Spring 4-27-2022

# Optimizing Class Access - OwlGo

Matthew Pendley

Bright Uchehara

Renji Maliakal

Jorge Solorzano

# Optimizing Class Access – KSU Marietta Campus OwlGo


**ISYE 4900 - Senior Design**
**Dr. Adeel Khalid**

**Renji Maliakal**
*Data Analyst*

**Matthew Pendley**
*Project Manager*

**Jorge Solorzano**
*Project Engineer*

**Bright Uchehara**
*Engineering Coordinator*

## Executive Summary

Kennesaw State's Marietta campus is situated on 230 acres. Classes are not located in buildings that are the most effective and efficient for both students and professors. The spread of classes, as well as the size of the campus, results in long walk times, up to 10 to 12 minutes when walked at a normal pace (3 to 4mph) to or from class from certain parking lots. The optimization of class locations and walkways will reduce the travel time for single major students and professors by half. Kennesaw State's current infrastructure will be utilized to minimize costs.

Travel time is reduced by utilizing an assignment problem that has values assigned based on the location of each potential classroom location. The weighted variables are given a value according to its designated parking lot, reassigned major department building, and assigned classroom building for each class. The variables OwlGo utilizes includes parking lot, department building, classroom building, timeslot, and day of week, which is used to create a five-dimensional array that contains all the initial coefficient based on location combination. Information is input using a user filled excel sheet. The information is processed, and constraints are used to transform the five-dimensional coefficient array into a six-dimensional value array that can then be utilized for solving the problem. Additional constraints can then be added to ensure classrooms are assigned based on its department.

A major aspect of the project is the rearrangement of classes based on the department in which the class is under. This will result in a class schedule that allows for upper-level commuters or commuters that are taking major focused courses to reach their classes in less time. starting positions to reduce the variability of class locations. A current issue that some students experience is that their classes are in different buildings, even though the classes belong to the same major. This problem can be solved with the use of OwlGo.

# List of Contents

.

## List of Figures



## List of Tables

# Chapter 1 Preface

## 1.1 Introduction

Kennesaw State University's main campus is well organized and consists of buildings that are designated for specific colleges and majors. This results in an optimal environment for the student body and allows for optimal transportation to class from dorms or parking lots and in-between classes. However, the Marietta Campus differs from the main campus. The campus primarily consists of engineering majors and was purchased by Kennesaw State University from Southern Polytechnic State University. Due to the transaction, the infrastructure to and from class is not of the standard of the main campus, entire buildings are designated to niche majors, and class schedules are distributed in a way that results in longer travel times. The problem mainly affects students, because they have to walk miles to get from their designed parking lot to the building, they have class in, no matter how apart they are from each other. OwlGo intends to resolve the issues both students and professors face daily by rearranging course locations and housing similar majors under the same buildings, assigning parking locations to allow for students to reach their major courses within five minutes, all while implementing new infrastructure for students and faculty to reach their destinations faster and safer.

## 1.2 Overview

Kennesaw State University is continuously increasing the number of majors and courses offered on the Marietta campus. Because of this, the student population at Marietta is growing at a rapid pace. The resulting course timetable has yet to utilize the current facilities and infrastructure efficiently, leaving some of the smaller majors designated to some of the larger buildings. Rather than accommodating student travel time, students are often forced to walk back and forth across campus just to attend their major-related courses. As time is a key factor for student efficiency, the outlier in our university's success becomes satisfaction for the student. The average student should be able to walk to their department building from their designated parking lot within five minutes. They should also be able to make it to and from additional major related classes within two minutes and still be able to make it back to their parking lot within another five minutes.

## 1.3 Objective

- Create and utilize infrastructure that minimizes travel time through optimal routes.
- Move major-related classes under same building where possible (or <2 min walk).
- Recommend a parking lot that best suits each student's schedule (<5 min walk).

## 1.4 Justification

Pleasing the students should be one of the main objectives for Kennesaw State as a university. The happier the students, the higher the student approval and success rates. These not only make the university look better, but more students are going to select KSU as their college of choice. Since one of the biggest complaints with the Marietta campus is the course location

distribution, the need for bringing the campus up to standard with the Kennesaw location becomes blatant. The more time that students spend walking on campus, the less time there is for instruction and studying. Grouping major-related classes into designated buildings would benefit the students and teachers alike. In addition, ensuring a safe walk to class should be prioritized, regardless of what objectives or constraints are at hand.

Another point worth mentioning is how much more utilization (of classrooms) can be achieved in each department building if the class locations were better arranged. As students who have been on the campus for years, the researchers have noticed a need for change within the course scheduling. Some classrooms are rarely used whereas others only utilize half of the room capacity with a given course.

## 1.5 Project Background

Arriving at campus with a personal vehicle requires a parking pass designated to a specific lot that is not interchangeable. After parking and attending classes on opposite sides of the Marietta campus, a simple walk to class becomes traveling long distance. Classrooms are assigned below an optimal capacity whereas this is not as common on the Kennesaw campus. Another distinction between the two is the grouping of majors into designated buildings at the Kennesaw campus, whereas major-related classes are more spread in Marietta. The researchers desire a system that will allow major-specific students to park and learn in a vicinity that is efficient, by minimizing both the walking time to classes from a designated parking lot and walking time between classes.

