General Instructions

Submission instructions: You should submit your answers via GradeScope and your code via Snap submission site.

Submitting answers: Prepare answers to your homework into a single PDF file and submit it via [http://gradescope.com](http://gradescope.com). Make sure that answer to each question is on a separate page. This means you should submit a 7-page PDF (1 page for the cover sheet, 3 pages for the answers to question 1, and 3 pages for answers to question 2). On top of each page write the number of the question you are answering. Please find the cover sheet and the recommended templates located here:


Not including the cover sheet in your submission will result in a 2-point penalty.


Questions

1 Visualization [35 points - Caroline, Tim]

A network visualization should convey the properties of the network. You are tasked with visualizing the 2009-2013 US Census data on county-to-county commute flows in the US and Puerto Rico. The data was downloaded from: [http://www.census.gov/hhes/commuting/data/commutingflows.html](http://www.census.gov/hhes/commuting/data/commutingflows.html). There are 3 files: CenPop2010_Mean_CO.txt has the population and latitude/longitude of the county centers. 2010commuteflows.txt has the numbers of individuals commuting from county to county. commute.gdf has combined the data from the first two files into a format easily digestible by Gephi. You can use Gephi or any other network visualization software of your choice (GraphViz, InfoMap, etc.) to do the following:

(a) Create 1 or 2 network visualizations of the counties (or aggregations of nodes or just counties within a region of the US) and flows (or subsets of flows) between them. Use at least 3 of the following to convey information about the network: node color, node size, edge color (not an option in Gephi), node shape (not an option in Gephi), edge thickness, node position, (selective) labels. The visualization should effectively convey characteristics of the network and need not show all data exhaustively.

(b) Create a caption for your network that succinctly describes the visualization along with an explanation of the features used (e.g.: the nodes are colored red to blue, with red being X and blue being Y) and also any modifications you made to the raw network.
(c) Draw a 1-2 sentence conclusion about US commute flows that draws on what your visualization has made apparent.

The TAs will pick the 3 most interesting and informative visualizations, which will be awarded 10 bonus points each.

**What to submit?**

1-2 network visualizations (screenshot) along with captions describing the visualization and 1-2 sentence conclusions each.

★ **SOLUTION:**

(a) Commute flows in the US.
(b) Commute flows in Colorado, Utah, Wyoming, & Nebraska.

(b) In the figures above, the nodes are colored using communities detected by Gephi’s "modularity" algorithm. The size of each node corresponds to its indegree in the network, after edges with < 100 workers were removed.
(c) From the figures above, and especially the regional visualization, it is apparent that some commute flows form communities across state boundaries, that are nevertheless geographically proximate. One can also identify counties containing large cities, which not only draw from many surrounding counties, but also apparently have cross-country flows between rather distant cities (e.g. west coast and east coast).

2 Diffusion with NetLogo [10 points Paris, Omid]

Take the ER diffusion model http://web.stanford.edu/class/cs224w/NetLogo/ERDiffusion.nlogo. This model is an SI model (nodes are either susceptible or infected). Make it an SIS model: nodes are susceptible, they become infected, and then they have a chance of recovering from the infection (at which point they are immediately susceptible again). At each time step your model should do the following: 1) Examine each node. Each node that is infected infects each of its uninfected neighbors with probability INFECT-RATE. 2) If a node is infected, after it is done with its infection attempts, with probability RECOVER-RATE it goes back to susceptible. It may be simplest to answer this question if you do the following:

Add a global variable RECOVER-RATE by adding a slider to the GUI Add a monitor variable (call it e.g. cumul-av), which records the cumulative average number of infected nodes over time. Once done, your interface may look something like this.

NetLogo is likely unfamiliar to you. Browse its documentation. You may find the built-in models library a great source of example code.

