Group Project: COVID Scheduling Problem

CS 340, Fall 2020
Prof. Sorelle Friedler

Deadline: November 17th, 10am

1 Problem Description

You will be considering the issue of which classes should be online-only or in-person so that students are less likely to contract a disease (e.g., COVID-19) via in-class contacts. You will assume that the list of classes you are given is finalized (i.e., students aren't going to add or drop courses, and all students on your list will get into the listed courses). Your goal is to maximize the number of student seats in in-person classes (more specifics below) while keeping the number of people infected over a semester under a given threshold.

Specifically, you'll need to consider the following parameters to your SIR model:

1. **Schedule**: A list of classes with enrolled students is given. Every class must be scheduled either in-person or online-only and all students will attend in that form (i.e., we are disallowing the idea of hybrid courses). Classes are associated with given time slots that you do not have the option to change. You may assume that all classes fit in the given room if held in-person.
2. **Contagion probability**: Your algorithm should take a contagion probability ([0,1]) as an input. This is the probability that the virus spreads along any contact edge in the network.
3. **Infection length**: Your algorithm should take the infection length as an input. This is the length of time (in days) when people are in the infectious stage (I) of the SIR model.
4. **Semester length**: Your algorithm will take a number of days (the semester length) as an input. This is the number of days that students remain on campus.
5. **Acceptable number of infected students / professors**: Your algorithm will take as input a total number of students or professors who the administration thinks it's ok to have get infected at some point during the semester.

Your goal is to allocate classes as in-person or online-only so that all students are in the same courses at the same times as given; in-person courses are considered infection opportunities and all students and the professor in the course may be infected by any student / professor in the course currently in the infectious period of SIR. An allocation of in-person / online-only courses is considered optimal if it contains the maximum number of total student seats in in-person classes while remaining under or equal to the given acceptable number of infected students. We will call the total sum over all students of the number of courses they attend in-person the total number of in-person student seats. The maximum such value is 4 times the number of students if all students take 4 courses.

1.1 Your Tasks

You are *not* required to create an algorithm that achieves the optimal in-person/online-only allocation for the given inputs. Instead, your goal is to create an algorithm that consistently returns an allocation that is not "too far" from the optimal, and that runs "quickly." Quickly, in this context, means that given an instance about the size of Haverford, your algorithm should be able to run live during a 5 minute presentation (i.e., make sure it runs in under 3 minutes). The method of comparison to the optimal will be described in more detail in Section 3.2. The main portions of this project are as follows and will be described in more detail in the rest of this document:

1. Design the algorithm.
2. Analyze the algorithm's running time.
3. Do an experimental analysis to consider the quality of the allocation your algorithm creates.
4. Modeling and human impact analysis.
5. Write-up these results.
6. Present these results and do a demonstration of your program for your classmates.

Meeting these expectations will ensure that your group gets a C for the project. In order to get an A or B, you will need to do some additional work, described in Section 4. If you plan to do these tasks, you will likely find it useful to work on them from the beginning of the project and not wait to do them all at the end.

To ensure that your group does not wait until the last minute, which would be disastrous, there will be two project checkpoints during this process. Meeting these checkpoints will account for 20% of your project grade. The rest of the grade will be divided approximately as follows: Presentation - 10%, Paper - 70% (Algorithm description, analysis, and discussion 40%, Experimental and human impact analysis analysis 30%).

2 Programming Guidelines

You may choose to write the program implementing your algorithm in any programming language of your choice. Keep in mind that some programming languages are inherently faster than others, and that the programming language you choose should complement your algorithm design. You should prioritize a programming language everyone in your group is comfortable programming in. Your program should take as input (in order):

1. a constraints file giving the class times and days of the week. A sample tab-separated constraints file is given in the file `demo_constraints.txt`.
2. a file listing class, room, time, teacher, and student assignments per course. A sample tab-separated schedule input file is provided in the file `demo_schedule_input.txt`.
3. the contagion probability [0,1]
4. the infectious period length in (integral) days
5. the semester length in (integral) days
6. the acceptable total number of students or professors infected
7. an output filename where the resulting schedule should be written. A sample such file called `demo_schedule_output.` is provided. It contains the same information as the input schedule file, with an additional `Location` column indicating whether the course is `in-person` or `online-only`.

In addition, your algorithm should print the calculated in-person student seats value achieved in the form:

```
In-person Student Seats:  XXX
```

Where XXX is the final value. On the line following that, you should also print the best case student value - 4 times the number of students, or the student seats value if all students are enrolled in-person for all their requested classes:

```
Best Case Student Seats Value:  XXX
```

Since you may write in any programming language, you must provide a file called `run.sh` that when called with:

```
sh run.sh constraints.txt schedule_input.txt contagion_prob infection_days semester_days
    acceptable_infected schedule_output.txt
```

takes the two input files and four algorithm parameters, runs the algorithm, and outputs to the given output file name.

3 Paper Guidelines

The paper should be as long as it takes to sufficiently respond to the issues in the following sections. This is likely to be between 5 and 10 pages, with additional pages needed depending on the number of additional tasks attempted. It should be typed and a *single* pdf copy (i.e., one for the entire group) should be handed in by email by the deadline.

Your paper should also include a one to two paragraph *abstract* that emphasizes your findings (but not, in detail, how you came to them). This should be readable to someone who was not in the class, so that I could send it to the registrar as a set of recommendations!