## 1.6 Problem Statement

The Marietta campus of Kennesaw State University is roughly 0.9 buildings for every class based on a survey of 26 mixed engineering students, with an average major-related class count of 3.35 and a standard deviation of 1.32. This is a result of the different majors being spread across multiple buildings and leads to a diminished social and learning environment for students of the same major. The spread of ideas and teachings at a student-to-student level is not effective given the current infrastructure and class schedule. To bring a solution to this problem, our team is applying our knowledge in optimization, linear programming, Python coding, and input analyzer to find a solution where students and teachers will not struggle to walk between buildings and parking lots, and hopefully, there will be a better use of all the facilities.

# Chapter 2 Literature

## 2.1 Challenges of Course Scheduling

Course scheduling is recognized as a challenge by institutions and universities around the entire world. This is no new problem, but rather has been evaluated for years as organizations try to utilize their current facilities and resources as efficiently as possible. The issue, however, is becoming more complicated as student enrollment and course availability continues to grow [1].

Whether being evaluated from an economic/sustainability standpoint or based on professor/student preference, both approaches must consider the current capabilities of the university. This would include the available meeting times, classroom/lecture buildings, parking capacities, etc. One way to manage this data is by encoding it as an array. Doing so allows for the data to be manipulated more easily and across several dimensions [2].

Additionally, most studies do not explicitly account for the relationship between the course-scheduling and the parking demand. However, according to a case study done at the University of Louisville, optimizing course schedules always plays a viable role in managing the campus parking supply. By strategically assigning parking based on student registration, course assignment may become much more direct and user-friendly [3].

## 2.2 Excel for Data Management

Spreadsheets have been used for data processing ever since Excel, Minitab, and other software were created. When dealing with large groups of numbers, the potential for making errors increases as data set sizes increase. While not only ensuring quantitative data, but Excel also provides qualitative data by the variety of visual representations that are available. In comparison to other software, Excel is “just right,” as the functionality is between too easy and too hard [4].

Spreadsheet programs, such as Excel, allow each cell to be interacted with and manipulated independently. On the contrary, these cells can be combined to create a dependent formula, capable of solving a variety of problem types. With the ability to sort and filter large data sets, Excel is key to performing quick analysis [5].

## 2.3 Course Assignment Problem

The university course timetabling problem, otherwise known as UCTP, has been addressed by many academic researchers. Despite the effort, this issue is being confronted daily as the number of soft and hard constraints between each university will vary. Hard constraints are those that must be satisfied by the solution, whereas soft constraints are still highly favored, but not necessarily required by the solution. Because of the unique requirements of each university, researchers must determine which factors can be measured and interpreted within the system. One university might prioritize effective resource management over the requirements of the business. As with OwlGo, some universities will choose to prioritize the students through higher satisfaction and convenience of course scheduling [6].

UCTPs are often classified as having two distinct phases, the initial construction phase that lists all the workable solutions and then the optimization stage that selects the best out of all the possible solutions [7]. When beginning the course scheduling problem, researchers should maintain an adequate amount of utilization through enrollments. This is done to avoid wasting the current educational and physical resources. Professor workloads should remain evenly distributed if they are to be adjusted in the formulation. To mitigate these disruptions, capacity limits are often placed on each course section offered by the university. The result is a higher level of educational quality, but not always the highest level of student satisfaction. Students are forced to enroll in what is available at the time rather than what is preferred. Fortunately for the researchers, assigning compulsory courses is easier to manage due to the demand being more consistent from year to year [8].

## 2.4 Python as an Optimization Tool

The functionality of Python is credited with having an elevated level of abstraction. This allows users to implement the fundamental concepts of an algorithm with relative ease. For example, simple algorithms may be solved using only two functions, [*input, print*]. Rather than having to learn and express a large array of syntax, Python puts the attention of the user on problem-solving [9].

When tackling the assignment problem using Python, Reia Natu explains the process clearly with an example problem of four machines needing distinct job assignments. To begin, the user must first identify the sets to be assigned. In the researcher's case, this would be parking lots, the designated building for major classes, course locations by room, and the day/time slots for each. The next step in Python is inputting the data that will be used to determine the best solution. Included in the researcher's user-collected data are the transit times between each building and the transit times from each parking lot to each building. Once the model is defined, the user will then define the variables, which will be the specific locations to be assigned for each class, represented as either a 0 (does not meet constraints) or a 1 (meets constraints for given location).

Setting up the objective function comes next, and in the case of the researchers, it is aimed at maximizing the scores given for the building-to-building and parking lot-to-building data. Once this is established, all that remains is inputting the constraints that are to be satisfied in the algorithm. The specific constraints to be used by OwlGo will be discussed later in the problem-solving section. By issuing the '*print*' command on the defined model, Python will then find an optimal solution that is confirmed using sensitivity analysis [10].

