_{ij} independently.

We then define our embedding as follows:

$$f : X \to \mathbb{R}^{i_{\max} j_{\max}} \text{ where } f_{ij}(x) = d(x, A_{ij})$$

We note that every element in the space now has an associated vector of labels that is of length $O(\log^2 n)$, representing its location in $\mathbb{R}^{i_{\max} j_{\max}}$.

It is then clear that if this embedding achieves the desired distortion, along with the runtime, we will be done.

3. Proving Runtime

3.1. Claim. *Given d, the embedding f can be computed in $\tilde{O}(n^2)$ time. If (X, d) is a shortest path metric defined by an m-graph, then f can be computed in $\tilde{O}(m)$ time.*

Proof of Claim. First of all, it takes $\tilde{O}(n)$ time to sample all A_{ij}. To compute each entry of $f(x)$

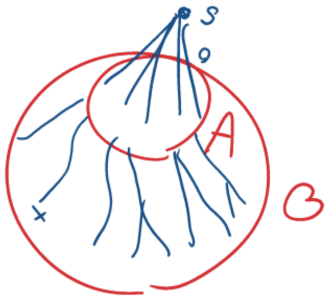
for all $x \in X$, when (X, d) is a shortest path metric, instead of fixing a point in X and compute the distance between itself and all possible A_{ij}, we switch gear by fixing a A_{ij} and computing its distance between all $x \in X$. To do this, for a fixed A_{ij}, we assign a source s that connects to each point in A_{ij} with distance 0, and running Dijkstra's (which takes $\tilde{O}(m)$ time) to compute the shortest path between any point x and s, which is exactly the distance between x and A_{ij}. Since there are $O(\log n^2)$ A_{ij}s that we have to do this with, computing the embedding will take $\tilde{O}(m)$ time, as desired.

When (X, d) is a general metric, the key observation is that we can treat any general metric as the shortest path metric on the weighted complete graph $G_d(X, E, \kappa)$ where $\forall \{x, y\} \in E, \kappa(\{x, y\}) = d(x, y)$. In this way, we could use the similar strategy as we described above to compute f in $\tilde{O}(n^2)$ time since G_d has $O(n^2)$ edges. It remains to show that d is indeed the shortest path metric on G_d. $\forall \{x, y\} \subseteq V$, on one hand, $\text{dist}_{G_d}(x, y) \leq d(x, y)$ since there exists the path $< x, y >$ with weight $d(x, y)$ between x and y; on the other hand, $\text{dist}_{G_d}(x, y) \geq d(x, y)$ since d satisfies

triangle inequality whence by an inductive argument we can see that $\forall P \in \mathcal{P}_{x,y}, d(P) \geq d(x,y)$, and therefore the distance of the shortest $(x,y)-$path is also greater than $d(x,y)$.

□

For an illustration of the process:



4. Analysis of Algorithm

Now, the goal is to prove $\Omega(\log n) \cdot d(x,y) \leq \|f(x) - f(y)\|_1 \leq O(\log^2 n) \cdot d(x,y)$. We begin with the upper bound.

4.1. Upper Bound

4.1. Claim. $\|f(x) - f(y)\|_1 \leq O(\log^2 n) \cdot d(x,y)$.

Proof of Claim. We start by breaking down the definition of $\|f(x) - f(y)\|_1$. It is simply the sum of the magnitudes of the element-wise differences of $f(x)$ and $f(y)$. We can then use the definition of f to arrive at:

$$\|f(x) - f(y)\|_1 = \sum_{ij} |f_{ij}(x) - f_{ij}(y)| = \sum_{ij} |d(x, A_{ij}) - d(y, A_{ij})|$$

We know that there are only $O(\log^2 n)$ elements in the final summation, which is exactly the multiplier of $d(x,y)$ in the desired statement. This tells us that proving $|d(x, A_{ij}) - d(y, A_{ij})| \leq d(x,y)$ is sufficient for all i, j.

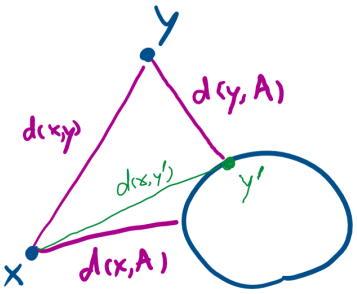
Without loss of generality, assume that $d(x, A_{ij}) \geq d(y, A_{ij})$. It suffices to prove that $d(x, A_{ij}) \leq d(x,y) + d(y, A_{ij})$

Let y' be such that $d(y, A_{ij}) = d(y, y')$.

By definition, we have that $d(x, A_{ij}) \leq d(x, y')$. Then, we get $d(x, y') \leq d(x,y) + d(y, y') = d(x,y) + d(y, A_{ij})$ by triangle inequality. Note that this actually works for any distance metric (X, d) with $A \subseteq X$.

□

Visually:



4.2. Lower Bound

4.2. Claim. $\|f(x) - f(y)\|_1 \geq \Omega(\log n) \cdot d(x, y)$.

Proof of Claim. We begin by defining a couple of useful objects.

4.3. Definition. For any point s and radius r, define $B(s, r) = \{z | d(z, s) \leq r\}$ to be the ball radius r around s. Similarly, $B^o(s, r) = \{z | d(z, s) < r\}$ will be referred to as the open ball around s with radius r.

We then fix $x, y \in X$, and define a couple more things.

Let $r_0 = 0$. For $j = 1, \ldots, j_{\max} = \lceil \log n \rceil$, r_j be the minimum r where both $|B(x, r)|, |B(y, r)| \geq 2^j$.

Define $\Delta_j = r_j - r_{j-1}$. Think of this as the amount that you have to grow the ball in order for the ball to encompass the next magnitude in size. Notice that, by telescoping, $\sum_{j=1}^{j_{\max}} \Delta_j = r_{j_{\max}}$.

We start off by assuming $r_{j_{\max}} < d(x, y)/4$ for now. Combinatorially, this means that the radius that we are considering is significantly less than the distance between them. This, of course, implies that $B(x, r_{j_{\max}})$ and $B(y, r_{j_{\max}})$ are disjoint. Note that there is no reason to expect that this assumption will hold. We will relax the assumption later in the proof.

Without loss of generality, assume that $|B(x, r_j)| \geq |B(y, r_j)|$. This means that y was the point that required r_j to be as large as it was, in fact, we can say then that $|B(y, r_j)| = 2^j$. If it is the case that multiple distances that are the same cause this assumption to be false, we can perturb each of the distances by a small constant ϵ, which will not cause any significant distortion, and give us the equality we are looking for. Thus, we can say that, $|B^o(y, r_j)| < 2^j$. Also, trivially $|B(x, r_{j-1})| \geq 2^{j-1}$.

4.4. Definition. We say a set A_{ij} is $good$ if $A_{ij} \cap B(x, r_{j-1}) \neq \emptyset$ and $A_{ij} \cap B(y, r_{j-1}) = \emptyset$

We use the term $good$ because it will help us definitively add to the minimum distance between the two points. In fact, we can observe that it will add at minimum Δ_j to $\|f(x) - f(y)\|_1$. This is because $d(y, A_{ij}) \geq r_j$ but $d(x, A_{ij}) \leq r_{j-1}$.

This is extremely powerful. To see why, let us see what happens when all A_{ij} are good. Then we would have

$$\|f(x) - f(y)\| = \sum_{ij} |d(x, A_{ij}) - d(y, A_{ij})| \geq \sum_{ij} \Delta_j = i_{\max} \cdot r_{j_{\max}} = \Omega(\log n) \cdot d(x, y)$$

The first equality comes from the definition of f. The second inequality comes from the justification provided above, and the rest is by definition (Noting that our assumption has $r_{j_{\max}} \approx d(x, y)/4$). This is exactly the result we are looking for, but unfortunately not all A_{ij} are good. But it should be clear that since we are aiming for $\Omega(\log n)$, a constant fraction of A_{ij} being good for each j will be sufficient for the bound. This is what we will next attempt to prove.

4.5. Proposition. *For every i, j, A_{ij} is good with probability at least $1/100$.*

Proof of Proposition. First, we calculate a lower bound the probability that $A_{ij} \cap B^o(y, r_j) = \emptyset$. Because of the way that we picked all of the points in A_{ij}, we have that the probability that every element in the open ball is avoided by A_{ij} is:

$$(1 - 1/2^j)^{|B^o(y, r_j)|} > (1 - 1/2^j)^{2^j} \geq 1/10.$$

The second inequality can be verified via graphing.

Similarly, $A_{ij} \cap B(x, r_{j-1}) = \emptyset$ occurs with probability:

$$(1 - 1/2^j)^{|B(x, r_{j-1})|} \leq (1 - 1/2^j)^{2^{j-1}} \leq 9/10.$$

Thus, $A_{ij} \cap B(x, r_{j-1}) \neq \emptyset$ occurs also with probability at least $1/10$. Because of our assumption that these two balls have an empty intersection, these two events are independent, and thus we can multiply their probabilities together to get a lower bound on the probability that A_{ij} is good:

$$\Pr[A_{ij} \cap B(x, r_{j-1}) \neq \emptyset \text{ and } A_{ij} \cap B^o(y, r_j) = \emptyset] \geq 1/10 \times 1/10 = 1/100$$

□

From the above, for each j, the expected total number of good sets A_{ij} over all i is at least $i_{\max}/100$.

However, before we proceed with the final part of the proof, we need to show that this occurs with high enough probability to get the high probability argument that we also want. This motivates the need for the following claim:

4.6. Claim. *For each j, the total number of good A_{ij} over all i is at least $i_{\max}/200$ with probability at least $1 - 1/n^{10}$.*

Proof of Claim. We will simplify this to the question of proving an upper bound on the probability of $i_{max}/200$ or fewer successes in i_{max} trials of Bernoulli random variables with success probability $1/100$. Let us define the random variables $X_1, .., X_{i_{max}}$ to represent the trials, and $Y = \sum_i X_i$.

Note that with Chernoff Bound, we have:

$$\Pr[Y \leq i_{max}(1/100 - 1/200)] \leq e^{-(1/8)(1/100)i_{max}}$$

Plugging in i_{max} with $10000 \log n$:

$$\Pr[Y \leq i_{max}/200] \leq e^{-(1/8)(1/100)10000 \log n} \leq 1/n^{10}$$

□

This gives us the following corollary:

4.7. Corollary. *For probability at least* $1 - 1/n^{10}$, $\|f(x) - f(y)\| \geq \frac{i_{\max}}{200} \cdot r_{j_{\max}} = \Omega(\log n) \cdot d(x, y)$.

Proof. This follows pretty much exactly the same as the argument used at the top of page 5. The only difference is that we can only sum over the good sets, which makes us lose a constant factor, which is still good enough for us.

$$\|f(x) - f(y)\| \geq \sum_{ij:A_{ij} \text{ is good}} |d(x, A_{ij}) - d(y, A_{ij})| \geq \sum_{ij:A_{ij} \text{ is good}} \Delta_j \geq \frac{i_{\max}}{200} \cdot r_{j_{\max}}$$

The last inequality holds with the probability desired, and so we are done. □

This corollary gives us the tools to prove the lower bound, however, we have to keep in mind that we are still operating under the assumption that $r_{j_{\max}} < d(x, y)/4$. In order to lift this assumption, we redefine $r'_j = \min\{r_j, d(x, y)/4\}$ and work with r'_j instead. By noticing that r'_j is a lower bound on r_j, the above analysis can still go through.

Now that we have addressed this point, we can finish the proof:

Since each distance $d(x, y)$ fails in being mapped to within the bounds with $1/n^{10}$ probability, we can use union bound to see that over all pairs, the probability of failure is less than or equal to:

$$\binom{n}{2}/n^{10} \leq 1/n^8$$

This allows us to say: For probability at least $1 - 1/n^8$, $\|f(x) - f(y)\| = \Omega(\log n) \cdot d(x, y)$ for all $x, y \in X$, which proves the original claim. □

Thus completes the proof of Bourgain's Theorem.

5. Examples

To motivate the way we have built the framework for the theorem, we will begin with a very basic attempt at embedding a distance into a ℓ_1 metric, and justifying why our approach needs to be as complicated as it is. This will be primarily done by testing ideas on shortest path metrics on various types of graphs. In addition to the discussion section, a large part of this section will be supported by the lecture notes of Luca Trevisan of UC-Berkeley[1].

5.1. Cycles

A natural precursor to the method illustrated in Bourgain's Theorem picks a point uniformly at random, and takes the embedding as labeling any point a with some constant factor times the distance between a and that point.

$$f(v) = k \times d(v, a)$$

Distances are then, naturally, the differences between these labels. The fact that this is a metric follows trivially. This is effective on cycles. We start by proving the following fact:

5.1. Claim. *Let a be a random node. $E[|d(x,a) - d(y,a)|] = \Theta(d(x,y))$.*

Proof of Claim. By the same argument used in the proof for 4.1, it is clear that $E[|d(x,a) - d(y,a)|] = O(d(x,y))$. Thus, we just have to justify that the expectation is not too small.

Consider the expected value of the distance when a is on the path between x and y. It is more or less uniformly distributed between 0 and $d(x,y)$ (depending on parity), with the low when the random point is the midpoint, and the high being when the point is one of the endpoints. This occurs with probability approximately equal to $d(x,y)/n$

Thus:

$$E[|d(x,a) - d(y,a)|] \approx d(x,y)^2/(2n) + \alpha$$

Where α represents the contribution in the other case. We also notice that the embedded distance is ranged from $d(x,y)$ to 0, with the high occurring whenever the paths representing x to a and y to a overlap, and the low happening whenever the random point is exactly the same distance to both x, y. This means that α contributes roughly (slightly more than) $d(x,y)/2$ on average per point. This clearly tells us that $E[|d(x,a) - d(y,a)|] = \Omega(d(x,y))$.

Thus finishes the proof. □

In much the same fashion as classic Bourgain's, we can then use a logarithmic number of f as described above, to get the desired probability for the accuracy of our bound.

[1] https://lucatrevisan.github.io/teaching/expanders2016/lecture10.pdf

5.1.1. Not Quite Enough

As is clear upon reading the proofs, this method is much easier. However, we will next see that it is not an effective way to embed an arbitrary metric.

Suppose that we have a metric that has distances between any two points be either 1 or 2. It is 2 if and only if z is one of the endpoints. All other distances are 1.

Looking at $E[|d(r,z) - d(r,v)|]$, reveals:

$$E[|d(r,z) - d(r,v)|]1 + 2/n$$

.
This means that the distortion has the potential to be linear, which is a problem. There are many steps between this simple argument with the proof discussed above, but hopefully this gives sufficient information to inform a little bit of the background of Bourgain's.

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 5: Approximate Max-flow Min-cut Theorem

September 14, 2021

Instructor: Thatchaphol Saranurak

Scribe: Yaowei Long

1. Overview and Preliminaries

We are going to prove a generalization of the max-flow min-cut theorem. Roughly speaking, for two undirected weighted graph G and H, we will show that the H-expansion of G and the maximum H-concurrent flow on G are approximately the same. In details, the gap between them is bounded by a $O(\log n)$ factor, and this factor is tight. Moreover, we will learn two partial orders based on cuts and concurrent flows respectively. The approximate max-flow min-cut theorem implies that these two orders are essentially the same.

The proof of the approximate max-flow min-cut theorem is based on the relationship between the H-expansion of G and its cut-metrics-to-all-metrics relaxation (Leighton-Rao relaxation).

1.1. Definition (H-expansion $\Phi(G,H)$)**.** Let G, H be two undirected weighted graphs on the same vertex set with edge weight functions ω_G, ω_H. The H-expansion of G is defined as

$$\Phi(G,H) = \min_S \frac{w_G(\partial_G S)}{w_H(\partial_H S)} = \min_{d_S:\text{cut metric over } V} \frac{\sum_{u,v} w_G(u,v)\cdot d_S(u,v)}{\sum_{u,v} w_H(u,v)\cdot d_S(u,v)}.$$

The latter equation follows the property of the cut metric, i.e. $\omega(\partial S) = \sum_{u,v} \omega(u,v)\cdot d_S(u,v)$.

1.2. Definition (LR relaxation)**.** Let G, H be graphs as in definition 1.1, the Leighton-Rao relaxation of $\Phi(G,H)$ is defined as

$$\text{LR}(G,H) = \min_{d:\text{metric over } V} \frac{\sum_{u,v} w_G(u,v)\cdot d(u,v)}{\sum_{u,v} w_H(u,v)\cdot d(u,v)}$$

$\text{LR}(G,H)$ is also equal to the optimal solution of the following LP.

$$\begin{aligned} &\text{(LR)} \\ &\min \sum_{u,v} w_G(u,v)\cdot d(u,v) \end{aligned} \tag{1}$$

$$\text{s.t.} \sum_{u,v} w_H(u,v) \cdot d(u,v) = 1$$
$$d(u,v) \le d(u,w) + d(w,v) \qquad \forall u,v,w$$
$$d(u,v) \ge 0 \qquad \forall \{u,v\} \in \binom{V}{2}$$

1.3. Theorem. *Let G, H be undirected weighted graphs.*

$$\mathrm{LR}(G,H) \le \Phi(G,H) \le O(\log n) \cdot \mathrm{LR}(G,H)$$

The detailed proof of theorem 1.3 is shown in lecture 2-1.

2. Concurrent Flow

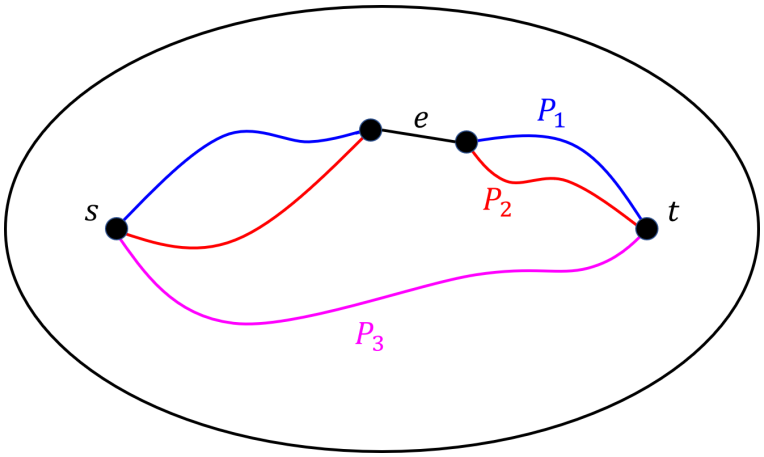
Concurrent flows are also called multi-commodity flows, which will route on an undirected weighted graph different types of commodities each of which has independent source, sink and demands.

The single commodity flow is the special one with just one type of commodity and single source and sink.

2.1. Definition ((s,t)-flows (single commodity flows))**.** Consider a graph $G = (V, E, \kappa)$ with edge capacity $\kappa : E \to \mathbb{R}_{\ge 0}$. Let s, t be the source and sink and let $\mathcal{P}_{st}$ be the set of all (s,t)-paths in G. An (s,t)**-flow** f is simply a function that assigns non-negative values to paths in $\mathcal{P}_{st}$. That is, $f : \mathcal{P}_{st} \to \mathbb{R}_{\ge 0}$.

- The **value** of f is $\|f\| = \sum_P f(P)$, i.e. total amount of flow sent through (s,t)-paths.
- The **congestion** of flow f on edge $e = (u,v)$ is the total amount of flow on e relative to the capacity of e:
$$\mathsf{cong}_f(e) = \frac{\sum_{P \ni e} f(P)}{\kappa(e)}.$$
- The **congestion of the flow** is the maximum congestion over all edges:
$$\mathsf{cong}_f = \max_{e \in E} \mathsf{cong}_f(e).$$

See figure 1 for an example.



1. ábra. In this example, let s and t denote the source and the sink respectively. There are three path P_1, P_2, P_3 connecting s and t. Let's say the (s,t)-flow f is defined by $f(P_1) = 10, f(P_2) = 3, f(P_3) = 2$. The value of f is $\|f\| = f(P_1) + f(P_2) + f(P_3) = 15$ and the congestion on edge e is $\mathsf{cong}_f(e) = (f(P_1) + f(P_2))/\kappa(e) = \frac{13}{\kappa(e)}$.

Actually, the definition of single commodity flows is equivalent to the classical definition of single-source and single-sink flows. In the classical definition, an (s,t)-flow $f : E \to \mathbb{R}_{\geq 0}$ assigns value on **edges** (not paths) such that for every node $u \neq s,t$, the total flow coming in to u equals the total flow going out of u (i.e. $\sum_v f(v,u) = \sum_v f(u,v)$). However, given any (s,t)-flow in the sense of this definition, we can always decompose it into a collection of (s,t)-paths and cycles. Now, by removing all cycles that does not contribute anything to the value of the flow. We obtain precisely the flow in the definition we are working with here.

2.2. Definition (Concurrent flows (multi-commodity flows))**.** Consider two graphs $G = (V, E_G, \kappa_G)$ and $H = (V, E_H, \kappa_H)$ with nonnegative edge capacities. An H**-concurrent flow** $\mathcal{F}$ is a collection of (s,t)-flows of value $\kappa_H(s,t)$ for all $(s,t) \in E(H)$:

$$\mathcal{F} = \{f_{(s,t)} \mid \|f_{(s,t)}\| = \kappa_H(s,t)\}_{(s,t)\in E(H)}.$$

$\mathcal{F}$ is also called a **flow-embedding** and we say $\mathcal{F}$ **embeds** H **into** G.

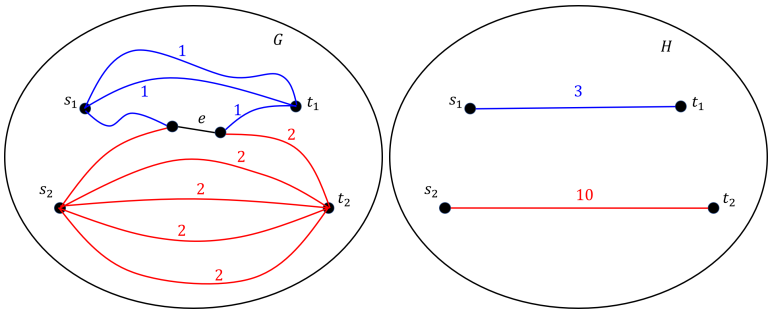
The **congestion of multi-commodity flow** $\mathcal{F}$ **on** e is

$$\mathsf{cong}_{\mathcal{F}}(e) = \sum_{(s,t)\in E(H)} \mathsf{cong}_{f_{(s,t)}}(e).$$

and **the congestion of** $\mathcal{F}$ is

$$\mathsf{cong}_{\mathcal{F}} = \max_{e\in E} \mathsf{cong}_{\mathcal{F}}(e).$$

If congestion $\mathsf{cong}_{\mathcal{F}} \leq 1$, we say that $\mathcal{F}$ has **no congestion** or $\mathcal{F}$ is **feasible**.



2. ábra. In this example, let $\mathcal{F}$ be the H-concurrent flow on G. We can see that $\mathcal{F}$ embeds blue edge (s_1, t_1) in H into three blue paths in G and embeds red edge (s_2, t_2) in H into five red paths in G. The congestion of edge e in G is $\mathsf{cong}_{\mathcal{F}}(e) = 1/\kappa(e) + 2/\kappa(e) = 3/\kappa(e)$.

Intuitively, H is the **demand graph** of $\mathcal{F}$ because each edge in H represents a unique type of commodity with its own source, sink and demand. Note that concurrent flows are not the same as the multiple-source multiple-sink flows, where the latter essentially has only one type of commodity but multiple sources and sinks.

3. Partial Order based on Concurrent Flow

3.1. Definition (Partial order based on concurrent flow)**.** Consider two undirected weighted graphs G and H.

We define $H \preccurlyeq^{\text{flow}} G$ if there is a feasible H-concurrent flow $\mathcal{F}$ in G. In this case, we say that **the demand** H **is routable in** G or H **is embeddable into** G. Notice that this defines a partial order between graphs.

We also say that **the demand** H **is routable in** G **with congestion** q or H **is embeddable into** G **with congestion** q if $\mathcal{F}$ has congestion q.

3.2. Fact. *For any graph G and G',*

- *let $G'' = G + G'$ be such that $\kappa_{G''}(u, v) = \kappa_G(u, v) + \kappa_{G'}(u, v)$ for all u, v.*
- *let $G' = \alpha \cdot G$ be such that $\kappa_{G'}(u, v) = \alpha \cdot \kappa_G(u, v)$ for all u, v.*

From the definition, H is embeddable into G with congestion q iff $H \preccurlyeq^{\text{flow}} q \cdot G$.
The partial order also has following properties.

- $G \preccurlyeq^{\text{flow}} G$
- *If $G_1 \preccurlyeq^{\text{flow}} G_2$ and $G_2 \preccurlyeq^{\text{flow}} G_3$, then $G_1 \preccurlyeq^{\text{flow}} G_3$.*
- $G_1 \preccurlyeq^{\text{flow}} G_2$ *iff* $G_1 + H \preccurlyeq^{\text{cut}} G_2 + H$.
- $G_1 \preccurlyeq^{\text{flow}} \alpha \cdot G_2$ *iff* $\frac{1}{\alpha} G_1 \preccurlyeq^{\text{flow}} G_2$.

4. The Maximum Concurrent Flow Problem

4.1. Definition (**Maximum concurrent flow** problem)**.** Given a graph G and a demand graph H, we need to compute a feasible $\tau \cdot H$-concurrent flow in G with maximum τ. We denote the optimal value τ^* by $\mathsf{mcf}(G, H) = \max\{\tau \mid \tau \cdot H \preccurlyeq^{\text{flow}} G\}$. In particular, we have $H \preccurlyeq^{\text{flow}} G \iff \mathsf{mcf}(G, H) \geq 1$.

Note that this problem is a generalization of the single-source single-sink maximum flow problem. Let H_{st} be the graph containing only one edge (s, t) with capacity 1. $\mathsf{mcf}(G, H)$ is exactly the value of maximum (s, t)-flow and $\Phi(G, H)$ is the value of minimum (s, t)-cut. The classical max-flow min-cut theorem states the following.

4.2. Theorem (Max-flow min-cut theorem)**.** *For any graph G where $s, t \in V(G)$, we have* $\mathsf{mcf}(G, H_{st}) = \Phi(G, H_{st})$.

5. An Approximate Max-flow Min-cut theorem

5.1. Theorem (Approximate Maxflow Mincut Theorem)**.** *Given undirected graphs G and H,*

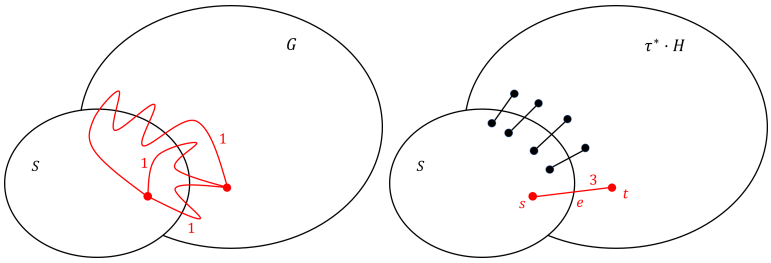
$$\mathsf{mcf}(G, H) \leq \Phi(G, H) \leq O(\log n) \cdot \mathsf{mcf}(G, H).$$

proof of one direction of theorem 5.1, $\mathsf{mcf}(G, H) \leq \Phi(G, H)$.

- Let $\tau^* = \mathsf{mcf}(G, H)$ be the optimal value of the concurrent flow. Let $\mathcal{F}$ be the multi-commodity flow that embeds $\tau^* H$ to G.
- Consider any cut $S \subset V$.
- For each edge $(s, t) \in \partial_H S$, $\mathcal{F}$ sends at least $\tau^* \cdot \kappa_H(s, t)$ unit of flow crossing the cut S.
- So the total unit of flow crossing the cut S is $\tau^* \cdot \kappa_H(\partial_H S)$.
- But the total capacity of the cut is $\kappa_G(\partial_G S)$. So

  $$\tau^* \cdot \kappa_H(\partial_H S) \leq \kappa_G(\partial_G S).$$

  Formally, let $\mathcal{F} := \{f_{s,t}\}_{(s,t)\in E(H)}$ be the feasible flow that embeds $\tau^* \cdot H$ into G. $\mathsf{cong}_{\mathcal{F}} \leq 1$ implies that $\forall e \in E(G), \sum_{(s,t)\in H} \sum_{P \ni e} f_{s,t}(P) \leq \kappa_G(e)$. Since $LHS = \sum_{(s,t)\in\partial_H S} \sum_{P\ni e} f_{s,t}(P) \leq \sum_{(s,t)\in H} \sum_{P\ni e} f_{s,t}(P) \leq \kappa_G(e) = RHS$, summing over all edges in $\partial_G S$ for both sides, we have $LHS \geq \sum_{(s,t)\in\partial_H S} \sum_{P\cap\partial_G S\neq\emptyset} f_{s,t}(P) = \sum_{(s,t)\in\partial_H S} \sum_{P\in\mathcal{P}_{s,t}} f_{s,t}(P) = \sum_{(s,t)\in\partial_H S} \tau^* \cdot \kappa_H(s, t) = \tau^* \cdot \kappa_H(\partial_H S)$; $RHS = \kappa_G(\partial_G S)$. $\implies \tau^* \cdot \kappa_H(\partial_H S) \leq \kappa_G(\partial_G S)$.
  See figure 3 for an intuitive proof.



3. ábra. Let the left figure represent G and the right figure represent $\tau^* \cdot H$. The capacity of the cut S in $\tau^* \cdot H$ is $\tau^* \cdot \kappa_H(\partial_H S)$, represented by five edges in the right figure. For each edge in $\tau^* \cdot H$ crossing the cut, it is embedded into several paths in G and each path will cross the cut in G at least on time. Take the red edge e with $\kappa_H(e) = 3$ as an example. It is embedded into three paths each of which has flow 1, which means that at least 3 units of capacity of the cut in G are owned by e. It turns out that $\tau^* \cdot \kappa_H(\partial_H S) \leq \kappa_G(\partial_G S)$.

- Therefore, we have

$$\mathsf{mcf}(G, H) = \tau^* \leq \min_S \frac{\kappa_G(\partial_G S)}{\kappa_H(\partial_H S)} = \Phi(G, H)$$

□

We prove another direction of theorem 5.1 (i.e. $\Phi(G, H) \leq O(\log n) \cdot \mathsf{mcf}(G, H)$) via LP duality. In fact, we will show that the optimal value of concurrent flow $\mathsf{mcf}(G, H)$ is exactly the same as the optimal value from the Leighton-Rao relaxation $\mathrm{LR}(G, H)$ of $\Phi(G, H)$ (see LP 1), as lemma 5.2 says. By lemma 5.2 and theorem 1.3, we immediately get

$$\Phi(G, H) \leq O(\log n) \cdot \mathrm{LR}(G, H) = (\log n) \cdot \mathsf{mcf}(G, H).$$

5.2. Lemma (Key lemma)**.** $\mathsf{mcf}(G, H) = \mathrm{LR}(G, H)$.

proof of one direction of theorem 5.1, $\Phi(G, H) \leq O(\log n) \cdot \mathsf{mcf}(G, H)$. Because $\mathsf{mcf}(G, H) = \mathrm{LR}(G, H)$ by lemma 5.2 and $\Phi(G, H) \leq O(\log n) \cdot \mathrm{LR}(G, H)$ by theorem 1.3, we immediately get $\Phi(G, H) \leq O(\log n) \cdot \mathsf{mcf}(G, H)$.

