18.409: An Algorithmist’s Toolkit
Lecture 8

Jonathan Kelner

Today

- Finish local and almost linear-time clustering and partitioning
- Start sparsification

Administrivia

- Problem set posted
  - Hints posted too
Where We Were Last Time

- Replaced each edge with two directed edges
- Given prob dist. \( p \), defined
  - \( \rho(u) = p(u)/d_u \)
  - \( \rho(u,v) = \rho(u) \)
    - So should all go to \( 1/m \) as walk converges
- Sorted edges s.t. \( \rho(e_1) \geq \rho(e_2) \geq \ldots \geq \rho(e_{2m}) \)
- Defined Lovasz-Simonovits curve by
  \[
  I(k) = \sum_{i=1}^{k} \rho(e_i)
  \]
  (and interpolate other points piecewise linearly)
- Goal was to prove converges to line:
  **Theorem:**
  \[
  I^t(x) \leq \min \left( \sqrt{x}, \sqrt{2m-x} \right) \left( 1 - \frac{1}{2} \phi_G^2 \right)^t + \frac{x}{2m}
  \]

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 \( p^t, I^t \), 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 \( x \in [0, \ldots, m] \)
  - L-S Theorem follows from this by simple calculation

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 \( c_i \)
    - 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 Claim 1

- See blackboard
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 $(u_i, v_i), (u_i, v_k)$
  $$\sum_{i=1}^{k} \rho^t(u_i, v_i) = \sum_{i=1}^{k} \rho^{t-1}(v_i, u_i)$$

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

Want $\sum_{(v,u) \in W_1} c(v, u) \leq x/2 - \phi x$

$c_i$’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 $\frac{1}{2}$ prob mass)

- Sequence looks something like 1,0,0,1,1,0,0,1

Tighten bound by putting $c_i = \frac{1}{2}$ on edges in $W_1$

- Sequence will now have a lot fewer zeros
- Can then double everything when applying Claim 2

So

$$\sum_{(v,u) \in W_1} \rho^{t-1}(v, u) = \sum_{(v,u)} c(v, u) \rho^{t-1}(v, u)$$

$$= \frac{1}{2} \sum_{(v,u)} 2c(v, u) \rho^{t-1}(v, u)$$

$$\leq \frac{1}{2} I^{t-1}(x - 2\phi x)$$
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
- **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 need starting vertex to be well-contained in cluster

Obstacles

- We need a bound that says this works
  - This is why we need the L-S theorem
- 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

General Strategy

- 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
  - Showed this worked with Lovasz-Simonovits Theorem
- Approximate these probabilities and take vertices with k highest vals as possible cut
- Keep trying until you get a good cut (or reach some predetermined limit)
- Use this as a primitive to construct almost linear global algorithm

Corollary of L-S Theorem

- Proof of L-S used cuts on level sets of ρ^t
- So if walk doesn’t mix quickly, know that one of them has bad conductance
- **Corollary:** For W a set of verts, \( x = \sum_{w \in W} d_w, \)

\[
\sum_{w \in W} \rho^t(w) - \pi(w) \leq \min(\sqrt{x}, \sqrt{2m-x}) \left(1 - \frac{1}{2} \phi_W^2\right)^t
\]
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
- 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

PageRank Vectors

- Google uses the directed graph version of these
- We’re going to use the undirected version
- Fix a “starting vertex” \( s \)
- Fix a “teleport probability” \( \alpha \)
- Consider the following process on \( G \), starting at \( s \):
  - **Repeat:**
    - With probability \( (1-\alpha) \), take a step of the lazy random walk on \( G \)
    - With probability \( \alpha \), jump back to \( s \)

