Zun Wang

1. (1 point) Metapopulation Model

We will model the SIR metapopulation model briefly introduced in Lecture 4 (Slide 5-6).

Let there be M regions with total populations $\{N_i\}_{i=1}^M$. We let $S_i(t), I_i(t), R_i(t)$ be the total population in respective component for region i at time t.

Assume that we σ_{ij} be the population flow from region i to j. Also, we assume the parameters β, γ are same across all regions.

Q 1.1 (6 points) Similar to Lecture 4 Slide 5, write down the equations to derive effective population sizes $S_i^{eff}(t), I_i^{eff}(t), R_i^{eff}(t)$ at time t from $\{S_j(t), R_j(t), I_j(t)\}_{j=1}^M$ and flow parameters σ_{ij}.

population S in j — flow from j to i — population S in i — flow from i to j — total population in j — total population in i — inflow — outflow

$$S_i^{eff}(t) = S_i(t) + \sum_j S_j(t)\frac{\sigma_{ji}}{N_j} - \sum_i S_i(t)\frac{\sigma_{ij}}{N_i}$$

$$I_i^{eff}(t) = I_i(t) + \sum_j I_j(t)\frac{\sigma_{ji}}{N_j} - \sum_i I_i(t)\frac{\sigma_{ij}}{N_i}$$

$$R^{eff}(t) = R_i(t) + \sum_j R_j(t)\frac{\sigma_{ji}}{N_j} - \sum_i R_i(t)\frac{\sigma_{ij}}{N_i}$$

Q 1.2 (6 points) Instead of 1 time-step, derive equations for change in Δt time; i.e., derive the formulas for $S_i(t+\Delta t) - S_i(t), I_i(t+\Delta t) - I_i(t), R_i(t+\Delta t) - R_i(t)$. *Hint:* This is similar in nature to Q1.2 in HW 1.

Since $S_i(t+1) - S_i(t) = S_i(t) - \beta S_i^{eff}(t)\sum_{j=1}^M \frac{I_j^{eff}(t)}{N_j} - S_i(t)$

$= -\beta S_i^{eff}(t)\sum_{j=1}^M \frac{I_j^{eff}(t)}{N_j}$

as for $S_i(t+\delta t) - S_i(t) = -\beta \cdot \delta t \cdot S_i^{eff}(t)\sum_{j=1}^M \frac{I_j^{eff}(t)}{N_j}$

Same for $I_i(t+\delta t) - I_i(t) = \beta \cdot \delta t \cdot S_i^{eff}(t)\sum_{j=1}^M \frac{I_j^{eff}(t)}{N_j} - \gamma\, \delta t \cdot I_i^{eff}(t)$

$R_i(t+\delta t) - R_i(t) = \gamma \cdot \delta t \cdot I_i^{eff}(t)$

Q 1.3 (3 points) Set $\Delta t \to 0$ and derive the ODE equations for the metapopulation model.

Divide δt on both side

$$\frac{S_i(t+\delta t) - S_i(t)}{\delta t} = -\beta S_i^{eff}(t)\sum_{j=1}^M \frac{I_j^{eff}(t)}{N_j} \qquad \frac{I_i(t+\delta t) - I_i(t)}{\delta t} = \beta S_i^{eff}(t)\sum_{j=1}^M \frac{I_j^{eff}(t)}{N_j} - \gamma I_i^{eff}(t) \qquad \frac{R_i(t+\delta t) - R_i(t)}{\delta t} = \gamma I_i^{eff}(t)$$

$\delta t \to 0$

$$\frac{dS}{dt} = -\beta S_i^{eff}\sum_{j=1}^M \frac{I_j^{eff}}{N_j} \qquad \frac{dI}{dt} = \beta S_i^{eff}\sum_{j=1}^M \frac{I_j^{eff}}{N_j} - \gamma I_i^{eff} \qquad \frac{dR}{dt} = \gamma I_i^{eff}$$