□

6. Proof of Lemma 5.2

We start by modeling the maximum concurrent flow problem as an LP.

- The problem is fully specified given **parameters** $\{C_{uv}\}_{(u,v)\in\binom{V}{2}}$ and $\{D_{uv}\}_{(u,v)\in\binom{V}{2}}$ defined as follows.
  - Let $C_{uv} = \kappa_G(u, v)$ be the capacity of edge (u, v) in G. If (u, v) is not in G, then $C_{uv} = 0$.
  - Let $D_{uv} = \kappa_H(u, v)$ be the demand of from u to v defined by H. If (u, v) is not in H, then $D_{uv} = 0$.

- Recall definition 4.1 of the maximum concurrent flow problem and definition 2.2 of concurrent flows. The **variables** of this LP are the scaling factor τ and flow assignments $f(P)$ to all (u, v)-paths for all u, v.

$$
\begin{aligned}
& \text{(Primal } \mathsf{P}_{LP}\text{)} \\
\max \quad & \tau \\
\text{s.t.} \quad & D_{uv} \cdot \tau - \sum_{P \in \mathcal{P}_{u,v}} f(P) \leq 0 && \forall u, v \\
& \sum_{P \ni (u,v)} f(P) \leq C_{uv} && \forall u, v \\
& f(P) \geq 0 && \forall P \in \mathcal{P}_{uv} \text{ over all } u, v
\end{aligned}
$$

We let $y(u, v)$ be variables in the dual corresponding to the first type of constraints in the primal LP and $x(u, v)$ be variables corresponding to the second type of constraints in the primal LP. We immediately get the dual LP as follows.

$$
\begin{aligned}
& \text{(Dual } \mathsf{D}_{LP}\text{)} \\
\min \quad & \sum_{u,v} C_{uv} \cdot x(u, v) \\
\text{s.t.} \quad & \sum_{u,v} D_{uv} \cdot y(u, v) \geq 1 \\
& -y(u, v) + \sum_{(u',v') \in P} x(u', v') \geq 0 && \forall P \in \mathcal{P}_{uv} \text{ over all } u, v \\
& x(u, v), y(u, v) \geq 0 && \forall u, v
\end{aligned}
$$

6.1. *Remark.* Deriving dual LPs is a very important skill for researching in TCS. To interpret the problem that D_{LP} is solving, we can treat $x(u, v)$ as "the length of edge uv", C_{uv} as "the capacity or width of the edge uv", and $y(u, v)$ as the "distance between u and v" since it plays the role of a lower bound of the length of paths between u and v. The objective function is trying to minimize the "total volume" of the graph, yet still guarantee the distances between vertices are big enough.

By definition, $\mathsf{mcf}(G, H)$ is exactly the optimal value of P_{LP}, and by strong duality, we have $\text{OPT}(\mathsf{P}_{LP}) = \text{OPT}(\mathsf{D}_{LP})$, which means $\mathsf{mcf}(G, H) = \text{OPT}(\mathsf{D}_{LP})$. In what follows, we will show that $\text{OPT}(\mathsf{D}_{LP}) = \text{OPT}(\text{LR})$, where LR is LP 1 in definition 1.2 (see below for an equivalent version).

$$
\begin{aligned}
& \text{(LR)} \\
& \min \sum_{u,v} C_{uv} \cdot d(u, v) \\
& \text{s.t.} \sum_{u,v} D_{uv} \cdot d(u, v) \geq 1 \\
& \quad d(u, v) \leq d(u, w) + d(w, v) && \forall u, v, w
\end{aligned}
$$

$$d(u,v) \geq 0 \qquad \forall u, v$$

By lemma 6.3 and definition 1.2, we immediately have

$$\mathsf{mcf}(G,H) = \mathrm{OPT}(\mathsf{D}_{LP}) = \mathrm{OPT}(\mathrm{LR}) = \mathrm{LR}(G,H).$$

6.2. Fact. *Let* $\mathrm{dist}_x(u,v)$ *be the distance metric defined by* x*. We have* $y(u,v) \leq \mathrm{dist}_x(u,v) \leq x(u,v)$ *for all* u, v

6.3. Lemma. $\mathrm{OPT}(\mathrm{LR}) = \mathrm{OPT}(\mathsf{D}_{LP})$.

Proof of $\mathrm{OPT}(\mathrm{LR}) \leq \mathrm{OPT}(\mathsf{D}_{LP})$.

Let (x, y) be an optimal solution for D_{LP}. Let $d(u,v) = \mathrm{dist}_x(u,v)$ for all u, v. We have $d(u,v)$ also a feasible solution for LR because

- $\sum_{u,v} D_{uv} \cdot d(u,v) \geq \sum_{u,v} D_{uv} \cdot y(u,v) \geq 1$ and
- the triangle inequality is trivially satisfied as d is a metric.

Furthermore,

$$\mathrm{OPT}(\mathrm{LR}) \leq \sum_{u,v} C_{uv} \cdot d(u,v) \leq \sum_{u,v} C_{uv} \cdot x(u,v) = \mathrm{OPT}(\mathsf{D}_{LP})$$

□

Proof of $\mathrm{OPT}(\mathrm{LR}) \geq \mathrm{OPT}(\mathsf{D}_{LP})$.

For convenience, we write $d(P) = \sum_{(u',v') \in P} d(u', v')$, and the constraint in D_{LP} can be rewritten as $x(P) \geq y(u,v)\ \forall P \in \mathcal{P}_{uv}$.

Obviously, for any function $d : V \times V \to \mathbb{R}_{\geq 0}$, if $d(u,v) \leq d(u,w) + d(w,v)$ for all u, v, w (i.e. d satisfies the triangle inequality), then for all u, v and $P \in \mathcal{P}_{uv}$, we have $d(u,v) \leq d(P)$.

Let d be an optimal solution for LR. Let $x(u,v) = y(u,v) = d(u,v)$ for all u, v. We have

- $\sum_{u,v} D_{uv} \cdot y(u,v) = \sum_{u,v} D_{uv} \cdot d(u,v) \geq 1$ and
- $y(u,v) = d(u,v) \leq d(P) = x(P)$ for all u, v and $P \in \mathcal{P}_{uv}$.

Furthermore,

$$\mathrm{OPT}(\mathsf{D}_{LP}) \leq \sum_{u,v} C_{uv} \cdot x(u,v) = \sum_{u,v} C_{uv} \cdot d(u,v) = \mathrm{OPT}(\mathrm{LR})$$

□

7. The Flow-Cut Gap

Actually, the factor $O(\log n)$ in the approximate max-flow min-cut theorem is tight. This is called the **flow-cut gap**.

7.1. Theorem. *There exist* G *and* H *where* $\mathsf{mcf}(G,H) \leq \frac{1}{\Omega(\log n)} \cdot \Phi(G,H)$.

Sketch of proof. The proof is from problem 5 in homework 1.

Let K be a complete graph with n nodes where each edge has weight $1/n$. Let G be an $\Omega(1)$-expander with constant maximum degree. We have $\Phi(G, K) \geq \Omega(1)$.

Because G is a graph with bounded degree, there will be at least $\Omega(n^2)$ pairs (u, v) of nodes where $\text{dist}_G(u, v) \geq \Omega(\log n)$. Let $\mathcal{F}$ be a concurrent flow that embeds K into G and define the volume of $\mathcal{F}$ as $\sum_{P \in \mathcal{F}} f(P) \cdot |P|$, where $f(P)$ is the flow value along the path P and $|P|$ is the number of edges in the path. The volume of $\mathcal{F}$ will be at least $\Omega(n \log n)$. Therefore, $\mathsf{mcf}(G, K) \leq O(1/\log n)$.

□

8. Partial Order based on Cut

In section 3, we define partial order based on concurrent flow. In this section, we define another partial order based on cut. We will see that these two partial orders are essentially equivalent.

8.1. Definition (Partial order based on cut)**.** Consider two undirected weighted graphs $G = (V, E_G, \kappa_G)$ and $H = (V, E_H, \kappa_H)$.

We define $H \preccurlyeq^{\text{cut}} G$ if for all $S \subset V$, we have $\kappa_H(\partial_H S) \leq \kappa_G(\partial_G S)$. In this case, we say that H **is cut-wise at most** G.

8.2. Fact. $H \preccurlyeq^{\text{cut}} G \iff \Phi(G, H) \geq 1$.

For any graph G and G'

- *let $G'' = G + G'$ be such that $\kappa_{G''}(u, v) = \kappa_G(u, v) + \kappa_{G'}(u, v)$ for all u, v and*
- *let $G' = \alpha \cdot G$ be such that $\kappa_{G'}(u, v) = \alpha \cdot \kappa_G(u, v)$.*

Similar to the partial order based on concurrent flow, the partial order based on cut also has following properties.

- *If $G_1 \preccurlyeq^{\text{cut}} G_2$ and $G_2 \preccurlyeq^{\text{cut}} G_3$, then $G_1 \preccurlyeq^{\text{cut}} G_3$.*
- *$G_1 \preccurlyeq^{\text{cut}} G_2$ iff $G_1 + H \preccurlyeq^{\text{cut}} G_2 + H$.*
- *$G_1 \preccurlyeq^{\text{cut}} \alpha \cdot G_2$ iff $\frac{1}{\alpha} G_1 \preccurlyeq^{\text{cut}} G_2$.*

We have seen two ways to compare graphs $\preccurlyeq^{\text{cut}}$ and $\preccurlyeq^{\text{flow}}$. The two relations are basically the same because of the approximate max-flow min-cut theorem. Roughly speaking, for two graphs G and H,

- If there exists a sparse cut in G w.r.t. H, then we cannot route the H-demand.
- If there is no sparse cut in G w.r.t. H, then we can route the H-demand with small congestion.

See lemma 8.3 for a formal description.

8.3. Lemma. *Consider two weighted undirected graphs G and H.*

- *If $H \preccurlyeq^{\text{flow}} G$, then $H \preccurlyeq^{\text{cut}} G$.*

- *If* $H \preccurlyeq^{\text{cut}} G$*, then* $H \preccurlyeq^{\text{flow}} O(\log n) \cdot G$.

Proof.

- Recall the approximate max-flow min-cut theorem.

$$\mathsf{mcf}(G, H) \leq \Phi(G, H) \leq O(\log n) \cdot \mathsf{mcf}(G, H).$$

- Recall that $H \preccurlyeq^{\text{flow}} G \iff \mathsf{mcf}(G, H) \geq 1$ and $H \preccurlyeq^{\text{cut}} G \iff \Phi(G, H) \geq 1$.
- If $H \preccurlyeq^{\text{flow}} G$, then $\Phi(G, H) \geq \mathsf{mcf}(G, H) \geq 1$ and so $H \preccurlyeq^{\text{cut}} G$.
- If $H \preccurlyeq^{\text{cut}} G$, then $\Phi(G, H) \geq 1$. So $\mathsf{mcf}(G, H) \geq \frac{1}{O(\log n)}$. So $H \preccurlyeq^{\text{flow}} O(\log n) \cdot G$.

□

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 6: Cut/Flow Equivalences Among Expanders

September 16, 2021

Instructor: Thatchaphol Saranurak

Scribe: Tian Zhang

1 Overview and Preliminaries

As is discussed in the previous lectures, an expander behaves like complete graph (the most well-connected graph) such that it is **robust against deletions**. This lecture is going to show expanders are the most well-connected graphs in the senses of **cut and flow**.

We first introduce some definitions.

Definition 1.1. Given a graph $G = \{V, E\}$ and a function $\boldsymbol{d} : V \to \mathbb{R}$, we say G has **degree profile** $\boldsymbol{d}$ if $\boldsymbol{d}(u) = \deg_G(u), \forall u \in V$. And graph is $\boldsymbol{d}$**-restricted** if $\deg_G(u) \leq \boldsymbol{d}(u), \forall u \in V$.

Definition 1.2. The $\boldsymbol{d}$**-product graph** $K = \{E, V\}$ is a complete graph such that the capacity/weight of each edge (u, v) is $\kappa_K(u, v) = \frac{\boldsymbol{d}(u)\boldsymbol{d}(v)}{\boldsymbol{d}(V)}$.

- Note that function $\boldsymbol{d}$ is exactly the degree profile of K, since $\forall u \in V, deg_G(u) = \sum_{v \in V} \boldsymbol{d}(u)\frac{\boldsymbol{d}(v)}{\boldsymbol{d}(V)} = \boldsymbol{d}(u)$.
- when $\forall u \in V$, $\mathbf{d}(u) = 1$, the graph is a normalized complete graph where each edge has weight $\frac{1}{n}$.

Definition 1.3. Given graphs $G = \{V, E\}$ and $H = \{V, E'\}$, $H \preccurlyeq^{\mathbf{cut}} G$ if $\kappa_H(\partial_H S) \leq \kappa_G(\partial_G S), \forall S \subseteq V$. $H \preccurlyeq^{\mathbf{flow}} G$ if H is embeddable into G with no congestion. And $H \preccurlyeq^{\mathbf{deg}} G$ if $\deg_H(u) \leq \deg_G(u)$ for all u.

2 Product Graphs and Their Cut/Flow Connectivity

Our goal of this section is to show that among all $\boldsymbol{d}$-restricted graph, the $\boldsymbol{d}$-product graph K is the most well-connected graph w.r.t. both $\preccurlyeq^{\text{cut}}$ and $\preccurlyeq^{\text{flow}}$.

2.1 Cut Connectivity of Product Graphs

Every cut of product graph is almost maximally big. Formally, we have the following lemma.

Lemma 2.1. *For all $\boldsymbol{d}$-restricted graphs H and the $\boldsymbol{d}$-product graph K, we have $H \preccurlyeq^{\text{cut}} 2K$.*

Proof. For any cut S where $\boldsymbol{d}(S) \le \boldsymbol{d}(V \setminus S)$, we have $\kappa_H(\partial_H S) \le \text{vol}_H(S) = \sum_{u\in S} \deg_H(u) \le \sum_{u\in S} \deg_K(u) = \boldsymbol{d}(S)$ since H is $\boldsymbol{d}-$restricted.

For each $u \in S$, we have $\kappa_K(E_K(u, V\setminus S)) = \sum_{v\notin S} \frac{\boldsymbol{d}(u)\boldsymbol{d}(v)}{\boldsymbol{d}(V)} = \boldsymbol{d}(u)\sum_{v\notin S} \frac{\boldsymbol{d}(v)}{\boldsymbol{d}(V)} = \boldsymbol{d}(u)\frac{\boldsymbol{d}(V\setminus S)}{\boldsymbol{d}(V)} \ge \boldsymbol{d}(u)/2$. So $\kappa_K(\partial_K S) \ge \boldsymbol{d}(S)/2$. □

- Note: this is the same as proving that $\Phi(K, H) \ge 1/2$.

2.2 Flow Connectivity of Product Graphs

All $\boldsymbol{d}$-restricted demands are routable on the $\boldsymbol{d}$-product graph. To see that, let us first get some intuition and discuss about routing demands with small congestion on arbitrary graphs.

Given any $\boldsymbol{d}$-restricted demand H and an arbitrary graph G,

- If there is a node u in G where $\deg_G(u) \ll \boldsymbol{d}(u)$, then clearly routing some H in G must cause large congestion.
- What if $\deg_G(u) \ge \boldsymbol{d}(u)$ for all u? Is this enough to route H in G in general? No, for example...

However, if G is a $\boldsymbol{d}$-product graph, condition "$\deg_G(u) \ge \boldsymbol{d}(u), \forall u$" is enough to make demand H routable in G.

Lemma 2.2. *All $\boldsymbol{d}$-restricted demands H are routable in the $\boldsymbol{d}$-product graph K with congestion 2, i.e. $H \preccurlyeq^{\text{flow}} 2K$.*

Proof. To embed H into K, we construct the following flow $\mathcal{F} = \{f_{(s,t)}\}_{(s,t)\in E(H)}$: For each $(s,t) \in E(H)$, the flow $f_{(s,t)}$ send flow through the two-hop paths $P_{svt} = (s, v, t)$ with value $f(P_{svt}) = \kappa_H(s,t) \cdot \frac{\boldsymbol{d}(v)}{\boldsymbol{d}(V)}$. Note that $\|f_{(s,t)}\| = \sum_v \kappa_H(s,t) \cdot \frac{\boldsymbol{d}(v)}{\boldsymbol{d}(V)} = \kappa_H(s,t)$. So, $\mathcal{F}$ indeed embeds H into K.

Now, we bound the congestion when we implement flow $\mathcal{F}$ in K. For each edge $(u, v) \in E(K)$, it is contained in flow paths $\{P_{suv}\}_{s\in V}$ and $\{P_{uvt}\}_{t\in V}$ with total flow value

$$\sum_{s\in V} \kappa_H(s,v)\cdot\frac{\boldsymbol{d}(u)}{\boldsymbol{d}(V)} + \sum_{t\in V} \kappa_H(u,t)\cdot\frac{\boldsymbol{d}(v)}{\boldsymbol{d}(V)} = \frac{\deg_H(v)\boldsymbol{d}(u)}{\boldsymbol{d}(V)} + \frac{\deg_H(u)\boldsymbol{d}(v)}{\boldsymbol{d}(V)} \le \frac{\boldsymbol{d}(v)\boldsymbol{d}(u)}{\boldsymbol{d}(V)} + \frac{\boldsymbol{d}(u)\boldsymbol{d}(v)}{\boldsymbol{d}(V)} \le 2\kappa_K(u,v)$$

□

Therefore, $\mathcal{F}$ embeds H into K with congestion 2.

3 Cut/Flow Connectivity of Expanders

Given an expander G, if you want to prove that $H \preccurlyeq^{\text{cut}} G$ or $H \preccurlyeq^{\text{flow}} G$ (within some constant factor), you can just show that **degree of G is big enough**! Formally, we have the following lemma.

Lemma 3.1. *Let G be ϕ-expander. Let H be any graph where $H \preccurlyeq^{\deg} G$. Then, $H \preccurlyeq^{\text{cut}} \frac{1}{\phi} G$.*

Proof. For any $S \subset V$ where $\text{vol}_G(S) \leq \text{vol}_G(V \setminus S)$, we have that

$$\kappa_H(\partial_H S) \leq \text{vol}_H(S) \leq \text{vol}_G(S)$$

because $H \preccurlyeq^{\deg} G$. At the same time we have

$$\kappa_G(\partial_G S) \geq \phi \text{vol}_G(S).$$

So $\kappa_H(\partial_H S) \leq \frac{1}{\phi}\kappa_G(\partial_G S)$ and so $H \preccurlyeq^{\text{cut}} \frac{1}{\phi} G$. □

By the approximate max-flow min-cut theorem, we know that $H \preccurlyeq^{\text{cut}} \frac{1}{\phi} G$ implies $H \preccurlyeq^{\text{flow}} \frac{O(\log n)}{\phi} G$. So we have

Corollary 3.2. *Let G be ϕ-expander. Let H be any graph where $H \preccurlyeq^{\deg} G$. Then, $H \preccurlyeq^{\text{flow}} \frac{O(\log n)}{\phi} G$.*

4 Cut/Flow Connectivity Equivalence of Expanders

We are going to show that expanders with same degree profile are approximately equivalent in some sense. We first formalize notations of equivalence.

4.1 Notations of Equivalence

Definition 4.1. G_1 and G_2 are **α-cut-equivalent** if there are $\alpha_1, \alpha_2 > 0$ where $\alpha_1 \cdot \alpha_2 \leq \alpha$ such that $G_2 \preccurlyeq^{\text{cut}} \alpha_1 G_1$ and $G_1 \preccurlyeq^{\text{cut}} \alpha_2 G_2$. Denote this by $G_1 \approx_\alpha^{\text{cut}} G_2$.

- Our definition is designed such that it measures how much the cut size of G_1 and G_2 are similar *modulo some scaling*. See the example below.

Example 4.2. G_1 and $G_2 = 100 \cdot G_1$ are 1-cut-equivalent, because $G_2 \preccurlyeq^{\text{cut}} 100 G_1$ but $G_1 \preccurlyeq^{\text{cut}} \frac{1}{100} G_2$.

Definition 4.3. G_1 and G_2 are **α-flow-equivalent** if there are $\alpha_1, \alpha_2 > 0$ where $\alpha_1 \cdot \alpha_2 \leq \alpha$ such that $G_2 \preccurlyeq^{\text{flow}} \alpha_1 G_1$ and $G_1 \preccurlyeq^{\text{flow}} \alpha_2 G_2$. Denote this by $G_1 \approx_\alpha^{\text{flow}} G_2$.

- Our definition is designed such that it measures the congestion when we try to embed G_1 and G_2 into each other.

Exercise 4.4. Show that:

- If $G_1 \approx_\alpha^{\text{flow}} G_2$, then $G_1 \approx_\alpha^{\text{cut}} G_2$.
- If $G_1 \approx_\alpha^{\text{cut}} G_2$, then $G_1 \approx_{O(\alpha \log^2 n)}^{\text{flow}} G_2$.

4.2 Equivalence and Well-Connectivity of Expanders

Based on previous conclusions given by simple proofs, We can obtain statements of strong conceptual messages.

In a sentence, **Expanders with Degree Profile d are all equivalent in the sense of cut/flow connectivity and they are the most well-connected d-restricted graphs.**

First, we have that $\tilde{\Omega}(1)$-expanders with the same degree profile are approximately equivalent to each other w.r.t. both $\preccurlyeq^{\text{cut}}$ and $\preccurlyeq^{\text{flow}}$.

Corollary 4.5. *Let G and G' be ϕ-expanders with degree profile $\boldsymbol{d}$. We have*

- *$G \preccurlyeq^{\text{cut}} \frac{1}{\phi} G'$ and $G' \preccurlyeq^{\text{cut}} \frac{1}{\phi} G$. So G and G' are $O(\frac{1}{\phi^2})$-cut-equivalent.*
- *$G \preccurlyeq^{\text{flow}} O(\frac{\log n}{\phi}) G'$ and $G' \preccurlyeq^{\text{flow}} O(\frac{\log n}{\phi}) G$. So G and G' are $O(\frac{\log^2 n}{\phi^2})$-flow-equivalent.*

Notice first that this statement is robust against perturbation of $\boldsymbol{d}-$ even if G and G' have different degree profile within a factor of some constant, which is going to add some extra factor on the equivalence factor, they are still $O\left(\frac{1}{\phi^2}\right)$-cut- and $O\left(\frac{\log^2 n}{\phi^2}\right)$-flow-equivalent.

This Corollary shows that, in other words, $\tilde{\Omega}(1)-$expanders with the same degree profile form a $\tilde{O}(1)$-cut- and -flow-equivalence class, where everything within the class have a similar cut and flow structure. Moreover, the most well-connected $\boldsymbol{d}$-restricted graph, i.e. the $\boldsymbol{d}$-product graph K, is also in this equivalence class, since it is an $\Omega(1)-$expander with degree profile $\boldsymbol{d}$. The approximation factor is a bit better too.

Corollary 4.6. *Let G be ϕ-expander with degree profile $\boldsymbol{d}$. Let K be the $\boldsymbol{d}$-product graph. We have*

- *$G \preccurlyeq^{\text{cut}} 2K$ and $K \preccurlyeq^{\text{cut}} \frac{1}{\phi} G$. So G and K are $O(\frac{1}{\phi})$-cut-equivalent.*
- *$G \preccurlyeq^{\text{flow}} 2K$ and $K \preccurlyeq^{\text{flow}} O(\frac{\log n}{\phi}) G$. So G and K are $O(\frac{\log n}{\phi})$-flow-equivalent.*

This is nice because G can have much fewer edges than K, yet G is as "good" as K.

Because of this, any expander with degree profile $\boldsymbol{d}$ is approximately the most well-connected $\boldsymbol{d}$-restricted graph w.r.t. both $\preccurlyeq^{\text{cut}}$ and $\preccurlyeq^{\text{flow}}$.

Corollary 4.7. *Let G be a ϕ-expander with degree profile $\boldsymbol{d}$.*

- *For all $\boldsymbol{d}$-restricted graphs H, we have $H \preccurlyeq^{\text{cut}} O(\frac{1}{\phi}) G$.*
- *For all $\boldsymbol{d}$-restricted graphs H, we have $H \preccurlyeq^{\text{flow}} O(\frac{\log n}{\phi}) G$.*

4.3 Certificate For Well-Connectivity via Expanders

- Lastly, suppose you want to show that some arbitrary $\boldsymbol{d}$-restricted graph G' is the most well-connected among all $\boldsymbol{d}$-restricted graphs.
- What is the *certificate* for this?

Corollary 4.8. *Let G be a ϕ-expander with degree profile $\boldsymbol{d}$. Let G' be any $\boldsymbol{d}-$restricted graph. If $G \preccurlyeq^{\text{flow}} G'$, then $H \preccurlyeq^{\text{flow}} O(\log(n)/\phi) \cdot G'$ for all $\boldsymbol{d}$-restricted demand H.*

- That is, a feasible flow embedding that embeds any expander G into G' is the certificate.

5 Summary

Let's fix the degree profile $\boldsymbol{d}$. We want to compare "well-connectivity" of graphs with this degree profile $\boldsymbol{d}$.

- First, we prove that the $\boldsymbol{d}$-product graph K is basically the most well-connected graph. In both cut and flow perspective, we have

  $$H \preccurlyeq^{\text{cut}} 2K \text{ and } H \preccurlyeq^{\text{flow}} 2K$$

  for any graph H where $\deg_H \leq \boldsymbol{d}$ (i.e. $\boldsymbol{d}$-restricted graph).

- Second, we show that if $H \preccurlyeq^{\deg} G$ and G is a ϕ-expander, then $H \preccurlyeq^{\text{cut}} \frac{1}{\phi} G$.

  - Therefore, for any two $\tilde{\Omega}(1)$-expanders G_1 and G_2 with the same degree profile $\boldsymbol{d}$ (i.e. $G_1 =^{\deg} G_2$), we have

    $$G_1 \approx^{\text{cut}} G_2$$

  - Combining with the approximate max-flow min-cut theorem (i.e. $H \preccurlyeq^{\text{cut}} G \overset{\approx}{\iff} H \preccurlyeq^{\text{flow}} G$), we have

    $$G_1 \approx^{\text{flow}} G_2$$

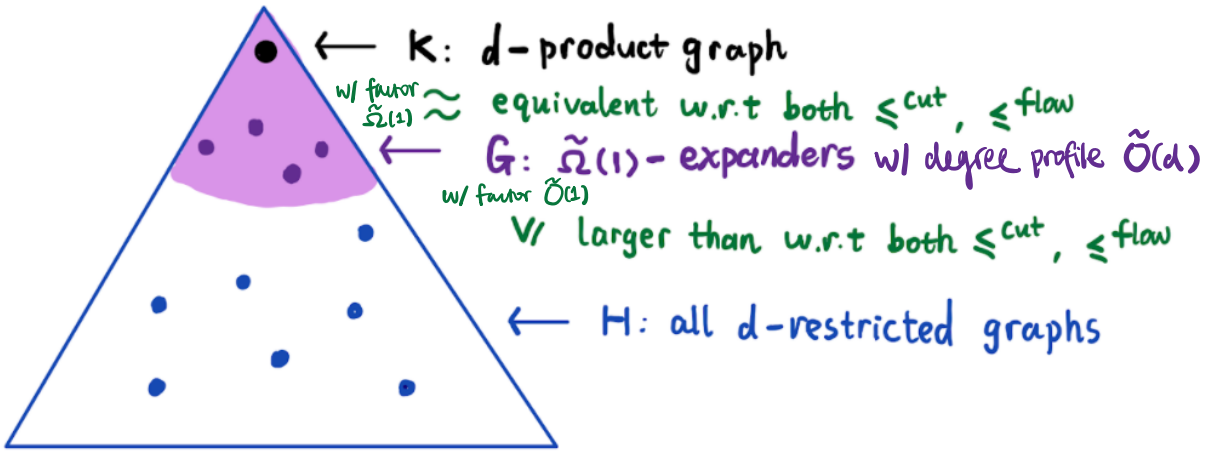
  - That is, expanders with the same degree profile are all approximately equivalent from both cut and flow perspectives.
  - Note that K is in the equivalent class too (as K is a $\Omega(1)$-expander).

- Now, as K is the most well-connected graph w.r.t $\preccurlyeq^{\text{cut}}$ and $\preccurlyeq^{\text{flow}}$, the whole class of expanders are basically most well-connected as well.

  - That is, for any $\tilde{\Omega}(1)$-expander G and a $\boldsymbol{d}$-restricted graph H, we have

    $$H \lessapprox^{\text{cut}} G \text{ and } H \lessapprox^{\text{flow}} G$$

- This is the picture you should have in mind



- To summarize, if you know $H \preccurlyeq^{\deg} G$ and G is an expander, then $H \lessapprox^{\text{cut}} G$ and $H \lessapprox^{\text{flow}} G$.
  - This is important.
  - For both flow and cut, all that "matters" for expanders is to compare degree cuts (when we allow some approximate)
- Lastly, suppose you want to show that some graph G' is the most well-connected among graphs with the same degree profile $\boldsymbol{d}$.
  - It is enough to show $K \lessapprox^{\text{cut}} G'$ or $K \lessapprox^{\text{flow}} G'$.
  - But it is enough to show $G \lessapprox^{\text{cut}} G'$ or $G \lessapprox^{\text{flow}} G'$ for **any** expander G too.
  - That is, if you can embed an expander G into G', then you are done.

6 The "Right" Way to Think of Expanders

- Let $G = (V, E)$ be a ϕ-expander.
  - By the previous discussion, if G has uniform degree, then G is cut/flow-equivalent to the corresponding $\boldsymbol{d}$-product graph, which is essentially the complete graph on V with unit weight on each edge (after scaling by some constant.) Hence we can think of G as an unweighted complete graph on V, and this is a nice picture to have in mind.
  - But, otherwise, it is equivalent to the $\deg_G$-product graph, which is quite hard to think about since it has different weights on different nodes.
- I find it much more convenient to think of an expander in the following way.
- Suppose that $G = (V, E)$ is unweighted for now (otherwise make parallel edges).
- Let G' be obtained from G by subdividing each edge $e = (u, v)$ into (u, x_e) and (x_e, v).
  - We call x_e the *split node* of e.
  - Let $X_E = \{x_e \mid e \in E\}$.
  - So $V(G') = V(G) \cup X_E$.
- Here is the "better" way to think about an expander. Let K_E be a **unweighted clique** on X_E (no edges on V).
  - Note: every node in $\frac{1}{m-1} K_E$ has degree 1.
- Essentially, G **is an expander iff an unweighted clique on split-nodes** X_E **(after scaling by** $\frac{1}{m}$**) is embeddable into** G'**.** Namely, we can think of G as an unweighted complete graph on X_E.

