CS/CMS/EE/IDS 144 **Gurus: Daniel & Matthew**

MiniProject 1: Pandemaniac

Assigned: 02/8/2023 Due: 02/22/2023

We encourage you to talk to other people about general strategies, but it's important to keep in mind that using the same exact strategy will end up hurting yourself. Keep a record of what things you tried; it'll make the report write up that much easier.

1 Pandemaniac [100 points]

Welcome to our first mini-project! Here we will crown the "pandemaniac".

This assignment is meant to give you a flavor of one of the "hot" topics in complex networks these days — competing cascades. We talked about cascades in lecture, but in that context we only had one new product which was looking to take over the network. Here, we'll consider a setting with multiple cascades competing. In fact, each group will be responsible for seeding a cascade with the goal of taking over as much of the network as possible, and we'll see which team wins!

Your Task: In groups of 2 or 3 (4 allowable, with correspondingly higher report expectations), create an algorithm that will take as input an undirected graph (in adjacency-list form) and output a list of N seed nodes where your epidemic will start (where N is given). You will be competing against other teams who are also trying to spread their epidemics on the same graph. Your team's goal is to take over more of the network (i.e., a larger number of nodes) with your epidemic than your competitors do with theirs.

The Epidemics:
As you can imagine, the key to this task is understanding the way the epidemic spreads. We'll be using a simple, deterministic model for the spread that is similar to what we studied in class. In particular, each node in the graph can be acquired by at most one team (color). Nodes start uncolored and then may be convinced to adopt a color based on the colors of their neighbors. After they adopt a color, they still may later be convinced to switch their color if their neighbors switch. We describe the process in detail below, and you can download the code to simulate the epidemic process (go to the Piazza page and click on the Resources tab).

Seed nodes

To begin, the entire graph is uncolored at iteration 0. Each team is then allowed to pick a set of nodes – called seed nodes – to acquire in iteration 1. **Note that, if more than one team picks the same node, then none of those teams gets that node!** These seed nodes will then be colored accordingly.

The spread of the epidemics

From the seed nodes, the epidemics spread iteratively. During each iteration, every node chooses its next color through a majority vote among itself and its direct neighbors. All colored direct neighbors of the node vote for their respective color with 1 vote; however, if the node itself is currently colored, it votes for its own color with 1.5 votes. Uncolored nodes do not vote, so only 1 neighbor needs to be colored to convert a node. If any color has a **strict majority** out of the total number of votes, then that becomes the next color for the node.

If there is no strict majority, then the node keeps its current color or remains uncolored. These cases are demonstrated in Figures 1 and 2. Note that it is (slightly) harder to convert a node after it has been acquired,

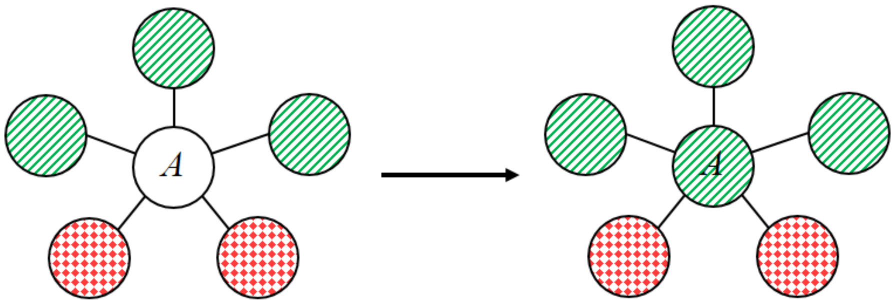

Figure 1: An uncolored node *A* converting to "striped" (since "spotted" gets 2 votes and "striped" gets 3 votes, which is a majority among the 5 votes).