More on PageRank

- **Repeat:**
  - With probability \( (1-\alpha) \), take a step of the lazy random walk on \( G \)
  - With probability \( \alpha \), jump back to \( s \)
- Converges to a stationary distribution \( pr_\alpha(s) \)
- Unique solution to:
  \[
  pr_\alpha(s) = \alpha s + (1 - \alpha)Wpr_\alpha(s)
  \]
  (where \( s \) is distrib. that’s 1 on \( s \), 0 elsewhere)
- Could (and will) use other starting distribts just as easily
- Weights shorter paths more than longer ones

L-S for PageRank

- Can show L-S theorem still holds for PageRank vector starting at \( s \)
- So if we knew PageRank vector starting at \( s \), could do same partitioning as with probability vector
  - \( \alpha \) corresponds to number of time steps
- If a set \( S \) contains more than a const factor more prob under \( pr_\alpha(s) \) than under stationary distrib, can find cut with conductance
  \[
  O \left( \sqrt{\alpha \log \left( \sum_{v \in S} d_v \right)} \right)
  \]
- Robust under small errors
- Get partial converse: if exists cut \( C \) of conductance \( \alpha \), at least \( 1/2 \) of verts in \( C \) will give cut of cond. \( O(\sqrt{\alpha}) \)
Approximating PageRank

- Use three properties:
  - \( pr_\alpha(cv + dw) = c \cdot pr_\alpha(v) + d \cdot pr_\alpha(w) \) [linearity]
  - \( Wpr_\alpha(s) = pr_\alpha(Ws) \) [commutes with W]
  - If \( 0 \leq r(v) \leq \varepsilon v \) for all v,
    \[ \frac{[pr_\alpha(s)](S)}{[pr_\alpha(s - r)](S)} \geq \frac{[pr_\alpha(s)](S) - \varepsilon \sum_{v \in S} d_v}{[pr_\alpha(s)](S)} \] [Error bound]
  - Algorithm will maintain two vects, p and r
  - p is approximate answer
  - r is error
  - Will maintain invariant \( p = pr_\alpha(s - r) \)
  - Start with p=0, r=s

Approximating PageRank (cont.)

- Repeat while \( r(u) \geq \varepsilon d(u) \) for some u
  - push(u)
- Moves a lot of prob. each step, so can’t happen too many times
  - Decreases \( ||r||_1 \) by \( \geq \alpha \cdot \varepsilon \cdot d_i \)
  - \( ||r||_1 = 1 \) at time 0
  - So does \( O(1 / (\varepsilon \cdot \alpha)) \) push ops
  - Support of p is \( O(1 / (1 - \alpha)) \)
    - Because at least \( O((1 - \alpha)d_i) \) prob remains in r(v) for any vert. in support
  - This gives us the approx we need, so get local partitioning algorithm
  - To find cut C, need \( \varepsilon = O(1/(\text{total degree of C})) \)
  - Running time proportional to \( (1/\alpha) \cdot (\text{total degree of C}) \)

A Caveat

- In random walk scheme, need to take number of steps like \( 1/\phi \) to get cut of conductance \( \phi^{1/2} \)
  - Actually even a little worse than this because of approximations necessary
  - So running time grows like
    - \( \text{(size of chunk we cut off)} \cdot \text{poly}(1/\phi) \)
- In PageRank scheme, need running time prop. to \( 1/\alpha \) to get cut of size \( \alpha^{1/2} \)
  - So again running time grows like
    - \( \text{(size of chunk we cut off)} \cdot \text{poly}(1/\phi) \)
  - This will make our algorithm run in time
    - (nearly linear) \cdot \text{poly}(1/\phi)
- So only get nearly linear algorithm for \( \phi = 1/\text{polylog}(n) \)
- Getting this to work better is still open
Almost Linear Partitioning