Intuitively this is saying that for a graph to be an expander, all of its edges should be pairwise "well connected", in the sense that there should be a collection of flows on G' that allows any edge in G to send a unit of flow to any other edge in G while this collection has small congestion. In this case X_E is representing all edges in G, and embedding K_E into G' is finding the collection of flows that we described above. To show this perspective is true, we state the following lemma formally,

Lemma 6.1. *We have:*

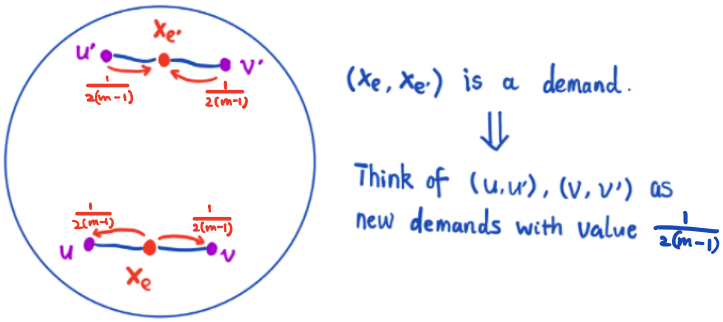
1. *If G is a ϕ-expander, then $\frac{1}{m-1}K_E \preccurlyeq^{\text{flow}} O(\frac{\log n}{\phi}) \cdot G'$.*
2. *If G is not a ϕ-expander, then $\frac{1}{m-1}K_E \not\preccurlyeq^{\text{flow}} \frac{1}{100\phi} \cdot G'$.*

Corollary 6.2. *For any 1-restricted graph H_E where $V(H_E) = X_E$, we have $H_E \preccurlyeq^{\text{flow}} O(\frac{\log n}{\phi}) \cdot G'$. That is, every edge can exchange 1 unit of flow between any other edges.*

Now, we prove the lemma.

Proof. There are two directions.

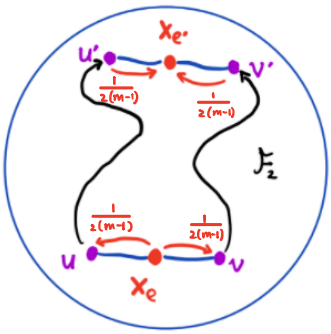
1. Suppose G is a ϕ-expander.
   - You just to construct a $\frac{1}{m-1}K_E$-concurrent flow $\mathcal{F}$ with small congestion.
   - We will construct $\mathcal{F}$ in several steps.
     (a) Fix a demand $(x_e, x_{e'})$ where $e = (u, v)$ and $e' = (u', v')$,
     (b) Think of this demand as directed demand so that we can draw the flow, but all we need is just a undirected flow.
     (c) x_e sends flow to u, v (each of value $1/2(m-1)$)
     (d) $x_{e'}$ receive flow from u', v' (each of value $1/2(m-1)$)



     (e) This induces new demand. Indeed, consider the demand graph $H = (V, E_H, \kappa_H)$ where $E_H = \{(u, v)\}_{u,v \in V(E)}$. Essentially H is a clique on V, and we will discuss the construction of κ_H in the next step.
     (f) We still look at the fixed demand $(x_e, x_{e'})$ first. For this demand $(x_e, x_{e'})$, we can treat sending $\frac{1}{m-1}$ unit of demand between x_e and $x_{e'}$ as a two-step process where firstly (1)x_e sending $\frac{1}{2(m-1)}$ unit of demand between $x_e$and u, x_e and v respectively; $x_{e'}$ sending $\frac{1}{2(m-1)}$ unit of demand between $x_{e'}$and u', $x_{e'}$ and v' respectively, then (2)sending $\frac{1}{2(m-1)}$ unit of demand between u, u' and v, v' respectively. In this way, we separate the demand sending process for all possible pairs $(x_{e_1}, x_{e_2}), x_{e_1}, x_{e_2} \in X_E$, and accumulate the unit of demand on each edge in E_H to form κ_H.
     Notice that the total demand each original node $u \in V$ needs to send/receive is $\deg_G(u)/2$, i.e. $\deg_H(u) = \deg_G(u)/2$. This is because u is adjacent to $\deg_G(u)$ vertices in X_E where each vertex is involved in $(m-1)$ edges in demand graph K_E, and each edge(representing $1/(m-1)$ unit of demand need to pass through) induces $\frac{1}{2(m-1)}$ unit of demand got send between u and some other vertex.

(g) Therefore, H is a $\frac{\deg_G}{2}$-restricted demand graph. We know $H \preccurlyeq^{\text{flow}} O(\frac{\log n}{\phi})G$. As $G \preccurlyeq^{\text{flow}} G'$, we have $H \preccurlyeq^{\text{flow}} O(\frac{\log n}{\phi}) \cdot G'$.

(h) So there is an H-concurrent flow $\mathcal{F}_2$ that embeds H into G' with congestion $q_2 = O(\frac{\log n}{\phi})$.



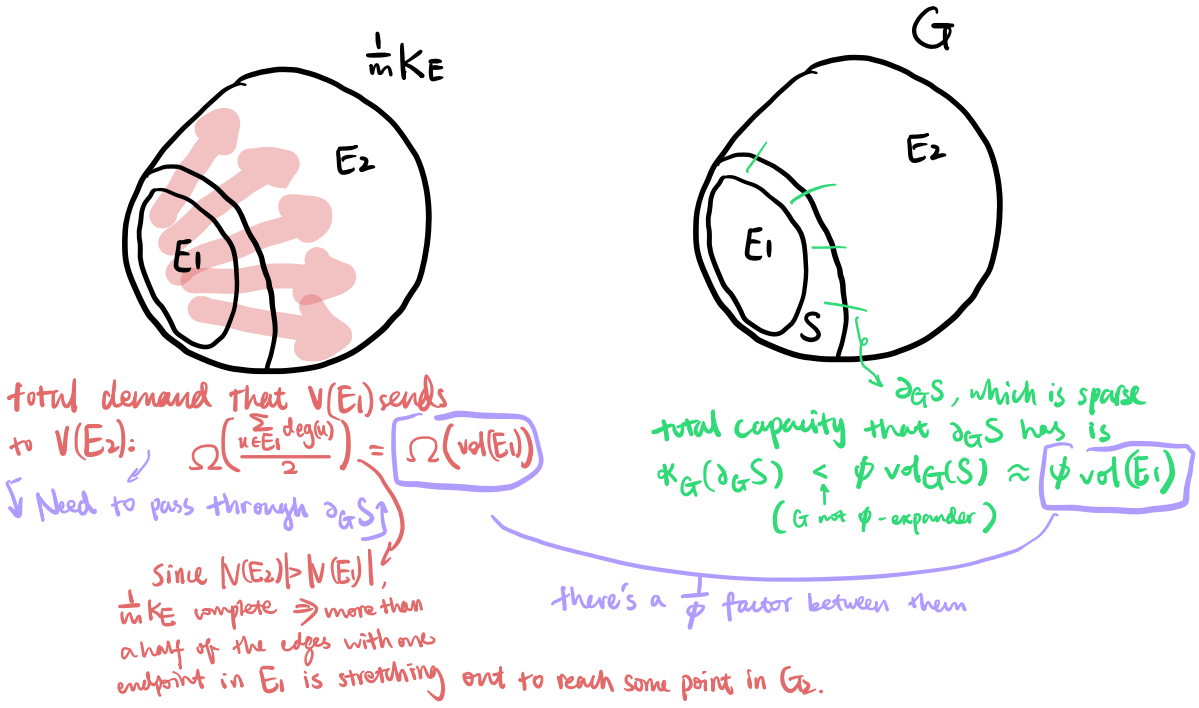
(i) Concatenate the $\mathcal{F}_2$ with the "red" flow for every demand. That would satisfy all demand of $\frac{1}{m-1}K_E$.

(j) What is the congestion? It is $1/2 + q_2 = O(\frac{\log n}{\phi})$. It is not just q_2! Why? This is because it's possible that there might be a flow in $\mathcal{F}_2$ with the largest congestion overlap with the flow between an endpoint in $V(G)$ and a split point in X_E on the edge with congestion q_2. In that case, the congestion on that edge should be $q_2 + 1/2$ where $1/2$ is the congestion generated by the flow between one endpoint to its adjacent split node.

(k) We have a $\frac{1}{m-1}K_E$-concurrent flow with congestion $O(\frac{\log n}{\phi})$ in G'.

2. Suppose G is not a ϕ-expander, where $\phi < 1/10$.

- Let $(S, \overline{S})$ be a cut such that $\text{vol}_G(S) \leq \text{vol}_G(\overline{S})$ and $\kappa_G(\partial_G S) < \phi \text{vol}_G(S)$.
- Consider $E_1 = E_G(S, S)$ and $E_2 = E_G(\overline{S}, \overline{S})$. Note that $|E_1| = \Theta(\text{vol}_G(S))$.
- The total demand that E_1 need to send to E_2 is $\Theta(|E_1|)$. Any concurrent flow for this demand must go through the cut S which has capacity $\kappa_G(\partial_G S)$.
- So the congestion must be least $\Theta(|E_1|)/\kappa_G(\partial_G S) = \Omega(1/\phi)$ (or $1/100\phi$ if we compute the constant explicitly).



□

- The proof above is quite simple for flow.
- You can give a tighter bound for cut. But it is a bit more tedious. This is an example where it can be easier to think about flow instead of cut (although they are essentially equivalent).

Exercise 6.3. Show that $\Phi(G) = \Theta(\Phi(G'))$.

Exercise 6.4. We have:

- If G is a ϕ-expander, then $K_E \preccurlyeq^{\text{cut}} \frac{100}{\phi} \cdot G'$.
- If G is not a ϕ-expander, then $K_E \not\preccurlyeq^{\text{cut}} \frac{1}{100\phi} \cdot G'$.

7 Optional: Computing Quality of Cut/Flow Equivalence

- Question: Given G and H, what is the smallest α where H and G are α-cut/flow-equivalent?

7.1 Cut

For any graph G and H,

- H and G are α-cut-equivalent where $\alpha = 1/(\Phi(G,H)\cdot\Phi(H,G)) \ge 1$.
- H and G are not α'-cut-equivalent for any $\alpha' < \alpha$.

Proof. Verify that $\Phi(G,H) \cdot H \preccurlyeq^{\text{cut}} G$ and $\Phi(H,G) \cdot G \preccurlyeq^{\text{cut}} H$. □

- As we know how to $O(\log n)$-approximate both $\Phi(G,H)$ and $\Phi(H,G)$ (even $O(\sqrt{\log n}\log\log n)$-approximation algorithm is known), we have the following:

Corollary 7.1. *Given graphs G and H, there is a polynomial time algorithm that* $\text{poly}\log(n)$*-approximates the minimum α cut-equivalence factor of G and H.*

Question 7.2 (Very Interesting Open Problem)**.** *Is there a polynomial time or even subexponential time algorithm that $O(1)$-approximates the minimum α cut-equivalence factor between G and H?*

7.2 Flow

For any graph G and H,

- H is a α-flow sparsifier of G where $\alpha = 1/(\mathsf{mcf}(G,H) \cdot \mathsf{mcf}(H,G)) \geq 1$.
- Moreover, H is not a α'-flow sparsifier of G for any $\alpha' < \alpha$.

Proof. Verify that $\mathsf{mcf}(G,H) \cdot H \preccurlyeq^{\text{cut}} G$ and $\mathsf{mcf}(H,G) \cdot G \preccurlyeq^{\text{cut}} H$. □

- In contract to the cut case, we can compute the embeddability factor α exactly because $\mathsf{mcf}(G,H)$ and $\mathsf{mcf}(H,G)$ are computable via LPs.

Corollary 7.3. *Given graphs G and H, there is an polynomial algorithm that exactly computes the minimum α flow-equivalence factor between G and H.*

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 7: Cheeger's inequality

September 21, 2021

Instructor: Thatchaphol Saranurak

Scribe: Jingyi Gao

1. Recap

So far, we've seen several characterization of expanders. Intuitively, expanders are graphs that are robust against any adversarial deletions. From the perspective of cut, when we compare all graphs with the same degree profile, the size of every cut of expander is almost maximum among them. On the other hand, regarding flows, for all graphs with a degree profile no greater than the expander, they are embeddable into expander with small congestion.
All of the three characterization of expander are combinatorial. Today, we are going to introduce an algebraic way to characterize expanders based on **eigenvalue**s. This invokes an area called **spectral graph theory.** Roughly speaking, in this area, we associate a graph G with some matrix M and relate eigenvalues of M to combinatorial properties of G.
A very nice resource is the textbook by Spielman [1].

2. Review of Linear Algebra

We start by reviewing some linear algebra facts.

2.1. Theorem. *Let $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ be a symmetric real matrix. Then there exists $\lambda_1, ..., \lambda_n \in \mathbb{R}$ and orthonormal vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^n$ such that*

- *For all i, $\boldsymbol{M}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$.*
- *In fact, $\boldsymbol{M} = \sum_i \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$*

We call $\lambda_1, ..., \lambda_n$ the eigenvalues *of $\boldsymbol{M}$ and each $\boldsymbol{v}_i$ is called the* eigenvector *corresponding to λ_i.*

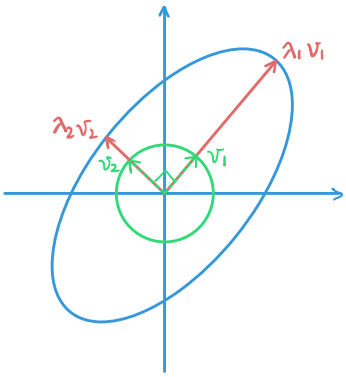
2.2. *Remark.*

- A useful way to think of the relation between $\boldsymbol{M}$ and its eigenvectors and eigenvalues is that any real symmetric matrix $\boldsymbol{M}$ corresponds to an ellipsoid. Consider the image of $\boldsymbol{M}C$, where $C = \{x \in \mathbb{R}^n : |x| = 1\}$. Let $x \in C$ and we can write x in terms of basis

[1] http://cs-www.cs.yale.edu/homes/spielman/sagt/sagt.pdf

$\{v_1, \cdots, v_n\}$ (an orthonormal basis consists of all eigenvectors of $\boldsymbol{M}$,) $x = x_1 v_1 + \cdots + x_n v_n$, where $\sum_{i=1}^n x_i^2 = 1$. Then $y = \boldsymbol{M}x = \sum_{i=1}^n x_i \lambda_i v_i$. This is to say that the component of the image point $y = \boldsymbol{M}x$ has components $y_i = x_i \lambda_i$ in terms of the basis v_i. Hence the image of $\boldsymbol{M}x$ is on the ellipsoid $\left\{ \sum_{i=1}^n y_i v_i : \sum_{i=1}^n \left(\frac{y_i}{\lambda_i}\right)^2 = 1 \right\} = \boldsymbol{M}C$ with axes along the vectors $v_i, i = 1, \cdots, n$ with length of the axes being $\lambda_i, i = 1, \cdots, n$, respectively.

- In short, any $\boldsymbol{M} \in \mathbb{SR}^{n \times n}$ corresponds to an ellipsoid, which is the image of unit circle under mapping $\boldsymbol{M}$. Each eigenvector v_i is the $i-$th axis of the ellipsoid and each eigenvalue λ_i describes how much we stretch the unit circle along the $i-$th axis. A picture for $n = 2$ is as below.



2.3. Theorem (Variational Characterization of Eigenvalues). *Let $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ be a symmetric real matrix with eigenvalues $\lambda_1 \leq \ldots \leq \lambda_n$ and eigenvectors $v_1, \ldots, v_n$. Then*

- $\lambda_1 = \min_{x \neq 0} \frac{x^\top \boldsymbol{M} x}{x^\top x}$
- $\lambda_n = \max_{x \neq 0} \frac{x^\top \boldsymbol{M} x}{x^\top x}$
- $\lambda_2 = \min_{x \perp v_1, x \neq 0} \frac{x^\top \boldsymbol{M} x}{x^\top x} = \min_{2\text{-}dim\, \mathcal{V}} \max_{x \in \mathcal{V} \setminus \{0\}} \frac{x^\top \boldsymbol{M} x}{x^\top x}$
- *In general, we have*

$$\lambda_k = \min_{k\text{-}dim\, \mathcal{V}} \max_{x \in \mathcal{V} - \{0\}} \frac{\boldsymbol{x}^\top \boldsymbol{M} \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}} = \min_{x \perp \text{span}\{v_1, \ldots v_{k-1}\}, x \neq 0} \frac{\boldsymbol{x}^\top \boldsymbol{M} \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}}$$

We call $R_M(\boldsymbol{x}) = \frac{x^\top \boldsymbol{M} x}{x^\top x}$ the Rayleigh quotient of x w.r.t. $\boldsymbol{M}$.

3. The Laplacian of Graphs

Given an undirected graph $G = (V, E, w)$ with non-negative weights $w : E \to \mathbb{R}_{\geq 0}$, the *adjacency matrix* $\boldsymbol{A}$ is such that $\boldsymbol{A}_{u,v} = w_{u,v}$ for all $(u, v) \in E$. Note that since the graph is undirected, the matrix is symmetric. The *degree matrix* $\boldsymbol{D}$ is a diagonal matrix such that $\boldsymbol{D}_{uu} = \boldsymbol{d}(u) = \sum_v w_{uv}$ for all $u \in V$, where $\boldsymbol{d}(u) = \deg_G(u)$ is the short hand for degree, and $\boldsymbol{d}(S) = \text{vol}_G(S)$ is the volume of

the vertex set S.
The **Laplacian matrix** is defined as the difference of these two matrices: $\boldsymbol{L}_G = \boldsymbol{D} - \boldsymbol{A}$.[2]

3.1. Fact. *$\boldsymbol{L}_G$ is a real symmetric matrix. Therefore, all the eigenvalues of $\boldsymbol{L}_G$ are real.*[3]

Let $\boldsymbol{L}_{(u,v)}$ be the Laplacian of the subgraph of G that contains only one unweighted edge (u, v). Then in this case, since

$$D = \begin{array}{c} \begin{array}{ccccccccccc} & \cdots & & u & & \cdots & & v & & \cdots & \end{array} \\ \begin{pmatrix} 0 & & & & & \cdots & & & & & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & & & & & \cdots & & & & & 0 \end{pmatrix} \end{array} \begin{array}{l} \vdots \\ \\ u \\ \\ \vdots \\ \\ v \\ \\ \vdots \end{array}$$

$$A = \begin{array}{c} \begin{array}{ccccccccccc} & \cdots & & u & & \cdots & & v & & \cdots & \end{array} \\ \begin{pmatrix} 0 & & & & & \cdots & & & & & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & & & & & \cdots & & & & & 0 \end{pmatrix} \end{array} \begin{array}{l} \vdots \\ \\ u \\ \\ \vdots \\ \\ v \\ \\ \vdots \end{array}$$

[2]Note that L_G does not change when we add self-loops.

[3]In fact, it is quite easy to see that L_G is positive semidefinite (this can be show by endowing the graph an arbitrary direction, then $\boldsymbol{L}_G$ is the incidence matrix multiply by its transpose) and hence all its eigenvalues are non-negative.

the Laplacian of a single-edge subgraph

$$\boldsymbol{L}_{(u,v)} = \boldsymbol{D} - \boldsymbol{A} = \begin{array}{c} \begin{matrix} & \cdots & & u & & \cdots & & v & & \cdots & \end{matrix} \\ \begin{pmatrix} 0 & & & & & \cdots & & & & & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & & & & & \cdots & & & & & 0 \end{pmatrix} \end{array} \begin{matrix} \vdots \\ \\ u \\ \\ \vdots \\ \\ v \\ \\ \vdots \end{matrix}$$

which looks very simple.

By decomposing the whole graph into edges, the Laplacian of the graph $\boldsymbol{L}_G$ is the sum of these simple matrices:

3.2. Proposition. $\boldsymbol{L}_G = \sum_{e \in E} w_e \boldsymbol{L}_e$.

Since the Laplacian is a linear combination of Laplacian of a single-edge subgraph, $\boldsymbol{L}_{(u,v)}$ is important and we should take a closer look. Observe that for $\boldsymbol{L}_{(u,v)}$, we have $\forall \boldsymbol{x} \in \mathbb{R}^V$

$$\boldsymbol{x}^\top \boldsymbol{L}_{(u,v)} \boldsymbol{x} = x_u^2 - 2x_u x_v + x_v^2 = (x_u - x_u)^2$$

whence

$$\mathbf{1}_S^\top \boldsymbol{L}_{(u,v)} \mathbf{1}_S = \begin{cases} 1 & \text{if } |\{u,v\} \cap S| = 1 \text{ (i.e. } \{u,v\} \text{ is cut)} \\ 0 & \text{otherwise} \end{cases}$$

Since $\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x} = \sum_{e \in E} w_e (\boldsymbol{x}^\top \boldsymbol{L}_e \boldsymbol{x})$, we can conclude that

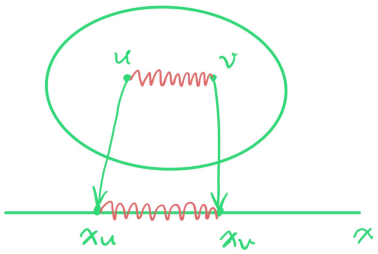
3.3. Proposition. *$\forall \boldsymbol{x} \in \mathbb{R}^V$, we have*

$$\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x} = \sum_{(u,v) \in E} w_{uv} (x_u - x_v)^2.$$

In particular, if $x = \mathbf{1}_S$, we have

$$\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x} = w_G(\partial_G S).$$

3.4. *Note.* A intuitive interpretation of this proposition is that we can think of in the original graph, all edges (u,v) are springs with spring constant $w_{(u,v)}$. We use $\boldsymbol{x}$ to keep track of each edge (u,v), in the way that $(x_u - x_v)^2$ scaled by the "spring constant," namely the edge weight, is the energy of the spring corresponds to (u,v). $\boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x}$ stands for the total spring energy in the graph G.



3.5. Proposition. *Let $\lambda_1(\boldsymbol{L}_G) \leq \cdots \leq \lambda_n(\boldsymbol{L}_G)$ be the eigenvalues of $\boldsymbol{L}_G$. We have*

1. *$\lambda_1(\boldsymbol{L}_G) = 0$ with a corresponding eigenvector $\boldsymbol{v}_1 = \mathbf{1}$.*
2. *$\lambda_k(\boldsymbol{L}_G) = 0$ if and only if G contains at least k connected components.*

Bizonyítás.

1. By definition,
$$\lambda_1(\boldsymbol{L}_G) = \min_{\boldsymbol{x}\neq\mathbf{0}} R_{\boldsymbol{L}_G}(\boldsymbol{x}) = \min_{\boldsymbol{x}\neq\mathbf{0}} \frac{\sum_{(u,v)\in E} w_{uv}(x_u - x_v)^2}{\boldsymbol{x}^\top \boldsymbol{x}}.$$
Since $(x_u - x_v)^2 \geq 0$, we have $\lambda_1(\boldsymbol{L}_G) \geq 0$. On the other hand, if we let $\boldsymbol{x} = \mathbf{1}$, then $R_{\boldsymbol{L}_G}(\mathbf{1}) = 0$. Therefore, as the minimum over all possible $\mathbb{R}_{\boldsymbol{L}_G}(x)$, $\lambda_1(\boldsymbol{L}_G) \leq R_{L_G}(\mathbf{1}) = 0$. Hence $\lambda_1(\boldsymbol{L}_G) = 0$. To find its corresponding eigenvector, as previously mentioned on page 3, the Laplacian of the subgraph with a single unweighted edge has exactly 2 non-zero rows containing exactly one 1 and one -1 each. Hence when we multiply this matrix by an all one vector, we will get zero vector. Therefore, $\boldsymbol{L}_G\mathbf{1} = \sum_{e\in E} w_e \boldsymbol{L}_e\mathbf{1} = \mathbf{0}$. Hence the eigenvector corresponding to the eigenvalue 0 is $\boldsymbol{v}_1 = \mathbf{1}$.

   Another way to find $\boldsymbol{v}_1 = \mathbf{1}$ is by eyeballing $\boldsymbol{L}_G\mathbf{1} = (\boldsymbol{D} - \boldsymbol{A})\mathbf{1} = \boldsymbol{D}\mathbf{1} - \boldsymbol{A}\mathbf{1}$. The i^{th} row of $\boldsymbol{D}\mathbf{1}$ is the degree of u, and the i^{th} row of $\boldsymbol{A}\mathbf{1}$ is the sum of the weight of all edges adjacent to u, which is also the degree of u. Therefore for any arbitrary i, the i^{th} row of $(\boldsymbol{D} - \boldsymbol{A})\mathbf{1}$ is 0, whence $(\boldsymbol{D} - \boldsymbol{A})\mathbf{1} = \mathbf{0}$.

2. For $\lambda_k(\boldsymbol{L}_G)$, recall that
$$\lambda_k(\boldsymbol{L}_G) = \min_{k\text{-dim }\mathcal{V}} \max_{\boldsymbol{x}\in\mathcal{V}-\{\mathbf{0}\}} R_{L_G}(\boldsymbol{x})$$
   " $\Longleftarrow$ " Suppose there are k connected components $C_1, \ldots, C_k$.
   Let $\mathcal{V} = \text{span}\{1_{C_1}, \ldots 1_{C_k}\}$. Note that $\dim(\mathcal{V}) = k$. For any $\boldsymbol{x} \in \mathcal{V}$ and for each component C_i, entries of $\boldsymbol{x}$ correspond to vertices in C_i are the same constant since by definition of $\mathcal{V}$, we can represent any $\boldsymbol{x} \in \mathcal{V}$ by using basis $\boldsymbol{x} = \sum_{i=1}^{k} \mathbf{1}_{C_i}$. Therefore $\sum_{(u,v)\in E(C_i)} w_{uv}(x_u - x_v)^2 = 0$. Since there's no edges between each connected components $R_{\boldsymbol{L}_G}(\boldsymbol{x}) = \sum_{i=1}^{k} \sum_{(u,v)\in E(C_i)} w_{uv}(x_u - x_v)^2 = 0$, and $\lambda_k(\boldsymbol{L}_G) \leq 0$. Combining with the fact that $\lambda_k(\boldsymbol{L}_G)$ is non-negative, it is 0
   " $\Longrightarrow$ " Suppose $\lambda_k(\boldsymbol{L}_G) = 0$.
   Let $\mathcal{V}$ be a k-dimensional space where $\max_{\boldsymbol{x}\in\mathcal{V}-\{\mathbf{0}\}} R_{\boldsymbol{L}_G}(\boldsymbol{x}) = 0$. For any $\boldsymbol{x} \in \mathcal{V}$ and for each component C_i, entries of $\boldsymbol{x}$ in C_i must be constant. Otherwise, $R_{\boldsymbol{L}_G}(\boldsymbol{x}) > 0$. Therefore $\mathcal{V} \subseteq \text{span}\{1_{C_1}, \ldots 1_{C_z}\}$ where z counts the connected components in G. So $k = \dim(\mathcal{V}) \leq z$, which in English means that the number of connected components of G is at least k.

□

3.6. *Remark.* Basically, $\lambda_1(\boldsymbol{L}_G)$ is not informative at all since it is always zero. But $\lambda_2(\boldsymbol{L}_G) > 0$ iff G is connected.

In other words, if $\lambda_2 = 0$, then the graph is not connected. A intuitive question to ask is that what if λ_2 is very closed to 0, would that means that the graph is very closed to be disconnected? And the answer is yes. We'll see the proof in the following section.

4. The Normalized Laplacian of Graphs

4.1. Definition. Excluding isolated vertices(namely $\boldsymbol{D}$ is of full rank,) the **normalized Laplacian** matrix is

$$\boldsymbol{N}_G = \boldsymbol{D}^{-1/2}\boldsymbol{L}_G\boldsymbol{D}^{-1/2} = I - \boldsymbol{D}^{-1/2}\boldsymbol{A}_G\boldsymbol{D}^{-1/2}.$$

Basically a scaled Laplacian.

4.2. Exercise. We have

1. $\lambda_1(\boldsymbol{N}_G) = 0$ with a corresponding eigenvector $\boldsymbol{v}_1 = \boldsymbol{d}^{1/2}$, where $\boldsymbol{d} = (\deg(u))_{u\in V(G)}$.
2. $\lambda_k(\boldsymbol{N}_G) = 0$ if and only if G contains at least k connected component.

4.3. Lemma.

$$\lambda_2(\boldsymbol{N}_G) = \min_{\boldsymbol{x}\perp\boldsymbol{d}} \frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}.$$

Bizonyítás.

Firstly, by definition

$$\lambda_2(\boldsymbol{N}_G) = \min_{\boldsymbol{v}\perp\boldsymbol{d}^{1/2}} \frac{\boldsymbol{v}^\top \boldsymbol{N}_G \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}}$$

since the first eigenvector of $\boldsymbol{N}$ is $\boldsymbol{d}^{1/2}$. Now, define $\boldsymbol{x} = \boldsymbol{D}^{-1/2}\boldsymbol{v}$, so $\boldsymbol{v} = \boldsymbol{D}^{1/2}\boldsymbol{x}$. We have $\frac{\boldsymbol{v}^\top \boldsymbol{N}_G \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}} = \frac{\boldsymbol{v}^\top \boldsymbol{D}^{-1/2}\boldsymbol{L}_G\boldsymbol{D}^{-1/2}\boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}} = \frac{\boldsymbol{x}\boldsymbol{L}_G\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}$. Also, $\boldsymbol{v}\perp\boldsymbol{d}^{1/2}$ iff $\boldsymbol{x}\perp\boldsymbol{d}$. Because $0 = \boldsymbol{v}^\top\boldsymbol{d}^{1/2} = \boldsymbol{x}^\top\boldsymbol{D}^{1/2}\boldsymbol{d}^{1/2} = \boldsymbol{x}^\top\boldsymbol{d}$. Since the mapping $\boldsymbol{v} \mapsto \boldsymbol{D}^{-1/2}\boldsymbol{v} = \boldsymbol{x}$ is a bijection, we have $\lambda_2(\boldsymbol{N}_G) = \min_{\boldsymbol{x}\perp\boldsymbol{d}} \frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}$ as desired. □

4.4. *Note.* The above lemma is the reason why we look at $\boldsymbol{N}_G$-the second eigenvalue of normalized Laplacian is closely related to the conductance since they are minimizing the same objective but over different domains. Indeed, we can see the similarity between $\lambda_2(\boldsymbol{N}_G)$ and $\Phi(G)$:

$$\lambda_2(\boldsymbol{N}_G) = \min_{\boldsymbol{x}\perp\boldsymbol{d}} \frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}$$

and

$$\Phi(G) = \min_{\boldsymbol{x}=\mathbf{1}_S : 0<\boldsymbol{d}(S)\le\boldsymbol{d}(V)/2} \frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}$$

where $\boldsymbol{x} = \mathbf{1}_S$, $\boldsymbol{x}^\top\boldsymbol{L}_G\boldsymbol{x} = w_G(\partial_G S)$ and $\boldsymbol{x}^\top\boldsymbol{D}\boldsymbol{x} = \boldsymbol{d}(S)$.

