Course Scheduling to Minimize Student Wait Times For University Buildings During Epidemics

1st Arvin Hekmati
Department of Computer Science
University of Southern California
Los Angeles, California, USA
hekmati@usc.edu

2nd Bhaskar Krishnamachari
Department of Computer Science
University of Southern California
Los Angeles, California, USA
bkrishna@usc.edu

3rd Maja J Matarić
Department of Computer Science
University of Southern California
Los Angeles, California, USA
mataric@usc.edu

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671629

***Abstract*—Epidemic diseases bring many challenges to universities. In the case of airborne contagious diseases like COVID-19, health agencies' guidelines recommend that people maintain a physical distance of about 2 meters from each other. Enforcing such physical distancing on a university campus means that it will potentially take longer for students to get into and out of classrooms and buildings on campus. We use real course registration data from a large US university to study wait times students would encounter to enter and exit campus buildings while keeping the recommended 2 meter physical distance, and show that peak wait times can be longer than 20 minutes. We propose LBCS, a load-balanced course scheduling algorithm that intelligently reduces the peak wait time while ensuring that conflicting classes are scheduled at different times. Through simulations we show that LBCS can reduce the peak wait time by a factor of 3×, better than naive alternatives such as shifting some classes to the weekend or randomly perturbing class start times.**

***Index Terms*—COVID-19, Epidemic Modeling, Wait Time**

I. Introduction and Related Work

The COVID-19 pandemic has had a profound impact on educational institutions around the world. More than 85 colleges and universities across the US have reported at least 1,000 cases of COVID-19, and over 680 institutions have reported at least 100 cases [1]. More than 124,000 public and private schools, colleges, and universities in the US closed in April 2020, impacting more than 55 million students [2]. Worldwide, similar disruptions have affected more than 1.7 billion students [3], [4].

In response to the initial COVID-19 outbreak in Spring 2020, a large number of colleges and universities across the US decided to cancel classes and close student housing [4]. Many universities and colleges moved instruction online. The transition from in person classes to online instruction brought many challenges for both students and instructors. For students, the transition to online instruction exacerbated challenges associated with access to technology and led to concerns regarding absenteeism and accommodation of special needs [4]. For instructors, the transition to online instruction led to increased concerns regarding student engagement and evaluation [5], [6].

In Fall 2020, many institutions of higher education in the US returned to in person instruction. However, this led to a significant increase in new infections. Several colleges and universities decided to reopen in Fall 2020 and provide hybrid classes where a portion of the students could attend classes in person while others attended online, providing a partial solution to the problems associated with purely online instruction. In addition, colleges and universities put in place rules about physical distancing and face-covering use, and limited social gatherings. Many institutions also put in place extensive population testing, contact tracing, and quarantining measures for on-campus students, staff, and faculty. This combination of measures has had some success in curbing the spread of COVID-19 on campuses [1].

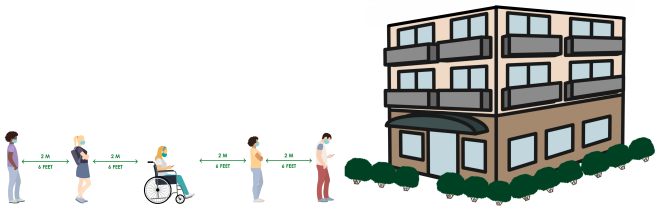


Fig. 1: Students waiting in line to enter a classroom building

Reopening a college or university brings many challenges. Hekmati *et al.* [7] analyzed risks associated with COVID-19 spread through classroom transmission on a university campus. In addition to the risk of spreading the virus by holding classes in person, students should also maintain physical distancing out of the classrooms to decrease the spread of the virus. This could potentially lead to long wait times for students entering and exiting buildings. Although vaccinated students are less likely to get infected with the original COVID-19 variant even at close physical distances, for the newer and more transmissible COVID-19 variants like the Delta variant, keeping the 2 meter physical distancing is recommended to slow the spread of the virus.

