Graphs and Networks Lecture 8

Random Walks and Diffusion

Daniel A. Spielman September 24, 2013

8.1 Disclaimer

These notes are not necessarily an accurate representation of what happened in class. They are a combination of what I intended to say with what I think I said. They have not been carefully edited.

8.2 Overview

I'll begin lecture with some explanation of the behavior of percolation that we saw last lecture.

We then introduce random walks and diffusion on undirected graphs. We will study random walks on directed graphs in a later lecture. These are very powerful, and are the basis for the famous PageRank algorithm, but are technically more complicated.

Random walks and diffusion are among the most-studied and best-behaved processes on graphs. They will play a crucial role in many of the analyses of graphs that we will perform in the last segment of the course. We will understand many other processes by comparison with these.

8.3 More on Percolation

I'd like to take a moment to follow up on our observations about percolation in real-world graphs from last lecture. First, I should say that the phenomenon that we observed–that these graphs did not evidence a sharp threshold–seems to be pretty well understood. Newman (the book) has an analysis of situtations like this under the configuration model, which I'll explain in a moment, but under node rather than bond percolation. That is, he considers removing vertices instead of edges.

Much of what we observed can be explained by the degree distributions. For example, the largest connected component of the protein-protein interaction network "Bg Homo sapiens" has 17796 nodes, and one node of degree 9704. That high-degree node alone can explain the curve for that graph. If we keep edges with probability p, than we expect $p(9704/17796)$ of the neighbors of that node to be connected. This is necessarily smooth with p until p is very small. Similarly, the largest connected component in "Bg S cerevisiae" has 6548 nodes and a vertex of degree 2873. While not all of the graphs have such dominant nodes, you can imagine that a few big ones can produce a similar effect once they land in the same component. The graph "web-Google" has around 900 thousand nodes. The largest has degree 6332. But, there are 135 nodes with degrees larger than 1000.

Another way of asking whether the percolation curves are a result of the structure of the graphs or merely their degrees is to ask how a random graph with the same degrees would behave. That is, we could fix the degrees in a graph, and then generate a random graph that has these degrees. The most natural way to do this is called the "configuration model". In this model, we view each edge as having two endpoints: one for each vertex. And, we view a vertex of degree d has having d *sockets* to which edges can be attached. Fixing the degrees of every vertex fixes its number of sockets. We would now like to place edges between sockets at random. The one difficulty in doing this is that if we choose a random matching between sockets then some edges will attach to the same vertex, forming self-loops, or will double-up, forming multi-edges. We can fix this by a process in which we pick a problematic edge (at random), pick another edge (again at random), and swap two of their endpoints. If we keep doing this, we will eventually fix all the problems and get a random graph of the desired degrees[1].

I implemented a slight variation of this: I chose to randomly re-wire the graphs I was given. That is, I went over the edges one-by-one and randomly swapped one of their endpoints with the endpoint of another randomly chosen edge. I then repeated the process to eliminate self-loops and multi-edges. Another way of implementing this is to only swap the endpoints of two edges when it will not create a self-loop or multi-edge. In fact, this process corresponds to a random walk on a giant graph. We view each choice of the edges in our graph as a vertex on a giant graph. When we swap two edges, we move to another vertex in the giant graph. We can then ask if this process will eventually give us a random vertex of the giant graph, which would be a random graph with the given degree distribution. In a moment, we will compare the curves from percolation on these graphs to percolation on our original graphs.

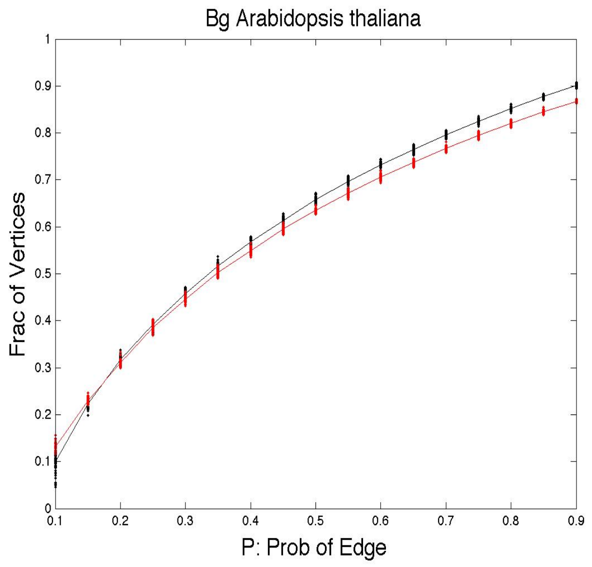
Random graphs generated this way often have a percolation threshold. I say "often" because I only know the theorem in the case that I keep the fraction of nodes of each degree fixed, and then let the total number of nodes grow. When we do this, the degree of every node grows small relative to the total number of nodes. In this case, Molloy and Reed [MR95] prove that the critical probability is