5. Cheeger's Inequality: Statement

It turns out that $\lambda_2(\boldsymbol{N}_G)$ does not only tell us if G is connected or not. It actually measures how much G is well-connected.

5.1. Theorem (Cheeger's Inequality)**.** *We have*

$$\frac{\lambda_2(\boldsymbol{N}_G)}{2} \leq \Phi(G) \leq \sqrt{2\lambda_2(\boldsymbol{N}_G)}.$$

Furthermore, there is an algorithm that find a cut S where $\Phi_G(S) \leq \sqrt{2\lambda_2(\boldsymbol{N}_G)} \leq 2\sqrt{\Phi(G)}$.

5.2. *Note.*

1. What this means - G **is expander if and only if the second eigenvalue of $\boldsymbol{N}_G$ is big.**
2. So when $\Phi(G) = \Omega(1)$, this gives $O(1)$-approximation algorithm for $\Phi(G)$.
3. But the approximation can be very bad when $\Phi(G)$ is small. For example, when $\Phi(G) = O(\frac{1}{n})$, the algorithm return a cut where $\Phi_G(S) \leq O\left(\sqrt{\frac{1}{n}}\right)$ which can be much larger than $O(\frac{1}{n})$.

5.3. Question. *Is there a $O(1)$-approximation algorithm $\Phi(G)$ for any graph G? If so, this would refute the small-set expansion conjecture.*

- Best known approximation: $O(\sqrt{\log n})$ via ARV.
  - We saw $O(\log n)$-approx algorithm (which is tight because of the flow-cut gap).
- The Cheeger's Inequality is tight.
  - the second inequality: for cycle of size n, $\Phi(G) = \Theta(1/n)$ and $\lambda_2(\boldsymbol{N}_G) = O(1/n^2)$
  - the first inequality: for d-dimensional hypercube on $n = 2^d$ nodes, $\Phi(G) = 1/d$ and $\lambda_2(\boldsymbol{N}_G) = 2/d$
  - See proofs: https://lucatrevisan.github.io/teaching/expanders2016/lecture15.pdf

6. Easy Direction: Sparse Cut implies Small Eigenvalue

6.1. Lemma. $\lambda_2(\boldsymbol{N}_G)/2 \leq \Phi(G)$

Bizonyítás. Consider any $S \subset V$ where wlog $0 < \boldsymbol{d}(S) \leq \boldsymbol{d}(V)/2$. Define $\boldsymbol{y}$ based on S,

$$\boldsymbol{y} = \mathbf{1}_S - \sigma\mathbf{1}$$

where $\sigma = \boldsymbol{d}(S)/\boldsymbol{d}(V)$ is chosen so that $\boldsymbol{y} \perp \boldsymbol{d}$. We have

$$\lambda_2(\boldsymbol{N}_G) \leq \frac{\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y}}{\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y}}.$$

Note that $\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y} = w_G(\partial_G S)$, since $\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y} = \sum_{(u,v)\in E} w_{uv}(y_u - y_v)^2$ is translation invariant whence is the same as $\mathbf{1}_S \boldsymbol{L}_G \mathbf{1}_S = w_G(\partial_G S)$.
Next, we compute $\boldsymbol{y}^\top \boldsymbol{D}\boldsymbol{y}$

$$\begin{aligned}
\boldsymbol{y}^\top \boldsymbol{D}\boldsymbol{y} &= \sum_{u\in S} \boldsymbol{d}(u)(1-\sigma)^2 + \sum_{u\notin S} \boldsymbol{d}(u)\sigma^2 \\
&= \boldsymbol{d}(S)(1-\sigma)^2 + \boldsymbol{d}(V\setminus S)\sigma^2 \\
&= \boldsymbol{d}(S) - 2\sigma \boldsymbol{d}(S) + \sigma^2 \boldsymbol{d}(V) \\
&= \boldsymbol{d}(S)(1-\sigma) \\
&= \boldsymbol{d}(S)\frac{\boldsymbol{d}(V\setminus S)}{\boldsymbol{d}(V)} \geq \frac{\boldsymbol{d}(S)}{2}
\end{aligned}$$

So we have

$$\lambda_2(\boldsymbol{N}_G) \leq 2 \cdot \frac{w_G(\partial_G S)}{\boldsymbol{d}(S)}$$

for all S where $0 < \boldsymbol{d}(S) \leq \boldsymbol{d}(V)/2$. Therefore,

$$\lambda_2(\boldsymbol{N}_G) \leq 2 \cdot \Phi(G).$$

□

7. Harder direction: Small Eigenvalue implies Sparse Cut

7.1. Theorem. *Given any $x \perp \boldsymbol{d}$, we can find a cut S in G where*

$$\Phi_G(S) \leq \sqrt{2\frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}}$$

7.2. Corollary. $\Phi(G) \leq \sqrt{2\lambda_2(\boldsymbol{N}_G)}$.

Sketch of proof of Theorem 7.1. We have $\lambda_2(\boldsymbol{N}_G) = \min_{x\perp \boldsymbol{d}} \frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}$.

- Remark that it will be necessary that the theorem works for any $x \perp \boldsymbol{d}$
  - because we will show how to find $x \perp \boldsymbol{d}$ where $\frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}$ is *almost* minimum in the next lecture.
- Given a vector $\boldsymbol{x}$, there are three main steps for constructing S.
  1. Firstly, we define a *centered* vector $\boldsymbol{y}$ where $\frac{\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y}}{\boldsymbol{y}^\top \boldsymbol{D}\boldsymbol{y}} \leq \frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}$,
  2. then, we further define *non-negative* vector $\boldsymbol{z}$ where $\frac{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}} \leq \frac{\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y}}{\boldsymbol{y}^\top \boldsymbol{D}\boldsymbol{y}}$
  3. $\boldsymbol{z}$ will have a nice enough structure such that it determines a *threshold cut* S where $\Phi_G(S) \leq \sqrt{2\frac{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}}}$

□

We proceed to elaborate on the three steps in the following sections.

7.1. Centering

Without loss of generality assume that

$$\boldsymbol{x}_1 \leq \boldsymbol{x}_2 \leq \cdots \leq \boldsymbol{x}_n.$$

7.3. Definition. Say y is *centered w.r.t.* $\boldsymbol{d}$ if

$$\sum_{u:y_u<0} \boldsymbol{d}(u) \leq \boldsymbol{d}(V)/2 \quad \text{and} \quad \sum_{u:y_u>0} \boldsymbol{d}(u) \leq \boldsymbol{d}(V)/2$$

Namely the total volume of both vertices with strictly less then zero value in $\boldsymbol{y}$ and vertices with strictly greater then zero value in $\boldsymbol{y}$ are less than a half.

Proof of the first step of Theorem 7.1. Given an arbitrary vector $\boldsymbol{x}$, we can make it into a centered vector by translation.

Let j be the minimum index where $\sum_{u=1}^{j} \boldsymbol{d}(u) \geq \boldsymbol{d}(V)/2$. Set

$$\boldsymbol{y} = \boldsymbol{x} - x_j \mathbf{1}$$

So we have $y_j = 0$, $\sum_{u:y_u<0} \boldsymbol{d}(u) \leq \boldsymbol{d}(V)/2$ and $\sum_{u:y_u>0} \boldsymbol{d}(u) \leq \boldsymbol{d}(V)/2$. Namely, $\boldsymbol{y}$ is centered w.r.t. $\boldsymbol{d}$. Also observe that

$$\frac{\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y}}{\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y}} \leq \frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D} \boldsymbol{x}}$$

because (a) the numerator is not changing at all since $\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y} = (\boldsymbol{x} - x_j \mathbf{1})^\top \boldsymbol{L}_G (\boldsymbol{x} - x_j \mathbf{1}) = \boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}$, as $\mathbf{1} \in \ker(\boldsymbol{L}_G)$;
(b) and the denominator can only possibly be larger, i.e. $\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y} \geq \boldsymbol{x}^\top \boldsymbol{D} \boldsymbol{x}$. To see this, let $\boldsymbol{v}_s = \boldsymbol{x} + s\mathbf{1}$, where s is a translating parameter. The minimum of $\boldsymbol{v}_s^\top \boldsymbol{D} \boldsymbol{v}_s$ is achieved at s for which $\boldsymbol{v}_s^\top \boldsymbol{d} = 0$. This is because if we take the derivative of this value $\frac{\partial(\boldsymbol{v}_s^\top \boldsymbol{D} \boldsymbol{v}_s)}{\partial s} = \mathbf{1}^\top \boldsymbol{D} \boldsymbol{v}_s + \boldsymbol{v}_s^\top \boldsymbol{D} \mathbf{1} = 2\boldsymbol{d}^\top \boldsymbol{v}_s$, we know that when $\boldsymbol{d}^\top \boldsymbol{v}_s$ reaches 0 the value reaches its minimum. But notice that $\boldsymbol{x}$ is orthogonal to $\boldsymbol{d}$, where we have $\boldsymbol{x}^\top \boldsymbol{d} = 0$ and hence we must have $\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y} \geq \boldsymbol{x}^\top \boldsymbol{D} \boldsymbol{x}$.

□

7.2. Non-negative

We next proceed to show the second step of the proof. Given the vector $\boldsymbol{y}$ above, we decompose it into two vectors, one comprising the positive entries and the other comprising the negative entries.

$$\boldsymbol{y} = \boldsymbol{y}^+ - \boldsymbol{y}^-$$

where

$$y_j^+ = \begin{cases} y_j & y_j > 0 \\ 0 & \text{otherwise} \end{cases} \text{ and } y_j^- = \begin{cases} -y_j & y_j < 0 \\ 0 & \text{otherwise} \end{cases}$$

Note that $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$ has several neat properties

- have disjoint non-zero entries,

- are non-negative, and
- are small w.r.t. $\boldsymbol{d}$. More precisely, $\sum_{u:y_u^+>0} \boldsymbol{d}(u) \leq \boldsymbol{d}(V)/2$ and $\sum_{u:y_u^->0} \boldsymbol{d}(u) \leq \boldsymbol{d}(V)/2$.

Suppose the following claim is true.

7.4. Claim.

$$\min\{\frac{(\boldsymbol{y}^+)^\top \boldsymbol{L}_G \boldsymbol{y}^+}{(\boldsymbol{y}^+)^\top \boldsymbol{D} \boldsymbol{y}^+}, \frac{(\boldsymbol{y}^-)^\top \boldsymbol{L}_G \boldsymbol{y}^-}{(\boldsymbol{y}^-)^\top \boldsymbol{D} \boldsymbol{y}^-}\} \leq \frac{\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y}}{\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y}}$$

Then we can simply take one of $\boldsymbol{y}^+$ and $\boldsymbol{y}^-$ that makes the inequality holds to be $\boldsymbol{z}$ where

$$\frac{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{D} \boldsymbol{z}} \leq \frac{\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y}}{\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y}}$$

with $\boldsymbol{z}$ being non-negative and $\sum_{u:z_u>0} \boldsymbol{d}(u) \leq \boldsymbol{d}(V)/2$. It remains to prove the claim.

Proof of claim 7.4. We first show $\forall uv \in E, ((y_u^+ - y_v^+) - (y_u^- - y_v^-))^2 \geq \left((y_u^+ - y_v^+)^2 + (y_u^- - y_v^-)^2\right)$.
We will prove this by considering the contribution of every edge. Consider any edge (u, v). If both u and v such that $y_u, y_v \geq 0$ or $y_u, y_v \leq 0$, then (u, v) contributes the exact same to both sides of inequality.
Otherwise, if $y_u > 0 > y_v$, then (u, v) contributes $(y_u^+ + y_v^-)^2$ to the left, but $(y_u^+)^2 + (y_v^-)^2$ to the right, where we know the inequality $(y_u^+ + y_v^-)^2 \geq (y_u^+)^2 + (y_v^-)^2$ always holds.
Therefore, we have

$$\begin{aligned}
\frac{\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y}}{\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y}} &= \frac{\sum_{uv\in E} w_{uv}(y_u - y_v)^2}{\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y}} \\
&\quad \text{by decomposing and rearranging} \\
&= \frac{\sum_{uv\in E} w_{uv} \left((y_u^+ - y_v^+) - (y_u^- - y_v^-)\right)^2}{(\boldsymbol{y}^+)^\top \boldsymbol{D} \boldsymbol{y}^+ + (\boldsymbol{y}^-)^\top \boldsymbol{D} \boldsymbol{y}^- - 2((\boldsymbol{y}^-)^\top \boldsymbol{D} \boldsymbol{y}^+)} \\
&\quad \text{according to what we've just shown} \\
&\geq \frac{\sum_{uv\in E} w_{uv} \left((y_u^+ - y_v^+)^2 + (y_u^- - y_v^-)^2\right)}{(\boldsymbol{y}^+)^\top \boldsymbol{D} \boldsymbol{y}^+ + (\boldsymbol{y}^-)^\top \boldsymbol{D} \boldsymbol{y}^-} \\
&\quad \text{since } \forall a_1, a_2, b_1, b_2, \frac{a_1 + a_2}{b_1 + b_2} \geq \min\left(\frac{a_1}{b_1}, \frac{a_2}{b_2}\right) \\
&\geq \min\left\{\frac{\sum_{uv\in E} w_{uv}(y_u^+ - y_v^+)^2}{(\boldsymbol{y}^+)^\top \boldsymbol{D} \boldsymbol{y}^+}, \frac{\sum_{uv\in E} w_{uv}(y_u^- - y_v^-)^2}{(\boldsymbol{y}^-)^\top \boldsymbol{D} \boldsymbol{y}^-}\right\} \\
&= \min\left\{\frac{(\boldsymbol{y}^+)^\top \boldsymbol{L}_G \boldsymbol{y}^+}{(\boldsymbol{y}^+)^\top \boldsymbol{D} \boldsymbol{y}^+}, \frac{(\boldsymbol{y}^-)^\top \boldsymbol{L}_G \boldsymbol{y}^-}{(\boldsymbol{y}^-)^\top \boldsymbol{D} \boldsymbol{y}^-}\right\}
\end{aligned}$$

□

7.3. Threshold Cuts

It suffices to show the following lemma,

7.5. Lemma. *Given a non-negative $\boldsymbol{z}$ as above, there is a number τ where $S_\tau = \{u \mid \boldsymbol{z}_u \geq \tau\}$ such that*

$$\Phi_G(S_\tau) \leq \sqrt{2\frac{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}}}$$

7.6. *Note.* Visually, when we set all entries of $\boldsymbol{y}$ on a real axis, if $\boldsymbol{z} = \boldsymbol{y}^-$, then the cut S_τ would be a cut includes all vertices correspond to points including everything from the left most on the axis all the way through the smallest u such that $\boldsymbol{z}_u \geq \tau$. This is because when we defined $\boldsymbol{y}^-$ we flipped the sign of each entry contained in this part in $\boldsymbol{y}$;
Conversely, when $\boldsymbol{z} = \boldsymbol{y}^+$, then the cut S_τ would includes all vertices correspond to points including everything from the right most on the axis all the way through the smallest u such that $\boldsymbol{z}_u \geq \tau$.

To show the Lemma, w.l.o.g we may assume $\max_u z_u = 1$ since scaling does not change the ratio $\frac{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}}$. We are going to use a probabilistic argument, where we first let $\tau > 0$ be sampled such that τ^2 **is uniformly distributed in** $[0, 1]$. Namely, $\Pr[\tau \leq \alpha] = \Pr[\tau^2 \leq \alpha^2] = \alpha^2$ for all $\alpha \in [0, 1]$.[4] Let $S_\tau = \{u \mid \boldsymbol{z}_u \geq \tau\}$, then the key claim is that

7.7. Claim.

$$\frac{\mathbb{E}_\tau w_G(\partial_G S_\tau)}{\mathbb{E}_\tau \boldsymbol{d}(S_\tau)} \leq \sqrt{2\frac{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}}}$$

Given the claim, even though

$$\mathbb{E}_\tau \left(\frac{w_G(\partial_G S_\tau)}{\boldsymbol{d}(S_\tau)} \right) \neq \frac{\mathbb{E}_\tau w_G(\partial_G S_\tau)}{\mathbb{E}_\tau \boldsymbol{d}(S_\tau)},$$

we are done since there must exist a τ s.t.

$$\min_\tau \frac{w_G(\partial_G S_\tau)}{\boldsymbol{d}(S_\tau)} \leq \frac{\mathbb{E}_\tau w_G(\partial_G S_\tau)}{\mathbb{E}_\tau \boldsymbol{d}(S_\tau)}$$

since $\frac{\sum_{i=1}^k a_i}{\sum_{i=1}^k b_i} \geq \min_i \frac{a_i}{b_i}$. Since we have $\boldsymbol{d}(S_\tau) \leq \sum_{u:z_u>0} \boldsymbol{d}(u) \leq \boldsymbol{d}(V)/2$,

$$\Phi_G(S_\tau) = \frac{w_G(\partial_G S_\tau)}{\boldsymbol{d}(S_\tau)}$$

which implies

$$\Phi_G(S_\tau) \leq \sqrt{2\frac{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}}}$$

as desired.

[4]For the existential argument, we consider the probability density function $f(x) = 2x$ because $\int_{x=0}^{\alpha} 2x = x^2$.

Proof of claim 7.7. To show $\frac{\mathbb{E}_\tau w_G(\partial_G S_\tau)}{\mathbb{E}_\tau \boldsymbol{d}(S_\tau)} \leq \sqrt{2\frac{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}}}$, for the denominator,

$$\begin{aligned}\mathbb{E}_\tau \boldsymbol{d}(S_\tau) &= \sum_u \boldsymbol{d}(u) \Pr[u \in S_\tau] \\ &= \sum_u \boldsymbol{d}(u) \Pr[\tau \leq z_u] \\ &= \sum_u \boldsymbol{d}(u) z_u^2 = \boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}\end{aligned}$$

For the numerator,

$$\mathbb{E}_\tau w_G(\partial_G S_\tau) = \sum_{uv} w_{uv} \Pr[\{u, v\} \text{ is cut}]$$

since τ is the exact value that separate the vertices within the cut and vertices that are not in the cut

$$= \sum_{uv} w_{uv} \Pr[z_u < \tau \leq z_v] \qquad \text{assuming } z_v > z_u$$

this probability is exactly $\Pr[\tau \leq z_v] - \Pr[\tau \leq z_u]$ i.e.

$$= \sum_{uv} w_{uv}(z_v^2 - z_u^2)$$

$$= \sum_{uv} w_{uv}(z_v - z_u)(z_v + z_y)$$

By Cauchy-Schwarz, we have

$$\begin{aligned}\sum_{uv} w_{uv}(z_v - z_u)(z_v + z_y) &\leq \sqrt{\sum_{uv} w_{uv}(z_v - z_u)^2} \cdot \sqrt{\sum_{uv} w_{uv}(z_v + z_u)^2} \\ &\leq \sqrt{\boldsymbol{z}^\top \boldsymbol{L}_G \boldsymbol{z}} \cdot \sqrt{2\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}}\end{aligned}$$

since

$$\begin{aligned}\sum_{uv} w_{uv}(z_v + z_u)^2 &\leq \sum_{uv} w_{uv}(2z_v^2 + 2z_u^2) \\ &= \sum_u 2\deg(u) \cdot z_u^2 \\ &= \sum_u 2\boldsymbol{d}(u) \cdot z_u^2 = 2\boldsymbol{z}^\top \boldsymbol{D}\boldsymbol{z}.\end{aligned}$$

[5] □

8. Conclusion: Fiedler's Algorithm

The proof above is in fact algorithmic - we can find a sparse cut S where $\Phi_G(S) \leq \sqrt{2\lambda_2(\boldsymbol{N}_G)} \leq 2\sqrt{\Phi(G)}$. Specifically, the algorithm is as follows:

[5] Note that it is this last step where we need that $\boldsymbol{d}(u) \geq \deg(u)$. So it would not work if we want to prove a generalized Cheeger's inequality for $\Phi(G, \boldsymbol{d})$ for arbitrary vector $\boldsymbol{d}$

1. Compute the second eigenvector $\boldsymbol{v}_2$ of $\boldsymbol{N}_G$. We have $\boldsymbol{v}_2 \perp \boldsymbol{d}^{1/2}$ and $\frac{\boldsymbol{v}_2^\top \boldsymbol{N}_G \boldsymbol{v}_2}{\boldsymbol{v}_2^\top \boldsymbol{v}_2} = \lambda_2(\boldsymbol{N}_G)$.
2. Compute $\boldsymbol{x} = \boldsymbol{D}^{-1/2}\boldsymbol{v}_2$. We have $x \perp \boldsymbol{d}$ and $\frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}} = \lambda_2(\boldsymbol{N}_G)$.
3. Sort indices such that $x_1 \leq \cdots \leq x_n$.
4. Define $S_i' = \{1, \ldots, i\}$ for all $i < n$, where $\forall j \in [i]$, j represents the vertex in G corresponding to entry x_j.
5. Return S^* with minimum $\Phi_G(S_i')$ among all $i \in \{1, \ldots, n-1\}$.

8.1. Lemma (Correctness)**.** *Consider the cut S_τ from the proof above. We have $(S_\tau, V \setminus S_\tau) = (S_i', V \setminus S_i')$ for some i. So $\Phi_G(S^*) \leq \Phi_G(S_\tau) \leq \sqrt{2\lambda_2(\boldsymbol{N}_G)}$.*

8.2. Exercise. If instead of $\boldsymbol{v}_2$, we are given a vector $\boldsymbol{v} \perp \boldsymbol{d}^{1/2}$ where $\frac{\boldsymbol{v}^\top \boldsymbol{N}_G \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}} \leq \lambda_2(\boldsymbol{N}_G) + \epsilon$. We would have $\Phi_G(S^*) \leq \sqrt{2\lambda_2(\boldsymbol{N}_G) + 2\epsilon}$

Solution. Let $\boldsymbol{x} = \boldsymbol{D}^{-1/2}\boldsymbol{v}$, then $\boldsymbol{x} \perp \boldsymbol{d}$. Theorem 7.1 shows that as long as $\boldsymbol{x} \perp \boldsymbol{d}$, the S^* found by Fiedler's algorithm satisfies $\Phi_G(S^*) \leq \sqrt{2\frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}}$. Since $\frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}} = \frac{\boldsymbol{v}^\top \boldsymbol{N}_G \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}} \leq \lambda_2(\boldsymbol{N}_G) + \epsilon, \implies \Phi_G(S^*) \leq \sqrt{2\lambda_2(\boldsymbol{N}_G) + 2\epsilon}$. □

8.3. Lemma (Time)**.** *Given the second eigenvector $\boldsymbol{v}_2$, we can compute S^* in $O(m + n \log n)$ time.*

- Modulo the time for computing $\boldsymbol{v}_2$, the algorithm is very fast and gives a good approximation when $\Phi(G)$ is big.
- In the next lecture, we will show how to obtain the vector which is almost as good as $\boldsymbol{v}_2$ in $\tilde{O}(m/\Phi(G))$ or even $\tilde{O}(m)$ time.

8.4. Question (Possible project)**.** *I think that using ideas from Section 7 of https://arxiv.org/pdf/1910.07950.pdf, we should be able to improve the above algorithm/analysis in two ways*

1. *The analysis based on "median cut" will be more intuitive.*
2. *The algorithm will only need to "query" only $O(\log^2 n)$ cuts of the form S_i' instead of $n-1$ cuts.*

The analysis in the paper is for PageRank but it can possibly work for Cheeger's cut too.

9. Extensions of Cheeger's Inequality

- Extension to Directed graphs [6]
- Improvement when vertex expansion is high[7]
- What about quantitative implications of λ_k for $k > 2$?
  - $\lambda_k(\boldsymbol{N}_G) = 0$ iff G has at least k connected components
  - What if $\lambda_k(\boldsymbol{N}_G) \approx 0$? We can also say that G can be clustered into k parts by deleting few edges.

[6] https://core.ac.uk/download/pdf/206608976.pdf
[7] https://arxiv.org/pdf/1504.00686.pdf

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 8: λ_n and The Power Method

September 22, 2021

Instructor: Thatchaphol Saranurak

Scribe: Peter Ly

We will mainly talk about two things for this lecture:

1. $\lambda_n(\mathbf{N})$ for finding max cuts (similar to $\lambda_2(\mathbf{N})$ for finding sparse cuts)
2. The power method: a fast algorithm for finding approximate eigenvectors.

1 $\lambda_n(\mathbf{N})$ vs Max Cut

We saw from Cheeger's Inequality that $\lambda_2(\mathbf{N}_G)$ corresponds to sparsest cut of the graph G. That is, $\lambda_2(\mathbf{N}_G)$ is small if and only if G has a sparse cut. In the extreme case where $\lambda_2(\mathbf{N}_G) = 0$, then we have that G is not connected.

There is an analogous result for $\lambda_n(\mathbf{N}_G)$ and the size of the maximum cut of G. $\lambda_n(\mathbf{N}_G)$ is large if and only if the size of the maximum cut is large - in other words, the graph is close to being bipartite. In the extreme case where $\lambda_n(\mathbf{N}_G) = 2$, then we have that G contains a connected component which is bipartite. We will show the proof of the extreme case and mention what happens if $\lambda_n(\mathbf{N}_G)$ is close to 2.

Lemma 1.1. *We have*

1. $\lambda_n(\mathbf{N}_G) \leq 2$. *So all eigenvalues of* $\mathbf{N}_G$ *are in the range of* $[0, 2]$.
2. $\lambda_n(\mathbf{N}_G) = 2$ *if and only if G contains a connected component which is bipartite.*

Proof. Recall

$$\lambda_n(\mathbf{N}_G) = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\mathbf{x}^\top \mathbf{N}_G \mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\mathbf{x}^\top \mathbf{L}_G \mathbf{x}}{\mathbf{x}^\top \mathbf{D}\mathbf{x}}$$

1. To prove $\lambda_n(\mathbf{N}_G) \leq 2$, observe that

$$2\mathbf{x}^\top \mathbf{D}\mathbf{x} - \mathbf{x}\mathbf{L}_G\mathbf{x} = \sum_u 2\deg(u)x_u^2 - \sum_{uv\in E} w_{uv}(x_u - x_v)^2 = \sum_{uv\in E} w_{uv}(x_u + x_v)^2$$

So

$$2 - \lambda_n(\mathbf{N}_G) = 2 - \max_{\mathbf{x}\neq\mathbf{0}} \frac{\mathbf{x}\mathbf{L}_G\mathbf{x}}{\mathbf{x}^\top\mathbf{D}\mathbf{x}} = \min_{\mathbf{x}\neq\mathbf{0}} \frac{2\mathbf{x}^\top\mathbf{D}\mathbf{x} - \mathbf{x}\mathbf{L}_G\mathbf{x}}{\mathbf{x}^\top\mathbf{D}\mathbf{x}} = \min_{\mathbf{x}\neq\mathbf{0}} \frac{\sum_{uv\in E} w_{uv}(x_u + x_v)^2}{\mathbf{x}^\top\mathbf{D}\mathbf{x}} \geq 0.$$

2. ($\Leftarrow$) Now, suppose there is a bipartite component C where $V(C) = L \cup R$ (see Figure 1 for the image to keep in mind). Define a vector $\mathbf{x}$ where

$$x_u = \begin{cases} 1 & u \in L \\ -1 & u \in R \\ 0 & u \notin V(C) \end{cases}$$

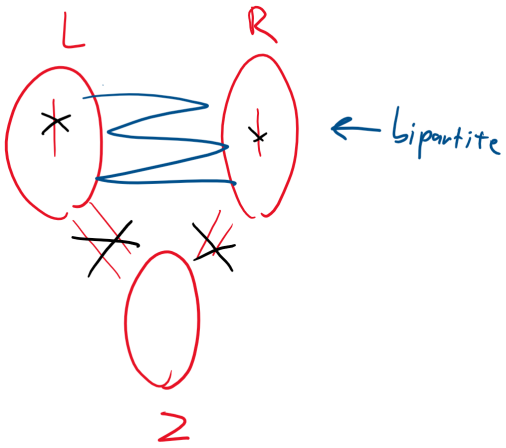
We have (from the proof of (1) in the lemma) that

$$2 - \lambda_n(\mathbf{N}_G) = \min_{\mathbf{x} \neq \mathbf{0}} \frac{\sum_{uv \in E} w_{uv}(x_u + x_v)^2}{\mathbf{x}^\top \mathbf{D}\mathbf{x}} = 0$$

So $\lambda_n(\mathbf{N}_G) = 2$.

($\Rightarrow$) Suppose $\lambda_n(\mathbf{N}_G) = 2$. Then there is some non-zero $\mathbf{x}$ such that $\sum_{uv \in E} w_{uv}(x_u + x_v)^2 = 0$. This happens if and only if $x_u = -x_v$ for all edges $(u, v) \in E$. Define $L = \{u \mid x_u < 0\}$, $R = \{u \mid x_u > 0\}$, and $Z = \{u \mid x_u = 0\}$ (see Figure 1 for an example of how the components relate). Since $\mathbf{x} \neq \mathbf{0}$, we have that $L \cup R$ is non-empty. Further, there are no edges between $L \cup R$ and Z. It follows that $L \cup R$ is a bipartite component of G.

Figure 1: $L \cup R$ form a bipartite component of G. There are no edges between $L \cup R$ and Z. Z may have edges inside of it, but those edges contribute 0 to the sum.



□

When we have that G is connected, then the only component is G itself, and Lemma 1.1 tells us that $\lambda_n(\mathbf{N}_G) = 2$ if and only if G is bipartite i.e. there is a cut that cuts all edges.

If we instead have that $\lambda_n(\mathbf{N}_G)$ is close to 2, then we can show that G is "close to being bipartite" - there is a cut which cuts almost every edge of G. This result was shown by Trevisan - more precisely, Trevisan showed a Cheeger's type inequality for λ_n:

$$\frac{2 - \lambda_n}{2} \leq \beta(G) \leq \sqrt{2(2 - \lambda_n)}$$

where $\beta(G)$ measures bipartiteness of G (see his lecture notes[1]).

[1] https://lucatrevisan.github.io/teaching/expanders2016/lecture05.pdf

2 The Power Method

Last time, we showed that we can construct a sparse cut S of G satisfying $\Phi_G(S) \leq \sqrt{2\lambda_2(\mathbf{N}_G)} \leq 2\sqrt{\Phi(G)}$ by using Fiedler's algorithm. The only slow step for constructing such an S was computing the second eigenvector $\mathbf{v}_2$ of $\mathbf{N}_G$. Now, we will show how to compute an "approximate second smallest eigenvector" in near-linear time. We can then use the approximate second smallest eigenvector in Fiedler's algorithm to obtain a sparse cut satisfying $\Phi_G(S) \leq \sqrt{2(\lambda_2(\mathbf{N}_G) + \varepsilon)}$.

We will begin by presenting how to find an approximation to the largest eigenvector and second largest eigenvector, and then we will show how to use the techniques for compuing the largest and second largest eigenvector to compute an approximate second smallest eigenvector. The technique we will use to compute these approximations is called the Power Method.

2.1 Largest Eigenvalue of M

This algorithm gives an approximation to the largest eigenvalue (and, in fact, its corresponding eigenvector).

