CSC 5450 Randomness and Computation

Week 10: Sublinear Algorithms

Plan: ① sublinear space algorithms: distinct elements, frequency moments

② random walk: eigenvalue, conductance, local graph partitioning

---

Sublinear algorithms In some applications where we have a massive data set, the data is too much that we can't afford to store them all or read them once. Yet we can design sublinear space or sublinear time algorithms to do something nontrivial. Randomess is crucial in these settings, as it can be shown that deterministic algorithms cannot guarantee anything nontrivial for most problems.

For sublinear space algorithms, we focus on the data streaming model, in which the algorithm can take one pass on the data but has very limited space for computation. A good motivating example is in computer networks, where routers would like to get good statistics of the traffic but could not afford to store all the data. We will study some basic problems such as estimating the number of distinct elements, the second moment of the data, etc.

For sublinear time algorithms, we have a very large database and we would like to support fast queries that return good estimates of some basic parameters of the database. One well studied area is property testing, and some examples include testing whether a sequence is sorted, a function is linear, a graph is bipartite, etc. There are also some sublinear time algorithms for some combinatorial optimization problems, e.g. minimum spanning tree and maximum matching. See L10 of 2011.

Instead of discussing these usual examples on sublinear time algorithms, today we will study random walk in some more detail. We will discuss the connections between mixing time, the second eigenvalue, and the graph conductance. Then we will mention how it can be used to design a local graph partitioning algorithm, which (sometimes) does not need to read the whole graph.

Distinct Elements [1,2]

The input is a stream of a sequence $a_1, a_2, \ldots, a_n$, where each a_i is chosen from $\{1,2,\ldots,m\}$. Our algorithm is allowed to take one pass of the stream, using very limited space (e.g. $O(\log(m+n))$), and return a good estimate on the number of distinct elements in the data stream.

We will give bounds in terms of $\log(m)$ instead of $\log(n)$. It is good in the case when $n \gg m$.

We will give bounds in terms of $\log(m)$ instead of $\log(n)$. It is good in the case when $n >> m$.

In the case when $m > n$, we can first hash each item to a table of size n^2 or larger, then with high probability there will be no collisions, and thus we can assume $\log m = O(\log n)$.

Since we use so little space, we cannot hope to compute the number of distinct elements exactly. What we could hope for is a $(1\pm\varepsilon)$-approximation with high probability, i.e. with prob $1-\delta$ for $\delta > 0$.

Let us first start with a simpler problem: given a $T > 0$, provide an algorithm with probability $1-\delta$,

- answers YES if # distinct elements $> (1+\varepsilon)T$
- answers NO if # distinct elements $< (1-\varepsilon)T$

(either answer is fine if the true answer is close to T, i.e. $(1-\varepsilon)T <$ # distinct elements $< (1+\varepsilon)T$.)

In the following let D = # distinct elements.

What would you do to solve this problem?

You'll expect to see random sampling and Chernoff bounds show up a lot, and also some hashing techniques we learned in last week.

The algorithm is very simple. Choose a random set $S \subseteq \{1, 2, \dots, m\}$ by adding each i to S with probability $1/T$. The algorithm will say <u>YES</u> if some values in S appear in the data stream; otherwise it says <u>NO</u> if the values in S never appear in the data stream.

<u>Analysis</u>: $\Pr(\text{no values in } S \text{ appear}) = (1-\frac{1}{T})^D$

We can assume that T is large enough so that $(1-\frac{1}{T})^D \approx e^{-\frac{D}{T}}$; otherwise if T is small we can store a list of T distinct elements appear in the data stream and solve the problem exactly.

If $D > (1+\varepsilon)T$, then the above probability $\approx e^{-(1+\varepsilon)} < \frac{1}{e(1+\varepsilon)} < \frac{1}{e}(1-\frac{\varepsilon}{1.1}) < \frac{1}{e} - \frac{\varepsilon}{3}$

(Taylor expansion; for small enough ε)

If $D < (1-\varepsilon)T$, then the above probability $\approx e^{-(1-\varepsilon)} > \frac{(1+\varepsilon)}{e} > \frac{1}{e} + \frac{\varepsilon}{3}$.

So there is a constant gap in the probability of saying NO in the above two cases.

Now, if we take the above experiments many times, i.e. generate $S_1, \dots, S_k$ and count how many times of "NO", then we expect to have an accurate answer.

Let Z be the number of times the algorithm says NO, among the k trials.