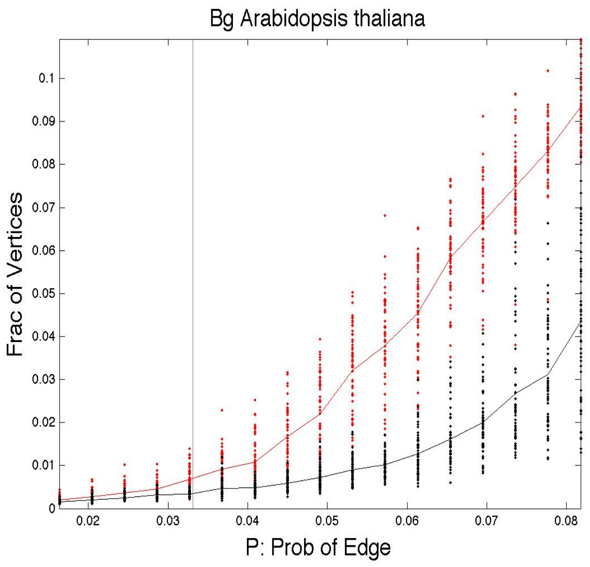
$$p_{crit} = \frac{\sum_i d_i}{\sum_i d_i^2 - \sum_i d_i}.$$

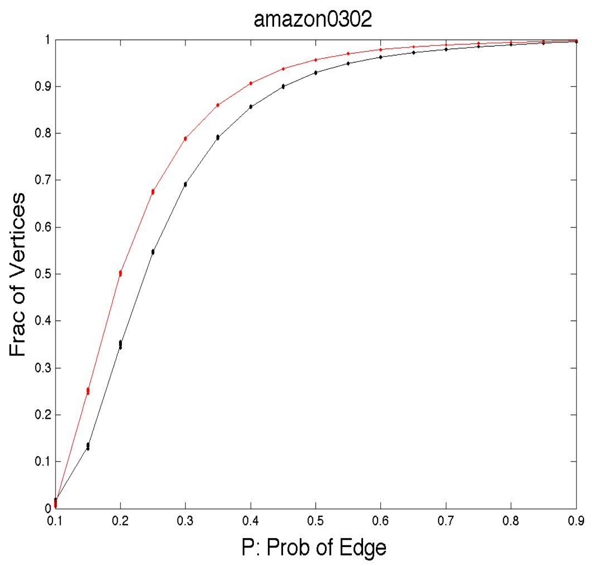
This is the threshold between having a largest component of a negligible fraction of the vertices and a constant fraction of the vertices. But, the constant fraction could be very small. This analysis essentially depends on the graph being locally tree-like almost everywhere. See [DGM08] for a much more extensive discussion of percolation in various graphs.

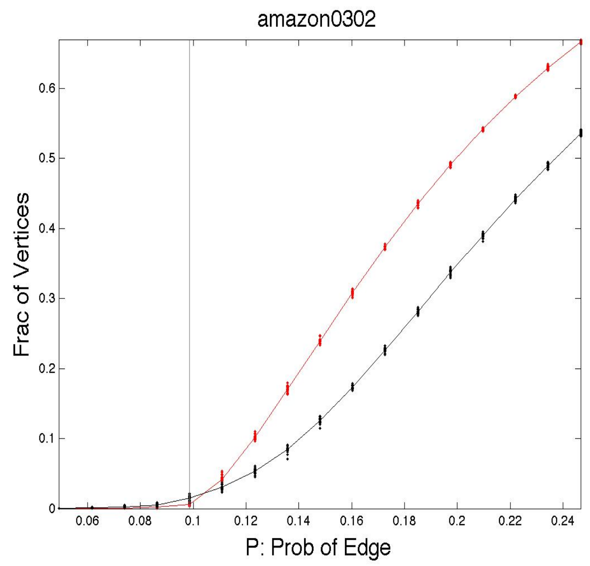
In the following demo, we will go through a few of the graphs, and I display fewer in these notes. The black curves are the originals, and the red curves come from the randomly rewired graphs. For each, we examine the curves on two scales: 0 to 1, and near the critical probability (which is drawn as a vertical line). The code to draw all this is `dataPercDemo2`.