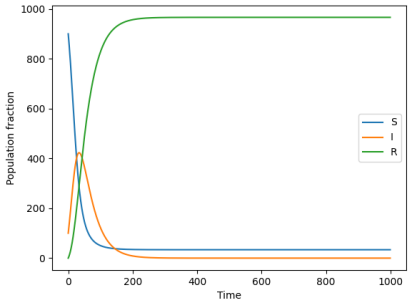
Q 1.4 (10 points) **[Bonus]** Implement the ODE of metapopulation model. The code should take the parameters β, γ, the flow parameters $\{\sigma_{i,j}\}_{i,j\in[1,M]\times[1,M]}$, total population $\{N_i\}_{i=1}^{M}$ and the initial population of each compartment for all regions $\{S_i(0), I_i(0), R_i(0)\}_{i=1}^{M}$ and `max_time` and output the population sizes of $\{S_i(t), I_i(t), R_i(t)\}_{i=1}^{M}$ till `max_time`.

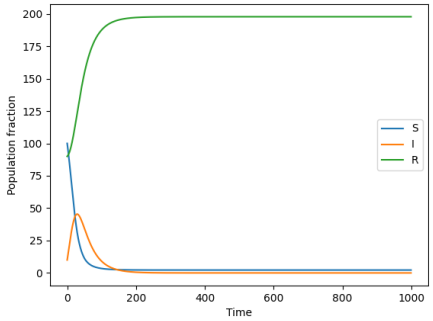
We have provided a starter code in python script `metapop.py` that implements the model. You have to fill in the missing parts of the method `def ode(self, times, init, parms):` that involves calculating the effective population.

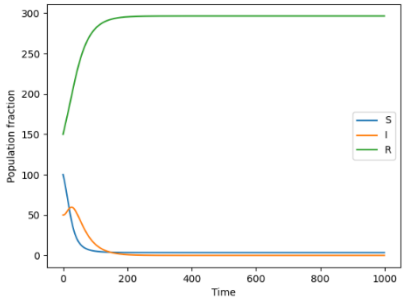
Once you have completed the method, run the script to generate the SIR plots for each population. Submit the completed code and the plots.

Note: You don't need to change other functions in the script, only fill in the code in the space indicated.

Note: In case you are not comfortable with python, you may translate the python script into your language of choice and complete the implementation of metapopulation model.







code in q1/metapop.py , plots in q1/plots/

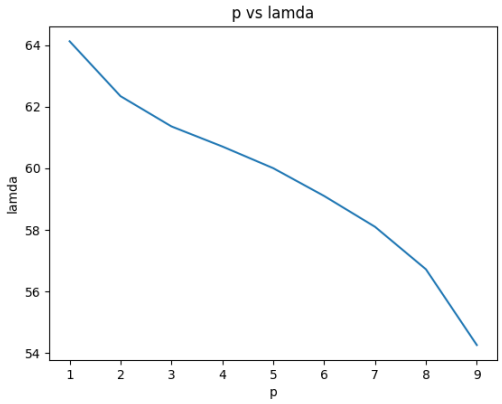
2. (36 points) Viral propagation You are managing a network of devices inside your company connected to each other via a local intranet connection. Occasionally, some of the computers in the network get infected with a virus which can spread across the network as users share information with each other. You job is to simulate the spread of the virus under different assumptions on the diffusion mechanism.

Network dataset: Your dataset contains multiple networks $\mathcal{G} = \{G_1, G_2, \ldots, G_9\}$ between routers (this dataset is sourced from actual internet router dataset [1]. Each of the networks are snapshots of the connections made across routers on some day collected over 3 months. The network files are named as `network1.txt, network2.txt,..., network9.txt`.

Aggregating all networks: We aggregate all the 9 snapshots of the networks into a single network $G(\rho)$. The aggregation process is simple: For each pair of nodes (u, v) if at least ρ networks in $\mathcal{G}$ have an edge between them, we add an edge (u, v) to $G(\rho)$. Here ρ is a parameter of our model.

We also assume an SIS network model with usual parameters β, γ on the graph $G(\rho)$.