In this work, we propose a model that estimates and *quantifies* student wait times, assuming a policy where students are encouraged to line up and move in an orderly fashion while keeping d meter distance from other students in order to orderly enter and exit university buildings. We use the course schedule of a major US university for evaluating our

model. We evaluate two naive approaches for reducing the wait times: randomly shifting class start times and moving classes from weekdays to weekends. Finally, we propose Load Balanced Course Scheduling (LBCS), an efficient load-balancing algorithm to distribute courses over the week and campus to reduce wait times, also considering minimizing schedule conflicts for the enrolled students. Our simulations show that the LBCS algorithm can reduce the ingress and egress wait times of the buildings by a substantial amount, up to 3× for very crowded buildings with many classrooms.

The rest of this paper is organized as follows. In section II, we present the general model for calculating ingress and egress wait times. Section III presents three naive algorithms for changing courses schedules in order to decrease wait times of the relevant buildings. LBCS is presented in Section IV. Section V describes the simulation methodology used for evaluating the algorithms. The results are presented and discussed in Section VI. Finally, we conclude the paper in Section VII.

II. The Wait Time Model

In this section, we present the model for estimating the wait times for students entering and exiting campus buildings in an orderly fashion. Various parameters are involved in estimating the ingress and egress rates of students. Here we assume the students walk with an average speed of $s_{in}[m/s]$ when entering a building and $s_{out}[m/s]$ when exiting a building, and have d meters physical distance between them. Given that, the average ingress rate r_{in} and egress rate r_{out} can be calculated by:

$$r_{in} = \frac{d}{s_{in}} \tag{1}$$

$$r_{out} = \frac{d}{s_{out}} \tag{2}$$

The average number of buildings ingress, and egress points are being considered as e_{in} and e_{out}, respectively. Additionally, for any given time period t of the day, the number of students entering and exiting the building is considered to be $n_{in}(t)$ and $n_{out}(t)$, respectively. The ingress and egress wait time of a building, given the time of day in the week, can be calculated as follows:

$$w_{in}(t) = \frac{n_{in}(t)}{r_{in} \cdot e_{in}} \tag{3}$$

$$w_{out}(t) = \frac{n_{out}(t)}{r_{out} \cdot e_{out}} \tag{4}$$

The time index in the equations (3) and (4) refers to the discrete time periods of the day of the week. For instance, we can divide all days in the week into slots with 20 minutes duration. Here t refers to the desired time slot on a weekday for calculating the ingress and egress wait time.

Given the ingress and egress wait times for the campus buildings, we define a new metric called *total wait time* as the maximum wait time between the ingress and egress wait times:

$$w_{total}(t) = \max\{w_{in}(t), w_{out}(t)\} \tag{5}$$

III. Naive Algorithms

In order to control and decrease the ingress and egress wait times, we present two naive algorithms:

- Class start time randomization: Each class time is moved by a random time chosen uniformly in $[-\delta, \delta]$.
- Weekend scheduling: Include one or two weekends in the schedule, such that, $n \cdot x$ percent of classes are moved to the weekend, where n is the number of weekends (1 or 2), and x is the ratio of the classes that in each day of the week will be moved to the weekend.

Any combination of these algorithms could be applied to reduce the wait times as much as possible. However, each algorithm has drawbacks. Class start time randomization can result in conflicts in student class schedules, i.e., some classes with the same students may be scheduled at overlapping times. Using weekends presents conflicts with other student (and instructor) activities.

In addition to these algorithms, reducing in person class occupancy significantly reduces wait times but presents the challenges of online education and managing appropriate selection of students for in person vs. online. In this work, we view reducing in person class occupancy as orthogonal and complementary to the scheduling algorithms we consider and evaluate. Therefore, our evaluations focus on in-person class attendance.