- Suppose $\phi = \text{polylog}(n)$
- Let $\text{vol}(C) = \sum_{v \in C} d_v$
  - Should have done this a few lectures ago...
- If pick random $v$ in a cluster $C$ with conductance $\phi^2$, with prob at least $1/2$, will find set of $\text{vol} \geq \text{vol}(C)/2$
  - If use appropriate $\alpha$ and matching $\epsilon$
    - But you don’t know what’s appropriate
    - Just binary search over possibilities
    - Only multiplies running time by log factor
- So can find globally optimal $\phi$ (up to usual squaring error times some log factors) by cutting off chunks of graph and repeating
  - Running time is almost linear since cut of $C$ in time almost linear in $\text{vol}(C)$

Motivation and What We’ll Cover

- Suppose you have a graph $G$ with $m = \Theta(n^2)$ and want to approximately solve a cut problem (e.g., sparsest cut, min cut, s-t min cut)
- Running time of most algorithms grow with $m$ and are much slower for dense graphs than sparse ones
- So would be really nice if we could somehow throw out a lot of edges and get a similar answer
  - Then running time would be like that of a sparse graph
  - But, of course, you only get an approximate answer
- Idea: randomized sampling
- This isn’t spectral, but later we’ll see how to use spectral techniques to improve it

A First Cut

- Create a new graph $G'$ by sampling every edge of $G$ with probability $p$
- Expected number of edges $= mp$
- Suppose have a cut $V = S \cup \bar{S}$ with $e_g(S) = c$
- Each edge in cut is kept with prob. $p$
- So expected value of $e_{G'} = pc$
- By Chernoff bound,
  $$\Pr \left[ |e_{G'}(S) - pc| \geq \epsilon pc \right] \leq e^{-\epsilon^2 pc/2}$$
- So very likely to get about the right answer for sufficiently large cuts
More on First Cut at Algorithm

Pr \([|e_{G'}(S) - pc| \geq \epsilon pc]\) \leq e^{-c^2 pc/2}

- **Theorem (Karger):** If G has min cut c, number of cuts less than \(\alpha c\) is less than \(n^{\alpha c}\)
- If \(p = \Omega \left( \frac{d \log n}{c^2} \right)\)
  
  cut of size \(\alpha c\) is within \((1+\epsilon)\) factor of expectation with probability \(n^{-O(1/\alpha^2)}\) for whatever constant in the exponent we want

So can choose constants so every cut of size \(\alpha c\) is right

- So can show every cut is within \((1+\epsilon)\) factor of correct value with probability \(1-n^{-d}\)
- If c is small all bets are off
  - A small cut may get badly distorted
  - So need to take very large p

A Slightly More General Chernoff Bound

- We previously had:
  - **Theorem (Chernoff Bound):** Let \(X_1, \ldots, X_n\) be independent \([0,1]\) random variables and \(X = \sum_i X_i\). Then:
    \[
    \Pr[|X - E[X]| \geq \epsilon X] \leq 2e^{-\Theta(1)\epsilon^2 E[X]}
    \]
  - If you look at a proof (or do the homework problem!), can replace \(X_i \in \{0,1\}\) with \(X_i \in [0,1]\) and theorem’s still true
    - Can’t make the \(X_i\)’s too big, or else one can dominate the sum
  - Can scale everything up w/o changing anything:
    - Let \(Y_i = c X_i\), \(Y = \sum Y_i\)
    \[
    \Pr[|Y - E[Y]| \geq \epsilon Y] \leq 2e^{-\Theta(1)\epsilon^2 E[Y]/c}
    \]
    - So just need to know bound c s.t. all \(Y_i \in [0,c]\)

How to Fix the Problem [B-K]

- Suppose graph has small cut c, but edge e is only involved in cuts of size at least \(k\)
  - Will actually need a slightly stronger assumption
- Somehow should only need to sample e as if graph had min cut of size \(k\)
  - Then every edge in a cut of size \(k\) would be sampled at least enough to give small probability of failure
- So maybe we should sample nonuniformly
  - See picture on blackboard
- **Problem:** Expectation isn’t right anymore
- **Solution:** When you do sample an edge, give it a weight of \(1/p\)
  - Importance sampling

