CS340 Final Project: COVID Scheduling Problem

Jiajie(Jason) Ma, Matthew Chen, Jiachen Yi

February 13, 2021

Abstract

Given the global pandemic (COVID-19) taking place in 2020, we are interested in how the spread of virus may affect class scheduling at Haverford. In particular, given a class schedule S, a semester length L, a contingency probability p, an infected length l, we want to see how many classes (at most) can be held *in person* while keeping the infected number of students/professors under an acceptable threshold n. To tackle this, we develop an $O(n^3 + n_p nL)$ algorithm that greedily moves classes with high(est) *connectivity* online. While this seems against our goal of maximizing number of in-person seats, it turns out to outperform the greedy algorithm on in-person seats by 148 seats on `Sample Enrollment Data 09282020` data. Further, the algorithm is always able to achieve at least 34% of the best case student seats value on `Sample Enrollment Data 09282020` data set. In an arbitrary selected case of 25 classes and standard inputs $L = 120$, $p = 0.062$, $l = 21$, $n = 50$; it yields 151 in-person seats while the optimal solution is 183. This is 82.5% of the optimal solution value.

With the existing in-person/online allocation of Haverford classes in Fall 2020, we also consider how COVID may progress through the SIR model with the standard contingency probability $p = 0.062$, infected length $l = 21$, and semester length $L = 81$ (Sept. 1st to Nov. 20th). The result shows a fast spread of virus that attains the peak on Day 29. Then, our baseline algorithm reports that 35 more hybrid classes need to be put *online* (with 734 remaining in-person seats) in order to keep the infection under the threshold. To improve the circumstance, we consider setting up a shelter for quarantine or advocating outdoor courses to alleviate the spread of virus. However, with a limit on the capacity of the shelter and the number of outdoor classes, neither of them is able to control the virus completely. Further, it turns out that the latter strategy is more effective with an observation that shelter merely helps with the situation. We also recognize that our baseline algorithm is developed under some naive assumption and propose some modified versions with respect to reality constraints. In particular, we model the situations where additional contacts increase the chance of infection and some classes such as art cannot be moved *online*. Our

algorithm is rather stable under these additional requirements with similar result outputs. However, we also find under the standard settings that our algorithm is not able to return a feasible plan where every in-person student has (at least) an in-person class. While this request seems hard, we also acknowledge that it is reasonable (as a request) and the algorithm should be used with this caution. Finally, we acknowledge that the algorithm may prefer students taking small classes to those taking large ones. The former population may include upper-level classmen, majors/minors in small departments etc. and the latter population contains freshmen (and sophomore), majors/minors in large departments. We discuss alternative methods to mitigate these negative consequences in Human Impact Analysis.

1 Description

Given an input schedule S (with a list of classes and their enrolled students), the contagion probability p, the infection length l, the semester length L, and the acceptable number of infected students/professors n; we propose a *greedy* algorithm that approximates the optimal allocation of in-person/online-only classes. The goal is to maximize the total number of in-person student seats.

Initially, all classes are labeled to be *in-person* in dictionary D. To set up, we construct a graph $G = (V, E)$ where vertices are students or professors, and edges connecting people who are in some same class. Further, we compute the degree of each vertex $v \in V$ and for each class, compute the sum of its participants' degrees (and store in dictionary D_c). For example, suppose there are three students s_1, s_2, s_3 and one professor p in CMSC340, whose degrees are $3, 4, 2, 3$ as vertices in G. Then, the sum of its participants' degrees is $3 + 4 + 2 + 3 = 12$. Note that this metric, indeed, reflects the connectivity of G (and thus, largely influences the infection rate). This motivates us to perform the greedy algorithm on this metric, with a hope that we delete minimum number of *in-person* classes (and possibly minimum number of students seats) to achieve a legal plan. Note that in the whole algorithm, we only keep track of this metric for *in-person* class (i.e. D_c only contains *in-person* classes and a class is deleted from D_c once it is online).

In each iteration, we assess if a semester stays under the acceptable infected threshold with the current plan given by D. To do this, we perform m simulations of the semester using SIR model and for each trial, record the number of infectants per day (as arrays). Given an expected threshold $0 \leq \tau \leq m$, if there are $m' \geq \tau$ number of simulations where the semester stays under the acceptable infection (by inspecting the infectant arrays), it implies that the plan given by D is legal and the algorithm terminates by returning D as well as the total number of *in-person* student seats (computed from D and S). Otherwise, if $m' < \tau$, it implies that the plan is not valid. In this case, we find the class C with the maximum sum of its participants' degrees and move C to be *online-only*. In particular, delete C from D_c. To maintain the graph, for each pair of participants u, v in C, we decrement the degrees of u, v by 1 if C is the only *in-person* class they have in common, implying that (u, v) is deleted from G (i.e. they no longer have offline contact). Then, for each (*in-person*) class in D_c, update the sum of its participants' degree using the updated value of degrees.[1]

Repeat the iteration above until the algorithm terminates with a valid plan, or all classes are moved *online* (i.e. D_c is empty). For the latter case, again, return D, where all classes

[1] In fact, this can be done in a smart way as shown in pseudocode. First, note that we need to update an *in-person* class in D_c only if it contains a participant u whose degree was just decremented. Thus, it suffices to update every *in-person* class u or v takes whenever we found that C is the only *in-class* class u, v have in common. Then, after traversing every pair of participants in C, we do not need to go through D_c again since every *in-person* class has been up to date.

are labeled *online*, and 0 as the total number of in-person student seats.

Two remarks should be noted in advance.

1. Since there are finitely many classes in S (the input schedule) and in each iteration, exactly 1 *in-person* class is moved *online* (with all classes *in-person* in beginning), we see that the algorithm must terminate with at most n_c iterations, where n_c is the total number of classes in S.
2. Since the algorithm either outputs a plan that passes the simulation test or a plan with all classes *online*, for which the semester must stay under the infected threshold (except if it is 0), we see that the output of the algorithm must be a valid plan.

2 Pseudocode

Let S be the input schedule, p the contagion probability, l the infection length, L the semester length, n the acceptable number of infected students/professors. To evaluate each proposed plan, recall that we run m simulations of SIR model and assess whether τ of them stay under the acceptable infected threshold.

