CS224W: Social and Information Network Analysis **Fall 2012**

Problem Set 1

Due 9:30am October 11, 2012

General Instructions

These questions require thought, but do not require long answers. Please be as concise as possible. Please fill the cover sheet (http://cs224w.stanford.edu/cover.pdf) and submit it as a front page with your answers. Include the names of your collaborators (see the course website for the collaboration or honor code policy). You are allowed to take maximum of 1 late day (see course website for a definition of a late day).

Regular (non-SCPD) students should submit hard copies of the answers either in class or in the submission cabinet (see course website for location). You should also include source code and any other files you used.

SCPD students should submit their answers through SCPD. The submission must include all the answers, the source code and the usual SCPD routing form (http://scpd.stanford.edu/generalInformation/pdf/SCPD_HomeworkRouteForm.pdf).

Questions

The purpose of this homework is to get you started with network analysis and get you install and try out graph analysis tools. We encourage you to use any graph analysis tools or packages you might want (SNAP for C++, NetworkX for Python, JUNG for Java, etc.). See http://cs224w.stanford.edu/resources.html for details.

1 Network Characteristics [35 points – Jacob]

One of the goals of network analysis is to find mathematical models that characterize real-world networks and that can then be used to generate new networks with similar properties. In this problem, we will explore two famous models—Erdős-Rényi and Small World—and compare them to real-world data from an academic collaboration network. Note that in this problem all networks are *undirected*.

- $G(n, m)$ *Random Network:* To construct this undirected network, use $n = 5242$ nodes and pick $m = 14496$ edges at random.
- *Small-World Network:* You can generate this undirected graph as follows: Begin with $n = 5242$ nodes arranged as a ring, i.e., imagine the nodes form a circle and each node

is connected to its two direct neighbors (e.g., node 399 is connected to nodes 398 and 400), giving us 5242 edges. Next, connect each node to the neighbors of its neighbors (e.g., node 399 is also connected to nodes 397 and 401). This gives us another 5242 edges. Finally, randomly select 4012 pairs of nodes not yet connected and add an edge between them. In total, this will make $m = 5242 \cdot 2 + 4012 = 14496$ edges.

- *Real-World Collaboration Network:* You can download this *undirected* network from http://snap.stanford.edu/data/ca-GrQc.txt.gz. Note that some edges may appear twice in the data, once for each direction. You should consider edges as undirected and store at most one edge between any pair of nodes, for a total of 5,242 nodes and 14,496 edges.

(a) Degree Distribution [10 points]

Generate or read in the above three networks. Plot and compare the log-log degree distribution[1] of all three networks. Superimpose the plots of the three networks. Briefly describe the shape of each distribution and explain the differences between the networks.

(b) Excess Degree Distribution [15 points]

An important concept in network analysis is the *excess degree distribution*, denoted as q_k, for $k \geq 0$. Intuitively, q_k gives the probability that a randomly chosen edge goes to a node of degree $k+1$. Excess degree can be calculated as follows:

$$q_k = \frac{q'_k}{\sum_i q'_i}, \qquad q'_k = \sum_{i \in V} \sum_{(i,j) \in E} I_{[k_j = k+1]},$$

where $I_{\text{condition}} = 1$ when condition is true and 0 otherwise. V denotes the set of nodes, E the set of edges and k_j the number of neighbors of j (equivalently, the degree of j). Additionally, the *expected excess degree* is defined as $\sum_{k \geq 0} k q_k$.

The degree distribution of the network is denoted by $\{p_k | k \geq 0\}$, i.e., p_k is the proportion of nodes having degree exactly k. The *expected degree* can consequently be computed as $\sum_{k \geq 0} k p_k$.

Your Tasks:

- **[10 points]** Plot and compare the log-log excess degree distributions of all three networks. Report the expected degree and the expected excess degree (the formulæ are given above) for each network. Also, focus on the tail of the distributions (large degrees) and briefly explain how and why these plots differ from the degree distribution plots (especially for the collaboration network) that you obtained in part (a).

[1] Generate a plot with the horizontal axis representing node degrees and the vertical axis representing the proportion of nodes with each degree. By log-log we mean that both the horizontal and vertical axis must be in logarithmic scale.

- **[5 points]** Can you calculate the excess degree distribution $\{q_k\}$ using the degree distribution $\{p_k\}$? Find this formula.

(c) Clustering Coefficient [10 points]

Recall that the local clustering coefficient for a node v_i was defined in class as

$$C_i = \frac{2|e_i|}{k_i \cdot (k_i - 1)},$$

where k_i is the degree of v_i and e_i is the number of edges between the neighbors of v_i. Find the *average clustering coefficient* for all the three networks, which is defined as

$$C = \frac{1}{|V|} \sum_{i \in V} C_i.$$

(if a node has 0 or 1 neighbor, we ignore it). Compare the average clustering coefficient of the three network types and explain why it is in the order you see. For this question, please write your own implementation of finding the clustering coefficient instead of using a built-in library function.

