Homework 1: Graph Theory in the Cornfields

36-720, Fall 2016

Due at 11:59 pm on 14 September 2016

One of the classic studies of network analysis and the diffusion of innovations concerned the spread of a new antibiotic among the doctors in a group of four small towns in the midwest in the 1950s. This assignment will use part of the data collected from this study to practice calculating basic summary statistics on networks, get a feel for the effects of sampling, and reasoning theoretically about network algorithms and dynamics.

The main data file, `http://www.stat.cmu.edu/~cshalizi/networks/16-1/hw/1/ckm_network.dat`, records a 246×246 binary adjacency matrix, indicating, for each pair of doctors in the four towns, whether or not they were socially linked[1]. We will come back to this network, and its covariates, in later assignments.

1. *Summaries* Read in the data file, into your favorite network analysis software, and make sure that it is a 246-node network.

   (a) (10) What is the average degree of the nodes? The standard deviation of the degree across nodes? The number of triangles? The number of four-cycles?

   Note: If you can count the number of isomorphisms from subgraphs of the network to a triangle, you will be over-counting the number of sets of nodes which form triangles by a factor of six. (Why 6?) Similarly, counting isomorphisms from sub-graphs to a four-cycle over-counts the number of four-cycles (by how much?).

   (b) (5) Plot degree against vertex betweenness for all nodes. What is the correlation between the rank of the two measures of importance?

   (c) (5) Find the number of connected components in the graph, and their sizes. You should see a very clear separation in the sizes. Suggest an explanation for the number of large clusters, and a way to check that explanation.

   (d) (8) For each of the large connected components, find the mean and standard deviation of the degree, the number of triangles, and the

[1] *For nit-pickers:* The original study actually collected three different social relations — advice-seeking, discussing medicine, and friendship — which I have collapsed to one, and made symmetric.

number of four-cycles. Repeat the plots of degree against vertex betweenness for each connected component. (One plot with multiple lines is acceptable.)

(e) (5) Plot the cumulative distribution function of the degree distribution for the full graph, and for the large connected components. Do the components seem to have the same degree distribution? Would it be valid to use a chi-square test for difference in distributions to test this formally?

2. *Induced subgraph sampling*

   (a) (10) Write code to take graph of n nodes, include each node independently with a probability p, and return the induced subgraph. How do you know that this code works?

   (b) (6) Using repeated simulation, plot the expected number of edges in the sampled graph as a function of p.

   (c) (6) Using repeated simulation, plot the expected number of triangles in the sampled graph as a function of p.

   (d) (6) Using repeated simulation, plot the expected number of 4-cycles as a function of p.

   (e) (4) Offer a conjecture on how the number copies of a given k-node motif appearing in the sampled graph varies with p, k, and the number of appearances of the motif in the full graph.

3. *Component-finding* Consider the following algorithm for finding connected components in an undirected graph. (You do not have to implement it, unless you find that helpful in doing the problems.) (I) Take the nodes in some arbitrary order. Label each node, initially, with its rank in this ordering. For notation, you may write $\ell(i,0)$ for the initial label of node i. (II) At each time step, each node replaces its label with the smallest label among its neighbors and itself, $\ell(i,t+1) = \ell(i,t) \vee \min_{j \in \mathrm{neighborhood}(i)} \ell(j,t)$. (Assume that all nodes do this minimization simultaneously, in parallel.) (III) Continue until no node changes labels. If this algorithm works, in the final state, all the nodes with a common label should be connected, and any two connected nodes should have the same label.

   (a) (5) Prove or refute the following claim: "For any node i, and any initial ordering of nodes, the label of i never increases."

   (b) (10) Prove or refute the following claim: "For any pair of nodes i, j, if $\ell(i,0) < \ell(j,0)$ and there is a path linking i and j, then there exists a time τ such that $\ell(j,t) \leq \ell(i,0)$ for all $t \geq \tau$".

   (c) (5) Will the algorithm always terminate? Either prove that it does, or provide a counter-example in the form of a graph and an ordering of its nodes where the algorithm will run forever.

(d) (5) If the algorithm terminates, will it always terminate correctly? Again, either prove that the final state always identifies the connected components, or provide a counter-example.

(e) *Extra credit (5)* Would the algorithm still work if in (II) the nodes update their labels in sequence, rather than simultaneously? Does the answer depend on the order of updating?

RUBRIC (10): The text is laid out cleanly, with clear divisions between problems and sub-problems. The writing itself is well-organized, free of grammatical and other mechanical errors, and easy to follow. Questions which ask for a plot or table are answered with both the figure itself and the command (or commands) use to make the plot. Plots are carefully labeled, with informative and legible titles, axis labels, and (if called for) sub-titles and legends; they are placed near the text of the corresponding problem. All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision. Code is properly integrated with a tool like R Markdown or knitr, and both the knitted file and the source file are submitted. The code is indented, commented, and uses meaningful names. All code is relevant to the text; there are no dangling or useless commands. All parts of all problems are answered with actual coherent sentences, and never with raw computer code or its output.