POWER
Input: a PSD matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ and parameter k

1. Sample uniformly $\mathbf{x} \in \{-1, 1\}^n$
2. Return $\mathbf{y} = \mathbf{M}^k \mathbf{x}$.

POWER has runtime $O(k(n + \text{nnz}(\mathbf{M}))$ where $\text{nnz}(\mathbf{M})$ is the number of non-zero entries of $\mathbf{M}$.

Let Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ be eigenvalues of $\mathbf{M}$ (we have that all eigenvalues are non-negative because $\mathbf{M}$ is PSD).

Theorem 2.1 (Performance Guarantee of POWER)**.** *For any $\epsilon > 0$, with probability at least 3/16, the algorithm* POWER *returns* $\mathbf{y}$ *such that*

$$\frac{\mathbf{y}^\top \mathbf{M} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \geq (1 - \epsilon)\lambda_1 \cdot \frac{1}{1 + 4n(1 - \epsilon)^{2k}}.$$

If we take $k = O(\log(n)/\epsilon)$, then we have that $\frac{\mathbf{y}^\top \mathbf{M} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \geq (1 - O(\epsilon))\lambda_1$ - that is we have a multiplicative approximation for the first eigenvector with constant probability. We can amplify this probability to at least $1 - \frac{1}{\text{poly}(n)}$ by repeating POWER independently $O(\log n)$ times and taking the vector y^* which maximizes $\frac{\mathbf{y}^\top \mathbf{M} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}}$.

Analysis of the POWER algorithm. Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be eigenvectors of $\mathbf{M}$. We can write

$$\mathbf{M} = \sum_{i=n} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top \text{ and } \mathbf{M}^k = \sum_{i=n} \lambda_i^k \mathbf{v}_i \mathbf{v}_i^\top$$

by spectral decomposition. We can then write

$$\mathbf{x} = \sum_i a_i \mathbf{v}_i$$

where $a_i = \langle \mathbf{x}, \mathbf{v}_i \rangle$ i.e. express $\mathbf{x}$ in the vector space spanned by the eigenvectors of $\mathbf{M}$. We can then write

$$\mathbf{y} = \mathbf{M}^k \mathbf{x} = \sum_i a_i \lambda_i^k \mathbf{v}_i.$$

Let us first see on a high level why the power method works.

Intuition. Suppose $\lambda_1 = 1$ and all $\lambda_2, \ldots, \lambda_n \leq 1/2$. Then we can write

$$\mathbf{y} = a_1 \mathbf{v}_1 + \sum_{i \geq 2} a_i \lambda_i^k \mathbf{v}_i.$$

When $k = \Theta(\log n)$, then each $\lambda_i^k \leq \frac{1}{\text{poly}(n)}$ and the second term becomes vanishingly small. Note that here we need $\mathbf{M}$ to be PSD to guarantee that each λ_i^k was non-negative so that each term with λ_i^k for $i \geq 2$ would actually go to zero - otherwise we might have a negative eigenvalue < -1 and some terms of the sum might not go to zero. Now suppose we can show that $|a_1| = |\langle \mathbf{x}, \mathbf{v}_1 \rangle|$ is not very small, then we get $\mathbf{y} \approx \pm \mathbf{v}_1$, so $\frac{\mathbf{y}^\top \mathbf{M} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \approx \lambda_1$.

Making it formal. There were two points we were informal about in our intuition:

1. Why should $|a_1| = |\langle \mathbf{x}, \mathbf{v}_1 \rangle|$ be not too small?

   This is very natural to expect - essentially we are projecting $\mathbf{v}_1$ in a random direction - the magnitude of this projection is likely to be comparable to $\mathbf{v}_1$. We prove this formally in the exercise session. To be exact, we show

   Lemma 2.2. *For any vector $\mathbf{v} \in \mathbb{R}^n$ where $\|\mathbf{v}\| = 1$, sample $\mathbf{x} \in \{-1, 1\}^n$. Then*

   $$\Pr\left[|\langle \mathbf{x}, \mathbf{v} \rangle| \geq \frac{1}{2} \right] \geq \frac{3}{16}.$$

   In particular, with good probability $|a_1|$ is not too small.

2. There is no reason for there to be a large gap between λ_1 and $\lambda_2, \ldots, \lambda_n$. What if λ_2 is close to λ_1? We will show that even if we do not have such a gap, we still obtain a large value for $\frac{\mathbf{y}^\top \mathbf{M} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}}$.

   Fix $\varepsilon > 0$, and define ℓ so that λ_i can be classified as follows: $\lambda_1, \ldots, \lambda_\ell \geq (1 - \varepsilon)\lambda_1$ and $(1 - \varepsilon)\lambda_1 > \lambda_{\ell+1}, \ldots, \lambda_n$.

   We have that $\mathbf{y} = \sum_i a_i \lambda_i^k \mathbf{v}_i$ (shown above) and

   $$\mathbf{y}^\top \mathbf{y} = \sum_i a_i^2 \lambda_i^{2k} \text{ and } \mathbf{y}^\top \mathbf{M} \mathbf{y} = \sum_i a_i^2 \lambda_i^{2k+1}$$

   from the orthogonality of the eigenvectors from the spectral decomposition of $\mathbf{M}$.

   Our goal is to show that $\frac{\mathbf{y}^\top \mathbf{M} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}}$ is big i.e. show a lower bound on the numerator and an upper bound on the denominator.

   By truncating terms in $\mathbf{y}^\top \mathbf{M} \mathbf{y}$ and then applying the bound $\lambda_\ell \geq (1 - \varepsilon)\lambda_1$, we obtain the following lower bound on $\mathbf{y}^\top \mathbf{M} \mathbf{y}$:

   $$\mathbf{y}^\top \mathbf{M} \mathbf{y} \geq \sum_{i=1}^{\ell} a_i^2 \lambda_i^{2k+1} \geq (1 - \epsilon)\lambda_1 \cdot \sum_{i=1}^{\ell} a_i^2 \lambda_i^{2k}.$$

Recall that

$$\mathbf{y}^\top \mathbf{y} = \sum_{i=1}^{\ell} a_i^2 \lambda_i^{2k} + \sum_{i=\ell+1}^{n} a_i^2 \lambda_i^{2k}$$

We can upper bound the $\sum_{i=\ell+1}^{n} a_i^2 \lambda_i^{2k}$ term (and therefore $\mathbf{y}^\top \mathbf{y}$) as follows:

$$\begin{aligned}
\sum_{i=\ell+1}^{n} a_i^2 \lambda_i^{2k} &< (1-\epsilon)^{2k} \quad {\lambda_1}^{2k} \quad \sum_{i=\ell+1}^{n} a_i^2 && (\ell \text{ defined so } (1-\varepsilon)\lambda_1 > \lambda_{\ell+1}, \ldots, \lambda_n) \\
&\leq (1-\epsilon)^{2k} \quad {\lambda_1}^{2k} \quad \|\mathbf{x}\|^2 \\
&\leq (1-\epsilon)^{2k} \quad a_1^2 {\lambda_1}^{2k} \quad \frac{\|\mathbf{x}\|^2}{a_1^2} \\
&\leq (1-\epsilon)^{2k} \quad \sum_{i=1}^{\ell} a_i^2 \lambda_i^{2k} \quad 4n && (\text{using } a_1^2 = |\langle \mathbf{x}, \mathbf{v}_1 \rangle|^2 \geq \tfrac{1}{4})
\end{aligned}$$

So we can upper bound $\mathbf{y}^\top \mathbf{y}$ by

$$\mathbf{y}^\top \mathbf{y} \leq \sum_{i=1}^{\ell} a_i^2 \lambda_i^{2k} \cdot (1 + 4n(1-\epsilon)^{2k})$$

Putting the bounds together, we have with probability at least 3/16

$$\frac{\mathbf{y}^\top \mathbf{M} \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \geq (1-\epsilon)\lambda_1 \cdot \frac{1}{1 + 4n(1-\epsilon)^{2k}}$$

2.2 Second Largest Eigenvalue of M

What if we want the second largest eigenvalue of $\mathbf{M}$? Just run the same algorithm after translation so that the vector we pick is in the space orthogonal to the first eigenvector $\mathbf{v}_1$.

POWER2
Input: a PSD matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$, vector $\mathbf{v}_1$, and parameter k

1. Sample uniformly $\mathbf{x}' \in \{-1, 1\}^n$
2. $\mathbf{x} \leftarrow \mathbf{x}' - \mathbf{v}_1 \langle \mathbf{x}', \mathbf{v}_1 \rangle$ (and so $\mathbf{x} \perp \mathbf{v}_1$)
3. Return $\mathbf{y} = \mathbf{M}^k \mathbf{x}$.

We can write

$$\mathbf{x} = \sum_{i=2}^{n} a_i \mathbf{v}_i$$

where $a_i = \langle \mathbf{x}, \mathbf{v}_i \rangle$ as above. We only need to sum for $i = 2$ to n because $\mathbf{x} \perp \mathbf{v}_1$ by step 2 of POWER2. We also have that $a_2 \geq 1/2$ with probability at least 3/16. Performing the same analysis as above for

$$\mathbf{y} = \sum_{i=2}^{n} a_i \lambda_i^k \mathbf{v}_i$$

and we have

$$\frac{\mathbf{y}^\top \mathbf{M}\mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \geq (1-\epsilon)\lambda_2 \cdot \frac{1}{1+4n(1-\epsilon)^{2k}}$$

with probability at least 3/16.

Theorem 2.3. *Suppose that* $\mathbf{v}_1$ *is the first eigenvector of* $\mathbf{M}$*. For any* $\epsilon > 0$*, with probability* $\geq 3/16$*, the algorithm* POWER2 *returns* $\mathbf{y} \perp \mathbf{v}_1$ *such that*

$$\frac{\mathbf{y}^\top \mathbf{M}\mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \geq (1-\epsilon)\lambda_2 \cdot \frac{1}{1+4n(1-\epsilon)^{2k}}.$$

2.3 Second Smallest Eigenvalue of $\mathbf{N}_G$

Now that we have an algorithm for the second largest eigenvalue of a PSD matrix, we can show the algorithm to get the second smallest eigenvalue of $\mathbf{N}_G$ to use it for Cheeger's cut. Recall that all eigenvalues of $\mathbf{N}_G$ are in $[0, 2]$. If we instead work with the matrix

$$\mathbf{M} = 2\mathbf{I} - \mathbf{N}_G = \mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}.$$

we can get the second smallest eigenvalue of $\mathbf{N}_G$ by finding the second largest eigenvalue of $\mathbf{M}$. If $0 = \lambda_1 \leq \cdots \leq \lambda_n \leq 2$ are eigenvalues of $\mathbf{N}_G$, then the eigenvalues of $\mathbf{M}$ are

$$2 = 2 - \lambda_1 \geq 2 - \lambda_2 \geq \cdots \geq 2 - \lambda_n \geq 0$$

and so $\mathbf{M}$ is PSD and we can apply POWER2 to $\mathbf{M}$.

To apply POWER2, we need the eigenvector $\mathbf{v}_1$ corresponding to the largest eigenvalue $2 - \lambda_1$ of $\mathbf{M}$. The eigenvector is the same as λ_1 of $\mathbf{N}_G$ which is $\mathbf{v}_1 = \mathbf{d}^{1/2}$. By running POWER2 on $\mathbf{M}$ with $\mathbf{v}_1$, we get $\mathbf{y} \perp \mathbf{d}^{1/2}$ where $\mathbf{y}^\top \mathbf{M}\mathbf{y} \geq (2 - \lambda_2 - \epsilon) \cdot \mathbf{y}^\top \mathbf{y}$.

However

$$\begin{aligned} & \mathbf{y}^\top (2\mathbf{I} - \mathbf{N}_G)\mathbf{y} \geq (2 - \lambda_2 - \epsilon) \cdot \mathbf{y}^\top \mathbf{y} \\ \iff & (\lambda_2 + \epsilon)\mathbf{y}^\top \mathbf{y} \geq \mathbf{y}^\top \mathbf{N}_G \mathbf{y} \end{aligned}$$

so we have that

$$\frac{\mathbf{y}^\top \mathbf{N}_G \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \leq (\lambda_2 + \epsilon)$$

in time $\widetilde{O}(m/\epsilon)$.

Given $\mathbf{y}$, Fiedler's algorithm computes a cut S where $\Phi_G(S) \leq \sqrt{2(\lambda_2 + \epsilon)}$ in time $O(m+n \log n)$.

However, using $\mathbf{M} = 2(\mathbf{I} - \mathbf{N}_G)$ gives an *additive* approximation, rather than a multiplicative approximation. If we want a $(1 + \epsilon')$ multiplicative approximation, then we need to choose $\epsilon = \epsilon'\lambda_2$. Choosing such ϵ gives a time complexity of $\widetilde{O}(m/(\epsilon'\lambda_2)) \leq \widetilde{O}(m/(\epsilon'\Phi(G)^2))$ because $\Phi(G) \leq \sqrt{2\lambda_2}$ - when $\Phi(G)$ is small, obtaining a $(1 + \epsilon')$ multiplicative approximation is slow.

Question 2.4 (Open Problem)**.** Given a graph G with a promise that either $\Phi(G) \geq 0.1$ or $\Phi(G) \leq 1/\log^{100} n$, can we decide which case it is in $\widetilde{O}(m)$ **deterministically**?

We presented the Fiedler's algorithm which runs in time $\widetilde{O}(m)$, but the algorithm is not deterministic because we sample a vector on the hypercube uniformly at random. The current best known deterministic algorithm runs in time $O(m^{1.01})$ [CGLNPS'20] [2]. If you can solve this problem, you will likely have a successful Ph.D.; Most fast, deterministic algorithms have this problem as a bottleneck, so if you can solve it, then it should lead to several improvements in existing algorithms!

2.4 Let's see examples and get more intuition

Let's interpret what the power method does.

Suppose that the graph G is d-regular. Then we have

$$\mathbf{M} = 2\mathbf{I} - \mathbf{N}_G = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{I} + \frac{\mathbf{A}}{d}.$$

Let's consider the matrix

$$\mathbf{M}' = \frac{\mathbf{M}}{2} = \frac{\mathbf{I}}{2} + \frac{\mathbf{A}}{2d}$$

This matrix is easier to work with because its eigenvalues are between $[0, 1]$. It is just a scaled version of $\mathbf{M}$.

Consider

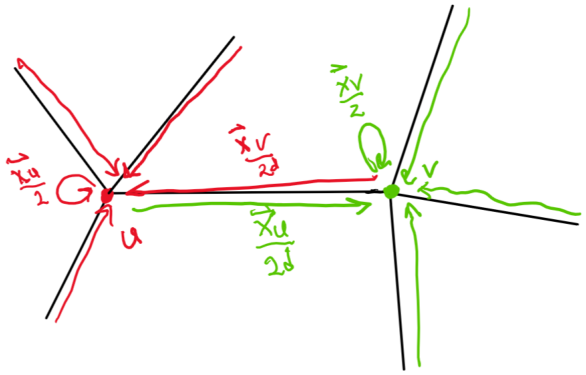
$$\mathbf{x}' = \mathbf{M}'\mathbf{x} = \left(\frac{\mathbf{I}}{2} + \frac{\mathbf{A}}{2d}\right)\mathbf{x}$$

which represents (up to scaling) one matrix multiplication in the power method. We can write

$$\mathbf{x}'_u = \frac{\mathbf{x}_u}{2} + \sum_{(u,v)\in E} \frac{\mathbf{x}_v}{2d}.$$

In total, there are k rounds in the power method and by the equation, in each round, each node u updates its value as a weighted average between its one value and its neighbors' values. This looks like a diffusion process like in Figure 2.

Figure 2: u and v are both updating their values. Because (u, v) are connected, they send some of their value to each other in addition to receiving value from their other neighbors. They also retain half of their value.

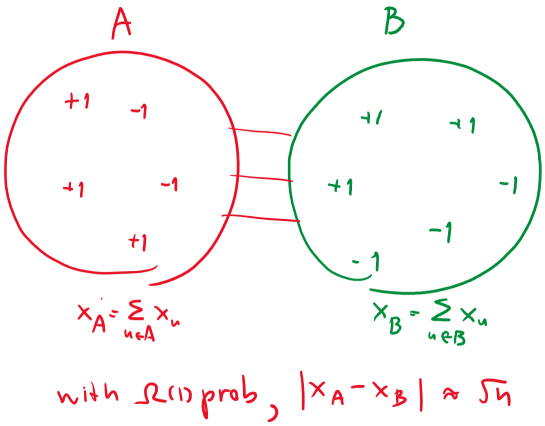


[2] https://arxiv.org/pdf/1910.08025.pdf

See this video[3] about the algorithm in action for cycles and hypercubes.

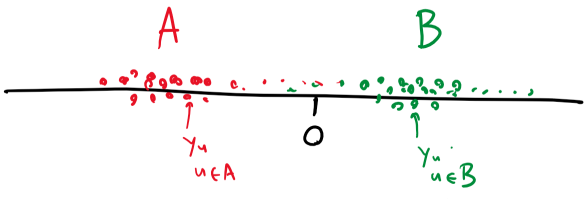
So, intuitively, why should this process give sparse cuts? Suppose there is a balanced sparse cut (A, B) (as in Figure 3) but A and B are themselves well-connected respectively. Denote $x_A = \sum_{u \in A} \mathbf{x}_u, x_B = \sum_{u \in B} \mathbf{x}_u$. In the beginning, since $\mathbf{x}$ is chosen uniformly from the hypercube, in expectation x_A and x_B are 0. Wlog we assume that $x_A \approx -\sqrt{n} < 0$ and $x_B \approx \sqrt{n} > 0$ (since $|x_A| = |x_B| = \Theta(\sqrt{n})$ with $\Omega(1)$ probability.) Then, after updating $\mathbf{x} = (\mathbf{M}')^k \mathbf{x}$, the sums won't change too much, i.e. $x_A \approx -\sqrt{n}$ and $x_B \approx \sqrt{n}$ since there are few edges connecting between A and B whence tiny amount of weight got send between A and B. What's more, since the process allow vertices to average their value with their neighbors, and vertices in A will mainly average with vertices within A, whence the value of vertices in A will be close to each other and most of them will be negative since we started with $x_A < 0$. Similarly, the values of vertices in B will still be close to each other and most of them will be positive.

Figure 3: (A, B) is a balanced sparse cut. Each vertex was assigned to -1 or 1 uniformly.



Fiedler's algorithm just sorts vertices according to $\mathbf{y} = (\mathbf{M}')^k \mathbf{x}$ (as shown in Figure 4) and sweeps through the order for the sparsest cut among the nested cuts. One of these cuts should be close to (A, B) so it should be sparse. This should also hold in the general case.

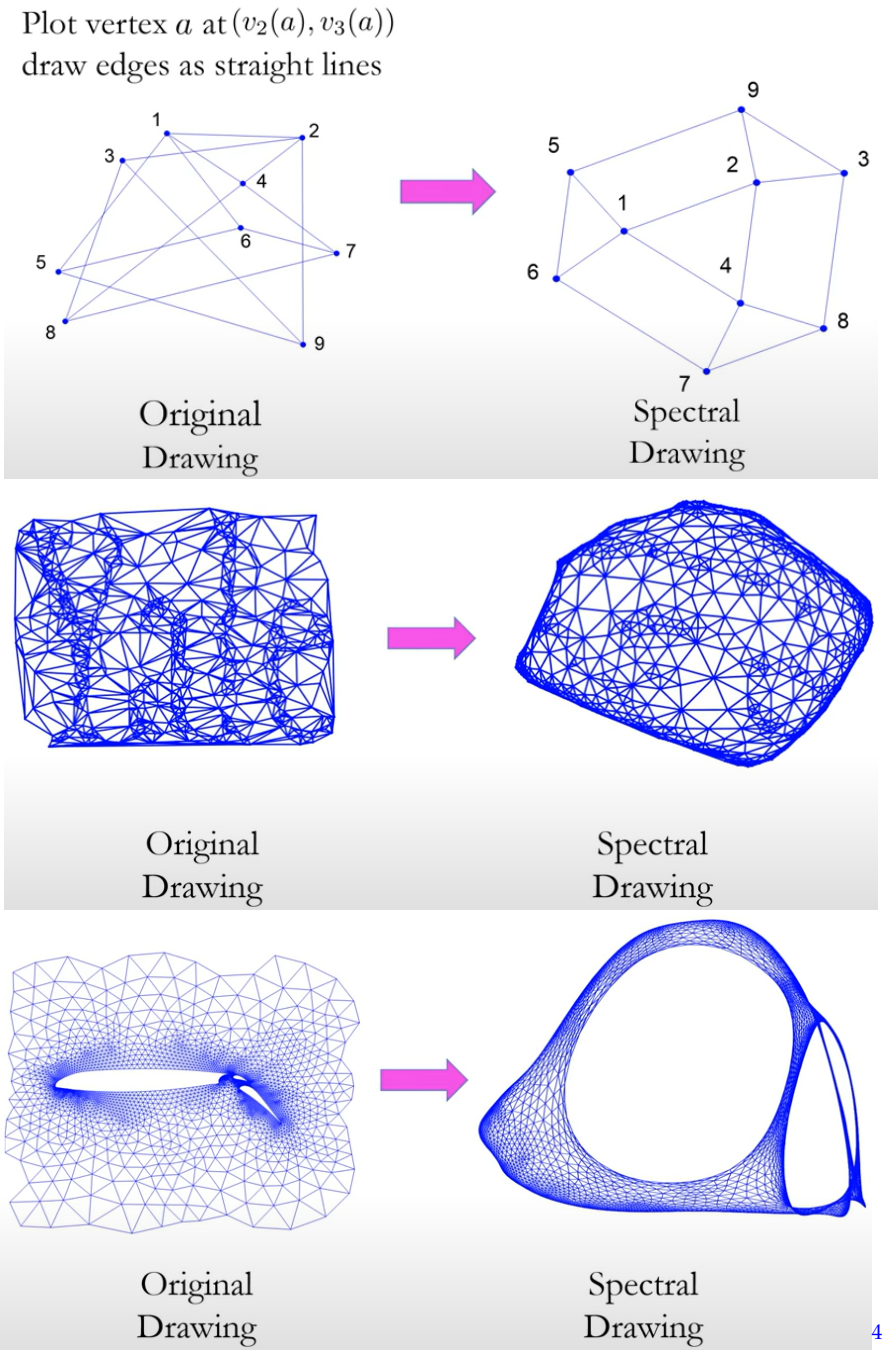
Figure 4: After computing $\mathbf{y} = (\mathbf{M}')^k \mathbf{x}$, Fiedler's algorithm sorts vertices according to $\mathbf{y}$ and sweeps across the nested cuts for the sparsest. Though there are some vertices in B that end with value less than 0 and vertices in A that end with value more than 0, there should be a cut such that there are not so many vertices on the wrong side - this cut should be close to (A, B).



[3] https://www.youtube.com/watch?v=cbYcn8o0Jfg

3 Digression: Spectral Graph Drawing

Let's look at the magic of $\mathbf{v}_2$ and $\mathbf{v}_3$ of L_G.



[4]

Automatically, the second and third eigenvectors tells us a lot about the topology of the graph.

[4]Image from https://www.youtube.com/watch?v=CDMQR422LGM

4 Additional Discussion in Exercise Session

4.1 Removing the $\frac{1}{\Phi(G)^2}$ Dependency from the Power Method

We obtained an additive approximation when we worked with $\mathbf{M} = 2\mathbf{I} - \mathbf{N}_G$ because we were approximating eigenvalues of the form $2 - \lambda_i$. The approximation is multiplicative for $2 - \lambda_i$, but when we view it to an approximation for λ_2, we obtain an additive approximation.

If we instead work with the inverse $\mathbf{N}_G^{-1}$, the eigenvalues should be of the form $1/\lambda_i$, but the first eigenvalue of $\mathbf{N}_G$ is $\lambda_1 = 0$ so $\mathbf{N}_G$ does not have an inverse. Instead, we will work with the **pseudo-inverse** $\mathbf{N}_G^\dagger$. $\mathbf{N}_G^\dagger$ is defined as

$$\mathbf{N}_G^\dagger = \sum_{i:\lambda_i \neq 0} \frac{1}{\lambda_i} v_i v_i^\top$$

where the v_i are from the spectral decomposition of $\mathbf{N}_G^\dagger$ and $\mathbf{N}_G^\dagger$ has eigenvalues

$$0, \frac{1}{\lambda_2} \geq \frac{1}{\lambda_3} \geq \cdots \geq \frac{1}{\lambda_n}.$$

We can directly use $\mathbf{N}_G^\dagger$ to get a multiplicative approximation for eigenvalues of the form $\frac{1}{\lambda_i}$ which can then be used to obtain a multiplicative approximation for λ_i.

Exercise 4.1. Show that $\mathbf{y} = (\mathbf{N}_G^\dagger)^k \mathbf{x}$ satisfies $\frac{\mathbf{y}^\top \mathbf{N}_G \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \leq (1+\varepsilon)\lambda_2$ for all $\varepsilon > 0$ and $k = O(\log \frac{n}{\varepsilon}/\varepsilon)$.

Proof. Similar to the analysis of POWER we upper bound $\mathbf{y}^\top \mathbf{N}_G \mathbf{y}$ and lower bound $\mathbf{y}^\top \mathbf{y}$. As before, we express $\mathbf{x}$ in the eigenvectors of $(\mathbf{N})_G^\dagger$ so $\mathbf{x} = \sum_{i:\lambda_i \neq 0} a_i \mathbf{v}_i$ where $a_i = \langle \mathbf{v}_i, \mathbf{x} \rangle$.

Observe that $\mathbf{y} = \sum_{i:\lambda_i \neq 0} a_i \frac{1}{\lambda_i^k}$, $\mathbf{y}^\top \mathbf{N}_G \mathbf{y} = \sum_{i:\lambda_i \neq 0} a_i^2 \frac{1}{\lambda_i^{2k-1}}$, and $\mathbf{y}^\top \mathbf{y} = \sum_{i:\lambda_i \neq 0} a_i^2 \frac{1}{\lambda_i^{2k}}$.

Fix $\varepsilon' = \varepsilon/2$. Again, we define ℓ to separate the eigenvalues such that $\lambda_2 \leq \cdots \leq \lambda_\ell \leq (1+\varepsilon')\lambda_2$ and $(1+\varepsilon')\lambda_2 < \lambda_{\ell+1} \leq \cdots \leq \lambda_n$.

Then we obtain the following bounds:

$$\begin{aligned}
\mathbf{y}^\top \mathbf{N}_G \mathbf{y} &= \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}} + \sum_{\substack{i:\lambda_i \neq 0 \\ i > \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}} \\
&\leq \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}} + \frac{1}{(1+\varepsilon')^{2k-1}\lambda_2^{2k-1}} \sum_{\substack{i:\lambda_i \neq 0 \\ i > \ell}} a_i^2 \\
&\leq \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}} + \frac{1}{(1+\varepsilon')^{2k-1}\lambda_2^{2k-1}} ||\mathbf{x}||^2 \\
&= \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}} + \frac{1}{(1+\varepsilon')^{2k-1}\lambda_2^{2k-1}} a_2^2 \frac{||\mathbf{x}||^2}{a_2^2} \\
&= \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}} + \frac{4n}{(1+\varepsilon')^{2k-1}} \frac{a_2^2}{\lambda_2^{2k-1}} \qquad \text{(using } a_2^2 = |\langle \mathbf{x}, \mathbf{v}_2 \rangle|^2 \geq \tfrac{1}{4}\text{)} \\
&\leq \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}} + \frac{4n}{(1+\varepsilon')^{2k-1}} \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} \frac{a_i^2}{\lambda_i^{2k-1}} \\
&= \left(1 + \frac{4n}{(1+\varepsilon')^{2k-1}}\right) \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} \frac{a_i^2}{\lambda_i^{2k-1}}
\end{aligned}$$

$$\begin{aligned}
\mathbf{y}^\top \mathbf{y} &= \sum_{i:\lambda_i \neq 0} a_i^2 \frac{1}{\lambda_i^{2k}} \\
&= \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k}} + \sum_{\substack{i:\lambda_i \neq 0 \\ i > \ell}} a_i^2 \frac{1}{\lambda_i^{2k}} \\
&\geq \frac{1}{(1+\varepsilon')\lambda_2} \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}}
\end{aligned}$$

So

$$\frac{\mathbf{y}^\top \mathbf{N}_G \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \leq \frac{\left(1 + \frac{4n}{(1+\varepsilon')^{2k-1}}\right) \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} \frac{a_i^2}{\lambda_i^{2k-1}}}{\frac{1}{(1+\varepsilon')\lambda_2} \sum_{\substack{i:\lambda_i \neq 0 \\ i \leq \ell}} a_i^2 \frac{1}{\lambda_i^{2k-1}}} = \left(1 + \frac{4n}{(1+\varepsilon')^{2k-1}}\right)(1+\varepsilon')\lambda_2.$$

When we take $k = O\left(\log\left(\frac{n}{\varepsilon'}\right)/\varepsilon'\right)$, we have that

$$\left(1 + \frac{4n}{(1+\varepsilon')^{2k-1}}\right)(1+\varepsilon')\lambda_2 \leq (1+2\varepsilon')\lambda_2 = (1+\varepsilon)\lambda_2.$$

□

However, to directly use $\mathbf{N}_G^\dagger$ to get a multiplicative approximation for eigenvalues of the form $\frac{1}{\lambda_i}$, we would need to compute $(\mathbf{N}_G^\dagger)^k\mathbf{x}$. Computing $\mathbf{N}_G^\dagger$ itself can be expensive - $\mathbf{N}_G^\dagger$ can be dense, even if G is sparse. Observe that $\mathbf{N}_G^\dagger\mathbf{x}$ is the same as finding $\mathbf{v}$ where

$$\mathbf{N}_G\mathbf{v} = \mathbf{x}.$$

This is a Laplacian linear system. Laplacian linear systems can be solved in near-linear time[5]! (I hope I will have time to cover this).

4.2 Proof of the Paley-Zygmund inequality

Lemma 4.2 (Paley-Zygmund inequality)**.** *If Z is a non-negative random variable with finite variance, then, for every $0 \le \delta \le 1$,*

$$\Pr(Z \ge \delta \cdot \mathbb{E}[Z]) \ge (1-\delta)^2 \cdot \frac{(\mathbb{E}[Z])^2}{\mathbb{E}[Z^2]}$$

Proof. $\mathbb{E}[Z] = \mathbb{E}[Z \cdot \mathbf{1}_{\{Z \ge 0\}}]$ where $\mathbf{1}_{\{Z\ge 0\}}$ is the indicator random variable taking value 1 when $Z \ge 0$ and value 0 otherwise. $\mathbb{E}[Z \cdot \mathbf{1}_{\{Z\ge 0\}}] = \mathbb{E}\left[Z \cdot \mathbf{1}_{\{Z \ge \delta\cdot\mathbb{E}[Z]\}} + Z \cdot \mathbf{1}_{\{Z<\delta\cdot\mathbb{E}[Z]\}}\right]$. We have $\mathbb{E}[Z \cdot 1_{Z<\delta\cdot\mathbb{E}[Z]}] \le \delta \cdot \mathbb{E}[Z]$, so $\mathbb{E}[Z \cdot 1_{Z\ge\delta\cdot\mathbb{E}[Z]}] \ge (1-\delta)\,\mathbb{E}[Z]$.