Our Case

- Every edge is random variable \(Y_e\)
- e gets a weight \(w_e\)
- If e is in a cut of size c, require \(w_e \leq c\)
- Earlier version:
  - Take all weights =size of min cut
  - Let \(p = \Omega(\log n/\epsilon^2)\)
  - Let \(Y_e = 1\) with prob. \(p/w_e\)
  - Guess that cut size = \(\sum w_e Y_e\)
  - Chernoff bound says get \(\epsilon\)-approx for given cut with prob \(n^{-\Theta(1/w_e)}\) where sum is over edges in cut
  - If have many more edges in cut than min cut, could take bigger \(w_e\)’s and smaller probs and still get this cut right with good prob.
Our Case (cont.)

- So, perhaps, take $w_e = \text{size of min cut containing } e$
- Several problems
  - Won’t cover details of how to fix
  - Will post original paper on web site
  - Will give details of stronger scheme next lecture
- **Problem**: Want to union bound over all cuts to show every cut is right
  - Doesn’t quite work, since have a lot of cuts of large size
- **Also**: Don’t know $w_e$!
- Rough ideas:
  - Can use approx $w_e$‘s instead of exact ones, and can compute these quickly
  - Can use a slightly more conservative weighting scheme

How Many Total Edges Do We Keep?

- Not many small cuts
- Not many edges in each small cut
- So not many edges in small cuts
- In fact, $\sum_i 1/w_i \leq n-1$
  - Depends on $n$ not $m$!
- Rough idea why:
  - Suppose have connected component of min cut $k$
  - Removing $k$ edges cuts it into two pieces
  - Total cost of edges is at most 1
  - Repeat until only have single vertices
  - Add a component each time, so can only do $n-1$ times
- So expect to keep only $\sum_i p/w_i = O(n \log(n)/\varepsilon^2)$ edges

The Result

- When an edge appears, we count it with weight $w_i$
- Size of every cut in $G'$ is $\varepsilon$-approx of size in $G$
- So we’ve produced a weighted graph $G'$ with $O(n \log n/\varepsilon^2)$ edges in which every cut is approx the same as in $G$
- Call this a *combinatorial sparsifier* of $G$

Back to Spectral Graph Theory

- Let $G$ be our original graph, $G'$ our (weighted) combinatorial sparsifier
- We didn’t really do weighted Laplacians, but I claim that everything is similar
  - Think of just having $w_e$ multiple edges
  - Off-diags $= -w_e$
  - $j$th diag $= \sum_{i \neq j} w_{ij}$
- Condition that all cuts are $\varepsilon$-approximated is just:
  $$ (1 - \varepsilon)x^T L_{G'} x \leq x^T L_G x \leq (1 + \varepsilon)x^T L_{G'} x $$
  for all $x = \{-1, 1\}^n$
- Would be even better if works for all $x$
  - By scaling, same as working for all $x \in [-1,1]^n$
- Call this a *spectral sparsifier* of $G$
- Reasonable conjecture: combinatorial sparsifiers are spectral sparsifiers
  - *Reasonable, but false!*
A Counterexample

- Edge \((i, j)\) if \(|i - j| \leq k \mod n\)
- Add edge from 0 to \(n/2\)
- Min cut = 2k
- So removing 1 edge gives combinatorial sparsifier with \(\varepsilon = 1/k\)

Let \(x = (0, 1, \ldots, n/2, n/2-1, \ldots, 1, 0)\)

\[
x^T L_G x = \sum_{(i, j) \in E} (x_i - x_j)^2 = \Theta(nk^3)
\]

\[
x^T L_G x = \sum_{(i, j) \in E} (x_i - x_j)^2 + \left(\frac{n}{2}\right)^2 = \Theta(nk^3) + \frac{n^2}{4}
\]

So need \(\varepsilon = \Omega(n/k^3)\), which can be very big (e.g., for constant \(k\))