Function `optimizeInPerson`(S, p, l, L, n)

Initially, all classes are labeled *in-person* (in dictionary D)
construct a graph $G = (V, E)$ where vertices are students or professors, and edges connecting people who are in some same class
for *each* $v \in V$ **do**
    compute the degree of v (stored in dictionary D_p)
end
for *each class in* S **do**
    compute the sum of its participants' degrees (stored in dictionary D_c)
end
while D_c *is not empty* **do**
    run m simulations of a semester using SIR model and the plan given by D. Record the number of successful simulations
    if $m' \geq \tau$ **then**
        break
    end
    else
        find an *in-person* class C with the maximum sum of its participants' degrees
        mark C to be *online* (in D)
        delete C from D_c
        for *each pair of participants* u, v *in* C **do**
            if C *is the only in-person class they have in common* **then**
                decrement the degrees of u, v by 1
                for *each* in-person *class* C' *that* u *or* v *takes* **do**
                    decrement C''s sum of its degree by 1
                end
            end
        end
    end
end
let N be the total number of seats for all *in-person* classes
return D, N

3 Time Analysis

Let n_c be total number of classes in S, n_p total number of participants, L the given length of the semester, and n the acceptable number of infected students/professors. Further, assume that S is given as a dictionary where keys are classes (id) and values are arrays of participants in that class.[2] This allows $O(1)$ access to a class and each participant in that class.

As noted in Description, the *while* loop (`Line` 7) has at most n_c iterations. We claim that the complexity of the *while* loop is $O(n_p^3 + n_p nL)$. To achieve this, the operations within the *while* loop must have complexity:

1. $O(L)$ for running m simulations of the SIR model and assess how many of them are under the infected threshold (`Line` 8-9),
2. $O(n_c)$ for finding the class C with the maximum sum of its participants' degrees (`Line` 11),
3. $O(1)$ for marking C to be online (`Line` 12),
4. $O(1)$ for removing C from D_c (`Line` 13),
5. $O(n_p^2)$ for accessing every pair of participants in C, updating their degrees, and updating the sum of degrees of the *in-person* classes they take. (`Line` 14-17)

3.1 Data Structures

To address the requirements above, we need the following data structure design:

- As specified in `Line` 1, we construct a dictionary D where keys are classes and values are the status of the classes (*in-person*/*online*). This is used to keep track of the up-to-date *in-person*/*online* class plan and is returned as the final solution in the end. For construction, it suffices to loop over S and insert {key : c, value: in-person} for each class $c \in S$. Since there are n_c classes in total and each insertion has complexity $O(1)$, it takes $O(n_c)$ to construct the dictionary.
- As specified in `Line` 2, we construct a graph G with vertex objects V and edge objects E. As demonstrated in Figure 9, each vertex v represents a student or a professor, which stores an identifier (i.e. $1, 2, 3, \ldots$ as given in S), a list of classes v takes, and a dictionary of edges, where keys are neighboring vertices u of v and values are edge objects that represent the edge (v, u). Note that the edge (v, u) reflects that v, u take some classes in common. To access the vertex objects in $O(1)$, we construct a dictionary D_v where keys are the identifiers of vertices and values are their corresponding vertex objects. At the same time, each edge object e stores its

[2] This is done in data pre-processing.

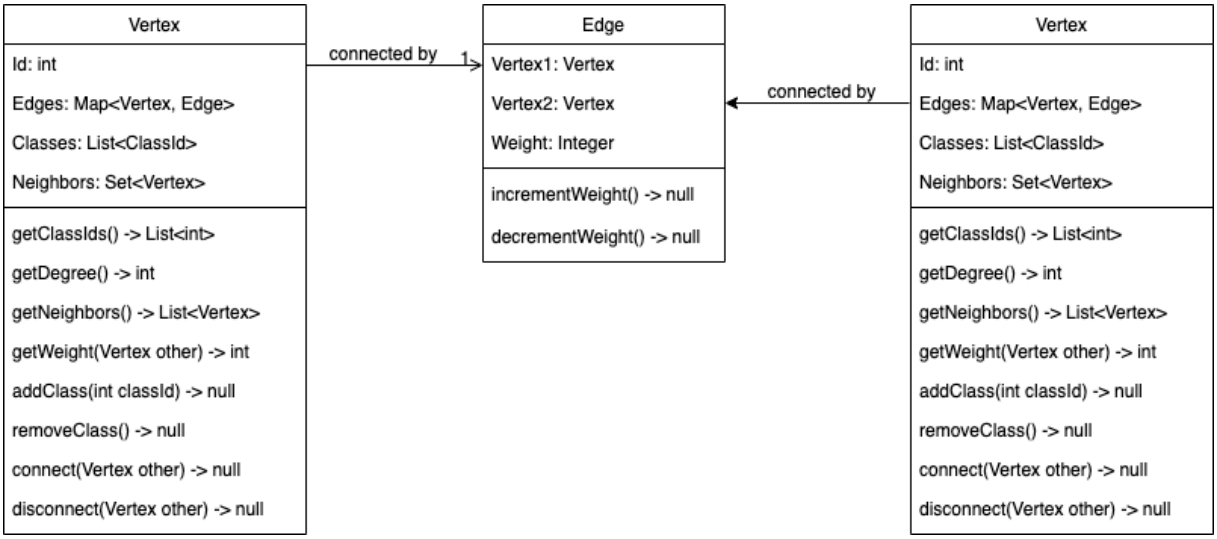


Figure 1: Class Diagram

endpoints u, v as well as a positive integer weight that shows the exact number of classes u, v take in common.

To construct G from S, we start by initializing an empty dictionary D_v. Then, we go through participants in every class in the following way. For each class C, we first go through all its participants and inspect if their identifiers i are in D_v. If not, initialize a key i and its corresponding vertex object as associated value. At the same time, for each participant in C, append C to their vertex object. Next, for each pair of participants (u, v) in C, we append an edge (u, v) to the vertex object of u and that of v. If there has already been an edge (u, v) (this may happen when u, v has more than 1 class in common), we increment its weight instead. Repeat these operations for every class in S. Now, since each student takes at most 4 classes, it takes $4n_p * O(1)$ operations to go through participants in every class for inspection (access to these students has complexity $O(1)$ by S). Further, since there are at most $n_p(n_p - 1)/2$ edges (when G is complete), each having a maximum weight of 4; we shall inspect at most $4n_p(n_p - 1)/2$ pairs of students (including repetitions) in total. In other words, it takes $4n_p(n_p - 1) * O(1) = O(n_p)$ operations to go through every pair of vertices in every class and append the edges for both/increment the weight. Since these two inspections are parallel, constructing G takes $O(n_p) + O(n_p^2) = O(n_p^2)$.

