









COL761 – HW2

Instructions

-  Deadline: **March 31, 2025.**
-  Anti-Plagiarism Policy: Plag detection equals F-grade in the course.
-  Cite your sources in the report.
-  Dataset: link.
-  Task 4 is competitive.

Submission related instructions:

-  Add to GitHub.
-  Only one submission (per team).
-  More in the document.

The Virality Problem: Optimizing Social Contagion

We want to understand the *diffusion* of a viral infection through the people of a town through this assignment, think COVID-19. This is an important problem in epidemiology. Let's say that we have a very expensive vaccine. We only have k copies of the vaccine that we must administer to the people of the town that are most likely to cause a lot of spread (such humans are called superspreaders). While other biological factors play a role here, for this assignment we will consider that a person who comes into contact with more people spreads the disease more. You're then tasked with finding the k people to vaccinate in order to control the spread of the disease.

To model this problem, you need to model how the disease spreads through your town. Fortunately, you had been working on a similar experiment as part of your scientific endeavors, due to which you have the exact data required for this. You decide to use a directed graph to model the diffusion process for this purpose. The people represent *nodes* and there are directed edges between them if they are likely to come in contact (due to being from the same profession, family, etc.) so that the disease spreads between them.

One solution could be to detect k largest communities (using some clustering algorithm) and choose a node from each; and hope that each of these nodes can spread the disease in their community maximally. But we can do better.

A level of sophistication comes from associating with each directed edge a number in the interval $(0, 1]$ representing the probability that the destination node will be infected by the source node. The goal then would be to determine k nodes such that the set of all infected nodes (after enough time has passed) is maximized.

This makes the problem very hard, therefore, you decide to lay some simplifying assumptions. First, if a node u gets infected at time t , then it gets only one chance to try and infect its *uninfected* neighbors. This means that if u tries to infect $v \in \mathcal{N}(u)$ and fails, then it never tries to do so again. Also, if $v' \in \mathcal{N}(u)$ is already infected, then u does not try to infect it. Second, if u succeeds in infecting $w \in \mathcal{N}(u)$ then w becomes infected at time $t + 1$. (This process of

diffusion happens at *discrete* time steps.) And finally, the order in which several competing nodes u_1, u_2, \dots try to infect any un-infected node v at a particular time step does not matter; the probabilities still remain the same, and an infected node doesn't get *more* infected.

Problem. Given a graph $G = (V, E, p)$, where $p : E \rightarrow (0, 1]$, and a budget k , determine an initial *seed* set A_0 of infected nodes such that

- $|A_0| = k$, and
- $\mathbb{E}(|A_\infty|)$ is maximized

where A_∞ is the set of infected nodes at time $t = \infty$, and $\mathbb{E}(|A_\infty|)$ is the *expected* size of such a set.

Remark. Notice that since the disease spreading is stochastic, we care about the expected size of the set of infected nodes.

Task 1: Reduce this problem to another well known problem in computer science which is NP-hard. [20 marks]

Task 2: Find an efficient *approximate* algorithm for the problem using the insight gained from task 1 and derive its complexity. [10 marks]

Task 3: Come up with a hypothetical dataset where the algorithm you came up as solution of task 2 selects sub-optimal nodes. [10 marks]

Task 4: Run the algorithm on the datasets provided and share the solutions. **This part is competitive.** The code with more spread gets higher marks. [60 marks]

Note. **For understanding our expectations of what we want you to provide in the solutions of these Tasks 1-3, please go through submission instructions; and more precisely, through what we expect you to put in the report. For Task 4, please go through the last two sections: competitive evaluation scheme and time limit. We detail out how the marking will be done there.**

Remark. Since we care about the expected size of the set of infected nodes, you decide to use randomized methods to estimate the expected sizes. More precisely, given a graph with stochastic edges, one can generate n instances of these by performing a coin-toss on each edge and converting each stochastic edge into a binary valued edge (either the edge exists, or it doesn't; no in-between). Then, one can find the size of infected nodes in every instance as $|A_\infty^{(1)}|, |A_\infty^{(2)}|, \dots, |A_\infty^{(n)}|$ where the superscript (j) is used for indexing. Finally, an estimate of $\mathbb{E}(|A_\infty|)$ can be computed as

$$\frac{1}{n} \sum_{j=1}^n |A_\infty^{(j)}|.$$

An example of this is shown in figure 1.

