**Optimal Class Deployment**

Group 5

 (Yutaro Sakai, Alexis Younkyung Shin)

**Introduction**

 Getting to class on time is crucial to a student’s focus throughout a whole lecture. The Carnegie Mellon campus in Pittsburgh is a relatively small sized campus, but there are still some unfortunate events where students do not have enough time to get to class because the buildings of two successive lectures are just too far away to reach within 10 minutes. Thus we have decided to explore this problem for possible solutions by assigning classes appropriately to the buildings on campus according to the designated timetables of the students. We figured that this might be a good place to start rather than to mess with the timetable itself, since that will very likely involve more complications.

**Benefits**

If fewer students are late for class, it will save a whole class from frequent head turns to see who just came in. It will mean less distraction for the professors as well. In normal contexts, one or two minutes might not seem much, but it is a very different story if they are between two consecutive classes. If students can come to class a minute or two early and get ready for class then the class can start right away. If professors can spare one or two minutes to finish explaining what they were explaining they will not have to start over in the next lecture just to mention that last point he/she could not make during the previous lecture. If students cannot go to the bathroom between classes in order to be at a class, and they end up leaving class to go during the lecture, this is not only a distraction to the individual student but to the whole class. Making quick important phone calls or printing documents out before lecture can make a lot of differences in an individual’s life. Little as it may seem, these minutes may add up to have a large effect cumulatively.

**Integer Problem Formation**

 We will try and minimize Z, which will denote the cumulative travel time for all the students. (Travel time: The time elapsed from when the student leaves the classroom to when he/she enters the next classroom) To achieve this, we form the following integer problem.

*Z* : total travel time for all the students (course k is in building p)

$$minimize Z=\sum\_{i=1}^{n}\sum\_{k=1}^{m}s\_{ik}\sum\_{\begin{array}{c}l=1\\l\ne k\end{array}}^{m}f\_{kl}c\_{lq}t\_{pq}$$

List of variables

$m$: number of courses ($k,l$)

* $n$: number of students ($i,j$)
* $b$: number of buildings ($p,q$)
* $t\_{pq}$ = travel time from building *p* to *q*

Integer Variables (0 if other than the stated conditions)

* $s\_{ik}$ = 1 if student *i* is taking course *k*
* $f\_{kl}$ = 1 if courses *k* and *l* are *respectively* *consecutive* for a given student’s time table

($f\_{kl}$ and $f\_{lk}$ are not the same)

* $c\_{kp}$ = 1 if course *k* is in building *p*

The constraints for this problem will be:

1. $\sum\_{p=1}^{b}c\_{kp}=1 \left(A given course should be assigned to only one building\right)$

$$2.\sum\_{l=1}^{m}f\_{kl}=1 for a given k $$

(For a given course, there will be only one course that follows)

$$3. For any given f\_{kl}, t\_{pq}<10$$

(All students should have travel time that is less than 10 minutes)

We used Java to implement a program that would find us the solution of this integer programming problem. However, we realized that we could not solve this problem through such brute force, iterative measures since the problem is considered a ‘Quadratic Assignment Problem’. QAPs such as our problem are known to be NP-hard (‘as hard as’ an NP problem, where NP stands for nondeterministic polynomial-time. The runtime solving these problems are known to be in the superpolynomial range.) Therefore, we decided it was more practical to devise a different algorithm to find other suboptimal solutions. This alternative approach will be explained in the next section.

**Alternative Options**

Due to the fact that solving our linear program for the general case is unsolvable due to the fact that it is NP-hard, we had to look for another way to solve our sample problem. First, we had to come up with a way to calculate the total travel time. The calculation was simple, as we would simply multiply the number of students going from some building to another by the estimated time needed to travel between those buildings. Then we would multiply by the number of days that course meets since we are trying to find the total travel time spent per week. We would do this per every timeslot and every building. In order to do so, it was necessary for us to make a table of the estimated travel times between the academic buildings. Also, since we were not able to access the data of how many students from a certain class goes to another math class, this data was made up for our solution.

