Aalto University, School of Science
CS-E5740 Complex Networks, Fall 2022
Saramäki, Hiraoka
Course project

Course project (52 pts)

- The deadline for handing in your solutions is December 20th 2022 20:00.
- Return your solutions (one `.pdf` file and one `.zip` file containing Python code) in MyCourses (Assignments tab). Additionally, submit your pdf file also to the Turnitin plagiarism checker in MyCourses.
- Check also the course practicalities page in MyCourses for more details on writing your report.

- Note that you need to get at least 25 points from the project in order to pass the course!
- You can use the Jupyter notebook available in Jupyterhub to get started. **Using the notebook is fully optional.**
- Discussing project solutions with other students is allowed when properly indicated in the report. Each student should nevertheless return their own report.

Preface

The topic of this project work that we have been running for a long time has, unfortunately, become all too real in the recent years. This shows that pandemics are not entirely unexpected, and the way how they spread through the globe has also been foreseen for a long time. In fact, if one would use global air transport data, a more complex disease model (SEIR), and the real geographic origin of spread, one would get a timeline that eerily matches early 2020.

Introduction

In the project work, your task is to implement a Susceptible-Infected (SI) disease spreading model and run it on top of a temporal network from air transport data, containing information on the departure and arrival times of flights. You will study the dynamics of spreading and how it depends on where the process starts as well as the infectivity of the disease, and use static-network centrality measures to understand the roles that specific nodes play.

Model specifications

In the SI model, each node is either Susceptible (S) or Infected (I). When an Infected node is in contact with a Susceptible node, the Susceptible node may become infected with some probability $p \in [0, 1]$, reflecting the infectivity of the disease. Infected nodes remain Infected forever.

In our model that mimics the spreading of disease through the air transport network, nodes are airports and time-stamped connections are flights between them. Initially, only one airport node (called *the seed* node) is set to the Infected state, while all other airports are Susceptible. Now, following the SI process, a flight from an Infected source airport infects its Susceptible destination airport with probability $p \in [0, 1]$. Note that a flight can carry the infection only if its source airport is infected at the time of the flight's departure! Infected airports remain infected for the rest of the simulation.

Data description

The data that are used in the project are available at the course MyCourses page in one `.zip` file as well as at `/coursedata/project` in Jupyterhub. The flight data that you will use to perform your simulation are located in the file `events_US_air_traffic_GMT.txt`, where each row contains the following fields:

```
1st column -> Source [0-278]
2nd column -> Destination [0-278]
3rd column -> Start Time (GMT) [seconds after Unix epoch time]
4th column -> End Time (GMT)
5th column -> Duration [Same as (EndTime-StartTime)]
```

The aggregated weighted network `aggregated_US_air_traffic_network_undir.edg` is constructed based on the event data, so that the weight of each link corresponds to the number of flights between the nodes. This network is static and undirected. Additionally, you can find the information about the airports in file `US_airport_id_info.csv`. `US_air_bg.png` is an image of the USA map used as a background image in some visualizations.

General hints on implementing the model:

- *Please familiarize yourself with all tasks before you begin coding*—this helps you to write your code so that the basic implementation works for all tasks, rather than writing code for Task 1 and only later noticing that Task 2 needs something else.
- What you want to get is a) the times of infection of each node, and b) the timeline of the number of infected, for each run.
- The fact that flights have a duration complicates things a bit. It helps if you use an np.array to keep a record of the infection times; the ith element of this array should always give the infection time of node i. Initialize the infection time of the seed node to the beginning of the simulation (that is, the departure time of the first flight of the data) and the infection time of all other nodes to `np.float('inf')`. Then, whenever a node becomes infected, set its infection time to the arrival time of the flight that infected it.
- The most transparent and straightforward way to implement the model is to first sort all flights in increasing order of departure time, and then go through them in sequence. Is the source node infected at the departure time of the current flight (i.e. is `infection_times[source]` smaller than or equal to the departure time)? If it is, there are two options: a) the target node is susceptible (if `infection_times[node]` is `np.float('inf')`), in which case you should infect the target node with probability p and update its infection time to the flight's arrival time, and b) the target node is already infected—if so, check if the current flight's arrival time is smaller than the node's infection time in the array, and if, update it to the arrival time with probability p. In the later case, the infection time of the target node has been due to an earlier-departing, longer flight, while the current flight departs later but is also shorter, thus reaching the target airport earlier.

- Pro tip: Build your implementation on two functions:
  - A function that perform the SI simulation, taking in the seed node id and p and returning the array of infection times.
  - A function that takes as input the infection times and returns the prevalence (see below). This function should read in the desired time intervals (see Task 2) for the prevalence curves, count the fraction of infected nodes at those time points from the infection times array, and return these fractions as a list.

