**18.409: An Algorithmist’s Toolkit**

**Lecture 7**

Jonathan Kelner

---

**Today**

- Finish nonblocking routing networks
- Local and almost linear-time clustering and partitioning
  - Lovasz-Simonovits Theorem
  - Will sketch some of the proofs
  - Posted readings with details on web site

**Administrivia**

- Problem set on web site tonight
  - Due two weeks from today
- Typos in Powerpoint
  - Someone asked if I could correct them online after lecture
  - I’d be happy to, but I might not remember them after lecture
  - Maybe scribe for lecture could make a list and send it to me?
**Definition Reminder**

- **Definition**: A d-regular bipartite graph is an \((\alpha, \beta)\)-expander if every set \(S\) on the left with \(|S| \leq \alpha n\) has \(N(S) \geq \beta |S|\).
- Whenever \(\alpha \beta < 1\), there exists some \(d\) s.t. almost all d-regular graphs on \(n\) nodes (\(n\) sufficiently large) are \((\alpha, \beta)\) expanders.

**Routing Networks**

- Useful for communications (e.g., phones), massively parallel computers, networks, etc.
- \(N\) inputs, \(N\) outputs, want to talk to each other simultaneously
- Signals need to be routed along vertex disjoint paths (can’t overload routers)
- Called “a nonblocking routing network” (exact name varies)
  - Pictures from here on taken from paper by Arora, Leighton, and Maggs
- Want graph with \(O(N \log N)\) verts that can route any permutation
- Also want fast, decentralized algorithm for finding routing

**Butterfly Networks**

- Butterfly network has \(\log(N)\) layers
- \(i\)th layer has blocks of size \(N/2^i\)
- A block splits into “up” and “down” blocks in next layer
- Each node has one edge to “up” block and one edge to “down” block in next layer
- Doesn’t work

**Failure of Butterfly Network**
Finding the Paths

- d-multibutterfly:
  - Each node has d edges out to “up” and d edges to “down”
  - Graph from, e.g. N to N/2 “up” block form an (α, β)-expander, β ≥ d/2, β = 1/2α
- Only 2Nα inputs/outputs
- Actual inputs/outputs connected to α/2 successive multibutterfly inputs/outputs
- Picture of a 2-multibutterfly (hard to draw bigger d)

Proof of Convergence Bound

- **Claim:** Multibutterfly can route any permutation of 2αN inputs to 2αN outputs
- **Proof will use** Halls theorem:
  - **Theorem:** A bipartite graph G has a perfect matching iff every set S of vertices on left has at least |S| neighbors on right
    - i.e., if G is a (1,1)-expander
  - Will prove claim layer by layer
    - Just talk about first layer, others identical
  - At most αN calls need to go to top half, αN to bottom half
    - Let S = set to go to top half, T = top half of next layer
  - Need a perfect matching between S and T
  - |S| ≤ αN, so guaranteed by Hall’s Theorem and expansion

- **Lemma 1:** Guarantee convergence
  - Follows from:
  - **Lemma 2:** For any set S on left of size ≤ αN, number of vertices on right with exactly 1 neighbor in S is at least (2β-d)|S|
  - **Proof:** Blackboard
  - Any node on left can receive at most d acceptances, from which lemma 1 follows
LOCAL AND ALMOST LINEAR-TIME CLUSTERING AND PARTITIONING

The Problem and Motivation
- Graphs these days are getting really really big
  - Circuit layout has 50 million transistors
  - Scientific computing has hundreds of variables
  - The Internet has billions of nodes
  - Amazon, Google, UPS, etc.
- \( n^2 \) is really bad
- Even \( n^{1.5} \) probably isn’t good enough
- Two types of goals
  - Algorithms (say, for finding balanced sparse cuts) that run in almost linear time
    - I.e., linear times polylog
  - Local algorithms

Local Clustering
- Given a vertex \( v \) of a graph and want to know if it is contained in a cluster
  - I.e., there’s a cut of some given conductance that cuts off a set of vertices containing \( v \)
- Want running time to depend on cluster size, not size of graph
  - E.g., find a cluster of web pages around mit.edu
- **Goal**: After running for time almost linear in \( K \), output a cluster of size at least \( K/2 \) around starting vertex, if it exists
  - Will define a cluster to be a set that can be cut off from the rest of the graph with low conductance
  - Will need starting vertex to be well-contained in cluster

General Strategy
- Cuts \( \leftrightarrow \) eigenvalues \( \leftrightarrow \) random walks
- Suppose you start in a cluster and run a random walk
  - Obstacle to mixing is a low conductance cut
  - Means you have trouble leaving the cluster
- So set of vertices that have highest probabilities after a given number of steps are a good guess at a cluster
- Approximate these probabilities and take vertices with \( k \) highest vals as possible cut
  - Like what we did with \( v_2 \), but with probability vector
- Keep trying until you get a good cut (or reach some predetermined limit)
- Use this as a primitive to construct almost linear global algorithm
  - Powerful primitive with many uses
Obstacles