We would like to set k just large enough so that with prob $1-\delta$:

- if $D > (1+\varepsilon)T$, then $Z < k/e$ (say YES)
- if $D < (1-\varepsilon)T$, then $Z > k/e$ (say NO)

- if $D < (1-\varepsilon)T$, then $Z > k/e$ (say No)

Then we can just use the value of Z to determine the output (YES/No) of the algorithm.

We can find a suitable k by using Chernoff bound.

If $D > (1+\varepsilon)T$, then $E[Z] < k\left(\frac{1}{e} - \frac{\varepsilon}{3}\right) = \frac{k}{e} - \frac{k\varepsilon}{3}$.

So, $\Pr(Z \geq k/e \mid D > (1+\varepsilon)T) \leq \Pr\left(|Z - E[Z]| > \frac{\varepsilon e}{3} E[Z]\right) \leq 2e^{-E[Z]\left(\frac{\varepsilon e}{3}\right)^2/3}$

By setting $k = O(\varepsilon^{-2} \log \delta^{-1})$, we can make this probability less than δ.

Similarly for the case when $D < (1-\varepsilon)T$.

So, we just need to use $O(\varepsilon^{-2} \log \delta^{-1})$ bits to solve the problem for a given T.

If T is not given, we can try $T = 1, 1+\varepsilon, (1+\varepsilon)^2, \ldots, m$ in parallel, and this multiplies the space used by a factor of $\approx \log m/\varepsilon$, so the total space used is $O(\varepsilon^{-3} \log \delta^{-1} \log m)$.

Are we done? Wait, how do we store $S_1, \ldots, S_k$? The space required to store them could be bigger than storing all the input! This is like the situation we were in hashing.

One idea is to use hash functions to solve the problem, e.g. having a table of size T and check whether say the first entry is empty or not.

But the analysis here used a strong property of a random function, namely $\Pr(\text{first bin is empty}) = \left(1 - \frac{1}{T}\right)^D$. No efficient hash family is known to have this property.

Anyway, this gives some intuition and a heuristic algorithm for the problem.

Distinct Elements (Take 2) [2]

We show how to use a 2-universal family to solve the problem.

The idea is to map the universe $\{1, 2, \ldots, m\}$ to $\{1, 2, \ldots, m^3\}$

As we have seen last week, there will be no collisions with high probability.

Ideally, we want the hash values $h(1), h(2), \ldots, h(m)$ to be evenly distributed in $\{1, 2, \ldots, m^3\}$.

Then, if there are D distinct elements, the t-th smallest hash value should be close to $t \cdot \frac{m^3}{D}$.

Therefore, if we know the t-th smallest non-empty bin is T, then $T \approx t \cdot \frac{m^3}{D}$ and thus $D \approx \frac{t \cdot m^3}{T}$.

We must be careful about the space requirement. First of all, we are not going to store the whole hash table, as it is really unnecessary.

We only need to store t numbers, corresponding to the t smallest non-empty bins.

Storing more numbers will help get a better estimate, but of course it requires more space.

We will find a good t to balance both the error probability and the space requirement.

We will find a good t to balance both the error probability and the space requirement.

Let's state the algorithm again.

- Pick a random hash function h from a 2-universal family.
- For each item a_i in the data stream, compute $h(a_i)$. Maintain the list of the t smallest hash values seen so far.
- Let T be the t-th smallest hash value after all the data read. Return $Y = \frac{tm^3}{T}$.

Claim: By setting $t = O(\frac{1}{\varepsilon^2})$, we have $(1-\varepsilon)D \le Y \le (1+\varepsilon)D$ with constant probability.
(This proof is similar to finding median in week 2.)

Case 1: We first bound the probability that $Y > (1+\varepsilon)D$.
This implies that $T < \frac{tm^3}{(1+\varepsilon)D}$, which is at most $\frac{tm^3(1-\frac{\varepsilon}{2})}{D}$ for small enough ε.
This means that there are at least t hash values that are smaller than $\frac{tm^3(1-\frac{\varepsilon}{2})}{D}$.

On the other hand, let X_i be the indicator variable that $h(a_i) \le \frac{tm^3(1-\frac{\varepsilon}{2})}{D}$.
Then $E[X_i] = \Pr(h(a_i) \le \frac{tm^3(1-\frac{\varepsilon}{2})}{D}) \le \frac{t(1-\frac{\varepsilon}{2})}{D}$ since $h(a_i)$ is equally likely to be in $\{1, \dots, m^3\}$.
Therefore, if there are D distinct elements, $E[\#\text{ of elements with hash values} \le \frac{tm^3(1-\frac{\varepsilon}{2})}{D}] \le t(1-\frac{\varepsilon}{2})$.