Q 2.1 (10 points) Decreasing ρ will increase the number of edges in the graph and make it more denser and connected. In class we saw that λ, the highest eigenvalue of the

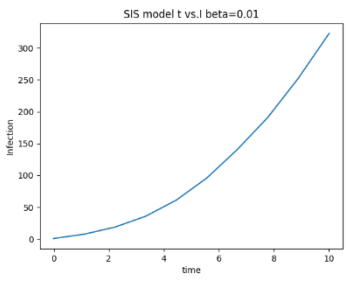


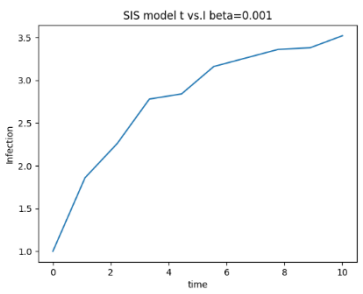
q2 code in q2/question2.ipynb

plots under q2/

Q 2.2 (10 points) Set $\rho = 1$. Use the EoN package [2] to set up an SIS model on $G(\rho)$. Specifically use the `EoN.fast_SIS` function [3]. Set $\gamma = 0.08, \beta = 0.01$ and `max_time` $T = 10$. Set node 1 as the only infected node at $t = 0$.

Now plot a curve with $I(t)$ on y-axis vs t on x-axis for $0 \leq t \leq T$ after simulating the model 50 times (similar to HW1 Q1.3). Now set $\beta = 0.001$ and plot the $I(t)$ vs t curve. Do you notice any difference? Explain.



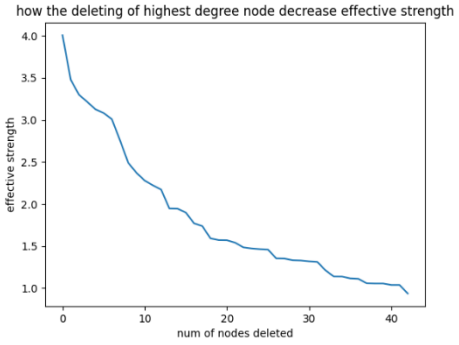


Since the infection rate is severely decreased, the infection count through time 10 also severely decreased, and when beta = 0.001, the shape of curve fluctuate more (instead of a steady exponential curve as in beta = 0.01), and it is because the tiny beta will increase the unpredictability.

Q 2.3 (10 points) Now set $\beta = 0.01, \gamma = 0.16$ and $\rho = 1$. We will try to reduce the effective strength of the graph by another method of iteratively removing nodes using the following rule:

Find the node with the largest degree among all nodes (in case of a tie, choose the node with the highest index). Then remove the node.

Using this rule find the number of node removals required to reduce the effective strength of the graph to less than 1. Also produce a plot of number of nodes removed in x-axis and effective strength on y-axis till effective strength reduces to less than 1.

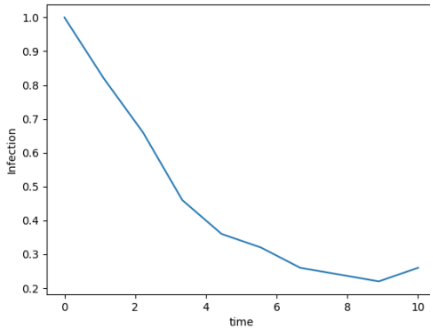


42 required for removal

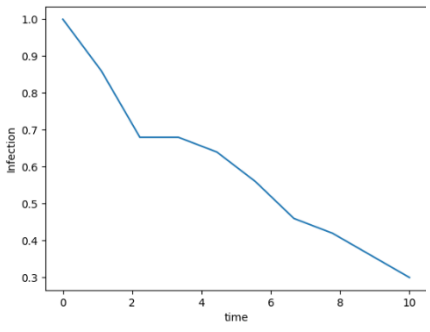
Q 2.4 (6 points) **[Bonus]** We will denote as K the number of nodes you had to remove in Q2.3 before effective strength decreased below 1. Let $[G_0, G_1, G_2, \ldots, G_K]$ be the sequence of graphs after each removal of a node from Q 2.3 where $G_0 = G(\rho = 1)$. The effective strength of all G_i for $i < K$ is greater than 1 and G_K has effective strength ≤ 1.

For each graph G_K, G_{K-1} and G_{K-2} submit a plot of $I(t)$ vs t (3 plots in total).