- As specified in `Line` 3, we construct a dictionary D_p where keys are vertices in G (i.e. students or professors) and values are the degrees of vertices. The values of D_p may be updated whenever a class is moved *online*. For construction, it takes $O(1)$ to access each vertex's edges (by construction of $G = (V, E)$). Since each vertex has at

most 4 edges, it takes $O(1)$ to access every edge of a vertex and compute its degree. Thus, since there are n_p vertices in total, constructing D_p takes $O(n_p)$.

- As specified in `Line` 5, we construct a dictionary D_c whose keys are classes and values are their sum of participants' degrees. Again, the values of D_p may be updated whenever a class is moved *online*. For construction, since there are n_c classes in total, it takes $O(n_c)$ to initialize an empty dictionary of size n_c. Then, for each class, we need to access its participants' degrees (and takes the sum), each of which takes $O(1)$ by D_p. Since there are at most $4n_p$ participants (including repetitions) across all classes, we need to access $4n_p$ times to D_p. Thus, initializing the values takes $4n_p * O(1) + n_p * O(1) = O(n_p)$ in total and the complexity of constructing D_c is $O(n_c) + O(n_p) = O(n_c + n_p)$.

With the data structure above, we see that:

1. To perform (1), we run the graph-based SIR model m times. In each simulation, we pick an infection seed node v randomly in $O(1)$. Further, we maintain a dictionary D_i of infected participants whose values are the remaining days for the participant to be recovered. Initially, D_i only contains $\{\text{key: } v,\ \text{value: } l\}$, where l is the total number of days for a participant to be recovered. At the same time, we maintain a set S_r of recovered participants who get cured after infected (and would not be infected again). Initially, S_r is the empty set. Thus, the constructions of D_i and S_r have complexity $O(1)$.

   Now, for each day, the algorithm checks if there are already more than n infectants (i.e. $|D_i| > n$) in $O(1)$, where n is the acceptable number of infected students/professors. If so, it terminates the simulation and reports a failure. Otherwise, the algorithm undergoes an infection step and a recovery step. For the infection step, it loops over all infectants $u \in D_i$ such that each (uninfected) neighbor of u is infected and added to D_i with the given contagion probability of p. Since each infectant has at most 4 (uninfected) neighbors and the infection can be determined in $O(1)$ (via a random number generator), the whole infection step has complexity $n \cdot 4 \cdot O(1) = O(n)$. For the recovery step, it loops over the infectant $u \in D_i$ and decrement each their recovery days by 1 in $O(1)$. If an infectant's recovery day reaches 0, they are popped out and added to S_r in $O(1)$. Again, it follows that the recovery step has complexity $n \cdot (O(1) + O(1)) = O(n)$. Overall, this implies that each day of the simulation has complexity $O(1)$ or $O(n) + O(n) = O(n)$, which are both bounded by $O(n)$. Further, since there are at most L days in a simulation, we see that one full simulation (including the preparations) above has complexity $O(1) + O(1) + O(1) + L \cdot O(n) = O(nL)$.

   Finally, since there are constant number of simulations (namely, m) and we may record the number of successful ones by maintaining an integer variable (increment it

by 1 for each successful simulation in $O(1)$), the operation of (1) has a total complexity $m \cdot O(nL) + m \cdot O(1) = O(nL)$.

2. To perform (2), it suffices to go through D_c. Since D_c has up to n_c keys, the linear search has complexity $O(n_c)$.

3. To perform (3), it suffices to update a value in dictionary D and it takes $O(1)$.

4. To perform (4), it suffices to remove a key from dictionary D_c and it takes $O(1)$.

5. To perform (5), for each pair of participants u, v in C, we first decrease the weight of edge (u, v) and it takes $O(1)$ with the constructed objects. If the weight turns to 0, it implies that they do not have any *in-person* class in common. In this case, we delete (u, v) from the vertex objects u, v, and decrement their degree in D_p by 1, both of which take $O(1)$. Further, for each class they take (this can be accessed in the vertex object in $O(1)$), we decrement their sum of degrees in D_c by 1. Since there are at most $2 \cdot 4 = 8$ classes, this can be done in $O(1)$ as well. Therefore, it takes $O(1)$ to maintain D_p and D_c for each pair of participants u, v in C. Since there are (at most) $n_p(n_p - 1)/2$ pair of participants (all of which can be accessed in $O(1)$ via S), the total complexity of (5) is $n_p(n_p - 1)/2 \cdot O(1) = O(n_p^2)$.

Therefore, as the operations above are all independent and there are at most n_c iterations, the *while* loop has complexity $n_c \cdot (O(nL) + O(n_c) + O(1) + O(1) + O(n_p^2)) = O(n_c nL + n_c^2 + n_c n_p^2)$. Since $n_c \leq 4n_p$, this is bounded by $O(n_p^3 + n_p nL)$. Now, since it takes $O(n_c) + O(n_p^2) + O(n_p) + O(n_c + n_p) = O(n_c + n_p^2) = O(n_p^2)$ to construct the data structures before the *while* loop and $O(n_p)$ to compute the total number of *in-person* student seats after the *while* loop,[3] the algorithm has complexity $O(n_p^2) + O(n_p^3 + n_p nL) + O(n_p) = O(n_p^3 + n_p nL)$ as claimed.

[3]Note that this requires us to go through D and for each *in-person* class labeled in D, count its seats by S. Since there are at most $4n_p$ seats across all classes, the number of counts is at most $4n_p$ and it has $O(n_p)$ operations.

4 Discussion

4.1 Greedy Algorithm: on which measure should we optimize

The proposed algorithm relies on a principle of greedily moving classes with highest connectivity (i.e. sum of its participants' degree) *online*. Since the infection rate is largely determined by the connectivity of the graph, this strategy is motivated by the goal to remove minimum number of *in-person* classes (and thus, possibly minimum number of students seats) in order for the plan to be valid. However, since what we ultimately want to optimize is the number of *in-person* student seats, an objection is that there is no guaranteed proof of how greedily removing classes with highest connectivity bounds the number of *in-person* student seats. In fact, since each student takes 4 classes at max, it is likely that classes with highest connectivity are those with more students seats – this seems to go against our objective. On the other hand, it should be recognized that while the variance of seats across each class tends to be small (10 to 40), the variance of connectivity across each class may be large. For example, it is possible that for two classes of 20 participants each, every participant in one class takes 2 courses, while every participant in the other class takes 4 courses. Then, the difference of their connectivity (i.e. sum of its participants' degree) is 40. This implies that the algorithm is likely to keep under the infected threshold efficiently with little sacrifice on the *in-person* student seats. In comparison, if we greedily move classes with minimum seats *online*, since the variance of connectivity across each class is large, we are very likely to remove a class with low connectivity and that has little effect on moving towards/under the threshold (this leads to more *online-only* classes). Further, this problem becomes more and more severe as we keep moving classes *online*, while the connectivity of *in-person* classes all decreases. In particular, those with small class size is expected to have low connectivity and it is a thankless work to remove them from graph.