To bound the derivation, we compute the variance. Let $X = \sum_{i=1}^{D} X_i$. Then $E[X] \le t(1-\frac{\varepsilon}{2})$.
Since X_i are pairwise independent, $\mathrm{Var}[X] = \sum_{i=1}^{D} \mathrm{Var}[X_i] \le t(1-\frac{\varepsilon}{2})$.
By Chebyshev's inequality, $\Pr(X > t) = \Pr(X > t(1-\frac{\varepsilon}{2}) + \frac{\varepsilon}{2}\cdot t) \le \Pr(|X - E[X]| \ge \frac{\varepsilon t}{2})$

$$\le \frac{4\,\mathrm{Var}[X]}{\varepsilon^2 t^2} \le \frac{4(1-\frac{\varepsilon}{2})}{\varepsilon^2 t} \le \frac{4}{\varepsilon^2 t} \le \frac{1}{6} \quad \text{for } t = O(\tfrac{1}{\varepsilon^2})$$

Case 2: We bound the probability that $Y < (1-\varepsilon)D$. This is very similar to case 1.
The condition implies that there are less than t distinct elements with hash values $\le \frac{(1+\varepsilon)tm^3}{D}$
By a very similar calculation, $E[X] \ge \frac{(1+\frac{\varepsilon}{2})t}{D}$ and $\mathrm{Var}[X] \le \frac{(1+\frac{\varepsilon}{2})t}{D}$.
By Chebyshev's inequality, $\Pr(X < t) \le \Pr(|X - E[X]| \ge \frac{t\varepsilon}{2}) \le \frac{4\,\mathrm{Var}[X]}{\varepsilon^2 t^2} < \frac{1}{6}$ for $t = O(\frac{1}{\varepsilon^2})$.

Therefore, with probability at most $1/3$, $(1-\varepsilon)D \le Y \le (1+\varepsilon)D$.
By running $O(\log \frac{1}{\delta})$ copies and taking the median, the error probability is at most δ.

The total space used is $O(\frac{1}{\varepsilon^2} \log m \log(\frac{1}{\delta}))$, since each copy stores $O(\frac{1}{\varepsilon^2})$ numbers each of $O(\log m)$ bits, and the hash function only requires $O(\log m)$ bits to store.

Note that the list can be implemented by balanced binary search tree or heap to support fast update.
Since there are at most $O(\frac{1}{\varepsilon^2})$ elements, each operation can be supported in $O(\log \frac{1}{\varepsilon})$ steps.

Since there are at most $O(\frac{1}{\epsilon^2})$ elements, each operation can be supported in $O(\log \frac{1}{\epsilon})$ steps.

Recently there is an optimal algorithm that uses only $O(\frac{1}{\epsilon^2} + \log m)$ space and $O(1)$ update time.
This is best possible ; see [3].

Frequency Moments [4,5]

Again, let $a_1, a_2, \ldots, a_n$ be the items in the data stream, where each a_i is from $\{1, 2, \ldots, m\}$.
For $1 \le i \le m$, let x_i be the number of items equal to i.
We would like to estimate $\sum_{i=1}^{m} x_i^p$ where p is a given constant.
Note that when $p=0$ this is just the distinct element problem, when $p=1$ it is trivial.
Here we focus on the problem when $p=2$, and later mention results for other p.

The algorithm is very simple and elegant. This is an important result in data streaming algorithms.
- Set $r_1, r_2, \ldots, r_m$ independently to be $+1$ with prob $1/2$ and -1 with prob $1/2$.
- Maintain $Z = \sum_{i=1}^{m} r_i x_i$ (this can be done by just storing the current value of Z, right?)
- Return $Y = Z^2$

We will prove that Y is a good approximation to $\sum_{i=1}^{m} x_i^2$ with high probability.

First we show that $E[Y] = \sum_{i=1}^{m} x_i^2$.
Since $Y = Z^2 = \left(\sum_{i=1}^{m} r_i x_i\right)^2$, $E[Y] = E[Z^2] = \sum_{1 \le i,j \le m} E[r_i x_i r_j x_j] = \sum_{1 \le i,j \le m} x_i x_j E[r_i r_j]$.
When $i=j$, $E[r_i r_j] = 1$; when $i \ne j$, $E[r_i r_j] = E[r_i]E[r_j] = 0$.
Therefore, $E[Y] = E[Z^2] = \sum_{i=1}^{m} x_i^2$.