We can define the inner product on the set of random variables by the expectation of their product i.e. $\langle X, Y\rangle = \mathbb{E}[XY]$. Then by the Cauchy-Schwarz inequality, we have

$$\begin{aligned}
&\langle Z, \mathbf{1}_{\{Z\ge\delta\cdot\mathbb{E}[Z]\}}\rangle^2 \le \langle Z, Z\rangle \cdot \langle \mathbf{1}_{\{Z\ge\delta\cdot\mathbb{E}[Z]\}}, \mathbf{1}_{\{Z\ge\delta\cdot\mathbb{E}[Z]\}}\rangle \\
\iff &\mathbb{E}[Z \cdot \mathbf{1}_{\{Z\ge\delta\cdot\mathbb{E}[Z]\}}]^2 \le \mathbb{E}[Z^2] \cdot \mathbb{E}[\mathbf{1}^2_{\{Z\ge\delta\cdot\mathbb{E}[Z]\}}] \\
\iff &\mathbb{E}[Z \cdot \mathbf{1}_{\{Z\ge\delta\cdot\mathbb{E}[Z]\}}] \le \sqrt{\mathbb{E}[Z^2]} \cdot \sqrt{\mathbb{E}[\mathbf{1}_{\{Z\ge\delta\cdot\mathbb{E}[Z]\}}]}
\end{aligned}$$

Then, $\mathbb{E}[\mathbf{1}_{\{Z\ge\delta\cdot\mathbb{E}[Z]\}}] = \Pr[Z \ge \delta \cdot \mathbb{E}[Z]]$ and by rearrangement and substitution, we obtain the inequality. □

4.3 Inner Product between a Vector with a Random Point from the Hypercube

We mentioned earlier a lemma about the inner product between a vector and a random point from the hypercube in order to prove the performance of POWER. This is a proof of the lemma.

Lemma 4.3. *For any vector $\mathbf{v} \in \mathbb{R}^n$ where $\|\mathbf{v}\| = 1$. Sample $\mathbf{x} \in \{-1, 1\}^n$. Then*

$$\Pr\left[|\langle \mathbf{x}, \mathbf{v}\rangle| \ge \frac{1}{2}\right] \ge \frac{3}{16}.$$

Remark 4.4. The proof of Lemma 4.3 works even if $x \sim \{-1, 1\}^n$ is selected according to a 4-wise independent distribution. This means that the algorithm can be derandomized in polynomial time.

[5] See the Spielman-Teng '04 paper

Proof. From Luca's lecture note[6]. We apply the Paley-Zygmund inequality to the random variable $\langle \mathbf{x}, \mathbf{v} \rangle^2$.

Define Z to be the random variable $\langle \mathbf{x}, \mathbf{v} \rangle = \sum_i x_i v_i$. We compute its first, second, and fourth moments to be:

$$\mathbb{E}[Z] = 0.$$

$$\mathbb{E}[Z^2] = 1.$$

$$\mathbb{E}[Z^4] = 3\left(\sum_i v_i^2\right) - 2\sum_i v_i^4 \leq 3.$$

Then observe that $\mathrm{Var}(Z) = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2 = 1$. Having the first, second, and fourth moment of Z, we can apply the Paley-Zygmund inequality to Z^2 and $\delta = 1/4$, so we obtain

$$\Pr\left[S^2 \geq \frac{1}{4}\right] \geq \left(\frac{3}{4}\right)^2 \cdot \frac{1}{3} = \frac{3}{16}$$

□

[6] https://lucatrevisan.github.io/teaching/expanders2016/lecture08.pdf

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 9: Spectral Equivalences, Expander Mixing Lemma, and Ramanujan Graphs

September 30, 2021

Instructor: Thatchaphol Saranurak

Scribe: Nikhil Shagrithaya

1 Introduction

Last week, we saw that the eigenvalues of the normalized Laplacian of a graph can contain useful information about the 'connectedness' of a graph. In particular, we saw that the second smallest eigenvalue of ($\mathbf{N}_G$) is closely related to the conductance of the graph. The precise relationship is given by Cheeger's inequality:

$$\lambda_2(\mathbf{N}_G)/2 \leq \Phi(G) \leq \sqrt{2\lambda_2(\mathbf{N}_G)}$$

Moreover, we also saw an algorithm which gave us a $O(1)$-approximation of the sparest cut, when the conductance of the graph was $\Omega(1)$. That is, using the power method, we were able to obtain a method cut for which the following was true:

$$\Phi_G(S) \approx \sqrt{2\lambda_2(\mathbf{N}_G)} \leq 2\sqrt{\Phi(G)}$$

2 Spectral equivalency of graph

We can also define a new notion of connectivity for graphs based on the eigenvalues of the normalized Laplacian itself. In order to do this, we need some definitions.

Definition 2.1. We define a partial order on graphs based on their eigenvalues. We say that:

$$H \preccurlyeq^{\text{spec}} G$$

if $\boldsymbol{x}^T\mathbf{L}_H\boldsymbol{x} \leq \boldsymbol{x}^T\mathbf{L}_G\boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^V$. Or equivalently, $\mathbf{L}_H \preceq \mathbf{L}_G$ or $\mathbf{L}_G - \mathbf{L}_H$ is psd (positive semi-definite).

Definition 2.2. We say that $\mathbf{H}$ and $\mathbf{G}$ are **α-spectral-equivalent** if $\mathbf{H}' \preccurlyeq^{\text{spec}} \mathbf{G} \preccurlyeq^{\text{spec}} \alpha\mathbf{H}'$ for some $\mathbf{H}' = c \cdot \mathbf{H}$. We write $\mathbf{H} \approx_\alpha^{\text{spec}} \mathbf{G}$.

Note that $\preccurlyeq^{\text{spec}}$ is stronger than $\preccurlyeq^{\text{cut}}$, because we can define the partial order based on cut in another, equivalent way: $H \preccurlyeq^{\text{cut}} G$ iff $\boldsymbol{x}^T \mathbf{L}_H \boldsymbol{x} \leq \boldsymbol{x}^T \mathbf{L}_G \boldsymbol{x}$ for all $\boldsymbol{x} \in \{0,1\}^V$. Because the partial order based on eignenvalues requires $\boldsymbol{x}^T \mathbf{L}_H \boldsymbol{x} \leq \boldsymbol{x}^T \mathbf{L}_G \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^V$, and not just $\boldsymbol{x} \in \{0,1\}^V$, $\preccurlyeq^{\text{spec}}$ is a stronger requirement than $\preccurlyeq^{\text{cut}}$, namely $H \preccurlyeq^{\text{spec}} G$ implies $H \preccurlyeq^{\text{cut}} G$.

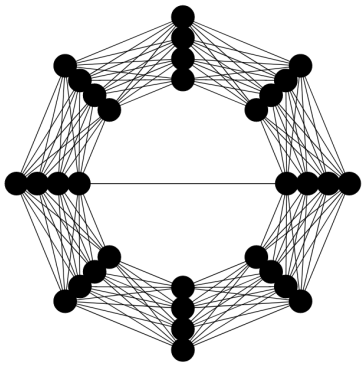
A natural question to then ask, is whether the converse is true, i.e. $H \preccurlyeq^{\text{cut}} G$ implies $H \preccurlyeq^{\text{spec}} G$? And if not, does it hold approximately?

We'll see that the answer is no to both of these questions, in the following lemma:

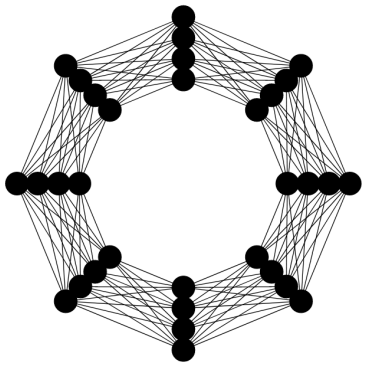
Lemma 2.3 ($\preccurlyeq^{\text{spec}}$ is strictly stronger than $\preccurlyeq^{\text{cut}}$)**.** *There exist graphs G and $\tilde{G}$ such that:*

- $\tilde{G} \preccurlyeq^{\text{cut}} G \preccurlyeq^{\text{cut}} (1+\epsilon)\tilde{G}$, *but*
- $\tilde{G} \preccurlyeq^{\text{spec}} G$ *but* $G \not\preccurlyeq^{\text{spec}} \frac{\epsilon^2 n}{10000}\tilde{G}$

Proof. We will construct unweighted graphs G and $\tilde{G}$ with two parameters: n and $k = \frac{50}{\epsilon}$. Both graphs will have $n \times k$ vertices, having n clusters with k vertices each. The vertex set of $\tilde{G}$ is $\{0, 1, \ldots, n-1\} \times \{1, \ldots, k\}$, where n is even. The graph $\tilde{G}$ will consist of n complete bipartite graphs, connecting all pairs of vertices $\{u, i\}$ and $\{v, j\}$ where $v = u + 1 mod n$. G is identical $\tilde{G}$, except that it has one extra edge going from vertex $\{0, 1\}$ to vertex $\{n/2, 1\}$. The graphs for $n = 8$ and $k = 4$ look like this:



G: $n = 8$ sets of $k = 4$ vertices arranged in a ring and connected by complete bipartite graphs, plus one edge across.



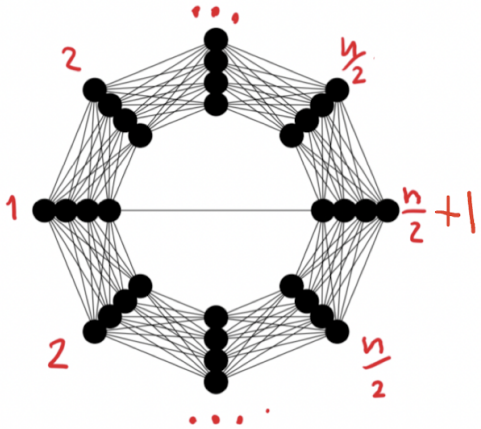
$\tilde{G}$: A good cut sparsifier of G, but a poor spectral sparsifier

[1]

It is easy to see that $\tilde{G} \preccurlyeq^{\text{cut}} G \preccurlyeq^{\text{cut}} (1+\epsilon)\tilde{G}$, because the only cuts whose value will change are those where the extra edge is crossing between them. Because any cut must have size at least k^2, adding one extra edge does not change the cut value by much.

Moreover, $\tilde{G} \preccurlyeq^{\text{spec}} G$ holds because $\tilde{G}$ is a subgraph of G. All that remains is to show $G \not\preccurlyeq^{\text{spec}} \frac{\epsilon^2 n}{10000}\tilde{G}$. Consider the following $\boldsymbol{x}$:

[1] Image from https://arxiv.org/abs/0808.4134



where $\forall u \in \{0, 1, \cdots, n/2\}, \forall i \in [k], x_{\{u,i\}} = u + 1; \forall u \in \{n/2 + 1, \cdots, n - 1\}\ \forall i \in [k], x_{\{u,i\}} = n + 1 - u$. Then, we have:

$$\boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} = nk^2 + \left(\frac{n}{2}\right)^2$$

but

$$\boldsymbol{x}^T \boldsymbol{L}_{\tilde{G}} \boldsymbol{x} = nk^2$$

So $\boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} - \frac{\epsilon^2 n}{10000} \boldsymbol{x}^T \boldsymbol{L}_{\tilde{G}} \boldsymbol{x} = nk^2 > 0 \implies G \not\preccurlyeq^{\text{spec}} \frac{\epsilon^2 n}{10000} \tilde{G}$. □

This lemma shows that spectral-equivalence between graphs are *strictly stronger* than cut-equivalence, and moreover, cut-equivalence cannot even imply a slightly weaker spectral-equivalence, as was the case for flow, because the lemma shows a gap of $\Omega(n)$, which is the largest possible gap.

As an aside, it is easy to check if $H \preccurlyeq^{\text{spec}} G$, because all we need to do is check whether $\boldsymbol{L}_G - \boldsymbol{L}_H$ is positive semi-definite, and this can be done in polynomial time.

Question 2.4 (Open)**.** Is there a **fast** algorithm for checking if $H \preccurlyeq^{\text{spec}} G$ or even $H \approx^{\text{spec}}_{(1+\epsilon)} G$?

Question 2.5 (Open)**.** Is there a **polynomial-time** algorithm for checking if $H \preccurlyeq^{\text{cut}} G$ or even $H \approx^{\text{cut}}_{(1+\epsilon)} G$?

From the previous lecture, checking if $H \preccurlyeq^{\text{flow}} G$ is checking the existence of feasible H-concurrent flow in G, which can be done by solving LP in polynomial time. Since $H \preccurlyeq^{\text{flow}} G \implies H \preccurlyeq^{\text{cut}} G$, Q2.5 can also be done in polytime. Remains to check the existence of faster algorithms for checking $H \preccurlyeq^{\text{flow}} G$ or $H \preccurlyeq^{\text{cut}} G$.

3 Spectral Equivalence between Expanders

In Lecture 03_2, we saw that expanders with the same degree profile are approximately equivalent, in both the cut and the flow sense. But are they also spectral-equivalent?

They indeed are. We begin with the following lemma:

Lemma 3.1. *Suppose G is a ϕ-expander. If $H \preccurlyeq^{\text{deg}} G$, then $H \preccurlyeq^{\text{spec}} O(\frac{1}{\phi^2})G$.*

Proof. We can increase the degree of any vertex in H to be equal to the degree of the corresponding vertex in G by adding self-loops, this will only make our task easier, and importantly, $\mathbf{L}_H$ remains the same. Now, both H and G have the same degree profile $\boldsymbol{d}$ (the degree profile is a vector whose entries are the degrees of the vertices of the graph). For any $\boldsymbol{x} \perp \boldsymbol{d}$, we have:

$$\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x} \geq \frac{\phi^2}{4} \boldsymbol{x}^\top \boldsymbol{L}_H \boldsymbol{x}.$$

The above statement is true because, by Cheeger's, $\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x} \geq \frac{\phi^2}{2} \boldsymbol{x}^\top \boldsymbol{D} \boldsymbol{x}$ for any $\boldsymbol{x} \perp \boldsymbol{d}$ (recall that $\lambda_2(\boldsymbol{N}_G) = \min_{\boldsymbol{x} \perp \boldsymbol{d}} \frac{\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D} \boldsymbol{x}}$). Also, for any $\boldsymbol{x}$, we claim $2\boldsymbol{x}^\top \boldsymbol{D} \boldsymbol{x} \geq \boldsymbol{x}^\top \boldsymbol{L}_H \boldsymbol{x}$ because:

$$\begin{aligned} 2\boldsymbol{y}^\top \boldsymbol{D} \boldsymbol{y} - \boldsymbol{y}^\top \boldsymbol{L}_H \boldsymbol{y} &= \sum_u 2 \deg_G(u) y_u^2 - \sum_{uv \in E} w_{uv} (y_u - y_v)^2 = \sum_u 2 \deg_H(u) y_u^2 - \sum_{uv \in E} w_{uv} (y_u - y_v)^2 \\ &= \sum_{uv \in E} w_{uv} (y_u + y_v)^2 \geq 0. \end{aligned}$$

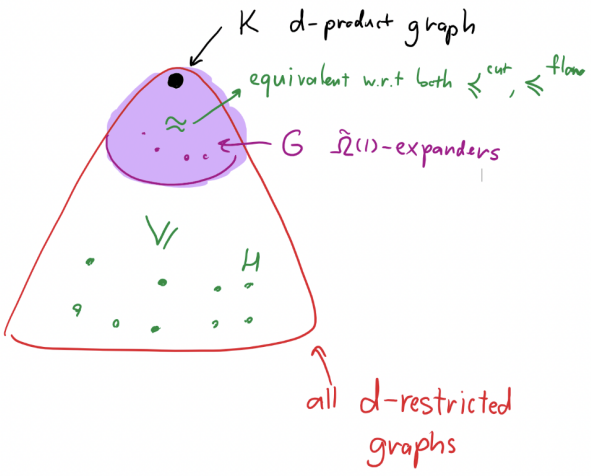
Now we have to show the same inequality also holds for all vectors $\boldsymbol{x}$ such that $\boldsymbol{x} \parallel \boldsymbol{d}$. Let $\boldsymbol{y} = \boldsymbol{x} - \sigma \mathbf{1}$ where $\boldsymbol{y} \perp \boldsymbol{d}$. Then:

$$\boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y} = (\boldsymbol{x} - \sigma \mathbf{1})^\top \boldsymbol{L}_G (\boldsymbol{x} - \sigma \mathbf{1}) = \boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} - \sigma \mathbf{1}^T \mathbf{L}_G \boldsymbol{x} - \boldsymbol{x}^T \mathbf{L}_G \sigma \mathbf{1} + \sigma \mathbf{1}^T \mathbf{L}_G \sigma \mathbf{1}$$

The last two terms are zero because $\mathbf{1}$ is in the kernel of L_G. The second term is also zero, because $\sigma \mathbf{1}^T \mathbf{L}_G \boldsymbol{x} = (\mathbf{L}_G^T \sigma \mathbf{1})^T \boldsymbol{x} = (\mathbf{L}_G^\sigma \mathbf{1})^T \boldsymbol{x} = 0$, where the second equality is true because $\mathbf{L}_G$ is symmetric. A similar argument holds for $\mathbf{L}_H$. Combining everything, we have that $\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x} = \boldsymbol{y}^\top \boldsymbol{L}_G \boldsymbol{y}$ and $\boldsymbol{x}^\top \boldsymbol{L}_H \boldsymbol{x} = \boldsymbol{y}^\top \boldsymbol{L}_H \boldsymbol{y}$, and because we've already shown that $\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x} \geq \frac{\phi^2}{4} \boldsymbol{x}^\top \boldsymbol{L}_H \boldsymbol{x}$ when $\boldsymbol{x} \perp \boldsymbol{d}$, we are done. □

Corollary 3.2. *Let G and G' be ϕ-expanders with degree profile $\boldsymbol{d}$. We have:*

- *$G \preccurlyeq^{\text{spec}} O(\frac{1}{\phi^2}) G'$ and $G' \preccurlyeq^{\text{spec}} O(\frac{1}{\phi^2}) G$. So G and G' are $O(\frac{1}{\phi^4})$-spectral-equivalent.*
- *For all $\boldsymbol{d}$-restricted graphs H, we have $H \preccurlyeq^{\text{spec}} O(\frac{1}{\phi^2}) G$.*
- We have same picture we saw. It holds for for $\preccurlyeq^{\text{spec}}$



4 Expander Mixing Lemma

4.1 Motivation

Let G be a graph with degree profile $\boldsymbol{d}$. If G has high conductance, then we have:

$$|E_G(S, V \setminus S)| \approx \boldsymbol{d}(S)$$

for all sets S such that $vol(S) \leq vol(V \setminus S)$. But the good conductance only guarantees us that the number of edges going out of a set is high. But what can we say about the number of edges **between two sets ?**

For the $\boldsymbol{d}$-product graph, it can be easily seen that:

$$|E(S, T)| \approx \frac{\boldsymbol{d}(S)\boldsymbol{d}(T)}{\boldsymbol{d}(V)}$$

for all $S, T \subseteq V$. Intuitively, one should think about $\frac{\boldsymbol{d}(S)\boldsymbol{d}(T)}{\boldsymbol{d}(V)}$ as the expectation of $|E(S, T)|$ in the random graph with degree profile $\boldsymbol{d}$. This is a stronger property than conductance, as conductance guarantees this only when $T = V \setminus S$.

A natual question to then ask, is: do expanders satisfies even this stronger property? We will see that this stronger property has applications.

4.2 The Normalized Adjacency Matrix

We begin by defining a variant of the Laplacian matrix, whose eigenvalues will give us a good approximation of the property we defined in the previous subsection. Recall that we previously defined:

$$\boldsymbol{N}_G = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}_G\boldsymbol{D}^{-1/2}$$

, which had eigenvalues satisfying:

$$0 = \lambda_1(\boldsymbol{N}_G) \leq \lambda_2(\boldsymbol{N}_G) \leq \cdots \leq \lambda_n(\boldsymbol{N}_G) \leq 2.$$

We will call

$$\boldsymbol{A}'_G = \boldsymbol{D}^{-1/2}\boldsymbol{A}_G\boldsymbol{D}^{-1/2}$$

as the normalized adjacency matrix. Note that $\boldsymbol{A}'_G = \boldsymbol{I} - \boldsymbol{N}_G$ has eigenvalues satisfying:

$$1 = \lambda_1(\boldsymbol{A}'_G) \geq \lambda_2(\boldsymbol{A}'_G) \geq \cdots \geq \lambda_n(\boldsymbol{A}'_G) \geq -1.$$

The following properties follow from the corresponding ones we proved earlier, about $\mathbf{N_G}$:

- $\lambda_2(\boldsymbol{A}'_G) = 1$ iff G is not connected.
- $\lambda_2(\boldsymbol{A}'_G) < 1 - \Omega(1)$ iff G is a $\Omega(1)$-expander.
- $\lambda_n(\boldsymbol{A}'_G) = -1$ iff G is bipartite.

We further define σ_2 as follows:

$$\sigma_2 := \max\{|\lambda_2(\boldsymbol{A}'_G)|, |\lambda_3(\boldsymbol{A}'_G)|, \ldots, |\lambda_n(\boldsymbol{A}'_G)|\} = \max\{|\lambda_2(\boldsymbol{A}'_G)|, |\lambda_n(\boldsymbol{A}'_G)|\}.$$

It is the second largest eigenvalue in absolute value of $\boldsymbol{A}'_G$. So from what we learned before, if $\sigma_2 \ll 1$ is small, then G is both "far" from being disconnected and bipartite. Moreover, in the following section, we will see that σ_2 describes when two arbitrary disjoint sets are chosen in the graph, how much the edges between them behave like these of $\boldsymbol{d}$-product graph. The smaller σ_2 is, the more similar they are. We will be formalizing this statement, and make it mathematically precise, in the following sections.

4.3 The Statement and the Proof

Lemma 4.1 (Expander Mixing Lemma)**.** *For a graph G with degree profile $\mathbf{d}$, and for every $S, T \subseteq V$, we have*

$$\left| |E(S,T)| - \frac{\boldsymbol{d}(S)\boldsymbol{d}(T)}{\boldsymbol{d}(V)} \right| \leq \sigma_2 \sqrt{\boldsymbol{d}(S)\boldsymbol{d}(T)}.$$

Therefore, when σ_2 is small, $|E(S,T)| \approx \frac{\boldsymbol{d}(S)\boldsymbol{d}(T)}{\boldsymbol{d}(V)}$ holds for all $S, T \subseteq V$.

Proof. We will show a slightly more general statement, which is:

$$\sigma_2 = \max_{\boldsymbol{x},\boldsymbol{y}} \frac{|\boldsymbol{x}^\top (\boldsymbol{A}_G - \boldsymbol{R})\boldsymbol{y}|}{\sqrt{\sum_v \boldsymbol{d}(v)x_v^2} \cdot \sqrt{\sum_v \boldsymbol{d}(v)y_v^2}}$$

where $\boldsymbol{R}$ is equal to $\frac{1}{\boldsymbol{d}(V)}\boldsymbol{d} \cdot \boldsymbol{d}^\top$. Looking at $\boldsymbol{R}$ entry-wise, we get $\boldsymbol{R}_{u,v} = \frac{\boldsymbol{d}(u)\boldsymbol{d}(v)}{\boldsymbol{d}(V)}$. So in other words, $\boldsymbol{R}$ is the adjacency matrix of the $\boldsymbol{d}$-product graph. Now, observe that if $\boldsymbol{x} = \mathbf{1}_S$ and $\boldsymbol{y} = \mathbf{1}_T$, we get $\boldsymbol{x}^\top \boldsymbol{A}_G \boldsymbol{y} = |E(S,T)|$, and $\boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{y}$ equals $\boldsymbol{d}(S)\boldsymbol{d}(T)/\boldsymbol{d}(V)$. Therefore, we have:

$$\sigma_2 \geq \frac{\left| |E(S,T)| - \frac{\boldsymbol{d}(S)\boldsymbol{d}(T)}{\boldsymbol{d}(V)} \right|}{\sqrt{\boldsymbol{d}(S)} \cdot \sqrt{\boldsymbol{d}(T)}}$$

which gives the lemma.

We can rewrite $\boldsymbol{A}'_G$ by making use of eigenvalue decomposition:

$$\boldsymbol{A}'_G = \boldsymbol{v}_1\boldsymbol{v}_1^\top + \lambda_2(\boldsymbol{A}'_G)\boldsymbol{v}_2\boldsymbol{v}_2^\top + \cdots + \lambda_n(\boldsymbol{A}'_G)\boldsymbol{v}_n\boldsymbol{v}_n^\top.$$

By how we defined it, $\lambda_1(\boldsymbol{A}'_G) = 1$. Moreover, $\boldsymbol{v}_1 = \frac{1}{\sqrt{\boldsymbol{d}(V)}}\boldsymbol{d}^{1/2}$. This we can see by combining the fact that $\boldsymbol{A}'_G = \boldsymbol{I} - \boldsymbol{N}_G$ has the same eigenvectors as $\boldsymbol{N}_G$, and that the first eigenvector of $\boldsymbol{N}_G$ is $\frac{1}{\sqrt{\boldsymbol{d}(V)}}\boldsymbol{d}^{1/2}$.

So we have:

$$\begin{aligned}
\sigma_2 &= \max_{z,w} \frac{|z^\top (A'_G - v_1 v_1^\top) w|}{\|z\| \cdot \|w\|} \\
&= \max_{x,y} \frac{|x^\top D^{1/2} (A'_G - v_1 v_1^\top) D^{1/2} y|}{\sqrt{\sum_v d(v) x_v^2} \cdot \sqrt{\sum_v d(v) y_v^2}} && z \mapsto D^{1/2} x, w \mapsto D^{1/2} y \\
&= \max_{x,y} \frac{|x^\top (A_G - R) y|}{\sqrt{\sum_v d(v) x_v^2} \cdot \sqrt{\sum_v d(v) y_v^2}}
\end{aligned}$$

The first equality is by the *variational characterization of singular values* (we prove this below). The last equality is because $A'_G = D^{-1/2} A_G D^{-1/2}$, and we can verify that

$$\begin{aligned}
D^{1/2} v_1 v_1^\top D^{1/2} &= D^{1/2} (\frac{1}{\sqrt{d(V)}} d^{1/2}) (\frac{1}{\sqrt{d(V)}} d^{1/2})^\top D^{1/2} \\
&= \frac{1}{d(V)} d \cdot d^\top \\
&= R
\end{aligned}$$

This completes the proof.

□

Corollary 4.2. *Let G be a d-regular graph. For every $S, T \subseteq V$, we have*

$$\left| |E(S,T)| - \frac{d \cdot |S| \cdot |T|}{dn} \right| \leq \sigma_2 d \sqrt{|S| \cdot |T|}.$$

It turns out that the converse of the Expander Mixing Lemma also holds.

Lemma 4.3. [2]*Let G be a d-regular graph. If*

$$\left| |E(S,T)| - \frac{d \cdot |S| \cdot |T|}{dn} \right| \leq \theta d \sqrt{|S| \cdot |T|}$$

for all two disjoint sets S, T. Then $\sigma_2 = O(\theta(1 + \log(\frac{d}{\theta}))$.

5 Ramanujan Graphs: The Best Expander w.r.t. Eigenvalues

To simplify the discussion, we will only talk about d-regular in this section. A natural question to ask at this point is: How small σ_2 can be? A smaller σ_2 means that $E(S,T)$ is closer to $\frac{d|S||T|}{dn}$, for all $S, T \subseteq V$.

It turns out that σ_2 can be as small as $\frac{2\sqrt{d-1}}{d} \leq \frac{2}{\sqrt{d}}$. Graphs that achieve this bound are called **Ramanujan graphs**. Furthermore, this bound is tight! Roughly, we have that:

[2]https://link.springer.com/article/10.1007/s00493-006-0029-7

Theorem 5.1. *For every n_0 and d_0, there exist $n \in [n_0, O(n_0)]$ and $d \in [d_0, O(d_0)]$ such that, there is a d-regular graph G with n vertices satisfying:*

$$\sigma_2 \leq \frac{2\sqrt{d-1}}{d} = O(\frac{1}{\sqrt{d}}).$$

That is, its second eigenvalue (in absolute values) of the normalized adjacency matrix is at most $\frac{2\sqrt{d-1}}{d}$.

Question 5.2 (Open)**.** Can we prove that this holds for all n and d?

Adam Marcus, Daniel Spielman and Nikhil Srivatsava [3] proved that it holds for bipartite graphs. That is, $\lambda_n(A'_G) = -1$, but $\max_{i \neq 1,n} \lambda_i(A'_G) \leq \frac{2\sqrt{d-1}}{d}$. This led to the resolution of the *Kardison Singer problem*[4]

5.1 Tightness of Ramanujan Graphs

One can ask whether Ramanujan expanders are the best expanders with respect to eigenvalues. That is: does there exist another family of graphs for which σ_2 is strictly smaller than $\frac{2\sqrt{d-1}}{d}$?

It is known that this is not the case, it has been shown that $\sigma_2 \geq \frac{2\sqrt{d-1}}{d}(1 - o(1))$.[5] That is, Ramanujan graphs are the best expanders w.r.t. eigenvalues. In this lecture, we will show a slightly less general version of this bound.

Lemma 5.3. *For any d-regular graph where $d \leq 0.99 \cdot n$, we have $\sigma_2 \geq \Omega(\frac{1}{\sqrt{d}})$.*

Proof. We have

$$\begin{aligned} \mathrm{Tr}[(A'_G)^2] &= \sum_{i=1}^{n} \lambda_i(A'_G)^2 \\ &\leq 1 + (n-1)\sigma_2^2 \end{aligned}$$

upon rearranging the terms, we get:

$$\sigma_2 \geq \sqrt{\frac{\mathrm{Tr}[(A'_G)^2] - 1}{(n-1)}}$$

Note $(A_G^2)_{u,v}$ is just the number of 2-step walks from u to v. So $\mathrm{Tr}[A_G^2] = nd$. Also, when G is d-regular, we have $A'_G = A_G/d$, and therefore, $\mathrm{Tr}[(A'_G)^2] = (nd)/d^2 = n/d$ holds. Substituting these values into the last inequality above, we get:

$$\sigma_2 \geq \sqrt{\frac{n/d - 1}{(n-1)}} = \sqrt{\frac{(n-d)}{d(n-1)}} = \Omega(1/\sqrt{d})$$

when $d \leq 0.99 \cdot n$. □

[3] https://arxiv.org/abs/1304.4132

[4] *https://arxiv.org/abs/1306.3969*

[5] https://www.sciencedirect.com/science/article/pii/0012365X9190112F?via%3Dihub

5.2 Some Properties of Ramanujan Graphs

Let G be a d-regular Ramanujan graph. Then:

Corollary 5.4. *For any $S, T \subseteq V$, if $|S| \cdot |T| \cdot d > 4n^2$, then $E(S,T) \neq \emptyset$.*

That is, if we take two vertex sets which are large enough, then in Ramanujan graphs, there is guaranteed to be an edge between those two sets.