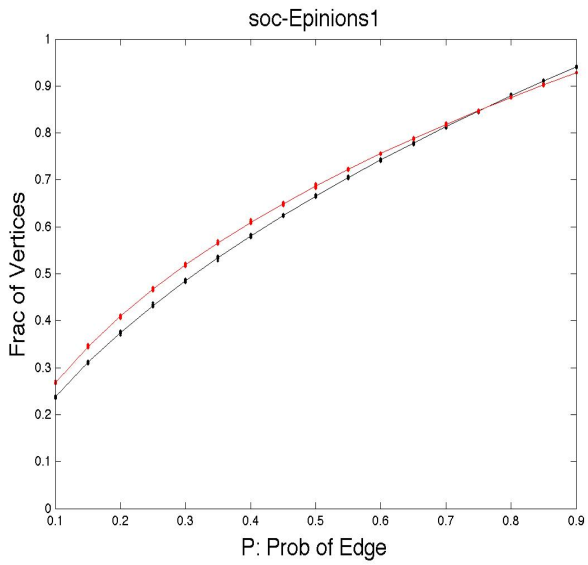
[1]You should ask whether we have sampled from the desired space uniformly. The answer is known, but I have not yet had a chance to look it up.

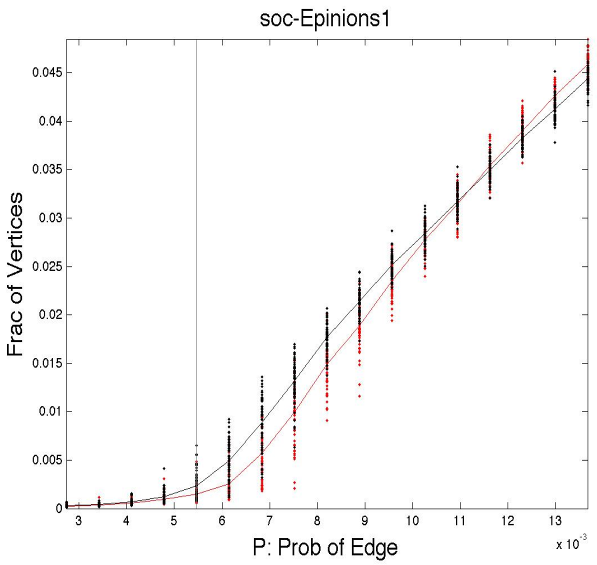

Bg Arabidopsis thaliana
Frac of Vertices
P: Prob of Edge

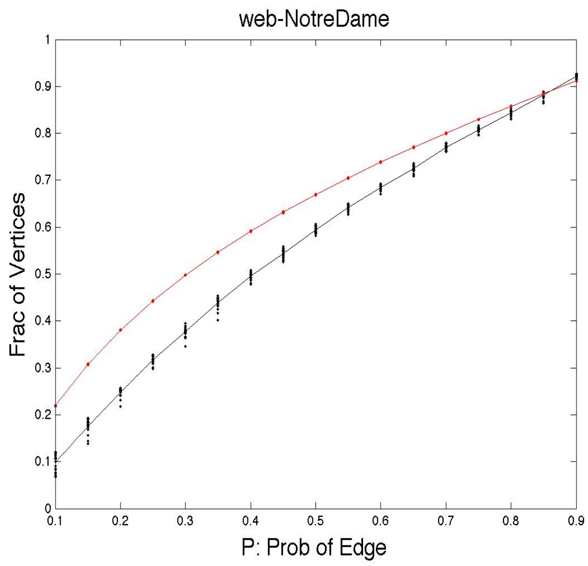

Bg Arabidopsis thaliana
Frac of Vertices
P: Prob of Edge

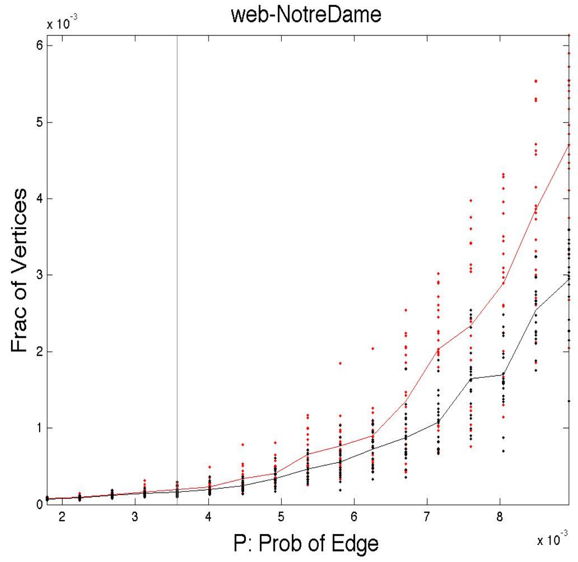

amazon0302
Frac of Vertices
P: Prob of Edge


amazon0302
Frac of Vertices
P: Prob of Edge









8.4 Random Walks

In this lecture, we will consider random walks on undirected graphs. Let's begin with the definitions. Let $G = (V, E)$ be an undirected graph. A random walk on a graph is a process that begins at some vertex, and at each time step moves to another vertex. The vertex the walk moves to is chosen uniformly at random among the neighbors of the present vertex. There are many things we can measure about a random walk, including

1. What is the limiting distribution of the random walk?
2. How long should it take before the walk approaches the limiting distribution?

3. How long should it take before the walk hits every vertex?

3. If we start a walk at s, what is the expected number of steps before it hits t?

Instead of keeping track of where a particular random walk is in a particular simulation, we usually measure the *probability distribution* of the random walk. That is, the probability at a given step that it is at any given vertex. We treat probability distributions on the vertices as vectors $\boldsymbol{p} \in \mathbb{R}^n$. I will sometimes write $\boldsymbol{p} \in \mathbb{R}^V$ to emphasize that $\boldsymbol{p}$ is a vector indexed by the vertices of the graph, or I may even write $\boldsymbol{p} : V \to \mathbb{R}$. I will write $\boldsymbol{p}(u)$ to indicate the value of $\boldsymbol{p}$ at a vertex u–that is the probability of being at vertex u in distribution $\boldsymbol{p}$. A probability vector $\boldsymbol{p}$ should satisfy $\boldsymbol{p}(u) \geq 0$, for all $u \in V$, and