To show that it is a good approximation with high probability, we need to show that Z^2 is concentrated around its expected value, for which we'll use the Chebyshev's inequality.
So we compute $E[Y^2] = E[Z^4] = \sum_{1 \le i,j,k,l \le m} x_i x_j x_k x_l E[r_i r_j r_k r_l]$.
Note that $E[r_i r_j r_k r_l] = 0$ whenever one index appears only once.
Therefore there are only two situations when $E[r_i r_j r_k r_l] \ne 0$:
- when $i=j=k=l$, and this contributes $\sum_{i=1}^{m} x_i^4$ to $E[Y^2]$.
- when two indexes appear twice, and this contributes $6 \sum_{i<j} x_i^2 x_j^2$ to $E[Y^2]$.

So $E[Y^2] = E[Z^4] = \sum_{i=1}^{m} x_i^4 + 6 \sum_{i<j} x_i^2 x_j^2$.
Thus $Var[Y] = E[Y^2] - (E[Y])^2 = E[Z^4] - (E[Z^2])^2$

$$= \sum_{i=1}^{m} x_i^4 + 6 \sum_{i<j} x_i^2 x_j^2 - \left(\sum_{i=1}^{m} x_i^2\right)^2$$

$$= \sum_{i=1}^{m} x_i^4 + 6\sum_{i<j} x_i^2 x_j^2 - \left(\sum_{i=1}^{m} x_i^4 + 2\sum_{i<j} x_i^2 x_j^2\right)$$

$$= 4\sum_{i<j} x_i^2 x_j^2 \leq 2\left(\sum_{i=1}^{m} x_i^2\right)^2 = 2\left(E[Y]\right)^2$$

By Chebyshev's inequality, $\Pr(|Y - E[Y]| \geq c\sqrt{\mathrm{var}[Y]}) \leq \frac{1}{c^2}$.

This already implies that $\Pr(|Y - E[Y]| \geq c\sqrt{2}\,E[Y]) \leq \frac{1}{c^2}$, getting a constant factor approximation.

To get a $(1\pm\varepsilon)$-approximation, we need to find a variable Y' with $E[Y] = E[Y']$ but the variance of Y' is smaller. The standard way to do this is to compute $Y_1, Y_2, \ldots, Y_k$ and take the average.

Let $Y' = \left(\sum_{i=1}^{k} Y_i\right)/k$. Then $E[Y'] = E[Y]$ and $\mathrm{Var}[Y'] = \sum_{i=1}^{k} \mathrm{Var}[Y_i/k] = \frac{1}{k}\mathrm{Var}[Y] \leq \frac{2}{k}\left(E[Y]\right)^2$

Therefore, $\Pr(|Y' - E[Y']| \geq c\sqrt{2/k}\,E[Y']) \leq 1/c^2$.

Setting c to be a constant and $k = O(1/\varepsilon^2)$ will get an $(1\pm\varepsilon)$-approximation with constant probability.

The space required to compute one Y is to store one number Z. So the total space required is $O(1/\varepsilon^2)$ numbers.

Wait, how do we compute Z if we don't store $r_1, r_2, \ldots, r_m$? This requires m bits, although it is not as bad as storing all the data, it is still undesirable if the universe is large (e.g. IP address).

A key point about this approach is that we only the variables to be 4-wise independent, so that the analysis about $E[r_i r_j r_k r_\ell]$ would hold.

m 4-wise independant bits can be generated from $O(\log m)$ random bits. So the extra storage required for the computation is only $O(\log m)$.

This approach is called sketching, which is very useful in data streaming algorithms. It is closely related to the idea of dimension reduction.

What about $\sum_{i=1}^{m} x_i^3$? It turns out that $\Theta(n^{1-\frac{2}{p}}\,\mathrm{polylog}(n))$ space is required to estimate $\sum_{i=1}^{m} x_i^p$ for any $p > 2$. On the other hand, for $0 \leq p \leq 2$ (p can be fractional), then $O(\mathrm{polylog}(m))$ space is enough.

Random Walk and Eigenvalues

We will discuss in more detail about what we mentioned at the end of week 7 about random walk.

In the following, we assume the graph is undirected and d-regular.

Recall that if the graph is bipartite, then a random walk may not converge to its stationary distribution.