  Take time to implement these two functions carefully; you can then use them for all tasks.
- For reading in the csv data, there are many options. The Jupyter notebook makes use of `read_csv` function from the `pandas` package. This function takes as input the file path and multiple optional arguments and returns the file contents as `pandas.DataFrame` object. For full documentation, check `https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html?highlight=read_csv#pandas.read_csv` (note that Aalto Jupyterhub runs `pandas` version 1.1.5).
  Other options for reading data include e.g. `numpy.genfromtxt`, or the built-in `csv` library in the Python standard library.
- Probably the fastest way to loop over flights is to first transfer the flights into an np.array and then iterate over rows (each row corresponds to a flight). Alternatively you can use the `iloc` function from `pandas`.
- Feel free to use any functions from `networkx`, `matplotlib`, `numpy` and `scipy`.

Tasks:

Task 1: Basic implementation (6 pts)

Implement the SI model using the temporal air traffic data in `events_US_air_traffic_GMT.txt`. Use the provided visualization module to check that your implementation works reasonably. Assume first that $p = 1$, *i.e.,* the disease is always transmitted.

a) (6 pts) If Salt Lake City (node-id=27) is infected at the beginning of the data set, at which time does Anchorage (ANC, node-id=41) become infected?

   Hint: The time point should fall within the range `1229283000-1229284000`.

Task 2: Effect of infection probability p on spreading speed (6 pts)

Run the SI model 10 times with each of the infection probabilities [0.01, 0.05, 0.1, 0.5, 1.0]. Again, let Salt Lake City (node-id 27) be the initially infected node. Record all infection times of the nodes, and answer the following questions:

a) (5 pts) Plot the averaged prevalence $\rho(t)$ of the disease (fraction of infected nodes) as a function of time for each of the infection probabilities. Plot the 5 curves in one graph. You should be able to spot stepwise, nearly periodic plateaus in the curves.

   Hints:

  - For averaging, things are easier if you divide the whole time span (from the first departure to the last arrival) into equal-sized steps using *e.g.* `np.linspace`).
  - For each of the 10 iterations, count the number of nodes infected at each time step (that is, nodes whose infection time is smaller than or equal to the time step) and normalize by the total number of nodes (see the *pro tip* above in general hints). Then, average over these values over the 10 iterations.

b) (1 pt) For which infection probabilities does the whole network become fully infected? What are the periodic "steps" in the curves due to?

Task 3: Effect of seed node selection on spreading speed (6 pts)

Next, we will investigate how the selection of the initially infected seed node affects the spreading speed.

a) (4 pts) Use nodes with node-ids [5, 38, 74, 134, 143] (DTW, SMF, VPS, SYR, ILM) as seeds and $p = 0.1$, and run the simulation 10 times for each seed node. Then, plot the average prevalence of the disease separately for each seed node as a function of time (recycling your code for Task 2).

b) (1 pt) The differences in spreading speed between seeds should be mostly visible in the beginning of the epidemic. Explain, why.

c) (1 pts) Why is it important to average the results over different seeds? What kind of problems could follow from using only a single seed, for example in the next task where we'll inspect the vulnerability of a node for becoming infected with respect to various network centrality measures? (A general-level answer is enough, no need to discuss e.g. any particular centrality measure in detail.)

Task 4: Where to hide? (10 pts)

Now, consider that you'd like to be as safe from the epidemic as possible. How should you select your refuge? To answer this question, run your SI model 50 times with $p = 0.5$ using different random nodes as seeds and record the *median* infection times for each node.

In this task, you can use the pre-built static network (`aggregated_US_air_traffic_network_undir.edg`) to compute the various centrality measures using the ready-made NetworkX functions.

a) (5 pts) Run the 50 simulations, and create scatter plots showing the median infection time of each node as a function of the following nodal network measures:

   i) *unweighted* clustering coefficient c
   ii) degree k
   iii) strength s
   iv) *unweighted* betweenness centrality

   Hint: If a node gets infected in more than half of the iterations, its median infection time is Inf. For visualization and correlation calculations (below), you need to replace these values with some large number, e.g. the largest observed non-Inf median infection time.

b) (2 pt) Use the Spearman rank-correlation coefficient [1] for finding out which of the measures is the best predictor for the infection times[1].

Hint: `scipy.stats.spearmanr`

c) (3 pts) Based on your results, answer the following questions:

- (1 pt) Which measure(s) would you use to pick the place to hide, i.e. which measure best predicts node infection time? Why are these measures good in predicting the infection time?
- (1 pt) Why does betweenness centrality behave differently than degree and strength?
- (1 pt) Why is clustering coefficient a poor predictor?

In your answers, explain the performance of different measures based on how they are defined (obviously, a measure is a good predictor if it has high correlation with the infection time but this is not the answer we are looking for). However, you don't need to use any mathematical notation if you don't want to – a verbal answer will do equally well.