IV. Load Balanced Course Scheduling (LBCS) Algorithm

In this section, we present LBCS, an efficient algorithm for distributing courses across the week and campus buildings to reduce building ingress and egress wait times. The algorithm must minimize conflicts in enrolled students' schedules. For that purpose, we present the "conflict graph" that aids in course scheduling with minimum conflicts.

A. Conflict Graph

Given students' course schedules, we design a conflict graph that shows how much courses conflict with one another in terms of having at least k enrolled students in common. We create a graph whose nodes represents the courses and edges that represent at least k students in common between the two nodes. Each edge has an attribute $conflict$ that indicates the number of students in common between two nodes/courses. Figure 2 shows the conflict graph for the Fall 2019 course schedule we used, with $k = 5$. For visual simplicity, we only show the conflict graph for $k = 5$[1]. Fall 2019 schedule has 5986 courses with 34042 students. Figure 2 shows 3676 courses that have a conflict with each other with $k = 5$, implying that creating a load-balanced course schedule in Fall

[1]Lower values of k show similar features but result in a denser graph that is harder to visualize.

2019 requires many considerations in terms of minimizing student course conflicts.

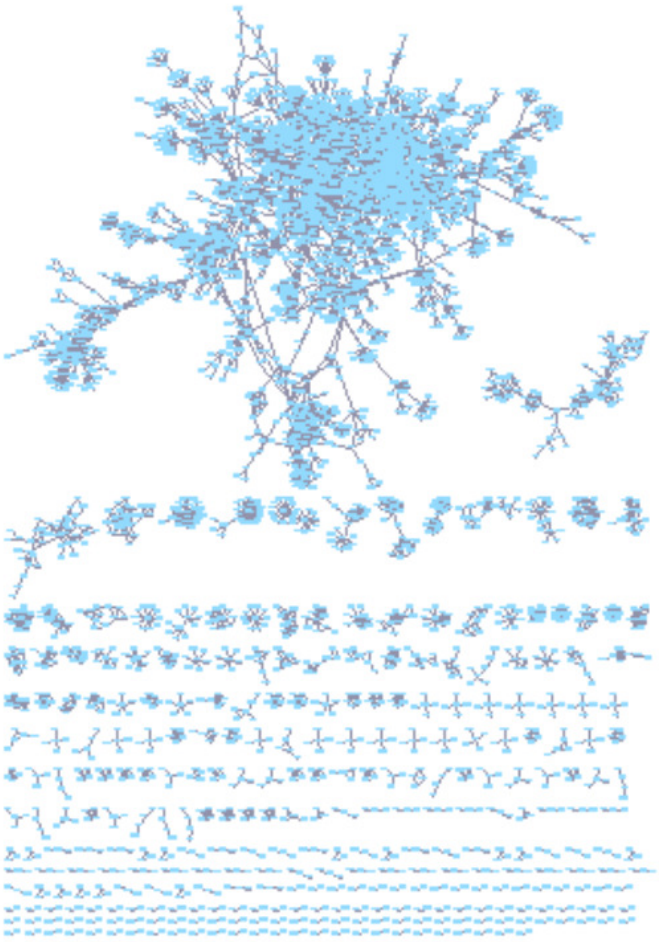


Fig. 2: Fall 2019 course conflict graph visualization

B. LBCS Steps

Here we present the steps of the LBCS algorithm. The pre-processing steps involve the following:

1) Create the conflict graph given the course schedule, based on the procedure described in section IV-A.
2) Create a list of graph nodes called $courses$, sorted according to their degree, in descending order. For nodes with the same degree, sort them according to the sum of the $conflict$ attribute that each node has with other nodes. Recall that the $conflict$ attribute shows the number of common students between two courses (nodes). In other words, the sum of the $conflict$ attribute for each course (node) is the number of students who have a conflict with other courses.
3) Create a list of classrooms with their associated capacities in an ascending order.