$$\sum_u \boldsymbol{p}(u) = 1.$$

We let $\boldsymbol{p}_0$ denote the initial probability distribution. That is, $\boldsymbol{p}_0$ encodes where we start the random walk. If we start the random walk at a vertex v, then

$$\boldsymbol{p}_0(w) = \begin{cases} 1 & \text{if } w = v \\ 0 & \text{otherwise.} \end{cases}$$

But, we could start the walk at a vertex chosen from some distribution.

We will let the vector $\boldsymbol{p}_t \in \mathbb{R}^n$ denote the probability distribution at time t. To derive a $\boldsymbol{p}_{t+1}$ from $\boldsymbol{p}_t$, note that the probability of being at a vertex u at time $t+1$ is the sum over the neighbors v of u of the probability that the walk was at v at time t, times the probability it moved from v to u in time $t+1$. Algebraically, we have

$$\boldsymbol{p}_{t+1}(u) = \sum_{v:(u,v)\in E} \boldsymbol{p}_t(v)/d(v), \tag{8.1}$$

where $d(v)$ is the degree of vertex v.

We will often consider lazy random walks, which are the variant of random walks that stay put with probability $1/2$ at each time step, and walk to a random neighbor the other half of the time. These evolve according to the equation

$$\boldsymbol{p}_{t+1}(u) = (1/2)\boldsymbol{p}_t(u) + (1/2) \sum_{v:(u,v)\in E} \boldsymbol{p}_t(v)/d(v). \tag{8.2}$$

8.5 Diffusion

There are a few types of diffusion that people study in a graph, but the most common is closely related to random walks. In a diffusion process, we imagine that we have some substance that can occupy the vertices, such as a gas or fluid. At each time step, some of the substance diffuses out of each vertex. If we say that half the substance stays at a vertex at each time step, and the other half is distributed equally among its neighboring vertices, then the distribution of the substance will evolve according to equation (8.2). That is, probability mass in the lazy random walk obeys this diffusion equation.