Proof. We prove the contra-positive. Suppose $E(S,T) = \emptyset$. The Expander Mixing Lemma for d-regular graphs gives us:

$$\left| |E(S,T)| - \frac{d \cdot |S| \cdot |T|}{dn} \right| \leq \sigma_2 d \sqrt{|S| \cdot |T|}$$

which implies that

$$\frac{d|S| \cdot |T|}{n} \leq 2\sqrt{d} \cdot \sqrt{|S| \cdot |T|} \iff |S| \cdot |T| \cdot d \leq 4n^2.$$

□

Example 5.5. Say $d = \sqrt{n}$. Consider S and T. If $|S|, |T| \geq 2n^{3/4}$, then $E(S,T) \neq \emptyset$. But in fact, if $|S|, |T| \geq 4n^{3/4}$ which are a bit bigger than what we required before, then there will be considerable amount of edges between then, i.e. $|E(S,T)| = \Omega(d|S||T|/n) = \Omega(n)$.

Corollary 5.6. *Any independent set S has size at most $2n/\sqrt{d}$.*

Proof. As $E(S,S) = \emptyset$ by the definition of an independent set, then $|S|^2 \cdot d \leq 4n^2$. So $|S| \leq 2n/\sqrt{d}$.
□

Exercise 5.7. Is there a d-regular graph that is better than Ramanujan graph? Say, for all S, T where $|S|, |T| \geq 2\sqrt{n}$, we have $E(S,T) \neq \emptyset$?

6 Expander: Conductance vs Eigenvalue

Recall Cheeger's inequality:

$$\frac{\lambda_2(\boldsymbol{N}_G)}{2} \leq \Phi(G) \leq \sqrt{2\lambda_2(\boldsymbol{N}_G)}.$$

Roughly, it claims the following fact: $\Phi(G)$ is big $\iff$ $\lambda_2(\boldsymbol{N}_G)$ is big. But in the *very well-connected* regime, the implication may not necessarily go both ways.

6.1 Eigenvalue can be stronger than Conductance

When the conductance $\Phi(G) = 1 - o(1)$ is almost maximum, we cannot always conclude that $|E(S,T)| \approx \frac{\boldsymbol{d}(S)\boldsymbol{d}(T)}{\boldsymbol{d}(V)}$ for all $S, T \subseteq V$. Take the case of the star graph: it has conductance 1. But for S, T such that they are equal sized partitions of the outer vertices, we get $|E(S,T)| = 0$, whereas $\frac{\boldsymbol{d}(S)\boldsymbol{d}(T)}{\boldsymbol{d}(V)}$ is roughly $O(n)$.

In general, by Cheeger's inequality, $\lambda_2(\boldsymbol{N}_G) \geq \Phi(G)^2/2 = \frac{1}{2} - o(1)$. Therefore, we can only bound $\sigma_2 \leq \frac{1}{2} + o(1)$. In this particular case, even when $\mathrm{vol}(S), \mathrm{vol}(T) = \mathrm{vol}(V)/2$, the Expander Mixing Lemma does not guarantee an edge between S and T. That is, $E(S,T) = \emptyset$ possibly.

6.2 Conductance can be stronger than Eigenvalue

Going the other way, there are cases when the eigenvalue can imply strong expansion, but the conductance of that graph might not be as good. To see one such case, suppose G is d-regular and $\sigma_2 \leq O(\frac{1}{\sqrt{d}})$ is almost minimum. Then, we have:

$$\lambda_2(\boldsymbol{N}_G) \geq 1 - \frac{1}{O(\sqrt{d})}.$$

By Cheeger's inequality, we only get:

$$\Phi(G) \geq \lambda_2(\boldsymbol{N}_G)/2 \geq \frac{1}{2} - o(\frac{1}{O(\sqrt{d})}).$$

Therefore, Cheeger's does not allow us to get strong lower bounds on the conductance, in this case. In particular, it does not imply $\Phi(G) \approx (1-\epsilon)$.

A d-regular graph G where $\Phi(G) \approx (1-\epsilon)$ is called a **lossless expander**, and it has lots of applications in:

- Complexity theory (pseudo-randomness)
- Streaming algorithms (deterministic sparse recovery).
- Dynamic algorithms (dynamic matching)[6]

Note that d should be small, and G should be regular. (Otherwise easy: consider clique and star). There is a bipartite version which is important as well.

6.3 Proof of variational characterization of singular values

Theorem 6.1. *Let $\mathbf{A'_G} = \mathbf{I} - \mathbf{N_G}$. Let σ_2 denote the second largest eigenvalue in terms of absolute value. Then,*

$$\sigma_2 = \max_{z,w} \frac{|z^\top (\boldsymbol{A}'_G - \boldsymbol{v}_1 \boldsymbol{v}_1^\top) w|}{\|z\| \cdot \|w\|}$$

Proof. For any matrix $\boldsymbol{M}$, we write the singular value decomposition of $\boldsymbol{M}$ as

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal = \sum_i \sigma_i(\boldsymbol{M}) \cdot \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

where $\boldsymbol{u}_i, \boldsymbol{v}_i$ are unit vectors and $\sigma_i(\boldsymbol{M})$ are the singular values of $\boldsymbol{M}$. Note that this generalizes eigenvalue decomposition, and holds for any $m \times n$ matrix $\mathbf{M}$. Chapter 3 of this book gives a gentle introduction to singular value decomposition and mentions a lot of applications:

https://home.ttic.edu/~avrim/book.pdf

We will need the following fact:

[6] https://arxiv.org/abs/2108.10461

Fact 6.2. *For each i, we have :*

$$\boldsymbol{M}\boldsymbol{v}_i = \sigma_i(\boldsymbol{M})\boldsymbol{u}_i.$$

Furthermore, the first (largest) singular value $\sigma_1(\boldsymbol{M})$ of $\boldsymbol{M}$ is such that

$$\sigma_1(\boldsymbol{M}) = \|\boldsymbol{M}\|_2 = \max_{\boldsymbol{x}} \frac{\|\boldsymbol{M}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}.$$

Lemma 6.3 (Variational Characterization of the First Singular Values)**.** *We have*

$$\sigma_1(\boldsymbol{M}) = \max_{\boldsymbol{z},\boldsymbol{w}} \frac{\boldsymbol{z}^\top \boldsymbol{M}\boldsymbol{w}}{\|\boldsymbol{z}\| \cdot \|\boldsymbol{w}\|}.$$

Proof. For any unit vector $\boldsymbol{z}$ and $\boldsymbol{w}$, by Cauchy-Schwarz we have

$$\boldsymbol{z}^\top \boldsymbol{M}\boldsymbol{w} \leq \|z\| \cdot \|\boldsymbol{M}\boldsymbol{w}\| \leq \|z\| \cdot \|\boldsymbol{M}\|_2 \cdot \|\boldsymbol{w}\| = \sigma_1(\boldsymbol{M})\|z\| \cdot \|\boldsymbol{w}\|.$$

That is,

$$\sigma_1(\boldsymbol{M}) \geq \max_{\boldsymbol{z},\boldsymbol{w}} \frac{\boldsymbol{z}^\top \boldsymbol{M}\boldsymbol{w}}{\|\boldsymbol{z}\| \cdot \|\boldsymbol{w}\|}.$$

But we know that this equality is attained for the unit vectors $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$ because

$$\boldsymbol{u}_1^\top \boldsymbol{M}\boldsymbol{v}_1 = \boldsymbol{u}_1^\top(\sigma_1(\boldsymbol{M})\boldsymbol{u}_1) = \sigma_1(\boldsymbol{M}).$$

This prove the lemma. □

Claim 6.4. *For symmetric matrix $\boldsymbol{M}$, $\sigma_1(\boldsymbol{M}) = \max\{|\lambda_1(\boldsymbol{M})|, |\lambda_n(\boldsymbol{M})|\}$.*

Proof. We have that

$$\begin{aligned}(\sigma_1(\boldsymbol{M}))^2 = \|\boldsymbol{M}\|_2^2 &= \max_{\boldsymbol{x}} \frac{\|\boldsymbol{M}\boldsymbol{x}\|^2}{\|\boldsymbol{x}\|^2} \\ &= \max_{\boldsymbol{x}} \frac{\boldsymbol{x}^\top \boldsymbol{M}^\top \boldsymbol{M}\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}} \\ &= \lambda_1(\boldsymbol{M}^2).\end{aligned}$$

But we know that $\lambda_1(\boldsymbol{M}^2) = (\max\{|\lambda_1(\boldsymbol{M})|, |\lambda_n(\boldsymbol{M})|\})^2$. This proves the claim. □

When we let $\boldsymbol{M} = \boldsymbol{A}'_G - \boldsymbol{v}_1\boldsymbol{v}_1^\top$, we have

$$\begin{aligned}\sigma_1(\boldsymbol{M}) &= \max\{|\lambda_1(\boldsymbol{A}'_G - \boldsymbol{v}_1\boldsymbol{v}_1^\top)|, |\lambda_n(\boldsymbol{A}'_G - \boldsymbol{v}_1\boldsymbol{v}_1^\top)|\} \\ &= \max\{|\lambda_2(\boldsymbol{A}'_G)|, |\lambda_n(\boldsymbol{A}'_G)|\} \\ &= \sigma_2\end{aligned}$$

by the definition of σ_2 from the lecture. (Basically, we defined $\sigma_2 = \sigma_2(\boldsymbol{A}'_G)$.) We can conclude now that

$$\sigma_2(\boldsymbol{A}'_G) = \sigma_1(\boldsymbol{M}) = \max_{\boldsymbol{z},\boldsymbol{w}} \frac{|\boldsymbol{z}^\top(\boldsymbol{A}'_G - \boldsymbol{v}_1\boldsymbol{v}_1^\top)\boldsymbol{w}|}{\|\boldsymbol{z}\| \cdot \|\boldsymbol{w}\|}$$

by the variational characterization of $\boldsymbol{M}$.

□

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 10: Deterministic Vertex Connectivity via Ramanujan Graphs

September 30, 2021

Instructor: Thatchaphol Saranurak

Scribe: Chaitanya Nalam

- Today, we will show an application of Ramanujan graphs to fast graph algorithms.

1 Vertex Connectivity

Let $G = (V, E)$ be an unweighted and undirected graph. We introduce some basic definitions related to vertex connectivity below.

Definition 1.1 (Vertex Connectivity)**.** Given a connected graph G, the smallest number of vertices needed to be deleted from G to disconnect G.

Definition 1.2 (Vertex cut)**.** For $G = (V, E)$, partition (L, S, R) on set of vertices V is said to be a vertex cut if $L, R \neq \phi$ and $E_{G \setminus S}(L, R) = \phi$, i.e. vertices in S separates L from R. $|S|$ is called the size of vertex cut.

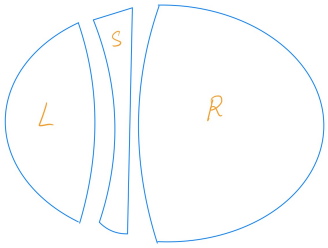


Figure 1: Vertex Cut

The size $|S^*|$ of the smallest possible vertex cut (L^*, S^*, R^*) is equal to the vertex connectivity of graph G denoted by $\kappa(G)$ or just κ.

Definition 1.3 ((s,t)−vertex cut)**.** A set $S \subset V$ is said to be a (s,t)− vertex cut for vertices $s, t \in V$ if removing the nodes in S will disconnect vertex s from t. It is also called (s,t)−separator.

Definition 1.4 ((s,t)− vertex mincut)**.** The smallest possible (s,t)− vertex cut is called (s,t)− vertex mincut or minimum (s,t)−separator and denote its size by $\kappa(s,t)$.

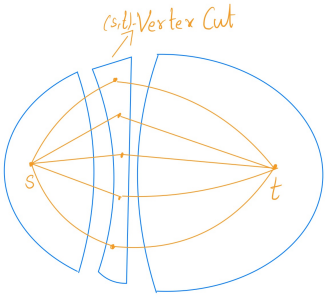


Figure 2: (s,t)-Vertex Cut

1.1 Naive Method

By max flow min cut theorem $\kappa(s,t)$ is the maximum number of vertex disjoint paths. Hence, we can compute $\kappa(s,t)$ in one max-flow call.

Hence, $\kappa(G)$ is obtained by taking minimum over $\kappa(s,t)$ all possible (s,t) pairs in the graph and finding the minimum separator of the graph.

$$\kappa(G) = \min_{s,t} \kappa(s,t)$$

However, this naive method takes $O(n^2)$ max flow calls. Today we will see a deterministic algorithm that takes $O(n^{1.5} \log n)$ max flow calls based on the paper by [Gabow'00].

1.2 State of the Art

There is a deterministic algorithm by [Gabow'00] that takes $O(m \cdot (n + \min\{\kappa^{5/2}, \kappa n^{3/4}\}) = O(mn^{1.75})$ time when κ is as big as $O(n)$. [1] It is the only fast algorithm that exploits Ramanujan Graphs as far as we know. This algorithm is much slower than the current algorithm that we present.

There is a randomized algorithm by [LNP**S**Y'21] which takes polylog(n) "max flow calls". Current best algorithm for max flow runs in $O(m^{4/3+o(1)})$ time. [2] The max flow calls are made on smaller graphs with fewer edges. The total size of the graph that they run max flow is still near-linear $\tilde{O}(m)$.

1.3 Overall Plan

Given an unweighted undirected graph G, the goal would be to return the minimum vertex cut (L, S, R) of size $\kappa(G)$.

Note: **We assume** $\delta \leq 9n/10$ to avoid annoying corner case where δ denote the minimum vertex degree.

[1] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.1674&rep=rep1&type=pdf
[2] https://arxiv.org/abs/2104.00104

2 Algorithm for vertex connectivity

There are two cases depending on how balanced the vertex cut is. We first assume that the minimum vertex cut is balanced and find it. Next we reduce the unbalanced case to balanced case.

Definition 2.1 ($\beta-$balanced vertex cut)**.** A vertex-cut (L, S, R) is said to be β**-balanced** if $|L| \geq \beta$ and $|R| = \Omega(n)$.

The main idea underlying the algorithm is as follows. Let (L^*, S^*, R^*) be the minimum vertex cut of size k. If we some how know a vertex $x \in L^*$ and $y \in R^*$, then we can do a max flow call to find minimum $(x, y)-$ vertex cut (L, S, R) whose size is same as $|S^*|$ (although exact cut may be different). This is because S^* is a $(x, y)-$vertex cut whence $\kappa(x, y) \leq k$; otoh since (L^*, S^*, R^*) is minimal, $\kappa(x, y) \geq k \implies \kappa(x, y) = k$.

However, we do not know which pair of vertices belongs to L^*, R^* respectively, whence the naive way is to try all possible pairs however this may take as large as $O(n^2)$ max flow calls.

But can we reduce the number of max flow calls using the "promise" that our vertex cut is $\beta-$balanced?

Yes, this is our next idea for an improved algorithm. By using the structural property of the Ramanujan Graphs we can decrease the number of pairs on which we need to call max flow, for finding the $\beta-$balanced cut. We will state the structural property of the Ramanujan Graph here without proof.

Corollary 2.2. *Let $H = (V, E)$ be a d-regular Ramanujan graph. For any $L, R \subseteq V$, if $|L| \cdot |R| \cdot d > 4n^2$, then $E(L, R) \neq \emptyset$.*

We want to identify a vertex pair such that they belong to either sides of the minimum vertex cut (L^*, S^*, R^*). From the above corollary we have that whenever $|L| \cdot |R| \cdot d > 4n^2$ then there always exist an edge from L to R.

Since we want to find a vertex pair between L^* and R^* when $|L^*| \geq \beta$ and $|R^*| = \Omega(n)$ we can as well set $d = \Theta(n/\beta)$ such that $|L^*| \cdot |R^*| \cdot d > 4n^2$ is satisfied and then by the property of Ramanujan Graph (H) we have a guaranteed edge between L^*, R^*.

However, we do not know which edge is between L^* and R^* (or really, we don't know where L^* and R^* are) so we use all edges of Ramanujan Graph (E_H) as candidate pairs for calling max flow and take the minimum value of the vertex cut between all of them. By the property of Ramanujan Graph and the balanced nature of minimum vertex cut we are guaranteed to find such a pair which gives us minimum cut.

Lemma 2.3. *If there exists a β-balanced vertex-mincut (L^*, S^*, R^*), then we can find a vertex-cut of size at most k in $O(n \times \frac{n}{\beta})$ max flow calls.*

Proof. We use the edges of H, E_H to get candidate pairs for finding the balanced cut. Since there can be at most $n \cdot d$ edges in a $d-$regular graph, $|E_H| = O(n^2/\beta)$ which implies at most $O(n^2/\beta)$ many max flow calls. □

Note that H is totally different graph from G but shares the same vertices.

3 Reduction to Balanced Case

We saw above that we can improve over the naive $O(n^2)$ max flow calls to $O(n^2/\beta)$ calls when we know that minimum vertex cut is balanced. However we do not know if the (L^*, S^*, R^*) is balanced. So in this section we present a way to reduce the unbalanced case to balanced case.

Our plan would be to gradually modify the graph to make it balanced and then solve it in the above mentioned way. We measure the balanced-ness of the minimum cut using a condition called the **gap condition**. We keep on improving gap condition until that it is sufficient to imply "every" minimum cut is $\beta-$balanced.

Note that balanced case requires that at least one of the all possible minimum cuts need to be balanced. But gap condition ensures that all minimum cuts are balanced.

3.1 Gap Condition

We know that $\kappa(G) = \kappa \leq \delta$ because a vertex with degree δ can be separated from the rest of the graph by removing its neighbours which are of size δ. Hence minimum cut can only be either smaller or equal.

Since $\kappa \leq \delta$, the difference between the min degree and min cut, $\delta - \kappa$ is called **Gap**. Why is this quantity useful? It is formalized in the lemma below.

Lemma 3.1. *Every mincut (L^*, S^*, R^*) is β-balanced where $\beta = \delta - \kappa$.*

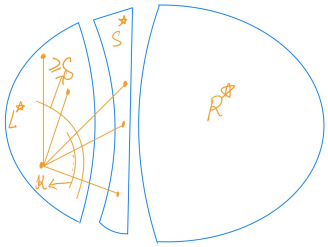


Figure 3: Gap Condition

Proof. Let $x \in L^*$ (we know L^* is non empty) and $N(x)$ be the set of x's neighbors. We have $N(x) \subseteq L^* \cup S^*$ as it cannot have edges to R^*.

$$
\begin{aligned}
|N(x)| &\geq \delta && \text{(minimum degree)} \\
|N(x)| &\leq |L^*| + |S^*| = |L^*| + \kappa \\
|L^*| &\geq \delta - \kappa
\end{aligned}
$$

There are always $\delta - \kappa$(**the gap** between minimum degree and minimum cut) vertices on both sides of the minimum cut. □

So to make the minimum cut balanced we need to increase the gap $\delta - \kappa$. We need more structural understanding of the vertex connectivity to see why we can do this and how to do this.

Definition 3.2 (x-rooted vertex connectivity)**.** The minimum number of vertices needed to disconnect x from any other vertex in the graph.

$\kappa(x) = \min_{y \neq x} \kappa(x, y)$ which can be computed in $n - 1$ max flows.

Lemma 3.3. *If* $\kappa(x) > \kappa(G)$*, then* $\kappa(G \setminus x) = \kappa(G) - 1$.

Proof. If $\kappa(x) > \kappa(G)$ then for all possible min cuts (L^*, S^*, R^*), $x \in S^*$ if not $\kappa(x) = \kappa(G)$.

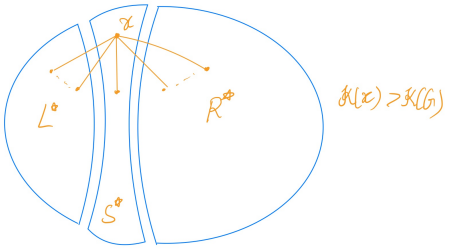


Figure 4: Remove vertex to decrease κ

$\kappa(G \setminus x)$ cannot be $< \kappa(G) - 1$ because then we can add x back and get a min cut for graph G of size $< \kappa(G)$ which is contradiction.

$\kappa(G \setminus x)$ need not be $> \kappa(G) - 1$ because we already have a cut of size $= \kappa(G) - 1$ which is $S^* \setminus \{x\}$ where S^* is any min cut separator and we also know that $x \in S^*$. □

We use the above structural lemma as follows. Compute $\kappa(x)$ along with the corresponding cut. If $\kappa(x) = \kappa$ then we have a found a vertex cut of optimal size. Otherwise $\kappa(x) > \kappa$ then remove x from the graph and compute $\kappa(G \setminus x)$ and according to above lemma we know that $\kappa(G) = \kappa(G \setminus x) + 1$.

From now on we use κ to always refer to the minimum vertex cut size of the current graph which is $G \setminus x$ after x is removed.

However, if we repeat the above step it may take $O(n)$ steps to find min-cut as κ can be $O(n)$. Since each step takes $O(n)$ max flows we are again at square one. Recall our target is to increase the gap between δ and κ.

Using the above step we have reduced κ by 1. So gap should increase if δ has stayed the same. However, by removing x, δ can also decrease by 1 if $N(x)$ contains some vertex of minimum degree, in which case the gap cannot be increased.

Let F be the set of neighbors of x such that $\forall x' \in F$, the degree of x' become $\delta - 1$ after removing x. So all nodes in F need to be "fixed" so that their degree become δ again. We can fix them without affecting the min cut size for which we need the following structural lemma.

Lemma 3.4. *If* $\kappa(x, y) > \kappa$*, then* $\kappa(G \cup (x, y)) = \kappa(G)$.

Proof. $\kappa(x, y) > \kappa$ then x, y always belong to same side either $L^* \cup S^*$ or $S^* \cup R^*$ for every possible min cut (L^*, S^*, R^*). If not $\kappa(x, y)$ would have been κ. So adding an edge between x, y (if it does not already exist) does not affect any minimum vertex cut. □

For any vertex w compute $\kappa(w)$ and if it is $= \kappa$ then we are done, else we can add edges to any other vertex from this without effecting the min cut as it belongs to S^* for every min cut (L^*, S^*, R^*).

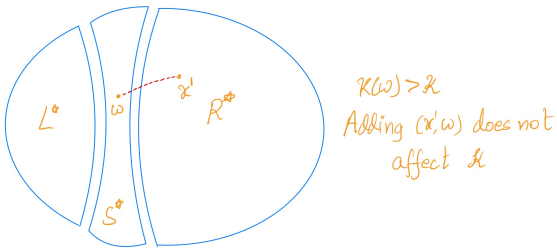


Figure 5: Add edge (w, x') to G

- Let $F_w = \{x' \in F \mid (x', w) \notin E\}$ be all nodes that need to be fixed, and **not** adjacent to w.
- For each $x' \in F_w$, we can add the edge (x', w) into the graph without changing κ.
- So all nodes in F_w are now fixed!

However, F_w might be very small when compared to F and may take many max flows to fix all vertices in F. We show below that it is not the case.

Lemma 3.5. *Let G' be the current graph after removing x. There exists $w \in V(G')$ such that*

$$|F_w| \geq |F| \cdot (1 - \frac{\delta - 1}{|V(G')|})$$

.

Proof. We have $\text{vol}_{G'}(F) \leq |F|(\delta - 1)$ as each vertex in F is of degree $\delta - 1$. So there is a vertex $w \in V(G')$ such that $|E_{G'}(w, F)| \leq \frac{|F|(\delta-1)}{|V(G')|}$.

That is, w is adjacent to at most $\frac{|F|(\delta-1)}{|V(G')|}$ vertices in F. So we can fix at least $|F| \cdot (1 - \frac{\delta-1}{|V(G')|})$ vertices in F by adding edges to w. □

We will see that the algorithm will guarantee $|V(G')| \geq n - \sqrt{n}$, and we assume $\delta \leq \frac{9}{10}n$ from the beginning. Hence, $|F_w| \geq |F| \cdot (1 - \frac{\delta-1}{|V(G')|}) \geq |F|/100$. So we fix at least $(1/100)^{th}$ fraction of vertices in F every time by incurring an overhead of n max flows for computing $\kappa(w)$.

We find w by scanning through all possible vertices and find the vertex w that fixes at least $(1/100)^{th}$ fraction of vertices in F, which is guaranteed to exist based on the above lemma. So it takes a total of $O(\log n)$ rooted vertex connectivity calls to fix all vertices. Once all vertices are fixed we have successfully increased the minimum degree to δ without increasing the minimum vertex cut $\kappa(G \setminus x) = \kappa(G) - 1$ and hence increasing the gap by 1. This took a total of $O(n \log n)$ max flow calls.

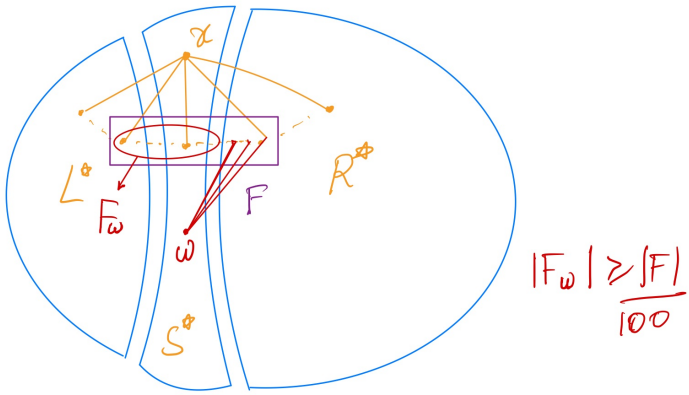


Figure 6: Vertex Cut

Remark 3.6. Note that there is an easy way to just find any arbitrary non-neighbour w of a vertex for a $x' \in F$ and compute $\kappa(x', w)$.

- If $\kappa(x', w) = \kappa$, since we track of minimum vertex cut we found so far, we are done.
- If $\kappa(x', w) > \kappa$, then we can fix $x' \in F$ by adding edge to w.

iHowever we do not choose the above said process and use $O(\log n)$ rooted vertex connectivity calls because, if later we can speed up the algorithm for problem of finding rooted vertex connectivity faster than $o(n)$ max flow calls which immediately implies an improved algorithm for vertex connectivity problem.

Finding rooted vertex connectivity in $o(n)$ max flow calls is an independent and very interesting open problem.

3.2 Summary of the Algorithm in the Unbalanced Case

Algorithm 1: $IncreaseGap(G)$

Result: Graph in which the gap, $\delta - \kappa$, is increased by 1
Choose any $x \in V$. Compute $\kappa(x)$ and the corresponding cut
Let $G' = G \setminus x$
Let $F = \{x' \in N(x) \mid \deg_{G'}(x') = \delta - 1\}$
while $F \neq \phi$ **do**
  Iterate over V to find w such that $|F_w| \geq |F|/100$
  Compute $\kappa(w)$ and the corresponding cut and set $\hat{\kappa}_{G'} \leftarrow \min\{\hat{\kappa}_{G'}, \kappa(w)\}$
  For each $x' \in F_w$, add edge (x', w) to G'
end

Note that if $\kappa(x) > \kappa$, then the gap is increased by 1 and our candidates for minimum cut are $\kappa(x)$ and $\hat{\kappa}_{G'} + 1$; otherwise if $\kappa(x) = \kappa$ or $\kappa(w) = \kappa$, then we had found the minimum vertex cut of G and $\kappa(G)$. If we repeated this process for β times and in each round we have $\kappa(x) > \kappa$ we could increase the gap by β, the vertex min cut in the resulting graph is β-balanced. Meanwhile we keep

track of the current smallest cut and update it using $\min\{\kappa(x), \hat{\kappa}_{G'} + 1\}$ and the corresponding cut. In this way, if at any round we found κ, then it will be recorded.

We finally use the balanced case to solve this graph for minimum vertex cut in the resulting graph and then ripple back through the deletions of vertices to find the minimum cut among all possible candidates.

3.3 Correctness

All the while we are comparing our rooted vertex connectivities with the minimum vertex connectivity κ however we do not know it before hand.

It is not an issue as we only over estimate the minimum cut. If we ever find the minimum cut exactly somewhere in our algorithm. Since we store all the cut values and take the minimum. Hence, we are guaranteed to find the minimum cut.

If we never found the minimum cut while reducing to balanced case, then according to our algorithm we will find the minimum cut after reducing it to balanced case. So, either way the minimum of all our cuts will give us the minimum cut.

3.4 Runtime

- We call $\beta \times O(n \log n)$ max flows to reduce the problem to the β-balanced case.
- In the β-balanced case, we can solve the problem in $O(n \times \frac{n}{\beta})$ max flows.
- So a total of $O(\frac{n^2}{\beta} + \beta n \log n)$ max flow calls. By choosing $\beta = \sqrt{n}$, you can solve the problem in $O(n^{1.5} \log n)$ max flow calls.

4 Exercises and Open Questions

Exercise 4.1. Show an algorithm for vertex connectivity that takes $O(n \cdot \kappa)$ max flow calls.

Proof. At the i^{th} iteration, pick an arbitrary point $x_i \in V$ and compute $\kappa(x)$ using $O(n)$ max flow calls and record $k_i = \min_{j \le i} \kappa(x_j)$. If $k_i \le i - 1$, then return k_i. The algorithm is going to terminate since as long as $i > \kappa$, then we are guaranteed to pick a point that is not in S^* where (L^*, S^*, R^*) is a minimum vertex cut, whence the algorithm halts in at most $\kappa + 1$ iterations. The algorithm is correct since if $i \le \kappa$ and $k_i > \kappa \ge i$ then the algorithm won't halt. □

Question 4.2 (Open)**.** *Can we deterministically solve vertex connectivity using $O(n)$ max flow calls or even less?*

Note the that for randomized case, it is just polylog(n) max flows now. Something should be possible!

Question 4.3 (Open)**.** *The state of the art for **weighted** vertex connectivity.*

- *Randomized:* $\tilde{O}(mn)$
- *Deterministic:* $\tilde{O}(m^2)$

- So if you can find
  - a randomized algorithm using $o(n)$ max flow
  - a deterministic algorithm using $o(n^2)$ max flow
- This would improve the state of the art.
- Any hardness from fine-grained complexity would be interesting too!

University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 11: Expander: Probabilistic view

October 5, 2021

Instructor: Thatchaphol Saranurak

Scribe: Aditya Anand

1 A probabilistic view of expanders

In this lecture, we shall study yet another characterization of expanders - **expanders are graphs on which random walks mix rapidly.** We shall see that there is a direct connection between the second eigenvalue of the normalized laplacian and the mixing time of a random walk on graphs.

2 Random Walks

Let G be a graph with degree profile $\boldsymbol{d}$. We define a random walk step starting from a given vertex as follows.

Definition 2.1. (Random Walk Step) Given an undirected unweighted graph G and a vertex u, the result of one step of a random walk from u is a uniformly random neighbour of u in G.

