Graphs and Networks	**out:** October 17, 2013
Problem Set 4	
Lecturer: Daniel A. Spielman	**due:** November 5, 2013

1 Introduction

This problem set has 4 theoretical problems and 1 experimental problem. To do the experimental problem, you must be enrolled in the course through Classes V2. See the rest of the instructions below.

The problem set is due **in class** on Tuesday, November 5. If you cannot come to class, you can put it in Dan Spielman's mailbox on the first floor of AKW before then.

I recommend that you start working on it as soon as you can.

2 Homework Policy

You may discuss the problems with other students. But, you must write your solutions independently, drawing on your own understanding. You should cite any sources that you use on the problem sets other than the textbook, TFs and instructor. This means that you should list your collaborators.

It goes without saying that you should write your own code for the experimental part.

You **may not** search the web for solutions to similar problems given out in other classes. If you think this policy needs any clarification, please let me know.

Problem 1: Symmetry in Personal PageRank

Let G be a connected, undirected graph and let u and v be two vertices in G. Fix α and let $\boldsymbol{p}_u$ and $\boldsymbol{p}_v$ be the Personal PageRank vectors starting at u and v respectively, with teleportation probability α. Prove that

$$\frac{\boldsymbol{p}_v(u)}{d(u)} = \frac{\boldsymbol{p}_u(v)}{d(v)}.$$

Problem 2: Symmetry in Electrical Flow

Let G be a connected, undirected, unweighted graph. Alternatively, you can think of every edge in G as having resistance 1. Treat G as a resistor network.

Let (x, y) and (w, z) be two different edges of G. Let $\boldsymbol{i}_{x,y}(w, z)$ be the amount of current that flows over the edge (w, z) in the electrical flow that sends one unit of current from x to y. Prove that

$$|\boldsymbol{i}_{x,y}(w, z)| = |\boldsymbol{i}_{w,z}(x, y)|\,.$$

Problem 3: Obvious Fact

Let G be a connected, undirected, unweighted graph. We will treat G as a spring network, so you can think of every edge as having spring constant 1. For any two vertices s and t, set the position of vertex s to 0, set the position of vertex t to 1, and let the others settle to the positions that minimize energy. Prove that every other vertex will have a position between 0 and 1.

Hint: This should be easy.

Problem 4: More Monotonicity

Show that if you just increase the resistance on an edge e then you will not increase the flow on that edge.

That is, let $G = (V, E)$ be an undirected, connected graph and let $\boldsymbol{r}$ be the edge resistances. Let e be an edge of E and let $\boldsymbol{r}'$ be another set of resistances such that

$$\begin{aligned}&\boldsymbol{r}'(f) = \boldsymbol{r}(f), \text{for all } f \neq e, \text{ and}\\ &\boldsymbol{r}'(e) \geq \boldsymbol{r}(e).\end{aligned}$$

Let (s, t) be any edge of G. Prove that when we flow one unit of current from s to t, the amount of current that flows on edge e under resistances $\boldsymbol{r}'$ is at most the amount that flows under resistances $\boldsymbol{r}$.

Hint: This is the trickiest. Problem 2 can help if you generalize it to arbitrary resistances.

Problem 5 : Semi-Supervised Learning

In this problem, you are going to try out the approach to semi-supervised learning that we covered in Lecture 12. Everyone is going to use the same graph: `amazon` (there will be a link on the web page for Problem Set 4). This is a co-purchasing graph in which the vertices represent products that Amazon sells. Amazon has also divided these products into categories. Each of you is going to try to reconstruct a different category.

Each of the categories that I have assigned have between 90 and 400 elements. For each category, I will give a list of 10 random products in that category and 100 random products that are not in that category. Your job is to construct a list of products that comes close to the category. You will do this by viewing the graph as a spring or resistor network. You should set $\boldsymbol{x}(a) = 1$ for every vertex a that is in the category, setting and $\boldsymbol{x}(0) = 0$ for every vertex that is not in the category, and letting the rest of the values of $\boldsymbol{x}$ settle. That is, they should satisfy the harmonic equations. These were labeled (12.1) in the lecture notes, with W being the set of 110 vertices whose status I have revealed to you.

Once you solve for this $\boldsymbol{x}$, you should report some subset of the vertices for which the value of $\boldsymbol{x}$ is highest. I recommend using the procedure from the last problem set: sort the vertices in decreasing order of their value in $\boldsymbol{x}$, and report the prefix that results in the sparsest cut. Since you know that you are looking for a set with between 90 and 400 vertices, it would be silly to report a set of 10 or 10,000 vertices. So, only consider sets with between 90 and 400 vertices. I will set up a server that will tell you if your answer is correct.

Warning: The difficult part of this problem is solving the system of linear equations. This is a very big system of linear equations, so you cannot do it using Gaussian Elimination. Rather, you should use an "indirect" method such as Conjugate Gradient. These go by names like "cg", "pcg", "cgs" and "gmres". They do not solve the system exactly, but rather can come up with better and better solutions the longer you let them run. On the Useful Software page, I give pointers to where you can find code that implements these. It is easiest in Matlab, where you can just call pcg.

You need to be careful when using these routines to solve a system, as they do not always give the right answer. If your system has the form $Mx = b$, you should check your solution by computing the norm of $b - Mx$. For this problem, I recommend driving it to below 10^{-5}. If you do not get a solution with small enough error, then let the algorithm run for more iterations and try again.

Solving these linear systems can take a while. A few minutes is reasonable on a fast machine. By simplifying the system you can help the code. When you set up the system of linear equations, you will get a system of form $Mx = b$ where M will not be symmetric. If M is not symmetric, you have to use either cgs or gmres. But, if you can convert the system into a form that is symmetric, then you can use the cg (or pcg) algorithm. You *should not* do this by multiplying by M^T, but rather by observing that your system will have a lot of rows with just one non-zero element. If you eliminate these manually, you can reduce the system to one in a sub-matrix of M that will be symmetric. Getting all of this to work can take a while.

I will also provide a smaller graph for you to play with.