People consider finer time steps in which smaller fractions of the mass leave the vertices. In the limit, this results in continuous random walks. Newman talks about these in Sections 6.13 and 6.14.

A variation of diffusion that I enjoy involves placing both matter and anti-matter at vertices. These cancel each other out when they meet. We can then ask question likes "if we put 1 unit of matter at v and 1 unit of anti-matter at u, how much matter will be left after time t?" The better connected vertices u and v are, the less mass that will be left.

8.6 Matrix form

The right way to understand the behavior of random walks is through linear algebra. First, we will re-write equation (8.1) in matrix form. It will help to identify the vertices with the set $\{1, \dots, n\}$. We then define the adjacency matrix, $\boldsymbol{A}$, of the graph by setting $\boldsymbol{A}(v,u)$ to 1 if $(u,v) \in E$ and 0 otherwise[2]. We then define the matrix $\boldsymbol{D}$ to be the diagonal matrix of degrees. That is $\boldsymbol{D}(u,u) = d(u)$, and $\boldsymbol{D}(u,v) = 0$ for $u \neq v$. Note that $\boldsymbol{D}^{-1}$ is also a diagonal matrix, and its uth diagonal entry is $1/d(u)$.

We now define

$$\boldsymbol{W} = \boldsymbol{A}\boldsymbol{D}^{-1}$$

to be the *walk matrix* of the graph. You can compute that

$$\boldsymbol{W}(u,v) = \begin{cases} 1/d(v) & \text{if } (u,v) \in E \\ 0 & \text{otherwise.} \end{cases}$$

We will now see that the update rule (8.1) is equivalent to

$$\boldsymbol{p}_{t+1} = \boldsymbol{W}\boldsymbol{p}_t.$$

This follows from the definition of matrix multiplication, and is obvious once you see it. From the definition of matrix multiplication, we know that

$$\begin{aligned} \boldsymbol{p}_{t+1}(u) &= \sum_v \boldsymbol{W}(u,v)\boldsymbol{p}_t(v) \\ &= \sum_{v:(v,u)\in E} (1/d(v))\boldsymbol{p}_t(v). \end{aligned}$$

You should check that this is identical to (8.1).

The matrix that realizes the lazy random walk is

$$\widehat{\boldsymbol{W}} \overset{\text{def}}{=} (1/2)(\boldsymbol{I} + \boldsymbol{W}).$$

[2]Note that I write $\boldsymbol{A}(v,u)$ to indicate entry of $\boldsymbol{A}$ in row v and column u. For undirected graphs, this entry is the same as $\boldsymbol{A}(u,v)$. In the directed case, I associate $\boldsymbol{A}(v,u)$ with edge (u,v) so that I can multiply by column vectors.

This formulation makes it easy to formalize the matter/anti-matter problem I gave a few moments ago. To start with 1 unit of matter at v and 1 unit of anti-matter at u, use the initial vector $\boldsymbol{x}_0$ where

$$\boldsymbol{x}_0(w) = \begin{cases} 1 & \text{if } w = v \\ -1 & \text{if } w = u \\ 0 & \text{otherwise.} \end{cases}$$

After t steps, the distribution of matter and anti-matter is given by $\boldsymbol{W}^t \boldsymbol{x}_0$. The cancellations happen automatically.

8.7 The steady-state distribution

A natural question to ask is whether the distribution of a random walk eventually converges to a steady state. It usually, but not always, does. For example, consider the graph with one edge and two vertices. The random walk will alternate vertices every other step, and never converge to one fixed distribution. A similar thing happens when the graph is bipartite: the walk goes from side to side every other step. It turns out that bipartite graphs are the only ones on which the random walk does not converge to a steady state. This is one motivation for studying lazy random walks. Lazy random walks always converge to a steady-state distribution. We will prove this next lecture.

