Network Analysis and Modeling
CSCI 5352, Fall 2019
Prof. Dan Larremore
Problem Set 5, due 11/6

1. (100 pts total) *Minimum Violations Rankings.* Here, you will implement the minimum violations algorithm and explore rankings in directed random networks.

   (a) (60 pts) *Coding up and testing the MVR algorithm.* Recall the MVR algorithm that we discussed in class: let π be an ordering of the network's n vertices, such that π_i is the ranking of vertex i, and using the convention that n is the best ranking and 1 is the worst ranking. Let $V(\pi)$ be the number of directed edges that violate the ordering by pointing from a lower ranked vertex to a higher ranked vertex. In other words, in a network where A_{ij} denotes an edge from i to j,

   $$V(\pi) = \sum_{ij} A_{ij} I_{\pi_i < \pi_j},$$

   where $I_{\pi_i<\pi_j}$ is an indicator function taking on the value 1 when $\pi_i < \pi_j$, and 0 otherwise.

   The algorithm begins with a random assignment of vertices to ranks. Then, at each step t, two nodes are chosen uniformly at random from the current ranking $\pi^{(t)}$ and we *propose* to swap them to create a new ordering $\pi^{(t+1)}$. If $V(\pi^{(t+1)}) \leq V(\pi^{(t)})$, the proposal is accepted, and we keep the swap and call the new ordering $\pi^{(t+1)}$. Otherwise, we reject the swap and let $\pi^{(t+1)} = \pi^{(t)}$. As a stopping condition, let the algorithm end only when $\binom{n}{2}$ proposals in a row have passed (accepted or unaccepted) with no decrease in V.

   After your algorithm is implemented, confirm that it works as you expect by testing two cases: (i) a directed chain of $n = 50$ vertices such that $A_{i,i+1} = 1$ for $i = 1, 2, \ldots, 49$. (ii) a $G(n = 50, p = 0.15)$ random network in which edges are made directed by forcing each undirected edge between i and j to point from i to j when $i < j$ and from j to i when $j < i$.

   First, write down what you expect the final ranking π of the algorithm to be, for each of the two cases.

   Second, run your algorithm on each of the two network types (starting from random initial π) 50 times. For case (ii), regenerate a new random network and new random initial π each time. For each of these two cases, produce two plots. Plot 1: time-series plot of the number of violations $V(t)$ vs the number of timesteps t for each repetition of the algorithm. In other words, there should be 50 lines. Plot 2: a histogram of the total number of timesteps elapsed when the algorithm stops.

   Please don't forget to include your code as an appendix to this homework file.

   (b) (40 pts) *Misled! Rankings in randomness!* Here, you will generate random directed networks, in which we expect no meaningful linear hierarchy to exist. However, just

like modularity maximization, we can always minimize violations, *even if the resulting communities or hierarchy are simply fluctuations in randomness.* You will explore how the parameters of a directed $G(n, p)$ network affect the apparent strength of hierarchies that the MVR algorithm finds.

First, write some code to create *directed* random networks. Let us place an edge from i to j independently for all *ordered* pairs of i and j with probability p. (This means that we will create $i \to j$ with probability p, and then independently make $j \to i$ with probability p.) Let us also prohibit self loops. Note that this method for forming a network is different from the method in the previous question. There, we created an undirected random network and then made it directed. Here, we create a directed network from the beginning.

Conduct an exploration of the effects of n and p on the expected number of violations found by the MVR algorithm when applied to directed $G(n, p)$ networks. Write a brief one-page summary of this exploration by telling me (1) how you attacked the problem, (2) what you found, (3) and what you expect to happen if we dramatically increase the size of the network or the probability of connection p. You may include any figures or mathematics that you wish!

This exploration is intended to be open-ended! Don't forget to repeat any numerical experiments a reasonable number of times so that your expected values of V as a function of n and p are not too noisy.

2. (20 pts extra credit) Using your *SI* simulation from Problem Set 4, construct a new centrality measure, which we will call *spreading centrality*. We define the spreading centrality s_i of a node i to be the average size of a cascade (number of infected nodes when the epidemic is complete) that is seeded at i and when the transmission probability is $1/c$, for the network's mean degree c. Thus, the bigger the average cascade that a node i tends to produce, the more important it is under this measure.

   Choose two moderately-sized networks ($n > 500$) from the *Index of Complex Networks* and numerically calculate s for all vertices in each network. Produce a table listing the names of the top 10 nodes, ordered by their spreading centrality, and report their centrality scores and their degree, for each network. Briefly comment on what you discover, with respect to the two networks you chose.

3. (10 pts extra credit) Suppose that there exists a directed and unweighted adjacency matrix A with n vertices. Suppose further that we have used SpringRank to compute the ranks $\hat{s}$ by minimizing the SpringRank Hamiltonian,

$$\hat{s} = \arg\min_s \left\{ \frac{1}{2} \sum_{ij} A_{ij} \mu (s_i - s_j - \ell)^2 \right\} \tag{1}$$

   Now, suppose that a new node t appears. Suppose further that t has neighbors in a set $\mathcal{N}_t$, and that t has out-degree of $|\mathcal{N}_t|$ and therefore t also has in-degree of 0. Given existing estimates of the ranks, $\hat{s}$, which are assumed to be fixed, where should we place the new node t in the embedding space? (Hint: imagine t to be a newcomer to a league in which we already

have SpringRank-derived rankings; if t wins all her games, and her opponents' rankings are given by $\hat{s}$, can you derive the SpringRank $\hat{s}_t$ of our new competitor t?) Your answer should be expressed in terms of existing ranks, i.e. elements of $\hat{s}$ and other variables above.