The approach we took to find the optimal schedule was to start off with some initial schedule and consider some possible swaps that can be done. By swapping, we mean swapping the locations of a pair of courses. For example, if course A is in building C and course B is in building D, when we swap course A and course B, we would consider the case where course A is in building D and course B is in building C. Then we would calculate the total travel time for this new schedule to see any improvements. By looking at all the possible combinations of possible swaps within the same class-time, we would find out what schedule would result in the lowest travel time. Since just randomly swapping two courses without any constraint would result in an excessive number of swaps, we decided to put some conditions of a “possible” swap. A “possible” swap would be such that two courses are in the same capacity group, not in the same building initially and offered on the same days. A capacity group is to take care of the capacities of each course. For example, a course with 200 students should not be swapped with a course with 30 students, as that would be unrealistic. Therefore we defined group 1 as a group of courses with 1 to 30 students, group 2 as a group of courses with 31 to 70 students, and group 3 as a group of courses with 71+ students. Also, courses that are offered in the same building should not be considered as that would not change the travel time. Lastly, two courses that are not offered on the same days cannot be swapped, as that has the possibility of not meeting the requirements for a course, such as not enough lectures, etc.

 However there were a number of key assumptions made in order to solve for our sample problem. We only considered the 50-minute math courses offered on Mondays, Wednesdays, and Fridays during the Fall 2010 semester. There were 39 such courses. Also, since we could not access the data for each course, we assumed that every course was full to their max capacity. During our process, we treated different sections of the same course since different sections would have lectures at different times in different buildings. Also, for the travel times, we assumed that a student who do not have class next timeslot or a student going to a class from no class is considered as not taking any time since we do not know where they would go and they would have time to go to their next course.

**Optimal Solution**

Given these conditions, we attempted to get an optimal schedule that would decrease the total travel time spent per week. For the travel times, we just the following table:



The red numbers are for times that take more than seven minutes.

Also, the following table lists an estimate of how many students are going to a course that is offered during the next timeslot: 

This data was estimated by consulting the undergraduate catalog and the suggested course schedules. (The numbers were also manipulated in order to make the results/iterations more ‘interesting’.)

The first element in the bracket is the number of students who do not have class after a certain course. Then the rest of the elements represent the number of students going to the first course listed in the next period, second course listed in the next period, etc.

 Using these data and our algorithm to calculate the total travel time per week, the initial schedule resulted in 6803.7 minutes. However, running our Java program that looks at all the possible combinations of possible swaps and calculates the travel time for each combination, this travel time can be reduced significantly. The following is the initial schedule with the arrows indicating the swaps that should be made to obtain the optimal schedule.

 (HH: Hammerschlag, Mar Mor: Margaret Morrison)

As shown in the table above, five two-way swaps and one three-way swap would result in optimal schedule. After these swaps have occurred, the final travel time would be 5527.8 minutes, which is a decrease of 1275.9 minutes. This is nearly 19% decrease, which is rather significant. However, note that these results are based on estimated data and for a very limited pool of students and courses. With real data and more courses, the results might turn out to be not so significant.

**Possible Improvements / How to Extend the Project**

There are many improvements that can be made for this sample problem. One thing that can make this solution more realistic is to get more accurate data. The travel times are estimates depending on our personal experience and the course selections are estimates depending on our personal experience and the undergraduate catalog. The data of the number of students going from a certain course to another can have a significant effect since that is one of the main variables that our solution depends on. Concerning the travel times, our solution can be improved if we take into account the time from the different classrooms within the same building. As of now, we assumed that it would take a minute moving within the same building. However, there is a case where a student must go from the lowest floor to the highest floor, or a student has two courses back-to-back in the same classroom.

 Also, we disregarded the recitations for each lectures as that would have given us about 50 more courses to be included in our program, and we did not have enough time to do so. Therefore, by regarding each recitation as a separate course, we would be able to find the optimal solution and consider the courses offered on Tuesdays and Thursdays. Moreover, there were a couple of 80-minute lecture courses held on Tuesdays and Thursdays that were not included in our sample program as we were not really sure how to include them in the program. Therefore, finding such way could increase our solution, even though just looking at math courses, it would not really make much impact since there are very few courses taught on Tuesdays and Thursdays.

 It would also be better if we can fix some classes to a classroom to begin with and disregard those classes for our swaps. This can be done with introductory level courses with large numbers of students. In this manner we also need to fix classes that need special facilities such as lab classes or sports classes.

 To extend this project, one can include the courses from other departments. For our sample problem, we only included math courses, and many non-Math-majors take math courses as well. Therefore, our sample problem is not very realistic. We can also just consider all the courses offered in Carnegie Mellon University this semester and find the optimal schedule regarding all of the courses. However, considering that our program took about two minutes to run with only 39 math courses, it may take a very long time to get the optimal solution.

**Conclusion / What We Have Learned**

 As students of math, we must always be prepared to try different approaches to a given problem. Through this project, we learned that just because the optimal solutions are hard or impossible to find, does not mean that we should give up on the problem. There will always be the ‘optimal’ suboptimal solution, and we just need to figure out ways to work around our limitations.