To avoid this problem, we consider lazy random walk, where we stay at the same vertex with

probability $1/2$. This won't change the mixing time much, and makes the random walk aperiodic.

So, let $W = \frac{1}{2}I + \frac{1}{2d}A$ be the transition matrix of the random walk, where A is the adjacency matrix.

Since W is a symmetric matrix (as the graph is undirected), it is known that all eigenvalues of W are real.

Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the eigenvalues of W. It can be shown that $1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n \geq 0$ if G is connected. Also, it is known that there is an orthonormal basis of eigenvectors $v_1, v_2, \ldots, v_n$ (i.e. $\langle v_i, v_j \rangle = 0$ if $i \neq j$, $\langle v_i, v_i \rangle = 1$ and any vector $x \in \mathbb{R}^n$ can be written as a linear combination of $v_1, \ldots, v_n$).

Let p_i be the probability distribution after i steps of random walk. Then $p_t = W^t p_0$.

Write $p_0 = c_1 v_1 + \ldots + c_n v_n$ as a linear combination of the eigenvectors.

Then $p_t = W^t p_0 = c_1 \lambda_1^t v_1 + \ldots + c_n \lambda_n^t v_n$, because v_i is an eigenvector with eigenvalue λ_i.

When $t \to \infty$, since $1 > \lambda_2 \geq \ldots \geq \lambda_n \geq 0$, $\lambda_i^t \to 0$ for $2 \leq i \leq n$, and thus $p_t \to c_1 v_1$.

Note that $v_1 = (\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}})$ and $c_1 = \langle p_0, v_1 \rangle = \frac{1}{\sqrt{n}}$, so we have $c_1 v_1 = (\frac{1}{n}, \ldots, \frac{1}{n})$, the stationary distribution when the graph is regular.

This proves that the random walk will converge to the stationary distribution.

Eigenvalue Gap

Now, suppose $\lambda_2 \leq 1 - \varepsilon$ for some constant $\varepsilon > 0$. Let π be the stationary distribution.

Then $\| p_t - \pi \| = \| \sum_{i \geq 2} c_i \lambda_i^t v_i \| \leq \sum_{i \geq 2} \lambda_i^t c_i \| v_i \| \leq n(1-\varepsilon)^t$.

So, when $t = \Omega(\log n)$, then $\| p_t - \pi \|$ is very small.

This proves that when there is a large gap between the first and the second eigenvalue, then the mixing time is only $\Theta(\log n)$.

Cheeger's Inequality

When is there a large gap between the first and the second eigenvalue?

Cheeger's inequality shows that it happens if and only if the graph is an expander.

Again, assume the graph is d-regular.

Define the conductance of a set S as $\phi(S) = |\delta(S)| / (d \cdot |S|)$, and $\phi(G) = \min_{S : |S| \leq n/2} \phi(S)$.

Then $0 \leq \phi(G) \leq 1$, and $\phi(G)$ is large (bounded away from zero) if and only if G is an expander.

Cheeger's inequality: $\frac{1}{2}(1 - \lambda_2) \leq \phi(G) \leq \sqrt{2(1-\lambda_2)}$

So, λ_2 is bounded away from one iff $\phi(G)$ is bounded away from zero.

Roughly speaking, every random walk has a small mixing time iff the graph is an expander.

Roughly speaking, every random walk has a small mixing time iff the graph is an expander.

So, to prove upper bound on mixing time, one can prove a lower bound on the graph conductance.

Graph Partitioning

Finding a set of small conductance, called a <u>sparse cut</u>, is an important algorithmic problem that is useful in different areas of computer science, e.g. image segmentation, clustering, community detection in social networks.

The proof of Cheeger's inequality actually provides an efficient algorithm to find a sparse cut with conductance $O(\sqrt{1-\lambda_2})$.

We won't prove Cheeger's inequality today.

Instead, we will prove a direct connection between conductance and mixing time, and this would imply a graph partitioning algorithm with similar guarantee as Cheeger's inequality

One advantage of this approach is that it gives a <u>local</u> graph partitioning algorithm that (sometimes) does not need to read the whole graph.

This is useful in applications when we have a massive graph, e.g. the social network.

[**Notice**: We provide more technical details in the notes, while in the lecture we only give the main ideas. We won't ask technical question about the following in homeworks.]

From now on, we work again on general undirected graphs, not necessarily regular.