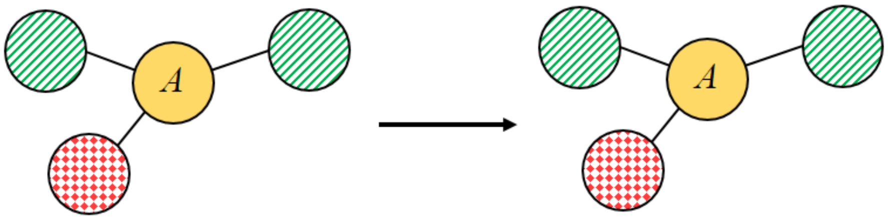

Figure 2: Node *A* doesn't change color because no color got the majority of votes ("striped" gets 2 votes, "dotted" gets 1 vote, "solid" gets 1.5 votes, none of them gets a majority among the 4.5 votes).

but nodes can switch colors multiple times.

Termination

The iteration spread of the epidemics will continue until it stabilizes, i.e. when no nodes of the graph change colors. Note that convergence is not guaranteed; however, cyclic behavior is unlikely. In the unlikely scenario in which the epidemics do not converge, we will terminate the simulation after a random number of iterations greater than 100. This number will be regenerated for every trial, so do not try to plan for some fixed cap on iterations.

The Details:

(a) *Website:* You will submitting your code for each round of this project through a website. **Please provide your team username and members of your team through a private post on Piazza so we can set your account up by Friday 11:59PM February 10th**.

Once you post your team username and list of team members, we will set up your account to give you website access: `http://54.176.230.217/`.

(b) *Graphs & Runs:* We will have a "regular season" during the week of the assignment and then a "tournament" at the end of the week. In particular, there will be several graphs to practice on over a period of 5 days prior to the tournament.

The graphs are named in the following (semantic) way:

competition_format . num_seeds . unique_id

- *competition_format* is a string "RR" or "J" representing the type of competition style (2 players or all players, see below),
- *num_seeds* is an integer representing the number of seed nodes that your strategy must select,
- *unique_id* is an integer that distinguishes between different types of graphs.

You must create a program that will accept as input an undirected graph in adjacency-list form via a JSON file. This file contains a JSON object with (string) keys representing the nodes and list of string values representing the direct neighbors. Note that if a node has no neighbors (albeit unlikely), then it is still present as a key in the JSON object, but with an empty array as a value. You may assume that the nodes of the graph are numeric strings. If there are V nodes, then they are numbered from 0 to $V-1$.

For example, the graph of a square, with vertices labeled 0 through 3 in a clockwise fashion, will be represented by the following:

```
{
  "0": ["1", "3"],
  "1": ["0", "2"],
  "2": ["1", "3"],
  "3": ["0", "2"]
}
```

We have given you example graphs to test your code with.

(c) *Submissions:* When your team is ready, you may download the graph, after which you will have a set amount of time (3-5 minutes, depending on the graph, **strictly enforced TIME LIMIT!**) to run your program and choose a set of seed nodes. Your team will then upload a file containing your list of seed nodes. **Each line of the file must contain only one node**. Since we are running **50 rounds for each graph** in order to reduce the variability of the result, you need to submit [$\textbf{num\_seeds} * \textbf{50}$] seed nodes. Groups are allowed to have different choices in each of the 50 rounds, so this will allow you to play mixed strategies. For example, a submission to a graph named `X.4.10` (i.e. a graph that requires choosing 4 seeds) could be:

$$1^{\text{st}}\text{ round}\begin{cases}0\\1\\2\\3\end{cases}$$

$$2^{\text{nd}}\text{ round}\begin{cases}0\\1\\2\\3\end{cases}$$

$$\vdots$$

$$50^{\text{th}}\text{ round}\begin{cases}0\\1\\2\\3\end{cases}$$

There should be total $4 * 50 = 200$ nodes in the text file. Every 4 nodes in your submission will be used to run the i^{th} round of the game for each graph. A sample submission text file can be found on the Piazza Page under Resources (in the General Resources box). You can make multiple submissions before your time is up. Once the timer expires, we will use your most recent successful submission. We have given you example submissions from last year to go along with the example graphs.