In answering the following questions, run the model repeatedly (setting up a new network topology, then reinfecting a few times, each time giving the virus a chance to infect a non-trivial fraction of the network). Once you have the virus wreaking havoc in the network, answer the following:

(a) Set INFECT-RATE to 0.15, and RECOVER − RATE to 0.40. The NUM − NODES should be 200. Repeatedly construct the network and reinfect it until you get an infection that is affecting a significant portion of the nodes. The way I like to do this is to construct the network once using SETUP-A-NEW-NETWORK, click on the SPREAD-REPEAT button, and then click on the SETUP-WITH-CURRENT-TOPOLOGY button several times until the infection takes hold. Let x be the average number of nodes infected in the long run (after a large initial infection) when AVG-DEGREE = 5. One approach is to keep a running average of the number of infected nodes, which you would reset after the infection establishes itself to avoid averaging the initial false starts (when the infection count is 0 because the infection has died out without achieving a steady state) Let y be the same av. # nodes infected in the long run when AVG-DEGREE = 4 (remember to click on SETUP-A-NEW-NETWORK when you change the average degree). Give the ratio x/y, accurate to within 0.2.

(b) Keeping the settings the same, again get an infection bouncing around your network by repeatedly reinfecting it until the infection spreads to some portion of the network. What is the highest value of AVG-DEGREE (to within .2) where the infection can die out after having infected a significant portion of the network?
★ SOLUTION:

(a) The ratio is 1.3: when the network is denser, a higher number of nodes that are infected at any
given time.

(b) The infection tends to die out when average degree is 3.3.

3 Network formation with NetLogo [20 points Paris, Omid]

Consider two network formation models.

(a) Consider a model http://web.stanford.edu/class/cs224w/NetLogo/InfoDiffNetwork.nlogo

of information diffusion on a network that changes the network itself. The network starts out
as a lattice with each node connected to its 4 closest neighbors. An origin is chosen at random,
and the gossip stops spreading at each step with probability STOP-PROB. The last node to
hear the gossip will connect to the origin with probability P-CONNECT-TO-ORIGIN. Add a
slider P-CLAIM that determines the probability that a node that hears the gossip will claim
that it is the origin. Add this action to the behavior of the gossipers (namely have them change
the global variable origin to be themselves). Observe how the long-term (much gossip has been
spread) degree distribution changes as you increase P-CLAIM from 0 to 0.3 while keeping P-
CONNECT-TO-ORIGIN at 0.5 and STOP-PROB at 0.05. Submit 2 screenshots, one of the
distribution at P-CLAIM = 0, the other at P-CLAIM = 0.3. Give a brief explanation of why
the distribution changes.

(b) Consider the “Connections Model from section 3.2.2 of Jacksons Social and Economic Networks:
http://web.stanford.edu/class/cs224w/NetLogo/JacksonUtilityModel.nlogo

Links are placed if the utility to both endpoints is greater with the edge present than without. This very
inefficient model calculates the utility $u_i$ for each node $u_i = \sum_j \delta_{p_{ij}} - c \ast k_i$, with $p_{ij}$ being
the link distance between $i$ and a reachable node $j$, and $k_i$ being the degree of $i$. That is,
there is a cost to each edge, but also a benefit in connectivity that falls with increasing path
length. Complete the model to keep or discard links according to the utility they bring to their
endpoints. With $n = 30$, $\delta = 0.5$ find the range of $c$ over which a non-empty, but non-complete
graph is stable (no pair of nodes wants to add an edge or delete one they already have). Submit
also a screenshot of your network in this intermediate range.

★ SOLUTION:

(a) After modifying our code so that gossiper can claim origin with certain probability we observe the
following behavior:

- When P-CLAIM is 0, the only way for a node to gain an incoming link and thus increase its
degree is for it to be ORIGIN. Since ORIGIN is picked uniformly at random at each round, the
difference we see in the degree distribution arises purely due to statistical fluctuations in the
number of times that each node is picked. Over $T$ rounds the number of times that any node
$i$ is picked $X_i \sim \text{Binomial}(T, \frac{1}{n})$ is a binomial random variable with probability $\frac{1}{n}$ and $T$ trials.
However, even if a node is picked there is 0.5 probability of obtaining a link. Thus, the degree distribution of the network should be approximately Binomial with $T$ trials and probability $\frac{1}{2n}$.