Pseudoode for the LBCS algorithm is given in Algorithm 1. The algorithm takes the following inputs:

- $conflict\_graph$: Conflict Graph
- $courses$: List of courses sorted according to their degree and sum of their edges' attribute, i.e. $conflict$
- $classrooms$: List of classrooms sorted according to their capacity
- $weekdays$: Weekdays to be used for scheduling
- $start\_time$: Start time of the day
- end_time: End time of the day
- th_1: Threshold 1 determines the maximum wait time for students to enter a building for the courses that are small enough to enter and exit the building in th_1 seconds. In other words, the number of students is less than $th_1 \cdot r_{in} \cdot e_{in}$ for the case of entering the building or is less than $th_1 \cdot r_{out} \cdot e_{out}$ for the case of exiting the building.
- th_2: Threshold 2 determines the additional possible wait time for each building in order to schedule large courses for which the number of students is so large that they can not enter and exit the buildings in th_1 seconds. In other words, the number of students is greater than $th_1 \cdot r_{in} \cdot e_{in}$ for the case of entering the building, or is greater than $th_1 \cdot r_{out} \cdot e_{out}$ for the case of exiting the building.

The LBCS algorithm schedules the courses in the conflict graph according to their degree in descending order, i.e., the courses with the highest number of conflicts with other courses. If two courses (nodes) have the same degree, the algorithm sorts them according to the sum of the $conflict$ attribute that each node has with other nodes. Recall that the $conflict$ attribute shows the number of common students between two courses (nodes). In other words, the sum of the $conflict$ attribute for each course (node) is the number of students who have a conflict with other courses. Note that in order to avoid conflict, the courses connected in the conflict graph will not be scheduled at the same time. For a given course, the wait time required for students in that classroom to enter and exit the building $course\_wait\_time$ is calculated. Next, the algorithm finds the classrooms that have enough capacity to accommodate $course$'s students. Then, a compatible classroom is selected with the lowest capacity that can accommodate $course$ at the earliest start time in the week and examine whether the selected ($room$, $time$) is available. If not, the next possible class time is selected and checked, and so on. If no time works, the classroom is updated to the next compatible one, and the search for class time starts at the earliest time again. Once we find a proper ($room$, $time$), if the total wait time of the $room$'s building at $time$ is less than th_1 or the total wait time of that building at $time$ is less than $th_2 + course\_wait\_time$, the schedule of the course at ($room$, $time$) is finalized. Suppose neither of the two conditions is valid for ($room$, $time$). In that case, the tuple and its corresponding wait time are stored, and the algorithm moves to the next possible time and classroom and calculates their wait times in order to find the ($room$, $time$) that has the minimum possible wait time. Note that th_2 helps with the case if the size of a class is so large that it cannot fit in th_1. In the case that no ($room$, $time$) are available to schedule $course$, the algorithm returns -1, indicating that the

load balancing schedule is not feasible for the given inputs.

V. Simulation Methodology

We performed simulations to evaluate the course scheduling algorithms based on their generated wait times. For evaluation, we used the Fall 2019 course registration data of a large US university. The dataset included information for all enrolled students at the university. We considered only classes that were held in person (most of them were, as the dataset predates the COVID-19 pandemic). In Fall 2019, there were 5986 courses with 34042 students on campus in the dataset.

We evaluated building wait times of four different algorithms. The first algorithm shows the original scheduling wait times used by the university. The next two are the naive algorithms: the random start time and transfer to the weekend. The fourth is LBCS. The presented evaluation examines each algorithms separately; as noted earlier, the algorithms could be combined to potentially achieve even lower building wait times.

We used two metrics to visualize the wait times. The first metric shows the distribution of the maximum building wait times during the week, allowing for readily identifying peak wait times and the most impacted buildings. The second metric is the plot of a specific building wait time over the course of the week. We plotted the wait time of all buildings, showing the peak values of wait times of all buildings and days of the week.