3.1 Algorithm Description, Time Analysis, and Discussion

You should write-up a description and analysis of your algorithm as described in the "Algorithm Write-up Guidelines," except that you may omit the proof of correctness. You should also include a discussion of your group's algorithmic choices. Consider answering some of the following questions: Why did you chose this algorithm? What complications did you encounter while creating it? What characteristics of the problem made it hard to create an algorithm for? What algorithmic category or categories does your algorithm fall into? What algorithms that we've studied is your algorithm similar to?

3.2 Experimental Analysis

Your group should also perform and write-up an experimental evaluation of your algorithm's performance with the goal of understanding how close to optimal the created schedule allocation is. You should do the following two analyses:

1. **Time Analysis**: Verify that your algorithm performs as expected based on the theoretical time analysis (included in the write-up described in Section 3.1). Describe your experiment design and explain how the resulting numbers verify the time analysis.

2. **Solution Quality Analysis**: The goal of this analysis is to compare your solution to the optimal, without needing to determine the optimal in all cases. You may use the best case student value as an upper bound on the optimal value. Experimentally justify that your algorithm achieves some lower bound on the optimal value. This lower bound should be expressed as a fraction of the upper bound. For example, you might say that your algorithm is always able to achieve at least half the best case student seats value.

4 Haverford scheduling

In order to receive an A or B on this project, you must also consider how this scheduling question relates to the schedule at Haverford. By looking at the real Haverford scheduling information, you should come up with **predictions**, given existing in-person/online allocation of classes, for how COVID may progress through campus. You should also analyze **mitigation strategies** that could have been put in place via scheduling or that could still be put in place now to reduce the spread of the virus through campus.

In order to receive a B on this project, your group must successfully study at least 2 additional questions. In order to receive an A, your group must successfully study at least 5 additional questions. You may want to check with the professor to be sure your additional tasks are sufficiently difficult. You will need to include a detailed description and analysis of these additional questions. You may choose whatever input format you would like for these modifications to the problem (as long as the previous unmodified version of your algorithm still works).

5 Human Impact Analysis

You should include a discussion of the potential human impact of your design decisions. Be sure to not only focus on people like you; this might mean thinking about other students as well as the faculty and staff experiences of your choices. Who have you preferenced in your algorithmic choices? Who might you have hurt? Your discussion should include potential *negative* consequences as well as possible alternative choices you could have made, and chose not to, to mitigate those negative consequences.

6 Presentation Guidelines

As part of your final presentation, you will be required to demonstrate your code on a given input instance. Therefore, your code must be fast enough to run during a demonstration! In addition to this demonstration, your presentation should include the information from the paper at a high-level. All group members should be equally involved in the presentation and it should last 5-10 minutes.

7 Checkpoints and Deadlines

All time deadlines below are listed in the U.S. Eastern Time zone.

7.1 Checkpoint 1: Tuesday, October 6th at 10am

By the first checkpoint, your group should have decided on an algorithm and written the algorithm description, analysis, and discussion section of your paper. A rough draft of this section should be printed out and submitted in class. Since this deadline is early, you will only have covered greedy algorithms and will likely need to take a greedy approach to this problem. That's ok! Greedy approaches will still vary from group to group and can be very successful.

7.2 Checkpoint 2: Friday, October 30th at 10am

By the second checkpoint, your group should have programmed your algorithm in your chosen programming language. The code should be complete enough so that it can run on a given instance. If possible, you should additionally have started the experimental analysis and appropriate debugging of your code. The code should be committed and you should make sure that it runs using `run.sh`.

7.3 Final Deadline: Tuesday, November 17th at 10am

The full write-up is due as an emailed pdf - *please be sure to email a single copy and cc all group members on the email.* The class period Thursday will be spent giving presentations on your algorithm and a demonstration of your code. The final version of the code should be committed by 10am via an emailed .zip file as well. **No late projects will be accepted.**

8 Frequently Asked Questions

How does the infection enter the system? Choose an initial single node randomly as the infection seed node for the basic algorithm. (If you're interested, you could consider how specific choices of a single seed node change the results and/or what happens if there are k initial infectious nodes.)

Do the specific time slots matter? No, they don't. I.e., you may assume that there are enough in-person "classrooms" for as many classes as want to meet in person. What matters is which students and professors are scheduled for the same classes. However, if you'd like to consider the additional constraint of there being a limited number of available classrooms for in-person courses (as is actually the case) as one extension, that could be an interesting examination.

Do professors and students have the same transmission probabilities? Yes. Similarly, you should assume that the transmission probabilities are the same for all student to student edges. If you'd like to examine what happens if these are different, that could be an interesting extension.

Is it possible for the semester to move to online-only part way through the semester? No. You should attempt to assign online-only / in-person course statuses so that those modes can be maintained for the whole semester.

Given that SIR is probabilistic, how do we stay under the acceptable infected threshold? You should make an algorithm that stays under this threshold in expectation. I.e., it's fine to run a simulation of this multiple times and make an assessment based on those results.

Do additional contacts increase the chance of being infected? There might be a student who is in two classes with an infected student while another student is only in one class with the infected student - does that additional class conflict increase the chance the student is infected? No. (Of course, if you're interested in changing this modeling assumption, that would be interesting to examine for an extension.)