Nevertheless, it should also be admitted that the metric/entire algorithmic strategy may be improved by taking the number of *in-person* student seats into account. For example, one way is to divide, for each class, the sum of its participants' degree by its number of student seats. That is, the class's average degree per participant. However, since every student takes 4 classes at max, this measure may be close across all classes and fails to give a sound selection. Another way is to imitate the Gradient Descent – as we start moving classes *online*, the number of infected people is monotone decreasing. Thus, in each iteration, we take a note of how far we are close to the infected threshold and accordingly, remove classes with appropriate connectivity. This potentially avoids the problem of overshoot.

We also note that the sum of degrees is not the best measure for connectivity. It is known from graph theory (and related epidemiology papers) that the connectivity of a graph (and the rate of infection) is highly related to the maximum eigenvalue λ_{max} of the graph's adjacency matrix. Then, it is desirable to delete nodes that reduce λ_{max} the most in each iteration. For example, it is possible that there are two groups in a graph that is connected

by one edge (so-called "bridge"). While the sum of degrees may not detect the endpoints of the bridge (it is likely that they have low degrees), the eigenvalue is a good reflection. However, since the detection of these nodes are computationally hard,[4] we choose the sum of degrees instead.

Finally, we remark that greedy algorithm is not the only way to approaching the problem. One method is to compute in advance how we can separate the graph into most connected components with least deletion of vertices (i.e. a k-clustering). In that case, we are able to "quarantine" the infected seed node within a small group and control the infection.

4.2 Complications

There are a few aspects that make the problem hard in nature. First, induced by a considerable amount of constraints, since the problem involves many trade-offs (e.g. connectivity vs. *in-person* student seats) and they seem to be input-dependent, it is hard to analyze the problem theoretically and produce a theoretical bound on the optimal solution (for any arbitrary input). Further, this is even harder without some in-depth knowledge of graph theory. That said, many algorithms are expected to rely on a closed-loop search for the optimal solution (by simulations). However, since the SIR model is stochastic, much more analyses can be done only probabilistically or rather, empirically (see Experimental Analysis section).

Besides, since we are required to delete classes of vertices rather than vertices, we cannot manipulate the graph as we rather desire under this granularity.

[4]Without further knowledge about graph theory, we may need to traverse all vertices in graph and check what λ_{max} would be if we remove the vertex.

5 Experimental Analysis

We evaluate our algorithm's performance, in terms of run time and solution quality, with experiments on `haverfordEnrollmentDataS14` and `Sample Enrollment Data 09282020` data sets. The former provides Haverford scheduling information (i.e. students' choices of (up to 4) courses) in Spring 2014 with a total of 1170 students, 311 professors, and 507 courses. The other data provides Haverford scheduling information in Fall 2020 with a total of 1961 students and 327 courses (the information of professors is not given). In addition, it also specifies the mode of each course being in-person/online due to the pandemic circumstance in 2020.[5] The algorithm is implemented in `Python` language and the code repository can be found at `https://github.com/MattChen2000/CS340-Final` with running instructions in `README.md`. Note that for the SIR simulation test, we run $m = 10$ times per test and take the successful threshold $\tau = 8$.

5.1 Run Time Analysis

Recall from (theoretical) Time Analysis section that the number of operations of our algorithm is a polynomial in n_c, n_p, n, L where n_c is the number of courses, n_p the number of students and professors, n the acceptable number of infected students/professors, and L the given length of semester. This polynomial is bounded by a polynomial in n_p, n, L whose dominating terms are $n_p^3 + n_p nL$. Assuming that the run time of each operation is constant, we expect that the n_p, n, L vs. run time plot is bounded by a polynomial whose growth rate is the same as $n_p^3 + n_p nL$. However, a direct verification confronts the problem of dimensionality: it is impossible to visualize a 4-axis plot and the growth rate/shape of a 3-variable polynomial $n_p^3 + n_p nL$ is hardly interpretable.

To tackle this, we consider the complexity to be bounded by a single-variable polynomial in $n_p^3 + n_p nL$. In other words, the $n_p^3 + n_p nL$ vs. run time plot is expected to be bounded by a linear function. To verify, we run the algorithm and measure the run time on `Sample Enrollment Data 09282020` data set. In particular, we fix a standard contingency probability of $p = 0.062$, $n = 50$ the acceptable number of infected students/professors (this number is the upper bound for a moderate level of COVID-19), and $l = 21$ the infection length (a moderate infection of COVID-19 takes 2 to 3 weeks for recovery).[6] Then, we use the whole scheduling information (i.e. $n_p = 1961$) and increment L by 5 starting at $L = 90$ (so that $n_p^3 + n_p nL$ grows with a fixed step size). For each set of inputs, we run the algorithm 100 times and take the average of their run time as final result.

[5] In this section, the problem definition provides us the freedom to manipulate the mode of classes, so we omit this information given in the data and will use it in Haverford Scheduling.

[6] As reported in Y. Chen et al. (2020) with confirmed cases, 0.062 is the infection rate of close contacts. The study can be found at `https://www.cebm.net/study/covid-19-epidemiological-characteristics-of-covid-19-close-contacts-in-ningbo-city/`.

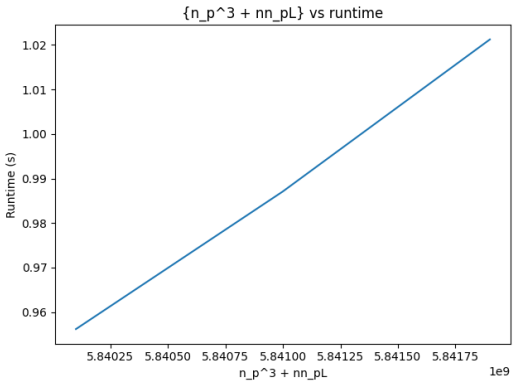
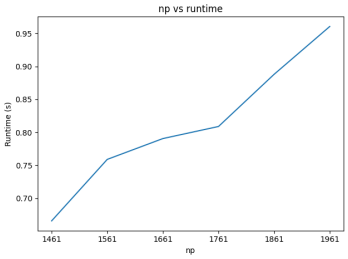
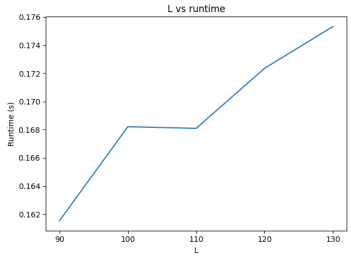