In the simulations, we assumed that each building has two ingress points and two egress points, i.e., $e_{in} = e_{out} = 2$; all four can have lines of students active at the same time. Additionally, we assumed that students go in and out of the building in an orderly fashion, maintaining a physical distance of 2 meters $d = 2[m]$ from students ahead and behind them in the line. In the simulation, each student moved at an average rate of 1 meter per second, $s_{in} = s_{out} = 1[m/s]$. This speed is slightly below the average walking speed of 1.38 meters per second for young persons from 20 to 29 years old [8], accounting for slower movement that may be anticipated in careful line behavior during physical distancing. Consequently, on average, 1 student will enter (or exit) per ingress (or, respectively, egress) point every 2 seconds, $r_{in} = r_{out} = 0.5$. Given these parameters, we calculate the ingress and egress wait time of each building using equations (3), (4).

In original scheduling, we present the building wait times from the original dataset. In the random start time scheduling algorithm, we simulated the algorithm for different time shifts of $[-t, +t]$ minutes for t from 10 to 90 minutes with incremental steps of 10 minutes. Figure 3 shows the peak waiting time of the random start time scheduling algorithm for different time shifts, i.e., t. As we can see, for $t = 30$ minutes, we are getting the minimum peak wait time. Therefore, we will use a uniform random time shift of 30 minutes for further analysis of this algorithm. In the transfer to the weekend scheduling algorithm, we are transferring $n \cdot x$ courses from each day of the week to the weekends, where n is the number of weekends (1 or 2), and x is the ratio of the classes that in each day of the week will be moved to the weekend. In this simulation, we used $n = 2$ to reduce the peak wait times as much as possible. Figure 4 shows the peak wait time of the transfer to the weekend scheduling algorithm by swiping the transfer ratio, i.e., x, from 0 to 0.5 with 0.1 incremental steps. As we can see, $x = 0.1$ provides the minimum peak wait time. Therefore, in further simulations of transfer to the weekend algorithm, we assume that $n \cdot x = 20\%$ of the courses in each day of the week will be transferred to the weekends.

Finally, in the LBCS algorithm, given operations of a university about 6000 courses and about 34000 students on campus, it is impossible to create a load-balanced schedule with much lower building wait times without conflicts. Therefore, we created the conflict graph by considering $k = 5$, only connecting nodes in the graph that have at least 5 students in common. Furthermore, thresholds 1 and 2 were

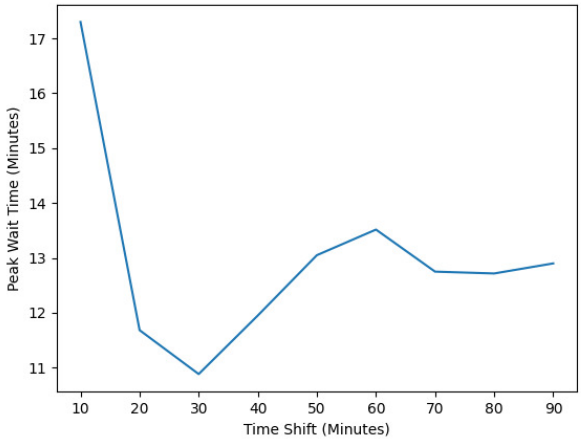


Fig. 3: Random start time scheduling algorithm analysis: peak wait time vs time shift

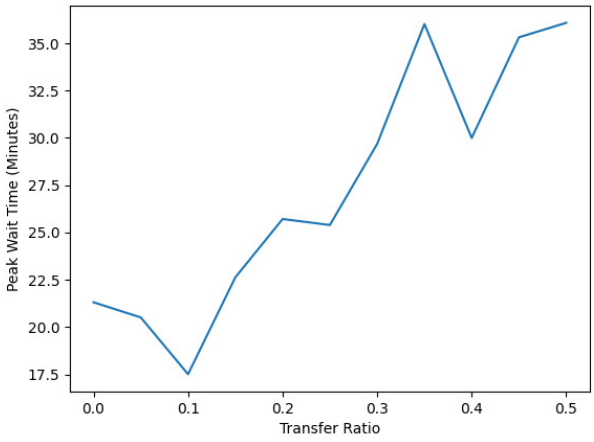


Fig. 4: Transfer to weekend scheduling algorithm analysis: peak wait time vs transfer ratio