(d) *The Regular Season:* The regular season matches will use 8 graphs every day from **February 12-16 (Sunday-Thursday).** These graphs will change each day. Submissions for each day's graphs will be open from **3:00am - 11:55pm**. The results will be available on the website by 10:00am the next day.

There will be two formats: Round-Robin and "Jungle" Free-For-All. In Round-Robin, each team will get paired against every other team, and scores will track the win-loss record. In the "Jungle", all teams will participate on one graph - see below for scoring.

More specifically, the graphs for each day are:

Day	**ER**	**PA**	**SSBM**	**Caltech**	**SNAP**
Feb 12	`RR.5.10`	`RR.5.20`	`RR.10.30`	`RR.10.40`	`RR.10.50`
Feb 13	`RR.5.11`	`RR.5.21`	`RR.10.31`	`RR.10.41`	`RR.10.51`
Feb 14	`RR.10.12`	`RR.5.22`	`RR.10.32`	`RR.10.42`	`RR.10.52`
Feb 15	`RR.10.13`	`RR.10.23`	`RR.10.33`	`RR.20.43`	`RR.10.53`
Feb 16	`RR.10.14`	`RR.10.24`	`RR.10.34`	`RR.15.44`	`RR.10.54`

Day	SSBM	Caltech	SNAP
Feb 12	`J.5.10`	`J.10.20`	`J.10.30`
Feb 13	`J.5.11`	`J.20.21`	`J.20.31`
Feb 14	`J.10.12`	`J.15.22`	`J.30.32`
Feb 15	`J.10.13`	`J.10.23`	`J.25.33`
Feb 16	`J.15.14`	`J.10.24`	`J.35.34`

The types of graphs are as follows:

- **ER Graphs:** the Erdős–Rényi model, with a value of p between $1/n$ and $\log(n)/n$.
- **PA Graphs:** the Preferential Attachment model
- **SSBM Graphs:** the Symmetric Stochastic Block model
- **Caltech Network Graphs:** Generated by crawling the 6 Caltech Division home pages.
- **SNAP Graphs:** Selected from Stanford Large Network Dataset Collection: `https://snap.stanford.edu/data/`

(e) *TA Graphs:* The TAs will also participate in the regular season and will compete 1-on-1 with each team on each SNAP graph of the round robins.

It is **highly recommended** you participate in the regular season in order to (1) get points for beating the TA teams and (2) practice and develop your strategy against other teams so that you can win the actual competition! TA strategies will be available for you to download and analyze after each day.

(f) *The Tournament:* The tournament will last 3 days, starting from **Friday, February 17th**. For each day, there will be a round-robin for 5 graphs which determines which teams will move on. The top 12 teams will proceed to day 2, and the top 4 will proceed to day 3. Scoring is the sum of win-loss records across the 5 games, and ties will be broken by the head-to-head record between two teams.

Results for each stage of the competition will be announced on the following day. Your team must submit your seed nodes for the competition graphs by **11:55pm** on each competition day. As before, once you download a graph, you will have a limited amount of time to submit your seed nodes for that graph. After you make your last submission, your algorithm (and code) is final and may not be modified.

For the Jungle format, there will be three final graphs on the first day of the tournament. **For each of the rounds** in a game, teams will earn points in the following olympic format:

Place	1st	2nd	3rd	4th	5th	6th	7th	8th	9th and beyond
Points	20	15	12	9	6	4	2	1	0

For each graph, the points across all 50 rounds will be summed. Teams will then receive points for the graph based on rank again according to the table above. The team with the most 'graph points' across the final 3 Jungle graphs will be crowned the Jungle Pandemaniac!