- We need a bound that says this works
  - All of our bounds are global, involving $\lambda_2$ of whole graph
- If we exactly compute all of these probs, will take too long
  - Just too many nonzero entries, so even most rough approximations will take too long
  - But if we don’t exactly compute, need an even stronger bound...
- How good an approx we need depends on cluster size, which we don’t know in advance

The Lovasz-Simonovits Theorem

- Will give us the bound we need for algorithm, but need some notation to define it
- Interesting idea: measure progress of walk, not by one number, but by a whole curve

Preliminaries

- Will study edges more than vertices
- Preprocess graph by replacing each edge by two directed edges, $(u,v)$ and $(v,u)$
- Suppose have prob. dist. $p$ on vertices
  - Let $\rho(u) = p(u)/d_u$
    - Goes to $1/m$ as walk converges
  - Let $\rho(u,v) = \text{probability mass about to be sent over } (u,v)$
    - $\rho(u,v) = \rho(u)/d_u$
      - So only depends on $u$
  - Order edges $e_1, \ldots, e_{2m}$ s.t.
    - $\rho(e_1) \geq \rho(e_2) \geq \ldots \geq \rho(e_{2m})$
From now on, will assume graph has $d_v/2$ self-loops at all verts $v$
- Treat them as normal edges in analysis
- Will define a $y$ coordinate for each $x=1,\ldots,2m$
  - Corresponding to first $i$ edges in sorted order
  - Note: lots of multiplicities in the $\rho(e_i)$ list
- Will then interpolate piecewise linearly
- Curve will measure how close we are to convergence, but contain more info too
- **Definition:** L-S curve given by

\[
I(k) = \sum_{i=1}^{k} \rho(e_i)
\]

- “How much probability mass is transported over $k$ most utilized edges”
  - Should eventually be a straight line
- $I(2m) = 1$
- Slope of $I$ between $k$ and $k+1$ is given by

\[
I(k+1) - I(k) = \rho(e_{k+1})
\]

- Since $\rho$ only depends on start vertex of edge, doesn’t matter how we order edges out of $u$
- Slope is nonincreasing, so curve is concave
- See Matlab
What We’ll Show

- Let $\rho^t$, $I^t$ be $\rho$ and L-S curve at time $t$
- **Claim 1:** For all $x$ and $t$, $I^t(x) \leq I^{t-1}(x)$
- **Claim 2:** For any $c_1, \ldots, c_{2m} \leq 1$
  $$\sum_{i=1}^{2m} c_i \rho(e_i) \leq I \left( \sum_{i=1}^{2m} c_i \right)$$
- **Theorem:** For all $p^0$, $t$, and every $x \in [0, \ldots, m]$,
  $$I^t(x) \leq \frac{1}{2} \left( I^{t-1}(x - 2\phi_G x) + I^{t-1}(x + 2\phi_G x) \right)$$
  For $x \in [m+1, \ldots, n]$,
  $$I^t(x) \leq \frac{1}{2} \left( I^{t-1}(x - 2\phi_G (2m - x)) + I^{t-1}(x + 2\phi_G (2m - x)) \right)$$
- Will just prove for $x \in [0, \ldots, m]$

Proof of Claim 1

- **Claim 2:** For any $c_1, \ldots, c_{2m} \leq 1$
  $$\sum_{i=1}^{2m} c_i \rho(e_i) \leq I \left( \sum_{i=1}^{2m} c_i \right)$$
  - Terms in sum on left are decreasing
  - So only makes sum bigger to add $\gamma$ to $c_i$ and subtract it from $c_j$ for $i<j$
    - I.e., move mass to the left
  - So sum is biggest when first bunch of $c_i$s are 1, next one is the remainder, and rest are 0
  - That gives the RHS

What This Means and Why We Care

- Proof will only use cuts on level sets of $\rho^t$
- So if walk doesn’t mix quickly, know that one of them has bad conductance
- **Theorem:**
  $$I^t(x) \leq \min \left( \sqrt{x}, \sqrt{2m-x} \right) \left( 1 - \frac{1}{2} \phi_G^2 \right)^t + \frac{x}{2m}$$
- **Corollary:** For $W$ a set of verts, $x = \sum_{w \in W} d_w$,
  $$\left| \sum_{w \in W} p^t(w) - \pi(w) \right| \leq \min \left( \sqrt{x}, \sqrt{2m-x} \right) \left( 1 - \frac{1}{2} \phi_G^2 \right)^t$$
- Sketch proof and draw on blackboard