Task 5: Shutting down airports (12 pts)

Now take the role of a government official considering shutting down airports to prevent the disease from spreading to the whole country. In our simulations, shutting down airports corresponds to immunization: an airport that has been shut down cannot become infected at any point of the simulation and therefore cannot transmit the disease.

One immunization strategy suggested for use in social networks is to pick a random node and immunize a random neighbour of this node. Your task is now to compare this strategy against five other immunization strategies: the immunization of random nodes and the immunization of nodes with the largest values of the four node properties calculated in task 4. In this exercise, use $p = 0.5$ and average your results over 20 runs of the model for each immunization strategy (120 simulations in total).

To reduce the variance due to the selection of seed nodes, use same seed nodes for investigating all immunization strategies. To this end, first select your immunized nodes, and then select 20 random seed nodes such that none of them belongs to the group of immunized nodes in any of the 6 different strategies.

a) (6 pts) Adapt your code to enable immunization of nodes, and plot the prevalence of the disease as a function of time for the 6 different immunization strategies (social net., random node, and 4 nodal network measures), always immunizing 10 nodes.

Hint: Add a list (or set) of immunized nodes as an optional input for your function for running the simulation, with an empty list (or set) as the default value. Then, when about to infect a node, check if the node is in the list/set, and if, do not infect.

b) (2 pts) Based on your results, answer the following questions:

- (1 pt) Which immunization strategy performs the best, and why?
- (1 pt) Why does betweenness centrality perform better as an immunization strategy than as a predictor for a safe hiding place?

[1] We use Spearman rank-correlation coefficient instead of the linear Pearson's coefficient, as we can not assume the dependency between the average infection time and different nodal network measures to be linear.

In your answers, explain the performance of different measures based on how they are defined.

c) (3 pts) The pick-a-neighbour immunization strategy probably worked better than the random node immunization[2]. Let us try to understand why.

- First, if the degree distribution of the network is $P(k)$, what is the probability of picking a random node of degree k?
- What is the expected outcome if you then pick a random neighbour of the random node (hint: see lectures 3 and 4)?
- Consequently, which of the strategies is expected to be more effective and why?

d) (1 pt) Although the social network immunization strategy outperforms the random immunization, it may be less effective as some other immunization strategies. Nevertheless, **explain** shortly, why it still makes sense to use this strategy in the context of social networks?

Task 6: Disease transmitting links (10 pts)

So far we have only analyzed the importance of network nodes—next, we will discuss the role of links. We will do this by recording the number of times that each link transmits the disease from one node to another. To this end, adapt your code for recording the static links which are used to transmit the disease. You can do this e.g. by storing for each node where the infection came from. For example, if node i gets infected by a flight arriving from node j, you should store the information of i being infected by j. This can be done by using a list `trans_nodes`, where `trans_nodes[i] = j`.

Run 20 simulations using random nodes as seeds and $p = 0.5$. For each simulation, record which links are used to infect yet uninfected airports (either by first infection-carrying flights arriving to susceptible airports or by infecting flights arriving before the already recorded infection time).

a) (5 pts) Run the simulations, and compute the fraction of simulations where each link transmitted the disease (f_{ij}). For example, if the disease spread from node a to node b in 5 simulations out of 20, $f_{ab} = 0.2$. Then use the provided function `plot_network_USA` to visualize the network on top of the US map. Adjust the width of the links according to the fractions f_{ij} to better see the overall structure. Compare your visualization with the maximal spanning tree of the network.

b) (1 pt) Explain why your visualization is similar to the maximal spanning tree.

c) (3 pts) Create scatter plots showing f_{ij} as a function of the following link properties:

i) link weight w_{ij}

ii) *unweighted* link betweenness centrality eb_{ij} (`edge_betweenness_centrality` in networkx)

Compute also the Spearman correlation coefficients between f_{ij} and the two link-wise measures.

d) (1 pt) How well do the two link properties predict f_{ij} and why? Explain their performance based on how they are defined.

[2] However, due to the randomness of the selection process there is a small probability that this was not the case for you.

Task 7: Discussion (2 pts)

Even though extremely simplistic, our SI model can readily give some insights on the spreading of epidemics. Nevertheless, the model is far from an accurate real-world estimate for epidemic spreading.

Discuss the deficiencies of the current epidemic model by listing at least four (4) ways how it could be improved to be more realistic.

Hint: Check *e.g.* Ref. [2] for ideas.

Feedback (1 pt)

To earn one bonus point, give feedback on this exercise set and the corresponding lecture latest two days after the report's submission deadline. You can find the feedback form at the Assignments tab in MyCourses.

References

[1] [Online]. Available: http://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

[2] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," *arXiv preprint arXiv:1408.2701*, 2014.