Algorithm 1 Load Balanced Course Scheduling (LBCS) Algorithm

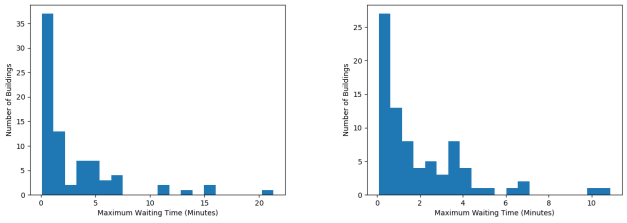
Input: $conflict\_graph$, $courses$, $classrooms$, $weekdays$, $start\_time$, end_time, th_1, th_2
for $course$ in $courses$ **do**
  $course\_wait\_time \leftarrow \#students \times rate/\#entrances$
  $course\_sessions \leftarrow$ $sessions$ in $course$
  for $session$ in $course\_sessions$ **do**
    $rooms \leftarrow$ compatible rooms in $classrooms$ with $course$
    $min\_slot.wait\_time \leftarrow \infty$
    $min\_slot.time \leftarrow start\_time$
    for $room$ in $rooms$ **do**
      for day in $weekdays$ **do**
        if day already used for scheduling $course$ **then**
          continue
        end if
        $time \leftarrow start\_time$
        for $time$ in $(start\_time, end_time)$ **do**
          $ingress\_time \leftarrow$ the ingress wait time of the classroom building at $time$
          $egress\_time \leftarrow$ the egress wait time of the classroom building at $time$
          $total\_wait\_time \leftarrow max(ingress\_time, egress\_time)$
          if $total\_wait\_time < min\_slot.wait\_time$ **and** $room$ is available at $time$ **and** $course$ does not conflict with the courses scheduled at $time$ **then**
            $min\_slot.wait_time \leftarrow total\_wait\_time$
            $min\_slot.time \leftarrow time$
            if $total\_wait\_time \leq th_1$ **or** $total\_wait\_time \leq course\_wait\_time + th_2$ **then**
              schedule $course$ at $time$
            end if
          end if
        end for
      end for
    end for
    if $min\_slot.wait\_time \neq \infty$ **then**
      schedule $course$ at $time$
    else
      return -1
    end if
  end for
end for

set to 120 and 30 seconds, respectively. These parameters were chosen empirically to be the smallest such that the LBCS algorithm results in feasible schedules. We used only weekdays (Monday to Friday) for evaluating load-balanced scheduling and considered the start and end of the days as 8 AM and 10 PM, respectively. Note that in the original and also random start time scheduling algorithm, we also used only weekdays from 8 AM and 10 PM. But, weekend scheduling algorithm had this advantage compared to others as it also used the weekends for the scheduling.

VI. Results and Discussion

Figure 5 shows the distribution of maximum wait times for the buildings using the four different scheduling algorithms for the Fall 2019 course registration dataset. Table I summarizes the information in Figure 5 by providing the peak wait times and the average of peak wait times for each algorithm. Further, Figure 6 shows the wait time versus the time of day for the buildings with the highest wait time.