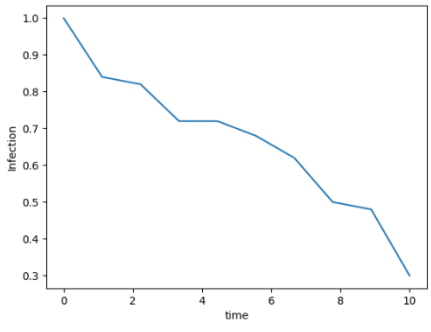
G_{k-2}



G_{k-1}



G_k



3. (30 points) Finding Culprits

Codes in q3/question3.ipynb

Output txt in q3/3.1out.txt and q3/3.3out.txt

Q3.2: 90% Q3.3: 44%

Q 3.4 (4 points) Does the accuracy of the estimator change (comparing the answer you get in Q3.2 and Q3.3)? Show your result and give the answer. If so, what does it tell you about the estimator on this type of graph (strengths and weaknesses; this is slightly an open-ended question). Do you think the rumor centrality metric we studied in class will probably perform better?

The accuracy changed a lot. When the first node is 0, the accuracy is around 88%, but when the first node is 1, the accuracy will decrease to around 45%. It tells me that the estimator tends to choose the center of the graph (0 in this case). The rumor centrality metric mentioned in the class does not count for such bias and will probably perform better in this graph.

4. (19 points) Steiner trees for Missing Infections

In this question we will explore the use of Steiner trees for finding missing infections. Steiner trees are classic objects used for many 'facility location' problems. As we know many infections go unrecorded in real-life, and it is helpful to understand which other nodes may be infected but not recorded (the missing nodes). In class we saw other types of methods for this problem. Here we will use the idea of minimum Steiner tree over the known infected nodes as a way to approximate the unknown infected nodes in the graph.

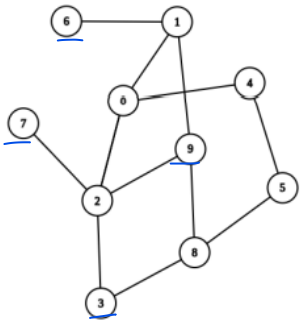
if $G = (V, E)$ is our undirected social network with edge weights, and a set of terminals $I \subseteq V$, then a Steiner tree is a tree in G that spans I (i.e., contains at least all nodes in I; it might contain extra nodes from V not in I too, but at least it has to span all of I).

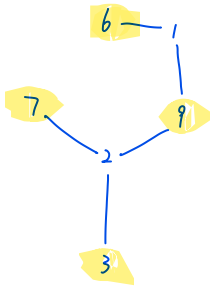
The minimum weighted Steiner tree, is the Steiner tree for a given I that has the least sum of the weights of all the edges included in it. There are several algorithms for finding such a Steiner tree given the set of terminals I and graph G. See this link for more details on the Steiner tree problem `https://en.wikipedia.org/wiki/Steiner_tree_problem`.

For the following question, assume that all the edges have equal weights.

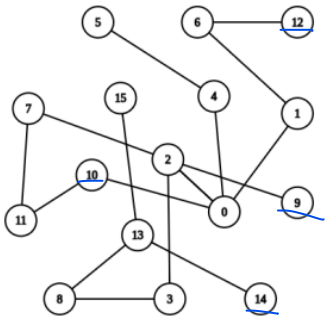
Q 4.1 (8 points) Let's try to compute a couple of minimum Steiner trees by hand. Feel free to use whatever method you want (except of course just using a software package). These can be done with intuition as well. Just draw and show us the final answer.

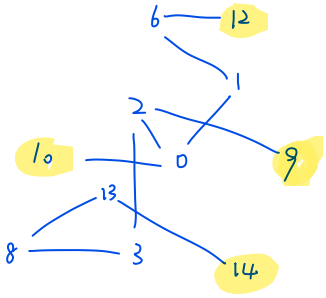
Q 4.1.1 (4 points) Terminal set $= \{7, 9, 3, 6\}$ for graph below.





Q 4.1.2 (4 points) Terminal set = $\{12, 14, 9, 10\}$ for graph below.