Figure 2: $n_p^3 + nn_pL$ vs. run time graph

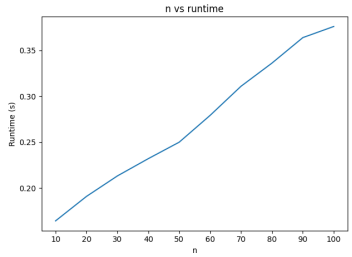
Observe that the graph is (almost) linear and thus, it is bounded by a linear function in $n_p^3 + nn_pL$. For further verification, we plot the n_p vs. run time graph by running the algorithm on `Sample Enrollment Data 09282020` data set with fixed inputs $L = 120$, $p = 0.062$, $n = 50$, $l = 21$. At the same time, n_p is incremented from 1461 to 1961 with a fixed step size of 50. Then, it is expected that the plot is bounded by a cubic function in n_p. Similarly, we plot the n vs. run time graph on `Sample Enrollment Data 09282020` data set with $n = 10$ to 100 (step size = 10) and fixed inputs $L = 120$, $p = 0.062$, $l = 21$, and the whole scheduling information (i.e. $n_p = 1961$). We also plot the L vs. run time graph on `Sample Enrollment Data 09282020` data set with $L = 90$ to 150 (step size = 10) and fixed inputs $n = 10$, $p = 0.062$, $l = 21$, and the whole scheduling information (i.e. $n_p = 1961$). Then, it is expected that both plots are bounded by a linear function (in n_p and L respectively). The results below verify our predictions. In particular, note that the latter two graphs are most almost linear and thus bounded by a linear function. While the first graph seems to be linear, it grows more rapidly than the other two (with a difference in the scale of y-axis). The (quadratic/cubic) curve is not apparent may be attributed to the fact that the algorithm is not run sufficiently long (i.e. the input size not large enough). However, it is still bounded by a cubic function in n_p.



(a) n_p vs. run time graph



(b) L vs. run time graph



(c) n vs. run time graph

5.2 Solution Quality Analysis

Recall that L is the given length of semester, n_p the number of students and professors, n the acceptable number of infectants, p the contingency probability, and l the infected length. We first verify the correctness of our algorithm in edge cases. For both `haverfordEnrollmentDataS14` and `Sample Enrollment Data 09282020` data sets, we run the algorithm with $p = 0$, $L = \{90, 100, \ldots, 150\}$, $n = \{10, 20, \ldots, 100\}$, $l = \{10, 14, 18, 21, 24\}$, and the whole scheduling information. Since the infected rate is 0 and $n > 1$, there is no spread of the virus and all classes can be set in-person. This corresponds to our obtained results. With $p = 1$, $L = \{120, \ldots, 150\}$, $n = \{10, 20, \ldots, 100\}$, $l = \{10, 14, 18, 21, 24\}$, and the whole scheduling information; the infected rate is high enough so that within a semester everyone in an in-person class would be infected. Thus, the optimal solution is to force all classes online and this corresponds to our results again.

To assess our solution in more complex settings, we start by comparing it with the total number of possible in-person seats (as the upper bound of an optimal solution). First, note that there are 4465 total seats in `haverfordEnrollmentDataS14` data set and 5659 total seats in `Sample Enrollment Data 09282020` data set. With the standard/desired inputs (explained above) $p = 0.062$, $L = 120$, $n = 50$, $l = 21$, and the whole scheduling information, the algorithm yields a result of 999 in-person seats for `haverfordEnrollmentDataS14` data and 2074 in-person seats for `Sample Enrollment Data 09282020` data. The ratio $\frac{\text{result in-person seats}}{\text{total seats}}$ are 0.224 and 0.366 respectively.[7]

For a wider range of validations, we run the algorithm on `Sample Enrollment Data 09282020` data set 300 times by randomly sampling 300 sets of inputs from $p \in \{0.062, 0.1, 0.162, 0.2, 0.262\}$, $L = \{90, 100, \ldots, 150\}$, $n \in \{50, \ldots, 100\}$, $l \in \{10, 14, 18, 21, 24\}$ (and using the whole scheduling information for every execution). Then, we see that all of them have a ratio $\frac{\text{result in-person seats}}{\text{total seats}} > 0.34$ with an average ratio 0.352. Thus, we may conclude that our algorithm is always able to achieve at least 34% of the best case student seats value on `Sample Enrollment Data 09282020` data set.

It is worth noting that the ratio above only provides the lower bound of $\frac{\text{result in-person seats}}{\text{optimal in-person seats}}$ for a set of given inputs. Since total seats remains unchanged for any set of inputs, the ratio $\frac{\text{result in-person seats}}{\text{total seats}}$ may vary depending on the difficulty of the inputs. For example, for small n and large p, the ratio $\frac{\text{result in-person seats}}{\text{total seats}}$ is expected to be small. However, this should not indicate a low quality of the solution – the in-person seats in the optimal solution is expected to be small as well.

For a more accurate assessment of our algorithm, we run our algorithm and a brute force algorithm (i.e. check every possible in-person/online class plan) on a subset scheduling of `haverfordEnrollmentDataS14` with (randomly chosen) 25 classes and inputs $L = 120$, $n =$

[7] For all the results presented (below), we run the algorithm 100 times and took the average.

100, $l = 21$, and $p = 0.3$.[8] Then, the optimal solution (given by the brute force algorithm) is 183 in-person seats, while our algorithm yields a plan with 151 in-person seats. The ratio $\frac{\text{result in-person seats}}{\text{optimal in-persno seats}} = \frac{151}{183} = 0.825$ (where the ratio $\frac{\text{result in-person seats}}{\text{total seats}} = \frac{151}{240} = 0.629$. This also validates some quality of our algorithm's performance.

[8]The reason for taking 25 classes results from the high complexity of brute force method – It took 15 hours to run the brute force algorithm on the 25 classes.

6 Haverford Scheduling

Given the existing in-person/online allocation of Haverford classes in Fall 2020 (`Sample Enrollment Data 09282020`), we consider how COVID may progress through the SIR model with a standard contingency probability $p = 0.062$ and infected length $l = 21$. Further, given the acceptable number of infected students/professors $n = 50$ (so that the COVID level is at most moderate), we are interested in how many more (and which) courses should be put *online* based on our algorithm. To do this, note that the modes of Haverford courses are either remote or hybrid (rather than in-person). At the same time, students have the choice of studying remotely or in person. Thus, we may assume that only students studying in person (in hybrid classes) may infect one another. That said, we pick out hybrid classes and students with in-person study mode in our algorithm to construct the graph G. Then, with the semester length (before thanksgiving break) $L = 81$ in reality (Sept. 1st to Nov. 20th), we run an SIR simulation and plot the days vs. number of infected students/professors graph as follows.