The same definition can naturally be extended to weighted graphs.

Definition 2.2. (Random Walk Step in Weighted Graphs) Given an undirected weighted graph G with weight function $w : V(G) \to \mathbb{R}$ and a vertex u, the result of one step of a random walk from u is the random vertex obtained by sampling a single vertex from $N_G(u)$ where $x \in N_G(u)$ is sampled with probability $\frac{w(u,x)}{\sum_{v \in N_G(u)} w(u,v)}$.

Today's lecture is motivated by the following natural questions, which can be better understood with an example.

Example 2.3. Given a cycle of length n, if we start at some vertex u and do a random walk for a sufficiently large number of steps, what is the distribution of vertices we end up with?

Intuitively, we understand that the answer is the uniform distribution. But can we prove this? We can also ask how many steps does it take so that $\Pr[\text{end at } u] = \frac{1 \pm \epsilon}{n}$ for all u - that is, how fast do we converge to the uniform distribution?

For a general graph, these questions become

- Do random walks converge?

Figure 1: Random walk illustration

- If so, to what distribution does the random walk converge?
- What is the rate of convergence - how fast does the random walk "mix" or converge to the final distribution?

The goal of this lecture will be to answer these questions, and understand these in the context of expanders - we will show that expanders have small mixing times.

3 Walk Matrices

Let us now formalize the notion of a random walk. Let $\boldsymbol{p}_t \in \mathbb{R}^V$ denote the probability distribution over vertices after time t. Suppose initially we start at a vertex u, so $\boldsymbol{p}_0 = \mathbf{1}_u$. Then $\boldsymbol{p}_1 = \frac{1}{\deg(u)} \cdot \mathbf{1}_{N(u)}$ where $N(u)$ is the set of neighbors of u. It is clear that

$$\boldsymbol{p}_{t+1}(u) = \sum_{(v,u)\in E} \frac{w(u,v)}{d(v)} \boldsymbol{p}_t(v).$$

Definition 3.1. (Walk matrix) The walk matrix $\boldsymbol{W}$ is the matrix $\boldsymbol{W}$ that satisfies $\boldsymbol{p}_{t+1} = \boldsymbol{W}\boldsymbol{p}_t$. From the above expression, we can see that

$$\boldsymbol{W} = \boldsymbol{A}\boldsymbol{D}^{-1}.$$

After t random walk steps, we have

$$\boldsymbol{p}_t = \boldsymbol{W}\boldsymbol{p}_{t-1} = \boldsymbol{W}^t \boldsymbol{p}_0.$$

We can now formalize the questions from the previous section as follows.

- Convergence - What is the limit of $\boldsymbol{p}_t$ when $t \to \infty$ (if it exists)?
- Rate of convergence - How fast does $\boldsymbol{p}_t$ converge to that limit?

Unfortunately, it turns out that the above limit may not exist for a general graph. In fact, consider a trivial graph consisting of only one edge (u, v). If we do a random walk starting from u, then at every odd step we are at v with probability 1 and at every even step we are at u with probability 1 - from which it is clear that the limit $\boldsymbol{p}_t$ when $t \to \infty$ does not exist.

In fact this limit does not exist for any bipartite or disconnected graph[1] - at every step the total probability mass on one partite set is zero. This suggests that we need a different notion of random walk to ensure convergence for every graph. Does there exist a natural random walk that converges for any graph? The answer is indeed yes, as we shall see in the subsequent section.

4 Lazy Random Walk

We define a lazy random walk step as follows.

Definition 4.1. (Lazy Random Walk Step) Given an undirected unweighted graph G and a vertex u, the result of one step of a lazy random walk is u with probability $\frac{1}{2}$, otherwise it is the result of a random walk step with u.

Let $\boldsymbol{p}_t$ be the probability mass vector after t lazy random walk steps. We now define a new walk matrix $\widetilde{\boldsymbol{W}}$ corresponding to the lazy random walk, so that we have

$$\boldsymbol{p}_{t+1} = \widetilde{\boldsymbol{W}}\boldsymbol{p}_t,$$

What is the matrix W? Since we stay put with probability $\frac{1}{2}$ and do a random walk with probability $\frac{1}{2}$, it follows that

$$\begin{aligned}\widetilde{\boldsymbol{W}} &= \boldsymbol{I}/2 + \boldsymbol{W}/2 \\ &= \boldsymbol{I}/2 + \boldsymbol{A}\boldsymbol{D}^{-1}/2\end{aligned}$$

Let us now return to our trivial example graph with one edge (u, v) and ask the question if the limit of $\boldsymbol{p}_t = \widetilde{\boldsymbol{W}}^t \boldsymbol{p}_0$ when $t \to \infty$ exists. It is clear that $p_t(u) = p_t(v) = \frac{p_{t-1}(v)+p_{t-1}(u)}{2}$, and hence the lazy random walk converges to $p_t(u) = p_t(v) = \frac{1}{2}$ in just one step.

5 The Stable Distribution

Theorem 5.1. *In a connected undirected graph G, for any $\boldsymbol{p}_0 \in \mathbb{R}^V$,*

$$\lim_{t\to\infty} \widetilde{\boldsymbol{W}}^t \boldsymbol{p}_0 = \frac{1}{\boldsymbol{d}(V)} \cdot \boldsymbol{d}$$

Essentially, the above result says that irrespective of the initial distribution, the lazy random walk converges on any graph. Further, it converges to the same distribution, where each vertex has probability mass proportional to its degree.

Proof. We have

$$\begin{aligned}\boldsymbol{p}_t &= \widetilde{\boldsymbol{W}}^t \boldsymbol{p}_0 \\ &= \boldsymbol{D}^{1/2}\boldsymbol{D}^{-1/2}\widetilde{\boldsymbol{W}}^t\boldsymbol{D}^{1/2}\boldsymbol{D}^{-1/2}\boldsymbol{p}_0\end{aligned}$$

[1] It can be shown that this random walk does not converge if and only if G is disconnected or bipartite.

$$= \boldsymbol{D}^{1/2}(\boldsymbol{D}^{-1/2}\widetilde{\boldsymbol{W}}\boldsymbol{D}^{1/2})^t\boldsymbol{D}^{-1/2}\boldsymbol{p}_0$$

Recall that

$$\widetilde{\boldsymbol{W}} = \frac{1}{2}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{D}^{-1})$$

So

$$\boldsymbol{D}^{-1/2}\widetilde{\boldsymbol{W}}\boldsymbol{D}^{1/2} = \frac{1}{2}(\boldsymbol{I} + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}) = \boldsymbol{I} - \frac{1}{2}\boldsymbol{N}.$$

Now, we write

$$\boldsymbol{p}_t = \boldsymbol{D}^{1/2}(\boldsymbol{I} - \frac{1}{2}\boldsymbol{N})^t\boldsymbol{D}^{-1/2}\boldsymbol{p}_0.$$

Note that $\boldsymbol{N}$ and $\boldsymbol{I} - \frac{1}{2}\boldsymbol{N}$ have the same eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ and since

$$0 = \lambda_1(\boldsymbol{N}) \leq \cdots \leq \lambda_n(\boldsymbol{N}) \leq 2,$$

we have

$$1 = \lambda_1(\boldsymbol{I} - \frac{1}{2}\boldsymbol{N}) \geq \cdots \geq \lambda_n(\boldsymbol{I} - \frac{1}{2}\boldsymbol{N}) \geq 0$$

where

$$\lambda_i(\boldsymbol{I} - \frac{1}{2}\boldsymbol{N}) = 1 - \lambda_i(\boldsymbol{N})/2$$

We can make use of the eigen-decomposition as follows

$$\boldsymbol{D}^{-1/2}\boldsymbol{p}_0 = \sum_i c_i\boldsymbol{v}_i \qquad \text{where } c_i = \left\langle \boldsymbol{v}_i, \boldsymbol{D}^{-1/2}\boldsymbol{p}_0 \right\rangle$$

Therefore it follows that

$$\begin{aligned}
\boldsymbol{p}_t &= \boldsymbol{D}^{1/2}(\boldsymbol{I} - \frac{1}{2}\boldsymbol{N})^t \sum_i c_i\boldsymbol{v}_i \\
&= \boldsymbol{D}^{1/2} \sum_i \left(\lambda_i(\boldsymbol{I} - \frac{1}{2}\boldsymbol{N})\right)^t c_i\boldsymbol{v}_i \\
&= \boldsymbol{D}^{1/2}c_1\boldsymbol{v}_1 + \boldsymbol{D}^{1/2} \sum_{i\geq 2} \left(1 - \frac{\lambda_i(\boldsymbol{N})}{2}\right)^t c_i\boldsymbol{v}_i
\end{aligned}$$

Since $\lambda_i(\boldsymbol{N}) > 0$ for $i \geq 2$ because G is connected, we have

$$\boldsymbol{p}_t = \boldsymbol{D}^{1/2}c_1\boldsymbol{v}_1 \text{ when } t \to \infty$$

Recall that the first eigenvector $\boldsymbol{v}_1$ of $\boldsymbol{N}$ is $\boldsymbol{v}_1 = \frac{\boldsymbol{d}^{1/2}}{\sqrt{\boldsymbol{d}(V)}}$. So, we have

$$c_1 = \left\langle \frac{\boldsymbol{d}^{1/2}}{\sqrt{\boldsymbol{d}(V)}}, \boldsymbol{D}^{-1/2}\boldsymbol{p}_0 \right\rangle = \left\langle \frac{\boldsymbol{1}}{\sqrt{\boldsymbol{d}(V)}}, \boldsymbol{p}_0 \right\rangle = \frac{1}{\sqrt{\boldsymbol{d}(V)}}$$

and

$$\boldsymbol{p}_t = \boldsymbol{D}^{1/2}\frac{1}{\sqrt{\boldsymbol{d}(V)}}\frac{\boldsymbol{d}^{1/2}}{\sqrt{\boldsymbol{d}(V)}} = \frac{1}{\boldsymbol{d}(V)} \cdot \boldsymbol{d}$$

when $t \to \infty$. □

6 Rate of Convergence

We call $\boldsymbol{\pi} = \lim_{t\to\infty} \widetilde{\boldsymbol{W}}^t \boldsymbol{p} = \frac{1}{\boldsymbol{d}(V)} \cdot \boldsymbol{d}$ the **stable/stationary distribution.**

Observation 6.1.

$$\boldsymbol{\pi} = \widetilde{\boldsymbol{W}}\boldsymbol{\pi}.$$

That is, doing a random walk from the stationary distribution keeps it intact.

Proof. Suppose that $\pi_{t-1}(v) = \frac{d(v)}{d(V)}$ for each vertex v. Fix a vertex x. The total probability mass that lands at x from y after one more random walk step is $\frac{1}{d(y)}\frac{d(y)}{d(V)}$ if $(x, y) \in E(G)$. Now summing over all such y we get $\pi_t(x) = \sum_{y\in N_G(x)} \frac{1}{d(V)} = \frac{d(x)}{d(v)} = \pi_{t-1}(x)$. □

This brings us to our next question on the rate of convergence. How fast does $\boldsymbol{W}^t\boldsymbol{p}$ converge to $\boldsymbol{\pi}$ as a function of t?

Theorem 6.2. *For all $a, b \in V$ and t, suppose $\boldsymbol{p}_0 = \mathbf{1}_a$. Then*

$$|\boldsymbol{p}_t(b) - \boldsymbol{\pi}(b)| \le \sqrt{\frac{\boldsymbol{d}(b)}{\boldsymbol{d}(a)}}\left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t.$$

That is, $\lambda_2(\boldsymbol{N})$ dictates how fast $\boldsymbol{p}_t$ converges to the stable distribution $\boldsymbol{\pi}$. On $\Omega(1)$ expanders, we have $\lambda_2(\boldsymbol{N}) \ge \Omega(1)$ and the bound decreases exponentially.

Proof. From previous analysis we have

$$\boldsymbol{p}_t = \widetilde{\boldsymbol{W}}^t\boldsymbol{p}_0 = \boldsymbol{D}^{1/2}c_1\boldsymbol{v}_1 + \boldsymbol{D}^{1/2}\sum_i \left(1 - \frac{\lambda_i(\boldsymbol{N})}{2}\right)^t c_i\boldsymbol{v}_i$$

where

$$c_i = \left\langle \boldsymbol{v}_i, \boldsymbol{D}^{-1/2}\boldsymbol{p}_0 \right\rangle$$

So

$$\begin{aligned}\boldsymbol{p}_t(b) = \mathbf{1}_b^\top\boldsymbol{p}_t &= \mathbf{1}_b^\top\boldsymbol{D}^{1/2}c_1\boldsymbol{v}_1 + \mathbf{1}_b^\top\boldsymbol{D}^{1/2}\sum_i \left(1 - \frac{\lambda_i(\boldsymbol{N})}{2}\right)^t c_i\boldsymbol{v}_i \\ &= \boldsymbol{\pi}(b) + \mathbf{1}_b^\top\boldsymbol{D}^{1/2}\sum_i \left(1 - \frac{\lambda_i(\boldsymbol{N})}{2}\right)^t c_i\boldsymbol{v}_i\end{aligned}$$

That is,

$$\boldsymbol{p}_t(b) - \boldsymbol{\pi}(b) = \mathbf{1}_b^\top\boldsymbol{D}^{1/2}\sum_i \left(1 - \frac{\lambda_i(\boldsymbol{N})}{2}\right)^t c_i\boldsymbol{v}_i$$

and we only need to bound the right hand side. To do this, observe that

$$c_i = \left\langle \boldsymbol{v}_i, \boldsymbol{D}^{-1/2}\boldsymbol{p}_0 \right\rangle = \frac{1}{\sqrt{\boldsymbol{d}(a)}}\boldsymbol{v}_i^\top\mathbf{1}_a$$

also

$$\mathbf{1}_b^\top \boldsymbol{D}^{1/2} \boldsymbol{v}_i = \sqrt{\boldsymbol{d}(b)} \mathbf{1}_b^\top \boldsymbol{v}_i$$

We have

$$\mathbf{1}_b^\top \boldsymbol{D}^{1/2} \sum_i \left(1 - \frac{\lambda_i(\boldsymbol{N})}{2}\right)^t c_i \boldsymbol{v}_i = \sqrt{\frac{\boldsymbol{d}(b)}{\boldsymbol{d}(a)}} \sum_i \left(1 - \frac{\lambda_i(\boldsymbol{N})}{2}\right)^t \cdot \mathbf{1}_b^\top \boldsymbol{v}_i \boldsymbol{v}_i^\top \mathbf{1}_a$$

Now, we are ready to bound the second term

$$\begin{aligned}
&\left| \sum_i \left(1 - \frac{\lambda_i(\boldsymbol{N})}{2}\right)^t \cdot \mathbf{1}_b^\top \boldsymbol{v}_i \boldsymbol{v}_i^\top \mathbf{1}_a \right| \\
&\leq \left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t \sum_i |\mathbf{1}_b^\top \boldsymbol{v}_i| \cdot |\boldsymbol{v}_i^\top \mathbf{1}_a| \\
&\leq \left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t \sqrt{\sum_i (\mathbf{1}_b^\top \boldsymbol{v}_i)^2} \cdot \sqrt{\sum_i (\mathbf{1}_a^\top \boldsymbol{v}_i)^2} && \text{by Cauchy-Schwartz} \\
&= \left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t \|\mathbf{1}_b\| \cdot \|\mathbf{1}_a\| && \text{as } \boldsymbol{v}_i \text{ form an orthonormal basis} \\
&= \left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t
\end{aligned}$$

So we can now conclude that

$$|\boldsymbol{p}_t(b) - \boldsymbol{\pi}(b)| \leq \sqrt{\frac{\boldsymbol{d}(b)}{\boldsymbol{d}(a)}} \left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t.$$

□

7 Mixing Time

Definition 7.1. We say that a walk after step t has ϵ**-mixed** if

$$|\boldsymbol{p}_t(b) - \boldsymbol{\pi}(b)| \leq \epsilon \boldsymbol{\pi}(b)$$

for all vertices b.

In particular, if the walk at step t is ϵ-mixed, the **total variation** between $\boldsymbol{p}_t$ and $\boldsymbol{\pi}$ which is defined as$\|\boldsymbol{p}_t - \boldsymbol{\pi}\|_{TV} = \frac{1}{2} \sum_b |\boldsymbol{p}_t(b) - \boldsymbol{\pi}(b)| \leq \frac{1}{2} \sum_b \epsilon \boldsymbol{\pi}(b) = \frac{\epsilon}{2}$ is at most $\epsilon/2$.

Definition 7.2. (Mixing Time) Fix some random walk process. The **mixing time** $\tau_{\text{mix}}(G, \epsilon)$ of the graph G is given by

$$\tau_{\text{mix}}(G, \epsilon) = \max_{\boldsymbol{p}_0} \min_t \{t \mid \|\boldsymbol{p}_t - \boldsymbol{\pi}\|_{TV} \leq \epsilon\}.$$

In English, mixing time is the minimium amount of steps for total variation to be ϵ small starting with any possible initial distribution.

7.1 Mixing time and eigenvalues: $\tau_{\text{mix}}(G, \epsilon) \approx 1/\lambda_2(\boldsymbol{N}_G)$

The theorem below shows that $\tau_{\text{mix}}(G, \epsilon)$ is just another way to think about $\lambda_2(\boldsymbol{N}_G)$!

Theorem 7.3. *We have that*

$$\Omega\left(\frac{\log(\frac{1}{\epsilon})}{\lambda_2(\boldsymbol{N}_G)}\right) \leq \tau_{\text{mix}}(G, \epsilon) \leq O\left(\frac{\log(\frac{\boldsymbol{d}(V)}{\epsilon \boldsymbol{d}_{\min}})}{\lambda_2(\boldsymbol{N}_G)}\right)$$

where $\boldsymbol{d}_{\min} = \min_u \boldsymbol{d}(u)$.

Proof. We omit the proof of lower bound, and only prove the upper bound below.

First we show that the "worst" initial distribution that takes longest to mix the walk is

$$\boldsymbol{p}_0 = \mathbf{1}_{a^*}$$

for some vertex a^*.

Observation 7.4. *Pick any initial probability distribution $\boldsymbol{p}_0$ on the vertices. Then there exists a vertex a^* such that the probability distribution $\mathbf{1}_{a^*}$ mixes slower - that is, $\tau_{mix}(\boldsymbol{p}_0) < \tau_{mix}(\mathbf{1}_{a^*})$.*

Proof. We get $\|p_t - \boldsymbol{\pi}(t)\|_{TV} = \frac{1}{2}\sum_b |\boldsymbol{p}_t(b) - \boldsymbol{\pi}(b)|$. Let $p_t(a, b)$ denote the fraction of mass starting from a landing on b. Then clearly

$$\begin{aligned}
\|p_t - \boldsymbol{\pi}(t)\|_{TV} &= \frac{1}{2}\sum_b |\sum_a p_0(a)\boldsymbol{p}_t(a, b) - \boldsymbol{\pi}(b)| \\
&= \frac{1}{2}\sum_b |\sum_a (p_0(a)\boldsymbol{p}_t(a, b) - p_0(a)\boldsymbol{\pi}(b))| \\
&\leq \frac{1}{2}\sum_b \sum_a |p_0(a)\boldsymbol{p}_t(a, b) - p_0(a)\boldsymbol{\pi}(b)| \text{(Triangle inequality)} \\
&= \frac{1}{2}\sum_b \sum_a p_0(a)|\boldsymbol{p}_t(a, b) - \boldsymbol{\pi}(b)| \\
&= \frac{1}{2}\sum_a p_0(a) \sum_b |\boldsymbol{p}_t(a, b) - \boldsymbol{\pi}(b)| \\
&\leq \frac{1}{2}\max_a \sum_b |\boldsymbol{p}_t(a, b) - \boldsymbol{\pi}(b)|.
\end{aligned}$$

Suppose $a = a^*$ maximizes $\sum_b |\boldsymbol{p}_t(a, b) - \boldsymbol{\pi}(b)|$. Then it is clear that the random walk starting at a^* takes more time to mix than the random walk starting with the initial distribution p_0, and hence we are done.

□

We are now ready to prove the result. Fix a vertex a^* from which we perform the lazy random walk. That is, fix $p_0 = \mathbf{1}_{a^*}$ We have

$$|\boldsymbol{p}_t(b) - \boldsymbol{\pi}(b)| \leq \sqrt{\frac{\boldsymbol{d}(b)}{\boldsymbol{d}(a^*)}} \left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t.$$

So we ask that is the smallest t where

$$\begin{aligned}
\sqrt{\frac{\boldsymbol{d}(b)}{\boldsymbol{d}(a^*)}} \left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t &\leq \frac{\epsilon \boldsymbol{d}(b)}{\boldsymbol{d}(V)} && \iff \\
\left(1 - \frac{\lambda_2(\boldsymbol{N})}{2}\right)^t &\leq \epsilon \frac{\sqrt{\boldsymbol{d}(a^*)\boldsymbol{d}(b)}}{\boldsymbol{d}(V)} && \iff \\
\exp(-\Theta(\frac{t\lambda_2(\boldsymbol{N})}{2})) &\leq \epsilon \frac{\sqrt{\boldsymbol{d}(a^*)\boldsymbol{d}(b)}}{\boldsymbol{d}(V)} && \iff \text{using } 1 - x \approx \exp(-x) \\
-\Theta(\frac{t\lambda_2(\boldsymbol{N})}{2}) &\leq \ln(\epsilon \frac{\sqrt{\boldsymbol{d}(a^*)\boldsymbol{d}(b)}}{\boldsymbol{d}(V)}) && \iff \\
t &\geq \Theta(\frac{2}{\lambda_2(\boldsymbol{N})}) \ln(\frac{\boldsymbol{d}(V)}{\epsilon\sqrt{\boldsymbol{d}(a^*)\boldsymbol{d}(b)}}) && \iff
\end{aligned}$$

from which we get

$$\tau_{\text{mix}}(G, \epsilon) \leq O\left(\frac{\log(\frac{\boldsymbol{d}(V)}{\epsilon \boldsymbol{d}_{\min}})}{\lambda_2(\boldsymbol{N}_G)}\right).$$

□

7.2 Examples

- Let's try to answer the question about cycles from the beginning of the lecture.
  - Why uniform distribution? - we saw that the probability mass on a vertex is proportional to its degree, and hence we have the uniform distribution as the stationary distribution.
  - How fast does the distribution converge? For a cycle G, $\lambda_2(N_G) = \frac{1}{n^2}$. Hence the mixing time is $O(n^2 \log n)$.
- When it is not clear what is $\tau_{\text{mix}}(G, \epsilon)$ or $\lambda_2(\boldsymbol{N}_G)$, knowing one implies $O(\log n)$-approximation of the other.
  - When G is a $\tilde{\Omega}(1)$-expander, we know $\lambda_2(\boldsymbol{N}_G) = \tilde{\Omega}(1)$. So $\tau_{\text{mix}} = \tilde{O}(1)$.
  - When G is a path, we know $\tau_{\text{mix}} \approx n^2$ (why?), so $\lambda_2(\boldsymbol{N}_G) \approx 1/n^2$.
    We can see this from the conductance of path. Another good reason is that the expected transition distance after n steps of a random walk is of order $\sqrt{n}$,[2] whence it takes roughly n^2 steps to move n units of distance away from the starting point.

[2] https://mathworld.wolfram.com/RandomWalk1-Dimensional.html

- When G is a dumbell, $\tau_{\text{mix}} \approx n^2$ (why?), so $\lambda_2(\boldsymbol{N}_G) \approx 1/n^2$
  The idea is that there is roughly $1/n$ of probability for a the current point to move from one clique to an endpoint of the bridge, and it takes another $1/n$ of probability to move from one endpoint of the bridge to the other (so that the point will enter the other clique.)
- When G is a Bolas graph, $\tau_{\text{mix}} \approx n^3$ (why?), so $\lambda_2(\boldsymbol{N}_G) \approx 1/n^3$.
  Similarly, it takes roughly $1/n$ of probability to move from one clique to an endpoint of the bridge, and another $1/n^2$ of probability to move to the other endpoint by previous result for path.

- Actually, the Bolas graph is the *unweighted* graph which is hardest to mix.

Exercise 7.5 (Worst mixing time in unweighted graphs)**.** In the homework, you will show that for any unweighted graph G, $\lambda_2(\boldsymbol{N}_G) \geq \Omega(1/n^3)$. So $\tau_{\text{mix}}(G, 1/\text{poly}(n)) = O(n^3 \log n)$.

8 Applications of Random Walks

8.1 Short path in $\tilde{O}(\sqrt{n})$ time on Expanders

- **Question:** Given a ϕ-expander (undirected, unweighted) $G = (V, E)$ and $s, t \in V$, can we find a "short" path from s to t in sublinear time?

- Algorithm SHORTPATH
  - Perform a lazy random walk from s for $\tau_{\text{mix}}(G, \frac{1}{n^2}) = O(\frac{\log n}{\phi^2})$ steps.
  - Repeat for $O(\sqrt{n} \log n)$ times.
  - Let S be the set of vertices where the walks end.
  - We now do the same walk from t. Let T be the similar set of vertices.
  - If a there is a vertex $v \in S \cap T$, return a path from s to t as the union of paths between s and v and v and t.

  Note that if $S \cap T \neq \emptyset$, then we obtain an (s, t) path of length $O(\tau_{\text{mix}}(G, \frac{1}{n^2}))$. The running time of the algorithm is clearly $\tilde{O}(\sqrt{n} \cdot \tau_{\text{mix}}(G, \frac{1}{n^2}))$.

  Observation 8.1. *$S \cap T \neq \emptyset$ whp.*

  Proof. (Sketch) Essentially birthday paradox. Since the walks have mixed sufficiently, S and T are essentially uniformly random vertices (since G is connected). Thus the probability of their intersection being non-empty is the probability that given n bins, if we independently throw $O(\sqrt{n} \log n)$ balls uniformly at random, two of the balls land in the same bin. To analyze this formally, note that the probability that no two of the balls collide is at most $(1 - \frac{1}{n})(1 - \frac{2}{n}) \ldots (1 - \frac{\sqrt{n} \log n}{n})$ which is at most $e^{-\sum_{i=1}^{\sqrt{n} \log n} \frac{i}{n}} \sim e^{-\log^2 n} \ll \frac{1}{n^c}$ for any constant c. □

  Corollary 8.2. *If G is an $\Omega(1)$-expander, then algorithm* SHORTPATH *returns a $O(\log n)$-length (s, t)-path in $\tilde{O}(\sqrt{n})$ time.*

8.2 Connectivity in $O(\log n)$ bits of space

- **Question:** Given a (undirected and unweighted) graph $G = (V, E)$ and $s, t \in V$, is there a $s - t$ path?
- We shall assume that the adjacency lists of G are *read-only* and we can use only $O(\log n)$ bits of *working memory*.
- **The algorithm**
  - Start from s.
  - Perform lazy random walks for $O(n^5 \log^2 n)$ steps.
  - Whenever we visit t, return YES.
  - If we never visit t, return NO.
- If s and t are not connected, clearly the algorithm return NO.

Observation 8.3. *If s and t are in the same connected component, then the random walk will meet t whp.*

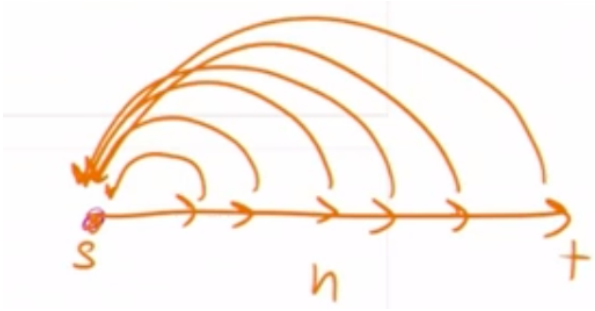
Proof. Let C be the connected component containing s and t.

We have $\tau_{\text{mix}}(C, \frac{1}{n^2}) \leq O(n^3 \log n)$ (as C is an unweighted graph - by exercise 7.5)

Therefore after every $O(n^3 \log n)$ steps, we visit t w.p. $\frac{d(t)}{d(C)} \geq \frac{1}{n^2}$. It follows that after $O(n^5 \log^2 n)$ steps, we never visit t w.p. at most $(1 - \frac{1}{n^2})^{O(n^2 \log n)} \leq 1/\text{poly}(n)$. □

Question 8.4. *Is it possible to improve the time to $\tilde{O}(n^2)$ or even $\tilde{O}(n)$ while using $O(\text{polylog}(n))$ space?*

For directed graph, the above algorithm won't work. Indeed, consider the graph in the below figure-in expectation it takes 2^n steps for s to reach t. An interesting open problem is that: is there an (even randomized) s-t connectivity algorithm use $O(n^{0.99})$ space runs in polylog(n) time? The current best known deterministic algorithm uses $O(n/2^{\sqrt{\log n}})$ space.



8.3 Sampling a Spanning Tree

The problem of sampling (almost uniformly) a spanning tree has been studied extensively in the literature. However, the "right" algorithm has been recently shown last year [3]. We will

[3] https://arxiv.org/abs/2004.07220

only discuss the high level ideas involved in the sampling process and skip the technical details.

- **Question:** Given a connected graph G, sample (almost)-uniformly at random a spanning tree T of G.
- **Key Idea:**
  - Given a spanning tree T, consider the following "flip" operation that
    - Choose one of the $m - (n-1)$ non-tree edges e
    - Look at the unique cycle C in $T \cup e$
    - Choose e' in C and remove e'.
    - We obtain another spanning tree $T' = T \cup e \setminus e'$.

Figure 2: The flip operation

- Consider the super graph $\mathcal{G}$ where
  - each node is a spanning tree
  - We add an edge between T and T' iff T can be flipped to T'.
- **Key idea (roughly speaking)**: although $\mathcal{G}$ is exponentially big, $\mathcal{G}$ is an expander. So it takes only $O(\log |V(\mathcal{G})|)$ random walk steps for get to sample a uniform spanning tree.

- Algorithm:
  - Start with any spanning tree.
  - Randomly flip for $O(m \log m)$ steps.
  - Return the tree.
- We can simulate the tree flipping using the link-cut tree data structure in $O(\log n)$ time. So the overall running time is $O(m \log^2 n)$.

9 Conclusion

Exercise 9.1. Suppose that the graph G is d-regular. Recall from the previous lecture that the power method for computing $\lambda_2(\boldsymbol{N})$ just computes $\widetilde{\boldsymbol{W}}^{O(\log(n)/\epsilon)}\boldsymbol{x}$ where $\boldsymbol{x}$ is a some random vector.

9.1 Alternate notions of expansion

Recall the various characterizations of expanders we have seen:

1. Robustness view: Robust towards edge deletions
2. Cut view: No sparse cuts, high conductance
3. Flow view: Can embed any degree restricted demand with small congestion
4. Spectral view: Small second eigenvalue of normalized laplacian
5. Probabilistic view: Random walks mix fast

Most of the theory we have seen is about conductance (i.e. edge expansion) in undirected graphs. But what about other notions?

Question 9.2. *Survey the equivalence we have seen so far for vertex expansion and/or directed graphs.*