Q 4.2 (9 points) In the missing infections problem you are given a set of observed infections (the *known* set) K. For using Steiner trees to compute missing infections, the idea is very simple. Set K as the terminal nodes set I in the minimum weighted Steiner tree problem over the graph G. The resulting minimum weighted Steiner tree is a good rough approximation of how the disease might have spread. In particular the *extra* nodes you find in your min. Steiner tree are your missing nodes.

First generage the graph G as using the following function [5]:

`G = networkx.random_graphs.extended_barabasi_albert_graph(50, 1, 0.2, 0.1, seed=10)`.

Obtain an approximate minimum Steiner tree using this existing Networkx function [6].

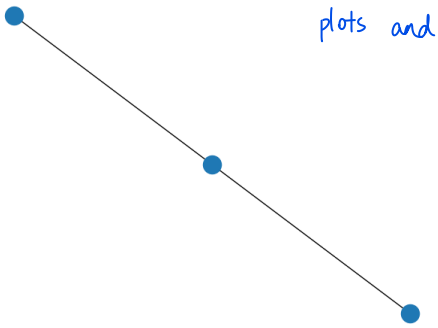
Submit both a visualization and the adjacency list of the minimum Steiner tree for each of the following initial known infected nodes.

1. $K = \{10, 14, 3\}$
2. $K = \{10, 14, 3, 4, 2\}$
3. $K = \{10, 14, 3, 4, 2, 31, 49\}$
4. $K = \{10, 14, 3, 4, 2, 31, 49, 21, 25, 36, 43\}$

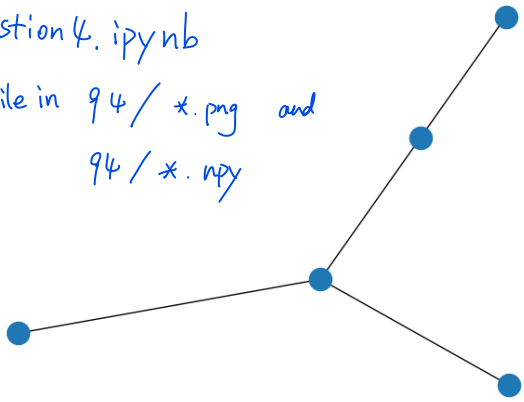
Plot the visualization in the pdf and submit the adjacency file as a .npy 2-D numpy array file.

code in 94/question4.ipynb

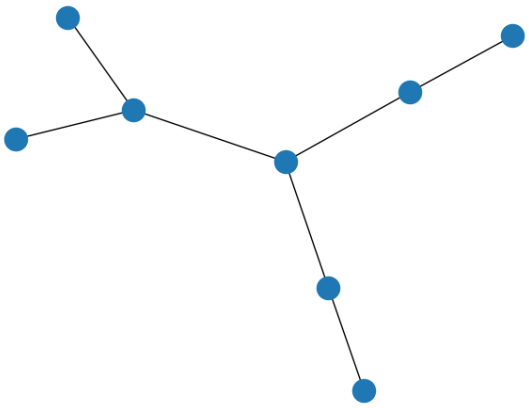
plots and adjacency file in 94/*.png and 94/*.npy



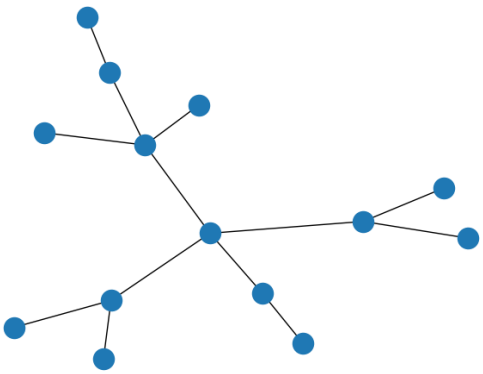
K1



K2



K3



K4

Q 4.3 (2 points) Briefly in 1-2 lines, what maybe the pros and cons in using a min. Steiner tree to identify missing infections? Suggest improvements for the same. This is an open-ended question so feel free to give intuitive answers.

Pros: Given the nature of this algorithm, it can be applied to many cases/ different kinds of data and it does compute very fast

Cons: It will bias towards the nodes with higher degree, and it will affect the output severely depending on how the original graph is constructed and it may shift a lot from the reality data.