For now, we just observe that there is a steady-state distribution and prove that it is unique. The steady-state distribution appears at each vertex with probability proportional to its degree. It is given by the vector $\boldsymbol{\pi}$ which we define by

$$\boldsymbol{\pi}(u) = \frac{d(u)}{\sum_{v \in V} d(v)}.$$

It is easy to verify that $\boldsymbol{\pi}$ is a probability vector, and we will now observe that

$$\boldsymbol{W}\boldsymbol{\pi} = \boldsymbol{\pi}. \tag{8.3}$$

So, if the walk ever reaches the distribution $\boldsymbol{\pi}$, it will remain in that distribution. To see this, it suffices to ignore the denominator and just show that $W\boldsymbol{d} = \boldsymbol{d}$, where $\boldsymbol{d}$ is the vector of degrees. To prove this, note that $D^{-1}\boldsymbol{d} = \mathbf{1}$, the all-1's vector, and $\boldsymbol{A}\mathbf{1} = \boldsymbol{d}$, as the u-th coordinate of $\boldsymbol{A}\mathbf{1}$ is

$$\sum_v \boldsymbol{A}(u,v) = \sum_{v:(v,u) \in E} 1 = d(u). \tag{8.4}$$

This is where we use the fact that the graph is undirected. In the directed case, the above expression would evaluate to the in-degree of u.

Note that equation (8.3) says that $\boldsymbol{\pi}$ is a right-eigenvector of $\boldsymbol{W}$ of eigenvalue 1. We will exploit this formulation more in the next lecture.

8.8 Uniqueness of the steady-state distribution

I'll now prove to you that if an undirected graph G is connected then the steady-state distribution is unique.

Lemma 8.8.1. *Let G be a connected graph and let $\boldsymbol{p}$ be a probability vector such that $\boldsymbol{W}\boldsymbol{p} = \boldsymbol{p}$. Then $\boldsymbol{p} = \boldsymbol{\pi}$.*

Proof. First, let u be any vertex that maximizes

$$\frac{\boldsymbol{p}(u)}{d(u)}.$$

From the assumption that $\boldsymbol{W}\boldsymbol{p} = \boldsymbol{p}$, we have

$$\begin{aligned}\boldsymbol{p}(u) &= \sum_{v:(v,u)\in E} \frac{\boldsymbol{p}(v)}{d(v)} \\ &\leq \sum_{v:(v,u)\in E} \frac{\boldsymbol{p}(u)}{d(u)} \\ &= \boldsymbol{p}(u).\end{aligned}$$

Now, the only way we can achieve this equality is if all of the inequalities on the second line are in fact equalities. This means that

$$\frac{\boldsymbol{p}(v)}{d(v)} = \frac{\boldsymbol{p}(u)}{d(u)} \tag{8.5}$$

for all neighbors v of u.

We can use this argument to show that this ratio is fixed for every vertex in the graph. To do that formally, let S be the set of vertices v for which (8.5) holds. Assume by way of contradiction that there is some vertex w not in S. As the graph is connected, it contains a path between u and w. Thus, there must be some edge (y, z) for which $y \in S$ and $z \notin S$. To obtain a contradiction, repeat the argument from the beginning of the proof with y in the role of u. It tells us that

$$\frac{\boldsymbol{p}(y)}{d(y)} = \frac{\boldsymbol{p}(z)}{d(z)},$$

which contradicts our assumption that $y \in S$ and $z \notin S$. □

8.9 Types of Convergence

In the next lecture, we will analyze how quickly the distribution of a random walk converges to the steady state and we will examine the obstacles to fast convergence. But first, we need to figure out what it means to converge.

The simplest measures of convergence reduce this to one number measuring how far $\boldsymbol{p}_t$ is from $\boldsymbol{\pi}$. One can measure this as a norm of the vector $\boldsymbol{p}_t - \boldsymbol{\pi}$. Different norms are useful in different applications. The standard is the Euclidean norm, also called the 2-norm:

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_u \boldsymbol{x}(u)^2}.$$

This one is so standard that the subscripted 2 is often dropped.

Another standard norm is the 1-norm:

$$\|\boldsymbol{x}\|_1 = \sum_u |\boldsymbol{x}(u)|.$$

This gives a distance called the *total variation* distance between $\boldsymbol{p}_t$ and $\boldsymbol{\pi}$.