Day	ER	PA	SSBM	Caltech	SNAP
Feb 17	`RR.10.15`	`RR.10.25`	`RR.10.35`	`RR.15.45`	`RR.10.55`
Feb 18	`RR.10.16`	`RR.10.26`	`RR.10.36`	`RR.15.46`	`RR.10.56`
Feb 19	`RR.10.17`	`RR.10.27`	`RR.10.37`	`RR.10.47`	`RR.10.57`

Day	SSBM	Caltech	SNAP
Feb 17	`J.15.15`	`J.25.25`	`J.20.35`

Grading:

- **Report [65 points]:** The most important part of this assignment is the thought that goes into your algorithm. So, the bulk of your grade comes from a report detailing your design process. Specifically, your team must submit one detailed report describing your strategy and the reasoning behind it. It must also describe the contributions of each team member as well as citations to any papers or other resources that helped in the formulation of your strategy.

  A good report will show that you put significant thought into whether and how you could use different centrality measures, clustering measures, and game theoretic principles. We also ask that you include suggestions on how we could change/improve it for next year! **This report, along with your final code, is due 5:00 pm, Wednesday, February 22.** The code should be available on a private github repository. Please add the TAs with usernames *d-nee* (Daniel) and *matthewhajjar* (Matthew) as collaborators. You should submit the report through Gradescope and add all members of the group to the submission.

- **Start early! [5 points]:** Participate in the first round on **February 12**.

- **Beat the TAs [25 points]:** The TAs will participate in the regular season every day and will compete in the round-robin SNAP graphs. You only have to beat each TA team once to get credit.
  - Beat the TA_baseline. (**15 Points**)
  - Beat the TA_target. (**10 Points**)
  - Beat the TA_hard. (**Optional, 2 Bonus Points Per Day Beaten**)

- **Play in the tournament! [5 points]:** Participate in the first round of the tournament.

- **Class leaderboard [Bragging Rights]:** At the end of the project the team with the best score will be crowned "Pandemaniacs", and be listed in perpetuum on the course website.

Hints and Notes:

- Feel free to use any libraries that may help you. The `json` library in Python is very useful for parsing JSON. You have also created some algorithms in Homeworks 1 and 2 that may be of use!
- A natural idea for your algorithm is to pick seed nodes that can easily spread your epidemic to other nodes. Using centrality measures might be a good way to pick these nodes, but which centrality measures are best? Also, be sure to take into account the fact that other teams might be using the same strategy as you (in which case your seed nodes will cancel out). This will be especially important in the Jungle format.
- Don't forget about clustering. Remember from class that clusters provide a key predictor to how far a cascade will spread.
- You only have a short amount of time to pick seed nodes, so your algorithm must run within that time. However, do you *really* need to examine all nodes? Is there any other information you could use about the graph to approximate your calculations? Most graphs are below 1000 nodes, and no graph has more than 10,000 nodes.
- Successful teams in the past have spent a great deal of time analyzing other group's strategies, especially after the first few days. See if you can tweak your strategy to win previous days' games. All strategy JSONs are available on the website. Download, copy, or counter your competitors! Be warned, teams are known to shield their strategies after beating the TA teams.
- We give you a fixed amount of time, so if your code is easily parallelizable, it may be useful to do so since the time you have is independent of how much compute power you have.
- We are providing you the epidemic simulation script that will allow you to run the simulations on your own. This can be useful for tuning your algorithms, but it could also be part of the algorithm itself! Could you calculate the effectiveness of competing strategies by pitting them against each other? Could a game tree be useful?
- For the Jungle format, you can win overall across 50 rounds without winning any of the individual rounds (i.e. get 2nd place in many rounds vs. getting 1st in a few). Carefully consider how incentives change between Round-Robin and Jungle.
- Ties of submitting the exact same strategy as another team results in both teams losing. Ties of even points across all 50 rounds will result in a win for both teams.
- A strategy which works for one type of graph may not work well for others.

Visualizations:

- After a run, you can see animations for each of the 50 rounds of each game for each graph by going to "Results" and clicking on your score. Tutorial linked here. The animation does not work well for the larger graphs used but should give you a feel for how your pandemic cascades.