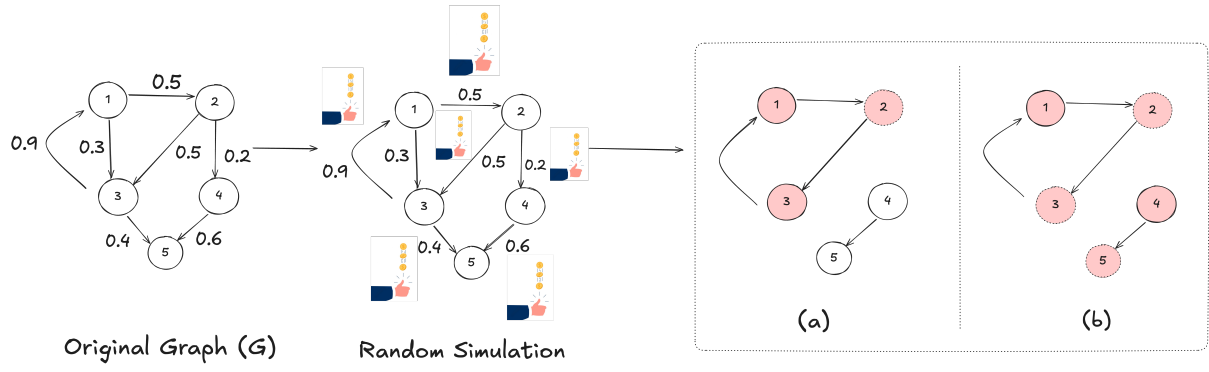


Figure 1: Here we demonstrate how we get a discrete graph using random simulations (we only show one random simulation, *i.e.*, $n = 1$). For $k = 2$, (a) shows the spread in this discrete graph when the seed set is $\{1, 3\}$ while (b) shows the spread when $\{1, 4\}$ is the seed set. Since the spread is more in (b), A_0 should be $\{1, 4\}$. **Note:** The red nodes with a solid outline constitute a seed set, while the red nodes with a dotted outline get infected. The white ones remain uninfected.

Submission Instructions

To ensure smooth evaluation of assignments, please ensure that you adhere to the following guidelines.

1. Create a script `solution.sh`. It should generate the output file with k lines, each containing exactly one integer. These k integers represent the node IDs of the k infected nodes at $t = 0$ such that the disease spread is maximised (that is they constitute A_0). We will run the script as follows:

```
bash solution.sh <absolute_path_to_graph> <absolute_output_file_path> <k> <#_random_instances>
```

2. You should also submit a `report.pdf`. The report should contain the following:
 - (a) The solution to Task-1, which should be a rigorous mathematical reduction of the given problem to an NP-Hard problem.
 - (b) For Task-2, provide the pseudocode for your proposed algorithm, state any optimizations over the naïve solution and derive your algorithm's time complexity.
 - (c) For Task-3, describe the dataset graph in the format $G = (V, E, p)$. Describe what each of the sets V and E are and the function p . Also show in detail why the solution your method provides is sub-optimal.
 - (d) Mention all team members' names, entry numbers, and contributions in the report as well.
3. Add <https://github.com/2023col761> as a collaborator to your GitHub repository. We'll count your latest git push as your submission time. (Do not create a new repository for this submission, use your repository from HW1.)
4. Upload your assignment under the `hw2` directory on *GitHub*.

5. The directory structure on *GitHub* should be as follows:

```
hw2
├── src/
├── solution.sh
└── report.pdf
```

6. On **Moodle**, submit a text file containing the link to your GitHub repository. It should be named `github_repo.txt`.
7. Do not submit data files.
8. Your code will be evaluated on HPC. Make sure to test it properly for reproducibility.

Competitive evaluation

The evaluation of Task-4 is competitive. It will have two components.

- For 30 marks you will compete among your class-mates. Let x be the highest spread any student achieves, and let y be the spread you achieve. Your marks will be $30y/x$.
- For the other 30 marks you will compete against a strong baseline that we will keep secret. Let z be the spread it achieves, then your marks will be $30y/z$.

Note. We have shared two datasets with you, and we have kept one dataset a secret. For each dataset you will get $30y/x \div 3$ marks and $30y/z \div 3$ marks, respectively. They will be then summed up to the full marks per component. Further, we will use the script `infection.cc` shared along with the code for evaluating the spread.

Time limit

!! Your code will be given **45 minutes** to run per dataset. Please periodically write the nodes your algorithm discovers in the output file so that we can compute the spread based on partial selection in case of a timeout and assign marks that way.