Lovász Simonovits Curve [6]

In the stationary distribution, the probability at a vertex v is $d(v)/2m$, and so the probability at an edge is $1/m$. It is uniformly distributed on the edges.

For the analysis, it is more convenient to think of the graph as a directed graph, where each undirected edge is replaced by two opposite directed edges (i.e. replace uv by $u \to v$ and $v \to u$).

Since we consider lazy random walk, we also add one self-loop for each outgoing edge.

So there are totally $4m$ directed edges in the graph, and for each vertex v it has $d(v)$ outgoing edges, $d(v)$ incoming edges, and $d(v)$ self-loops.

Let p_t be the probability distribution of the lazy random walk after t steps.

We let q_t denote the induced distribution on the directed edges, i.e. $q_t(u,v) = p_t(u)/2d(u)$.

We consider the cumulative distribution function of q_t, which we call C^t.

We define $C^t(k)$ to be the sum of the largest k values of q_t.

Let $V=\{1,2,\dots,n\}$.

If we order the vertices so that $p_t(1)/2d(1) \geq p_t(2)/2d(2) \geq \dots \geq p_t(n)/2d(n)$, then

$C^t(2d(1)) = p_t(1)$, $C^t(2d(1)+2d(2)) = p_t(1)+p_t(2)$, and $C^t(2d(1)+\dots+2d(k)) = p_t(1)+\dots+p_t(k)$.

The function will be piecewise linear between these points. We call these points the extreme points.

Let $x_k^t = 2d(1)+\dots+2d(k)$ be the location of the k-th extreme point.

Note that we need to re-order the vertices for each t.

Here are some basic properties of the curve C^t.

Lemma Extend $C^t(x)$ to all real $x\in[0,4m]$ by making it piecewise linear between integral points.

Then ① the function $C^t(x)$ is concave.

② for every x, every s with $x\pm s \subset [0,4m]$ and every $r<s$, we have

$$\frac{1}{2}\left(C^t(x+s)+C^t(x-s)\right) \leq \frac{1}{2}\left(C^t(x+r)+C^t(x-r)\right).$$

③ for every subset F of directed edges with no self-loops, we have $q_t(F) \leq \frac{1}{2}C^t(2|F|)$.

Proof We prove ②, and note that ① follows immediately by setting $r=0$.

② is equivalent to $C^t(x+s)-C^t(x+r) \leq C^t(x-r)-C^t(x-s)$, which follows from the definition as the r-s edges appear later have probabilities at most the r-s edges earlier, as we order the edges in non-increasing probabilities.

③ is also easy as the self-loops corresponding to F have the same probabilities, and so $C^t(2|F|) \geq 2q_t(F)$. □

Curve to Line

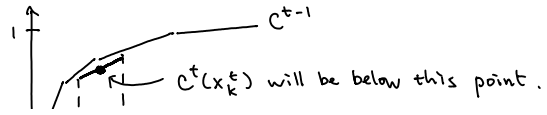
In the stationary distribution every edge has the same probability, and so the curve becomes a straight line.

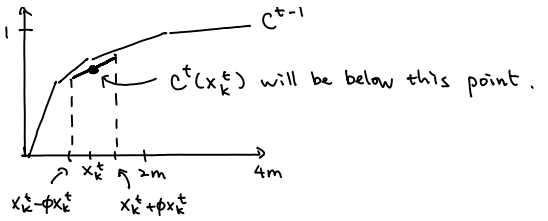
The following theorem says that the curve converges to the straight line faster if the conductance is larger.

Theorem (Lovász-Simonovits) Let G be a graph with conductance at least ϕ. Then, for every initial probability distribution p_0, every $t\geq 0$ and every x,

- if $x \leq 2m$, then $C^t(x_k^t) \leq \frac{1}{2}\left(C^{t-1}(x_k^t-\phi x_k^t)+C^{t-1}(x_k^t+\phi x_k^t)\right)$.
- if $x \geq 2m$, then $C^t(x_k^t) \leq \frac{1}{2}\left(C^{t-1}(x_k^t-\phi(4m-x_k^t))+C^{t-1}(x_k^t+\phi(4m-x_k^t))\right)$.

Pictorically, the theorem shows that





Intuitively, if the "chord" drawn is longer, then the new point will be lower, and thus the curve will drop faster.

When $x \le 2m$, the length of the chord is proportional to the conductance and x; when $x \ge 2m$ the length is proportional to ϕ and $4m - x$, that is, the smaller side of the cut.