2 Fighting Reticulovirus Avarum [20 points – Ashton]

One of the main applications of network analysis is the study of how viruses spread. By modeling hosts as nodes and host A infecting host B as a directed edge from A to B, we have a very natural and useful way to represent the spread of a virus as a network. To understand how a virus might spread, one first needs to understand the underlying network on which it is spreading.

In this problem, we will apply the network analysis techniques that you've learned in class to gain insight into how an email virus might propagate. We concentrate on email viruses because we have very accurate information on the underlying networks through which they spread. The data we'll use is an anonymous email network available on the handouts page (available for download at http://www.stanford.edu/class/cs224w/email_network.txt).

An evil TA (Bob) has written a nasty email virus called Reticulovirus avarum (R. avarum) that can propagate through this network. Your job is to analyze the worst-case scenario to obtain an upper bound on how bad the spread could be: Our working assumption will be that once R. avarum infects a host, it always infects all of the host's contacts if they are not already infected.

(a) **[5 points]** If you know the first node to contract the virus, which network analysis concept captures the set of nodes that will eventually also be infected?

(b) [**5 points**] In this part, we'll compute a simplified version of the bow-tie structure of the email network. How big (relative to the entire graph) are: (1) the largest SCC, (2) the in-component of the largest SCC, (3) the out-component of the largest SCC, and (4) the disconnected components (the components that aren't connected to the "bow-tie" components you just computed). Don't worry about the tendrils or tubes.

(c) [**5 points**] Assume R. avarum is only introduced once into the email network. What is the probability that R. avarum becomes a large-scale epidemic (assume "large-scale" means at least 30% of the population gets infected)? Explain your reasoning. [Note: use *only* the numbers you computed in part (b) (i.e., no additional computation is needed to answer this question].

(d) [**5 points**] Again assume that R. avarum is only introduced some random one node in the email network. What's the worst-case (biggest) possible outbreak size? Explain your reasoning. [The same note as in the previous part also applies here: no additional computation is needed to answer this question.]

3 Decentralized Search [45 points – Wayne]

In class, we saw a decentralized search algorithm based on geography that seeks a path between two nodes on a grid by successively taking the edge towards the node closest to the destination.

Here we will examine an example of a decentralized search algorithm on a network whose nodes reside in the leaves of a tree. The tree may, for instance, be interpreted as representing the hierarchical organization of a university where one is more likely to have friends inside the same department, a bit less likely in the same school, and the least likely across schools.

Let us organize students at Stanford into a tree hierarchy, where the root is Stanford University and the second level contains the different schools (engineering, humanities, etc.). The third level represents the departments and the final level (i.e., the leaves) are the Stanford students. Tom, a student from the computer science department, wants to hang out with Mary, who is in sociology. If Tom does not know Mary, he could ask a friend in the sociology department to introduce them. If Tom does not know anybody in the sociology department, he will seek a friend in the Stanford humanities school instead. In general, he will try to find a friend who is "close" to Mary in the tree.

There are three parts in this problem. The first two parts explore an effective decentralized search algorithm on the hierarchical model in a specific setting. The third part involves simulation experiments on the model under a more general setting. There are many subproblems, but do not panic. A good understanding of the math involved in the lattice model we covered in class will make this problem easy and shorten each subproblem to just a few line proofs.

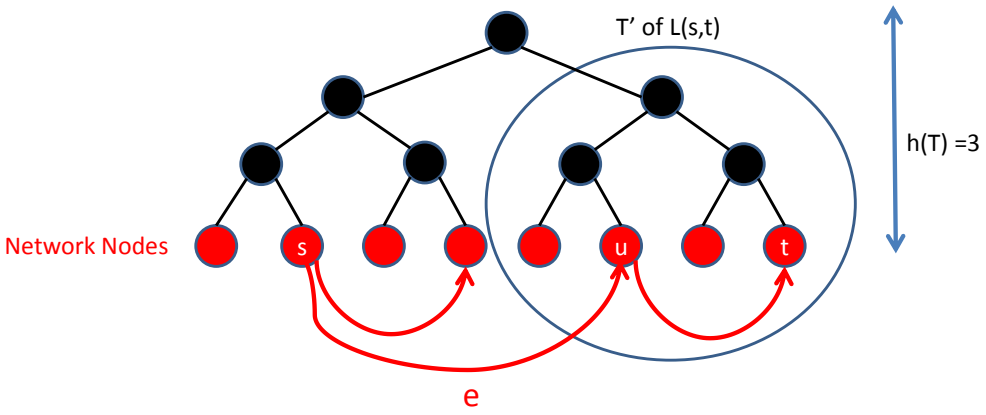

Figure 1: Illustration for Question 3 of the hierarchical graph. Black nodes and edges are used to illustrate the hierarchy and structure, but are not part of our network. Red nodes (leaf nodes) and red edges are the ones in our network. The lowest common ancestor of s and t is the root of the tree. The decentralized search proceeds as follows. Denote the starting node by s and the destination by t. At each step, the algorithm looks at the neighbors of the current node and moves to the one closest to t. In this graph, s will move to u.