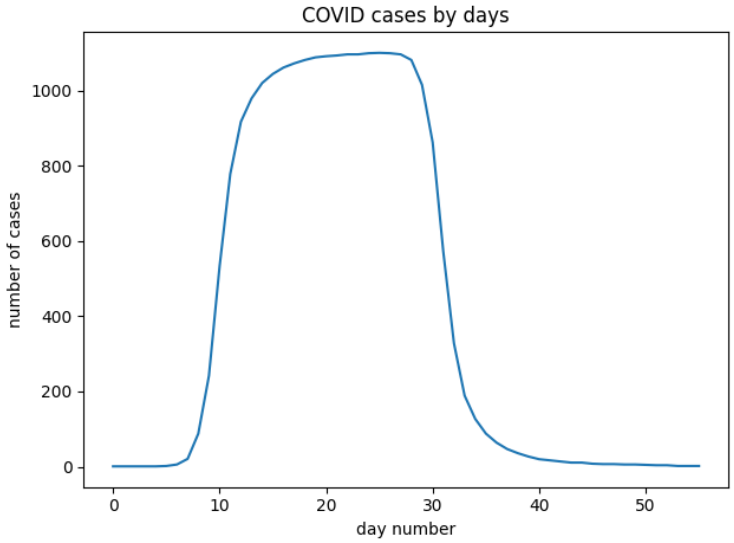


Figure 4: Number of infected cases per day

Since a recovered individual is not infected again, everyone is uninfected after Day 54 (and omitted from the graph). To generate a feasible plan under this setting, we run our algorithm with the inputs above ($n = 50$) and it shows that 35 more (hybrid) courses need to be put *online* with a result in-person seats (in hybrid classes) being 734. For comparison, the original in-person seats (in hybrid classes) is 2026 and the ratio $\frac{\text{result in-person seats}}{\text{total seats}} = 0.362$. To improve the number of in-person seats, we shall explore different mitigation strategies that can be put in place. In particular, we explore how setting up a shelter for

quarantine or advocating outdoor classes may help the circumstance in Extension 4 and 5.

It is also worth noting that our algorithm is developed under some naive assumptions. For example, while we assumed that additional contacts do not increase the chance of being infected, this may be false in the real world and the assumption eases the infection. We shall explore in Extension 1 how our algorithm reacts to a more complex contact-infection relation. In addition, our way of moving classes *online* confronts some reality constraints. It may be necessary for some subjects such as music or visual arts to be held in person. Further, for students who prefer study in person, it would be a nonsense if we move all their classes *online*. We handle these requirements in Extension 2 and 3.

6.1 Extension 1: Additional contacts increase infection rate

Recall that in the baseline algorithm, we construct a graph G where vertices are students/professors (with in-person study mode) and edges (u, v) indicate u, v share a hybrid class in common. Then, in each day of the SIR simulation, we only try one time of infection from an infected individual to its neighbors. In other words, each neighbor of the infected individual has the same chance of being infected (every day), namely the contingency probability p. However, consider a student s_1 who shares three hybrid classes in common with an infected student v whereas another student s_2 is only in one class with v. Then, s_2 is more likely to be infected with (potentially) more contacts with v. To model this circumstance, we assign each edge (u, v) a weight w that denotes the number of hybrid classes u, v have in common. Then, in each day of the SIR simulation, we try w times of infection from v to u where v is infected. In other words, the (number of) chances of infections corresponds to the number of contacts between u and v. The more contacts u and v have, the more likely that u is infected from v (where v is a already infected).

To see how this affects the COVID progress through the campus, again, we run an SIR simulation with semester length $L = 81$, contingency probability $p = 0.062$ and infected length $l = 21$. The days vs. number of infected students/professors graph is shown below.

We see from the graph below that the infection grows more rapidly than the naive model. Correspondingly, to maintain the number of infected students/professors under 50, our algorithm reports that 39 more classes have to be put *online* with 662 result in-person seats. While there are 4 more classes to be set *online* with a reduction of 72 in-person seats, we consider these loss as relatively acceptable and our algorithm is not sensitive to this reality constraint.

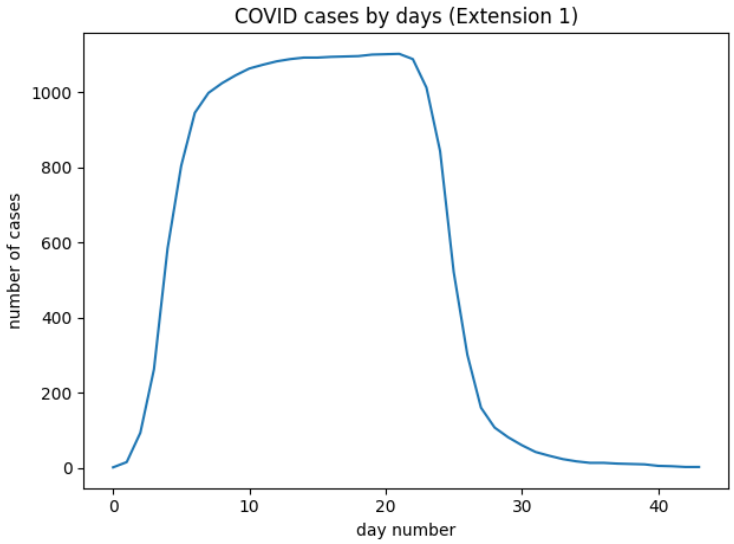


Figure 5: Number of infected cases per day (Extension 1)

6.2 Extension 2: Required in-person/hybrid classes

In the original algorithm, we are given complete freedom to move *any* classes *online* and our decision only depends on a single metric, namely the class connectivity. In reality, forcing a class *online* may accompany with various costs (e.g. technology and administrative efforts, etc.) and in certain cases, it may be impossible for a class to hold remotely. For example, music and visual arts classes may require special uses of instruments in the designated rooms. It is also important for professors to given some hands-on guidance in these classes. Under these considerations, we modify our algorithm such that (hybrid) music and visual arts classes cannot be set *online*. The algorithm returns -1 if there is no feasible plan under this constraint. Then, with the standard inputs $p = 0.062$, $l = 21$, $L = 81$, and $n = 50$, we see that 36 more classes have to be set *online* (none of which is a music or visual arts class) with 755 result in-person seats. This is roughly the same as the result given by the baseline algorithm and again, we consider our algorithm to be stable under this reality constraint.