<u>Proof</u> We will just prove the theorem for an extreme point x, and the rest follows from concavity.

Observe that $C^t(x_k^t)$ is the sum of the probabilities of the k largest vertices.

Let this set of k vertices be S.

Let S^{self} be all the self-loops attached to S. For those edges that are not self-loops, let S^{out} be the set of edges with their tails in S, and let S^{in} be the set of edges with their heads in S. Note that S^{in} and S^{out} can overlap; their intersection is the set of edges with both endpoints in S.

Note that the sum of the probabilities in S at time t is equal to the sum of the probabilities of the edges coming in S at time $t-1$.

That is, $p_t(S) = q_{t-1}(S^{self}) + q_{t-1}(S^{in})$.

As the probability at each self-loop is equal to its corresponding outgoing edge, we have

$$q_{t-1}(S^{self}) = q_{t-1}(S^{out}).$$

So, $p_t(S) = q_{t-1}(S^{out}) + q_{t-1}(S^{in}) = q_{t-1}(S^{out} \cap S^{in}) + q_{t-1}(S^{out} \cup S^{in})$.

Observe that $|S^{in} \cap S^{out}| = \text{vol}(S) - |\delta(S)|$, and $|S^{in} \cup S^{out}| = \text{vol}(S) + |\delta(S)|$.

So, by ③ of the previous lemma, we have

$$p_t(S) \le \frac{1}{2}\left(C^{t-1}(2\text{vol}(S) + 2|\delta(S)|) + C^{t-1}(2\text{vol}(S) - 2|\delta(S)|)\right),$$

where $\text{vol}(S)$ is defined as $\sum_{v \in S} d(v)$.

Assume $x \le 2m$. The other case is similar.

Then $|\delta(S)| \ge \phi\,\text{vol}(S)$. So, by ② of the previous lemma,

$$p_t(S) \le \frac{1}{2}\left(C^{t-1}(2(\text{vol}(S) + \phi\,\text{vol}(S))) + C^{t-1}(2(\text{vol}(S) - \phi\,\text{vol}(S)))\right).$$

Since $x_k^t = 2\text{vol}(S)$, this implies $p_t(S) \le \frac{1}{2}\left(C^{t-1}(x_k^t + \phi x_k^t) + C^{t-1}(x_k^t - \phi x_k^t)\right)$. □

Concrete Bound

We prove that the curve C^t is always below the curve U^t, defined by

$$U^t(x) = x/4m + \min(\sqrt{x}, \sqrt{4m-x})(1-\tfrac{1}{8}\phi^2)^t.$$

It is clear that $C^0 \le U^0$.

The proof will follow by induction once we show that

- for every $x \in (0, 2m]$, $\frac{1}{2}(U^{t-1}(x-\phi x) + U^{t-1}(x+\phi x)) \le U^t(x)$
- for every $x \in [2m, 4m)$, $\frac{1}{2}(U^{t-1}(x-\phi(4m-x)) + U^{t-1}(x+\phi(4m-x))) \le U^t(x)$,

because e.g. $C^t(x) \le \frac{1}{2}(C^{t-1}(x-\phi x) + C^{t-1}(x+\phi x)) \le \frac{1}{2}(U^{t-1}(x-\phi x) + U^{t-1}(x+\phi x)) \le U^t(x)$

For $x \in (0, 2m]$, $\frac{1}{2}(U^{t-1}(x-\phi x) + U^{t-1}(x+\phi x)) \le \frac{1}{2}(\sqrt{x-\phi x} + \sqrt{x+\phi x})(1-\frac{1}{8}\phi^2)^{t-1} = \frac{1}{2}\sqrt{x}(\sqrt{1-\phi} + \sqrt{1+\phi})(1-\frac{1}{8}\phi^2)^{t-1}$

Note that by Taylor series for $\sqrt{1+\phi}$ at $\phi = 0$, we have $\sqrt{1+\phi} = 1 + \frac{1}{2}\phi - \frac{1}{8}\phi^2 + \frac{1}{16}\phi^3 \le 1 + \frac{1}{2}\phi - \frac{1}{8}\phi^2$,

and so $\frac{1}{2}(U^{t-1}(x-\phi x) + U^{t-1}(x+\phi x)) \le \frac{1}{2}\sqrt{x}\left(1 - \frac{1}{2}\phi - \frac{1}{8}\phi^2 + 1 + \frac{1}{2}\phi - \frac{1}{8}\phi^2\right)(1-\frac{1}{8}\phi^2)^{t-1} = \sqrt{x}(1-\frac{1}{8}\phi^2)^t = U^t(x)$.