The third common norm is called the max-norm or the infinity-norm, and it measures the maximum difference between $\boldsymbol{p}_t$ and $\boldsymbol{\pi}$:

$$\|\boldsymbol{x}\|_\infty = \max_u |\boldsymbol{x}(u)|.$$

8.10 A more holistic approach

I often find it useful to examine a plot of the whole distribution $\boldsymbol{p}_t$. To set this up, I would first like to view the random walk as living on sockets rather than on vertices. Recall that I define a socket to be the endpoint of an edge where it attaches to a vertex. For example, the edge (u, v) has two sockets: one at u and one at v. If the graph has m edges then it has $2m$ sockets.

I will now describe the lazy random walk on a graph in terms of a walk on the sockets.

When the random walk is at u, it next moves to a random neighbor of u. We can view this as first moving to a random socket attached to u. If it follows this socket to node v, we can then immediately pick the socket attached to v that it will traverse next. Recall that the stead-state distribution $\boldsymbol{\pi}$ visits each vertex with probability proportional to its degree. If we transfer this distribution to the sockets, then we see that it visits each socket with *equal* probability.

Assign a number between 1 and $2m$ to each socket. I will denote a probability distribution on sockets by a vector $\boldsymbol{q} \in \mathbb{R}^{2m}$. Given a probability distribution on the vertices $\boldsymbol{p}$, we can derive the corresponding distribution on sockets by setting $\boldsymbol{q}(s)$ to be $\boldsymbol{p}(u)/d(u)$ when s is a socket attached to vertex u.

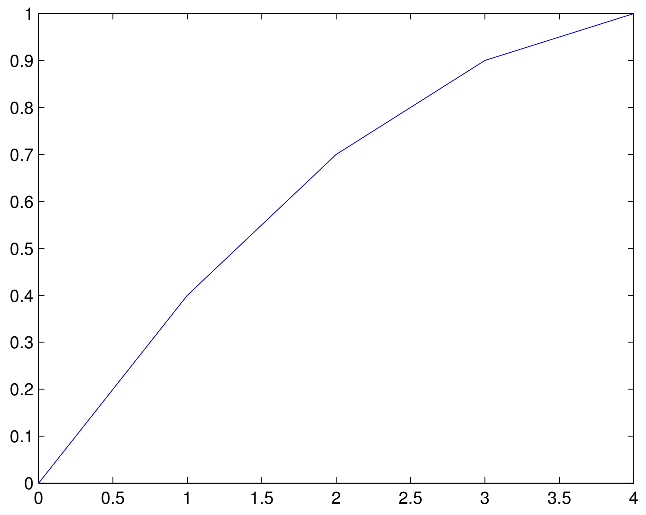
I will take this vector $\boldsymbol{q}$, sort it from largest to smallest, and measure its cumulative sum. To be concrete, I will do with a toy vector $\boldsymbol{q}$.

```
>> q = [.1, .2, .3, .4];
>> qs = sort(q,'descend')

qs =

    0.4000    0.3000    0.2000

>> plot([0:4],[0,cumsum(qs)])
>> hold on
>> plot([0:4],[0,cumsum(qs)],'o'
```



Let me state this formally. For each integer k between 0 and $2m$ I define $C_{\boldsymbol{q}}$ to be the sum of the largest k entries in the vector $\boldsymbol{q}$. I then plotted k against $C_{\boldsymbol{q}}(k)$, drawing straight lines between the integral points. When $\boldsymbol{q}$ is the uniform distribution, which is equal to $1/2m$ in each entry, $C_{\boldsymbol{q}}(k) = k/2m$. So, this plot is a straight line.

For every other probability distribution $\boldsymbol{q}$ on sockets, the plot of $C_{\boldsymbol{q}}$ is concave and lies above the straight line from $(0,0)$ to $(2m,1)$. One can show that $C_{\boldsymbol{q}_{t+1}}$ lies beneath $C_{\boldsymbol{p}_t}$ for every t. If I get to this material in today's lecture, then we will use this approach next lecture to show that $C_{\boldsymbol{q}_t}$ converges to the straight line.

References

[DGM08] Sergey N Dorogovtsev, Alexander V Goltsev, and José FF Mendes. Critical phenomena in complex networks. *Reviews of Modern Physics*, 80(4):1275, 2008.

[MR95] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random structures & algorithms*, 6(2-3):161–180, 1995.