6.3 Extension 3: Considering in-person students

Given the freedom to move *any* classes *online*, it is possible that in the result feasible plan, there is a student with in-person study mode who does not have any in-person classes at all. As a result, it does not makes sense for the student to study in person and the plan fails to meet everyone's preferences. On the other hand, there may exist a feasible plan in which every in-person student has a in-person/hybrid class. To find it, we modify our

choice of remote classes in the following way. First, we sort the (current) hybrid classes with respect to their connectivity in a list $\mathcal{L}$. To move a class C *online*, we first check if there is a student $s \in C$ such that C is the only in-person class for s. If not, we are safe to set C online; otherwise, we repeat this process with the next high-connectivity class in $\mathcal{L}$. The algorithm terminate with -1 if for all hybrid classes C, there is an in-person student $s \in C$ such that C is the only in-person class for s. In this case, no more class can be moved *online* to keep the infection under a threshold and satisfy all in-person students at the same time.

We run the modified algorithm with default inputs $p = 0.062$, $l = 21$, $L = 81$, and $n = 50$ for 100 times. However, all of the executions return -1 without a feasible plan. This may be interpreted as a result of severe pandemic situation and the strict case threshold $n = 50$ we have set (the algorithm starts returning feasible plans after p is lowered to 0.01 or n is lifted to 800). In turn, we see that with the given course schedules and the model, the request of enabling all in-person students to have an in-person/hybrid class is a hard one. In particular, there are 510 students with in-person study mode (on average) may not have an in-person class to keep the infection under the threshold. For fairness consideration, we do not further considering which courses should be forced *online* in order to minimize this loss and the original algorithm should be used with this caution.

6.4 Extension 4: Setting up a shelter

To help tackle with the COVID circumstance (i.e. reduce the number of additional remote classes and increase the number in-person seats), we need to control the spread of virus through the campus. One way is to contain the source of infection by setting up a shelter with capacity k ($k \in \mathbb{Z}^+$). Then, once an individual is observed infected and the shelter is not full, we send them to the shelter for quarantine. After the infected length l, the individual is recovered and discharged back to normal study life. In algorithm, this means that whenever an individual is observed infected and the shelter is not full, we disconnect the individual from the graph and add it to a data structure (list) that denotes the shelter. We pop the individual from the shelter once they are recovered. Further, follow the assumption of the original model that a recovered individual would not be infected again, we do not add them to the graph again.

It is worth noting that if an individual is observed infected when the shelter is full, they cannot be quarantined immediately and the spread of virus cannot be stopped. In other words, the effectiveness of this strategy depends on the capacity of the shelter k. At the same time, notice that the scheduling problem would become trivial if an individual can be observed infected immediately. In that case, the seed infected node would be quarantined in Day 1 and there would not be a spread of virus through the campus – that is, all hybrid classes can remain their mode of instructions. However, this does not agree with the fact that COVID has a 5 to 7-day incubation period, in which an infected individual can hardly

be tested positive. To model this situation, we require every infected individual to be observed positive after 7 days of infection.

Now, to see how this affects the COVID progress through the campus, again, we run an SIR simulation with shelter capacity $k = 30$, semester length $L = 81$, contingency probability $p = 0.062$ and infected length $l = 21$. The days vs. number of infected students/professors graph is shown below.

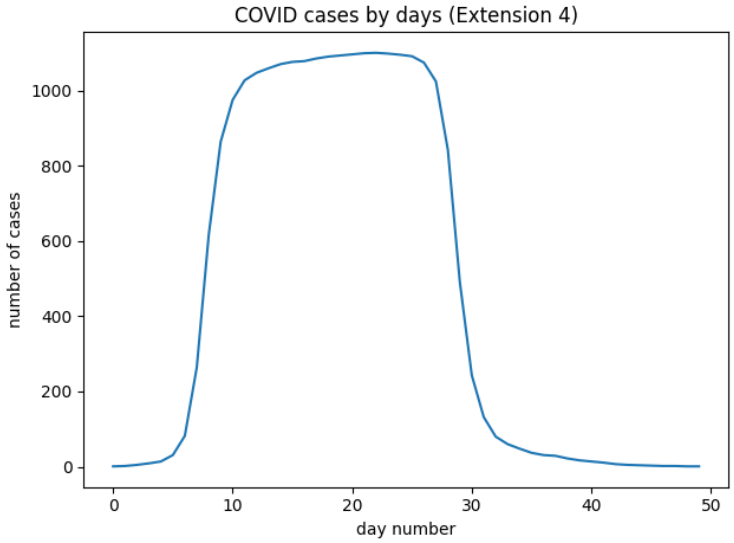


Figure 6: Number of infected cases per day (Extension 4)

Compared to the COVID progress given by the baseline SIR model, the existence of shelter helps stop the pandemic faster (under 50 days). However, there are still more than 1000 individuals infected and it does not lower the rate of spread of the virus. To maintain the number of infected students/professors under 50, our algorithm reports that 34 more classes have to be put *online* with 752 result in-person seats. This is only slightly improved from the result given by the in baseline. As mentioned above, the uneffectiveness may be attributed to the limited capacity of the shelter. In fact, it is also of administrative interest to see how the capacity of the shelter k affects the result in-person seats. The k vs. result in-person seats plot is shown below with standard inputs and k ranging from 30 to 100 (with a step size of 10, 100 times of experiments each). While we expected the number of in-person seats increase with respect to k, observe from the graph that it indeed oscillates with a maximum value attained at $k = 60$. On the one hand, it is likely that 60 is the best capacity for setting up a shelter. On the other hand, we infer that the existence of a shelter (with capacity under 100) is not effective as expected. This is possibly due to the high infection rate so that once an individual is infected and cannot be quarantined

immediately, the virus would spread through the campus in a striking manner.