(a) Basic Tree Properties [15 points]

This part covers some basic facts of the setting of the hierarchical model.

Consider a complete b-ary tree (each non-leaf node has b children and $b \geq 2$) T, and a network whose nodes are the leaves of T (the red nodes in the picture). Let the number of network nodes (equivalently, the number of leaves in the tree) be N and Let $h(T)$ denote the height of T.

One important thing to keep in mind: In this problem, the network nodes are only the leaf nodes in the tree. Other nodes in the tree are virtual and only there to determine the edge creation probabilities in the network.

(i) [5 points] Write $h(T)$ in terms of N.

(ii) [5 points] Given two network nodes (leaf nodes) v and w, let $L(v, w)$ denote the subtree of T rooted at the lowest common ancestor of v and w, and $h(v, w)$ denote its height (that is, $h(L(v, w))$). In Fig. 1, $L(u, t)$ is the tree in the circle and $h(u, t) = 2$. For a given node v, what is the maximum possible value of $h(v, w)$?

(iii) [5 points] Given a value d and a network node v, how many nodes satisfy $h(v, w) = d$?

(b) Network Path Properties [20 points]

This part helps you design a decentralized search algorithm in the network.

We assume that the directed network is k-regular. That is, all nodes (leaf nodes) have outdegree k. Each outgoing edge from a node v goes to a node $w \neq v$ with probability

$\frac{1}{Z}b^{-h(v,w)}$, where Z is the normalizing constant $Z = \sum_{w \neq v} b^{-h(v,w)}$.

(iv) [5 points] Show that $Z \leq \log_b N$. (Hint: use the results in (ii) and (iii).)

For two leaf nodes v and t, let T' be the subtree of $L(v,t)$ satisfying:

- T' is of height $h(v,t) - 1$,
- T' contains t,
- T' does not contain v.

(For instance, in Fig. 1, T' of $L(s,t)$ is the tree in the circle.)

(v) [5 points] Let us consider node v and an edge e from v to a random node u. We say that e points to T' if u is a leaf node of T'. Show that the probability of e pointing to T' is no less than $\frac{1}{b \log_b N}$.

(vi) [5 points] Let the out-degree k for each node be $c \cdot (\log_b N)^2$, where c is a constant. Show that when N grows very large, the probability of v not having any edge pointing to T' is asymptotically no more than $N^{-\theta}$, where θ is a positive constant which you will need to compute. (Hints: Use the result in (v) and recall that each of the k edges is independently created. Also, use $\lim_{x \to \infty}(1 - \frac{1}{x})^x = \frac{1}{e}$.)

The above claim indicates that for any node v, we can, with high probability, find an edge to a (leaf) node u satisfying $h(u,t) < h(v,t)$.

(vii) [5 points] Show that starting from any (leaf) node s, within $O(\log_b N)$ steps, we can reach any (leaf) node t. You do not need to prove it in a strict probabilistic argument. You can just assume that for any (leaf) node v, you can always get to a (leaf) node u satisfying $h(u,t) < h(v,t)$ and argue why you can reach t in $O(\log_b N)$ steps.

(c) Simulation [10 points]

In (i) to (vii), we have set the theory to find a decentralized search algorithm, assuming that for each edge of v, the probability of it going to w is proportional to $b^{-h(v,w)}$. Now we experimentally investigate a more general case where the probability is proportional to $b^{-\alpha h(v,w)}$. Here $\alpha > 0$ is a parameter in our experiments.

In the experiments below, we consider a network with the setting $h(T) = 10$, $b = 2$, $k = 5$, and a given α. That is, the network consists of all the leaves in a binary tree of height 10; the out degree of each node is 5. Given α, we create edges according to the distribution described above.

(h) [7 points] Create random networks for $\alpha = 0.1, 0.2, \ldots, 10$. For each of these networks, sample 1000 random (s,t) pairs ($s \neq t$). Then do a decentralized search starting from s as follows. Assuming that we are currently at (leaf) node s, we pick its neighbor u (also a leaf node) with smallest $h(u,t)$ (break ties arbitrarily). If $u = t$, the search succeeds.

If $h(s,t) > h(u,t)$, we set s to u and repeat. If $h(s,t) \leq h(u,t)$, the search fails. For each α, pick 1000 pairs of nodes and compute the average path length for the searches that succeeded. Then draw a plot of the average search time (number of steps it takes to reach t) as a function of α. Also, plot the search success probability as a function of α.

(i) [3 points] Briefly comment on the plots and explain the shape of the curve.