- when $P\text{--CLAIM}$ is 0.3, there are two ways to be origin either you were picked at random as the original source or you receive the gossip and with some probability you are the last person to claim origin. Since, each node that receives the gossip has constant probability to transmit the gossip to a random neighbor, the total number of individuals that are going to receive the gossip is distributed like a geometric random variable, with mean $\frac{1}{p} = 20$ where $p$ is the probability of failing to spread the gossip. The mode of this distribution is $\left\lceil -\frac{1}{\log_2(1-p)} \right\rceil = 14$, which is the most likely length of the path. Thus, the number of times that a node will be able to claim source is roughly proportional to the number of nodes that it are within distance 15 of it. This induces a rich-get-richer phenomenon, as each time a new link is added between a node and the “origin”, not only the node itself is coming closer to the origin but all the nodes that were close to the node are now close to the origin as well. Thus, for each link added the probability of being in a path from the origin in the future increases and thus it is more likely to receive new links in the future. This is why we see that the degree distribution has much heavier tails in the second case and resembles a power law.
(b) Modifying the NetLogo file such that we add links using:
ask n1 [create-link-with n2]
and remove links using:
ask new-link [die]
We will observe that the values of $c$ that produce a non-empty and non-complete stable graph lie between $(0.25, 0.50)$. A network in this intermediate range might look like this:

4 Games and networks [35 points Sameep, Nihit]

You are playing a networked game. Download the Netlogo file: http://web.stanford.edu/class/cs224w/NetLogo/GameCompetitionShell.nlogo.
The nodes of the network are turtles of different breeds (cooperators, defectors, randos and awesomes) living in the same world. The rules are as follows: two turtles will play a game if and only if they have bidirectional links to one another. When playing a game, a turtle decides which action it would like to use: S or T. The payoff matrix given on the Netlogo interface decides the gain/loss from the game based on the selected actions. You can check out the ‘Info’ tab of Netlogo model provided above to get more information. All turtles of same breed use same strategy to play the game. The various breeds and their strategies are:

- cooperators: Always select S
- defectors: Always select T
• randos: Select S or T at random
• awesomes: You need to implement this

For this assignment we will consider the prisoner’s dilemma payoff matrix (which you can obtain by clicking on the Prisoner’s dilemma button in the GUI). Your job is to implement the strategy for awesomes breed. For this, you need to have a strategy to decide: (i) which turtles to link to and (ii) which action to use when playing a game. Awesomes should be able to outperform the cooperators, defectors, and randos. Check out the existing strategies of the cooperators etc. and run the model a few times to get a sense of how the game works. (Search for ‘TODO’s in the code).

Note: Your awesome agents should not be accessing other turtle’s breeds (cooperator, defector, etc.), strategies, nor any play history with the exception of the last play they had with you.

You should submit the following on Gradescope:

(a) Code for your agent (ONLY the relevant functions you modified)
(b) A screenshot of the cumulative scores at \( t \sim 300 \). There is already a plot in the GUI that will show this.
(c) A screenshot of the network at this same point \( t \sim 300 \).
(d) An explanation in 1-4 sentences of your agents’ strategy.

Your agents should do better than any other by \( t \sim 300 \) in order to receive full credit on this part of the assignment. In addition, submit your code at http://snap.stanford.edu/submit/

★ SOLUTION:
Our awesome agents outperform other agents.

Our awesomes find a way to cooperate with each other and with cooperators while dodging the randos and defectors.

(d) This agent’s strategy was to not play with agents who had defected in the last interaction. It was able to quickly accumulate points by reciprocating incoming links, and therefore forming game-playing links earlier. As you can see in the plots above, the awesomes (little yellow men) in the long run are playing with the cooperators and with each other and racking up points. Also, eventually the cooperators also only play with other cooperating agents, and so the scores start to track exactly. Other potential strategies include defecting immediately before the last round to take advantage of unsuspecting cooperators.