By subtracting C^t from U^t, we have the following bound on the convergence rate of the random walk.

Theorem $p_t(S) - \pi(S) \le \sqrt{\mathrm{vol}(S)}(1-\frac{1}{8}\phi^2)^t$.

So, if ϕ is a constant, then after $\Theta(\log n)$ steps, then p_t is very close to the stationary distribution.

Sparse Cut by Random Walk [7]

We show how to apply the Lovász-Simonovits theorem to find a sparse cut.

Suppose S is a set of conductance ϕ.

Then, it is not difficult to show that there is a vertex v in S such that if the random walk starts at v, then $p_t(S) \ge 1 - t\phi$. That is, at each step $\le \phi$ probability mass leaves S.

Let $\pi(S) = \sum_{v \in S} \pi(v) = \mu \le \frac{1}{2}$.

Let's run the random walk at vertex v for $t = \frac{\mu}{2\phi}$ steps.

Then $p_t(S) \ge 1 - (\frac{\mu}{2\phi})\phi \ge 1 - \frac{\mu}{2} \ge 3/4$.

Notice that in the proof of the Lovász-Simonovtis theorem, we only use the fact that the conductance of S_k^i is at least θ, where S_k^i denotes the set of k highest probability vertices at time step i.

So, if $\min \phi(S_k^i) = \theta$ for $1 \le i \le \frac{\mu}{2\phi}$ and for all k, then by the theorem we have

$$1/4 \le p_t(S) - \pi(S) \le \sqrt{\mathrm{vol}(S)}(1-\tfrac{1}{8}\theta^2)^t \le \sqrt{2m}\, e^{-\frac{t\theta^2}{8}},$$

and this implies that $e^{t\theta^2/8} \le 4\sqrt{2m}$, hence $\theta \le \sqrt{8\ln(4\sqrt{2m})/t} = 4\sqrt{\ln(4\sqrt{2m}) \cdot \frac{\phi}{\mu}}$.

and this implies that $e^{t\theta^2/8} \le 4\sqrt{2m}$, hence $\theta \le \sqrt{8\ln(4\sqrt{2m})/t} = 4\sqrt{\ln(4\sqrt{2m})\cdot\frac{\phi}{\mu}}$.

It means that one of the sets in S_k^i must have conductance at most $O(\sqrt{\phi}\sqrt{\ln m})$.

We can check all sets S_k^i for $i \le \mu/2\phi$ in polynomial time, thereby obtaining an approximation algorithm for sparse cut almost as good as the spectral partitioning algorithm.

Local Graph Partitioning Algorithm [8]

It is possible to keep exploring a small part of the graph by <u>truncated</u> random walk.

That is, whenever the probability at a vertex v is at most $\varepsilon \cdot d(v)$, then we set the probability to be zero.

Note that the number of vertices with positive probability is at most $1/\varepsilon$.

By choosing an appropriate ε, we can keep the set small and the error is small.

Moreover, we can have some control over the output size using this approach.

The following is a more precise statement.

<u>Theorem</u> For an undirected graph $G = (V, E)$ and a set $U \subseteq V$, given $\phi \ge \phi(U)$ and $k \ge \mathrm{vol}(U)$, there exists initial vertices such that the truncated random walk algorithm can find a set S with $\phi(S) \le O(\sqrt{\phi/\varepsilon})$ and $\mathrm{vol}(S) \le O(k^{1+\varepsilon})$ for any $\varepsilon > 2/k$. The runtime of the algorithm is $\tilde{O}(k^{1+2\varepsilon}\phi^{-2})$.

References

[1] Piotr Indyk. Notes of 6.895, MIT.

[2] Chen, Sunkavally, Woodruff. Space efficient algorithms for distinct elements in a data stream

[3] Kane, Nelson, Woodruff. An optimal algorithm for the distinct elements problem.

[4] Piotr Indyk. Notes of 6.895, L2 norm estimation, MIT.

[5] Alon, Matias, Szegedy. The space complexity of approximating the frequency moments.

[6] Lovász, Simonovits. The mixing time of Markov chains, an isoperimetic inequality, and computing the volume.

[7] Spielman, Teng. A local clustering algorithm for massive graphs and its applications to nearly linear time graph partitioning.

[8] Kwok, Lau. Finding small sparse cut by random walk.