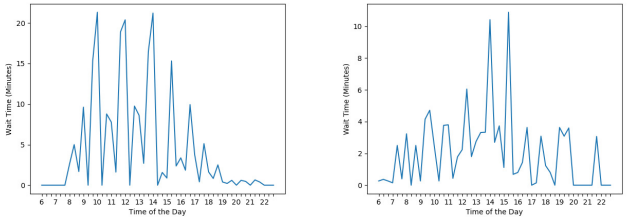
As can be seen in Figures 5 and 6 and in Table I, the original scheduling has a peak wait time of 21.32 minutes and average peak of 3.10 minutes. Of the three scheduling algorithms, LBCS performs the best by reducing the peak wait time to 7.16 minutes and an average peak of 1.17 minutes. This wait time is about three times lower than the original scheduling approach used by the university. The other two scheduling algorithms also reduce the wait time compared to the original approach, with the random start time algorithm performing better of the two, by reducing the peak wait time to 10.83 minutes and the average peak wait time to 1.97 minutes. Transfer to weekend strategy performed the worst and could not reduce the wait times effectively. Naturally, even lower wait times can be achieved by using the hybrid scheduling where classroom occupancy is reduced through online instruction.



((a)) Original ((b)) Random Start Time

((c)) Transfer to Weekend ((d)) LBCS

Fig. 5: Maximum peak wait time distributions



((a)) Original ((b)) Random Start Time

((c)) Transfer to Weekend ((d)) LBCS

Fig. 6: wait time over one day for the worst case building

TABLE I: Scheduling Algorithms' Building Wait Times

	Maximum Peak wait Time (mins)	Average Peak wait time (mins)
Original	21.32	3.10
Random Start Time	10.83	1.97
Transfer To Weekend	17.51	3.07
LBCS	7.16	1.17

VII. Conclusion

This paper discussed the challenges that health guidelines and policies may bring for universities. In the case of very contagious diseases like COVID-19, public health guidelines recommend maintaining physical distance of about 2 meters between people. Enforcing this in the context of colleges and universities results in potentially impractical ingress and egress wait times for classroom buildings. This paper compared four different classroom scheduling approaches in order to identify the approach that minimizes wait time. A comprehensive load-balanced course scheduling algorithm, LBCS, was presented to distribute courses across the week and campus so as to minimize the building wait times while respecting student course conflicts. We performed a comprehensive evaluation by using the Fall 2019 course registration data for a large US university. Our data demonstrate that the LBCS algorithm reduces the building wait time up to $3\times$ compared to actual pre-pandemic university scheduling.

VIII. Acknowledgments

This work was supported by a Provost's New Strategic Directions in Research and Scholarship Award at USC. We are grateful to the collaborators who provided the university data used in this work.

References

[1] "Tracking Covid at U.S. Colleges and Universities," https://www.nytimes.com/interactive/2020/us/covid-college-cases-tracker.html, accessed: 2020-10-12.

[2] "Map: Coronavirus and School Closures in 2019-2020," https://www.edweek.org/ew/section/multimedia/map-coronavirus-and-school-closures.html, accessed: 2020-10-12.

[3] V. Duong, P. Pham, T. Yang, Y. Wang, and J. Luo, "The ivory tower lost: How college students respond differently than the general public to the COVID-19 pandemic," *arXiv preprint arXiv:2004.09968*, 2020.

[4] "Impact of the COVID-19 pandemic on education," https://en.wikipedia.org/wiki/Impact_of_the_COVID-19_pandemic_on_education, accessed: 2020-10-12.

[5] O. B. Adedoyin and E. Soykan, "COVID-19 pandemic and online learning: the challenges and opportunities," *Interactive Learning Environments*, vol. 0, no. 0, pp. 1–13, 2020.

[6] D. C. Barton, "Impacts of the COVID-19 pandemic on field instruction and remote teaching alternatives: Results from a survey of instructors," *Ecology and Evolution*, vol. 10, no. 22, pp. 12 499–12 507, 2020.

[7] A. Hekmati, M. Luhar, B. Krishnamachari, and M. Matarić, "Simulation-based analysis of covid-19 spread through classroom transmission on a university campus," 2021.

[8] R. W. Bohannon and A. Williams Andrews, "Normal walking speed: a descriptive meta-analysis," *Physiotherapy*, vol. 97, no. 3, pp. 182–189, 2011.