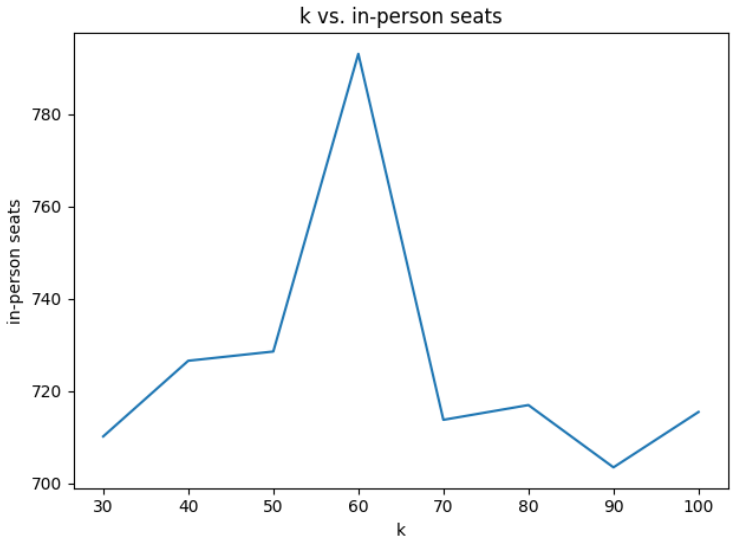


Figure 7: Capacity of the shelter vs. in-person seats graph

6.5 Extension 5: Advocating outdoor classes

Another way of controlling pandemic is to lower the infection rate by reducing the ways of transmissions. In particular, we may advocate professors to hold outdoor classes where there is a better air circulation (these are likely to be small humanity classes without special need of technologies in rooms). In those cases, COVID is less likely to spread in aerosols (floating in air). In algorithm, this means that for every two neighboring vertices u, v with v infected, v may infect u with the normal contingency probability p for every indoor (normal) classes u, v share in common and with a reduced contingency probability p' for every outdoor classes they have together. In our implementation, we assume that outdoor classes may alleviate the contingency probability by half (i.e. $p' = \frac{p}{2}$). Moving all English and Environmental Science classes outdoor, we run an SIR simulation with the standard inputs $L = 81$, $p = 0.062$ and $l = 21$. The days vs. number of infected students/professors graph is shown below. In particular, while the duration of the whole pandemic is longer, observe from the graph that the peak of infection lasts shorter than the original SIR model. Again, due to the high infection rate of COVID-19, there are still more than 1000 individuals infected and it does not lower the rate of spread of the virus.

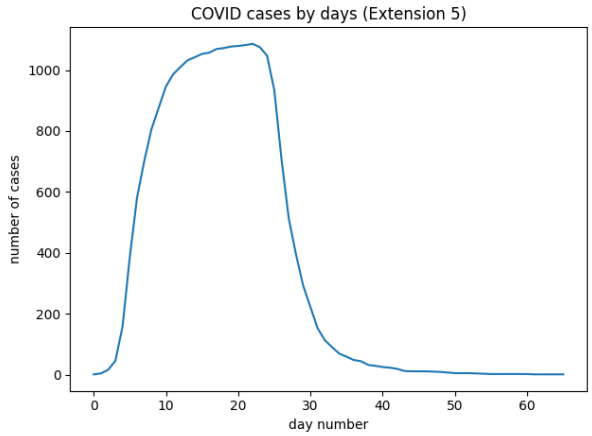


Figure 8: Number of infected cases per day (Extension 5)

To maintain the number of infected students/professors under 50, our algorithm reports that 35 more classes have to be put *online* with 772 result in-person seats. The result is rather better than set up a shelter. Similarly to the capacity of shelter k, the number of outdoor classes that take place is a hyperparameter that may affect the result. We run the algorithm with 2 to 10 subjects' courses moved outdoor (and with the standard inputs above). The number of outdoor subjects vs. result in-person seats graph is plotted below (each experiment is run 100 times). In general, we see an increasing trend of in-person seats as the number of outdoor subjects increase. This also shows that advocating outdoor classes is more effective than set up a shelter.

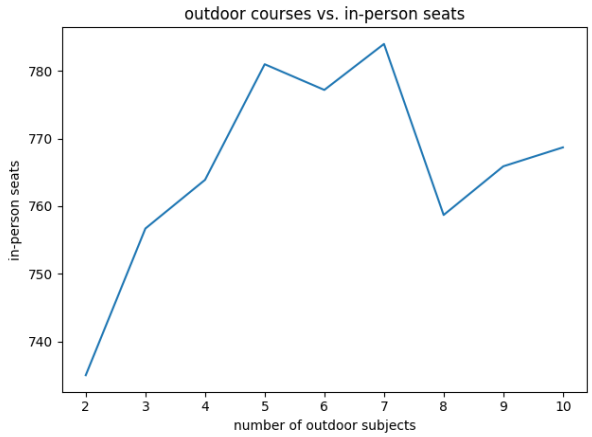


Figure 9: Number of outdoor subjects vs. in-person seats graph

7 Human Impact Analysis

Recall from Discussion section that our decision of moving classes with high connectivity *online* may result in most of lectures with large capacity remote and leaving small classes in-person/hybrid. On the one hand, this maintains the quality of small classes (especially seminars and arts classes with special mode of instructions) and prefers students who take small classes. According to the configuration of Haverford courses, these are likely to be students from upper-level classes or majors/minors in small departments. On the other hand, this algorithmic choice may hurt students who are in large classes such as intro lectures or popular major classes with a high demand. That said, experiences of freshmen (and sophomores) and majors/minors in large departments may be overlooked. Given the importance and difficulties of first-year life, this may add more burden to freshmen (with less convenient access to professors and other academic resources) and further, discourage them to study in person and adapt to an on-campus life. At the same time, not only does the algorithmic choice discourage students from their interested classes/majors due to the popularity, some multidisciplinary classes/subjects in which students are likely to take more classes together may be prioritized to set *online*. For example, biology students are also likely to take physics or chemistry courses. In that case, courses in these subjects may end up with a high connectivity and the algorithm discourages students from being multidisciplinary in this way. Both constraints on selection of courses betray the principle of a liberal arts college.

As mentioned in Discussion section, we could have chosen to greedily move classes with small(est) capacity *online*. However, this also suffers a dual problem of preferring large lectures/freshmen/courses in large departments and hurting upper-level classmen/seminars/small departments. In particular, it is likely that we need to set more remote classes in this case (to achieve the same result of in-person seats) and affect a wider range of people/departments. In fact, we have also implemented this greedy algorithm for experimental analysis. With the `Sample Enrollment Data 09282020` data set and standard inputs, the algorithm reports a result total seats of 586 (while our algorithm achieves 734). Thus, we see that this method is no better than our proposed algorithm except for reserving large in-person classes. Another potential method to mitigate the negative consequences is to loosen the impact of connectivity in our decision by redefining a metric. For example, we may add the factor of class capacity to the new metric. However, since there is a trade-off between the (in-person) class capacity and the spread of virus, this method may not end up with a good approximation of the optimal solution (of the in-person seats).