Proof of Claim 2

- **Claim 2:** For any $c_1, \ldots, c_{2m} \leq 1$
  $$\sum_{i=1}^{2m} c_i \rho(e_i) \leq I \left( \sum_{i=1}^{2m} c_i \right)$$
  - Terms in sum on left are decreasing
  - So only makes sum bigger to add $\gamma$ to $c_i$ and subtract it from $c_j$ for $i<j$
    - I.e., move mass to the left
  - So sum is biggest when first bunch of $c_i$s are 1, next one is the remainder, and rest are 0
  - That gives the RHS
Proof of Main Theorem

- From now on, assume \( x \in [0, \ldots, m] \). Want:
  \[
  I^t(x) \leq \frac{1}{2} \left( I^{t-1}(x - 2\Phi_G x) + I^{t-1}(x + 2\Phi_G x) \right)
  \]
  - WLOG let \( x = k \) cut off after all edges for some set \( W=(u_1, \ldots, u_k) \)
    - Call edges \( (u_i, v_i) \), \( i=1, \ldots, k \)
      \[
      \sum_{i=1}^{k} \rho^t(u_i, v_i) = \sum_{i=1}^{k} \rho^{t-1}(v_i, u_i)
      \]
  - Break edges into two sets:
    - \( W_1 = (v_i, u_i) \), \( u_i, v_i \in W \), \( v_i \neq u_i \)
    - \( W_2 = (v_i, u_i) \), \( u_i \in W \), \( v_i \notin W \)
      plus self-loops \( (w, w) \), \( w \in W \)
  - Claim: \( \sum_{(u,v) \in W_1} \rho^{t-1}(v, u) \leq \frac{1}{2} I^{t-1}(x - 2\Phi x) \)

  Will do \( W_1 \)

  \[
  \sum_{(u,v) \in W_2} \rho^{t-1}(v, u) \leq \frac{1}{2} I^{t-1}(x + 2\Phi x)
  \]

- Want \( \sum_{(u,v) \in W_1} \rho^{t-1}(v, u) \leq \frac{1}{2} I^{t-1}(x - 2\Phi x) \)
- Number of edges in \( W_1 \leq x/2 - \Phi x \)
  - \( x/2 \) edges are self-loops
  - At least \( \Phi x \) edges leave \( W \)
- So get easier bound from Claim 2:
  \[
  \sum_{(u,v) \in W_1} \rho^{t-1}(v, u) \leq I^{t-1}(x/2 - \Phi x)
  \]
  - Let \( c_i \) be 1 for \( e \in W_1 \), 0 otherwise
  - \( \sum c_i \leq x/2 - \Phi x \)
- Just need to move \( 1/2 \) outside of \( I^{t-1} \) somehow

\[
\sum_{(v,u)} c(v,u) \leq x/2 - \Phi x
\]

So
\[
\sum_{(v,u) \in W_1} \rho^{t-1}(v, u) = \sum_{(v,u)} c(v,u) \rho^{t-1}(v, u)
= \frac{1}{2} \sum_{(v,u)} 2c(v,u) \rho^{t-1}(v, u)
\leq \frac{1}{2} I^{t-1}(x - 2\Phi x)
\]

\[
\sum_{(v,u) \in W_2} \rho^{t-1}(v, u) \leq \frac{1}{2} I^{t-1}(x + 2\Phi x)
\]

- \( c_i \)'s can be anything \( \leq 1 \) for any \( i \in 1, \ldots, k \) (edges sorted by descending \( \rho \)) with correct sum
- Bound is tighter when more 1’s are at beginning
- We put 1 only on edges in \( W_1 \), no weight on self-loops (equals \( 1/2 \) prob mass)
  - Sequence looks something like 1,0,0,1,1,0,0,1
- Tighten bound by putting \( c_i = 1/2 \) on edges in \( W_1 \) and \( 1/2 \) on self-loops in \( W \)
  - Sequence will now have a lot fewer zeros
  - Can then double everything when applying Claim 2
Using this for Local Clustering

- So if after $O((\log m/\phi)^2)$ steps a set of vertices contains a constant factor more than would under stationary distribution, can get cut $C$ s.t. $\Phi(C) \leq \phi$
  - Use probs to map to real line, and cut like we did with $v_2$

- **Problem:** Computing all of the probabilities will be way too slow
  - Too many nonzero values
  - Need to somehow zero a lot of them out

- **One solution (S-T):** Zero out small ones and prove it doesn’t hurt too much
  - Analysis is pretty messy

- **Instead (ACL):** Use a slightly different vector: PageRank
  - Next lecture

- **Note:** Also need converse
  - Can show that if exists cut $C$ of cond. $\phi^2$, at least $C/2$ of its verts will give cut of cond. $\phi$, or else walk would mix too quickly