# Chapter 3 Define

## 3.1 Problem Solving Approach

1. What is the overall goal? Assigned parking lot to department buildings or classroom is within a five-minute walk, and department buildings to classroom is within a two-minute walk. Given user entered variables, which are listed in step 2, OwlGo will produce an optimized class schedule containing the following results:
    a. parking lot a department will primarily utilize
    b. department building, where classes within a department will be centralized around
    c. classroom location for each class
    d. time(s) in which each class occurs (not subject to change)
    e. day(s) of the week each class occurs

2. Using the Dynamic Schedule for Spring 2022, the researchers were able to extract the individual class times, class durations, and class sizes into a separate Excel sheet. Once the major-related courses were sorted by department, these values became the potential input variables for the assignment.

3. The OwlGo system will use a supply and demand approach, where the supply is determined by the potential combinations of the five variables: parking lot, dept. building, classroom, starting time, and days of week.

4. The next step was to define each of the variables
    a. Parking Lots
        i. Lot 51
        ii. Lot 35
        iii. Lot 36
        iv. Lot 37
        v. Lot 38
        vi. Deck 60
        vii. Deck 22

    b. Department Buildings
        i. Academic
        ii. Architecture
        iii. Atrium
        iv. Civil
        v. Crawford Lab
        vi. Design 1
        vii. Design 2
        viii. Engineering Technology Center
        ix. Joe Mack Wilson Student Center
        x. Mathematics
        xi. Science Lab Annex
        xii. W. Clair Harris Textile Center

   c. Classrooms
      i. (Found using Spring 2022 Dynamic Schedule (NOT full list of classes))

   d. Time
      i. 6:15 to 10:45 in increments of five minutes

   e. Days
      i. Sunday to Saturday

5. Determine the Building-to-Building values. The coefficients used will be created using two separate arrays. The first is a Department Building to Classroom Building array. This will determine if the buildings are within or outside of a two-minute walk. If the building-to-building time is outside two minutes, it is given a value of zero. If the value is within two minutes, it is given a value greater than zero. If the classroom is occurring in the department building, it will be assigned a value greater than the value given for building-to-building within two minutes.

6. Determine the Parking-to-Building values. The second array is Parking-to-Building. This will determine which buildings are within or outside the five-minute time limit. If it is outside five minutes, the value will be 0 to not utilize this building. If the value is within five minutes, it will be given a value of one.

7. Assignment problem formulation, please see Chapter 4 for detailed approach.

8. Create a multidimensional array to represent all the possibilities for the different class variables. (Five dimensions)

9. Use user input to create an array that provides the starting time, duration, days of week, class size, and department to narrow down the possibilities of combinations each class can utilize.

10. Create constraints to better narrow down the number of viable solutions.

11. Create and run a loop that will be used to loop through each possibility and find a maximum for the solution. Create a loop that will act like Tabu search.

In order to best solve the problem, OwlGo uses a multidimensional assignment problem.

$$maximize \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} \sum_{m=1}^{n} \sum_{p=1}^{n} C_{ijklmp} x_{ijklmp}$$

$$\sum_{i=1}^{n} x_{ijklmp} \leq 1 \quad \forall j,k,m,p = 1,\dots,n \quad (classes)$$

$$\sum_{j=1}^{n} x_{ijklmp} \leq 1 \quad \forall i,k,l,m,p = 1,\dots,n \quad (classrooms)$$

$$\sum_{k=1}^{n} x_{ijklmp} \leq 1 \quad \forall i,j,l,m,p = 1,\dots,n \quad (department\ buildings)$$

$$\sum_{l=1}^{n} x_{ijklmp} \leq 1 \quad \forall i,j,k,m,p = 1,\dots,n \quad (parking\ lots)$$

$$\sum_{m=1}^{n} x_{ijklmp} \leq 1 \quad \forall i,j,k,l,p = 1,\dots,n \quad (timeslots)$$

$$\sum_{p=1}^{n} x_{ijklmp} \leq 1 \quad \forall i,j,k,l,m = 1,\dots,n \quad (days)$$

$$x_{ij} \in \{0,1\} \quad \forall i,j,k,l,m,p = 1,\dots,n$$

*Figure 1 - Assignment Problem Model with 5 Dimensions*

This formulation above represents the mathematical formulation for the problem which OwlGo is attempting to solve.

## 3.2 Requirements

- OwlGo shall have students reach class in less than 5 minutes (on average) from their parking spot while walking at a 3-4 mph pace.
- OwlGo shall prioritize classes within the department building for specific and those within a 2-minute walk.
- OwlGo shall find a specified parking lot in which departments will primarily utilize.
- OwlGo shall find a specified department building that will house most of the department's classes.
- OwlGo shall utilize the previous infrastructure on campus and shall add infrastructure if there is a potential to increase the possibilities for the classroom supply combinations.
- OwlGo shall not move any specific equipment rooms.
- OwlGo shall consider class capacities when rearranging class locations (class size >= ½ * classroom capacity).

## 3.3 Gantt Chart

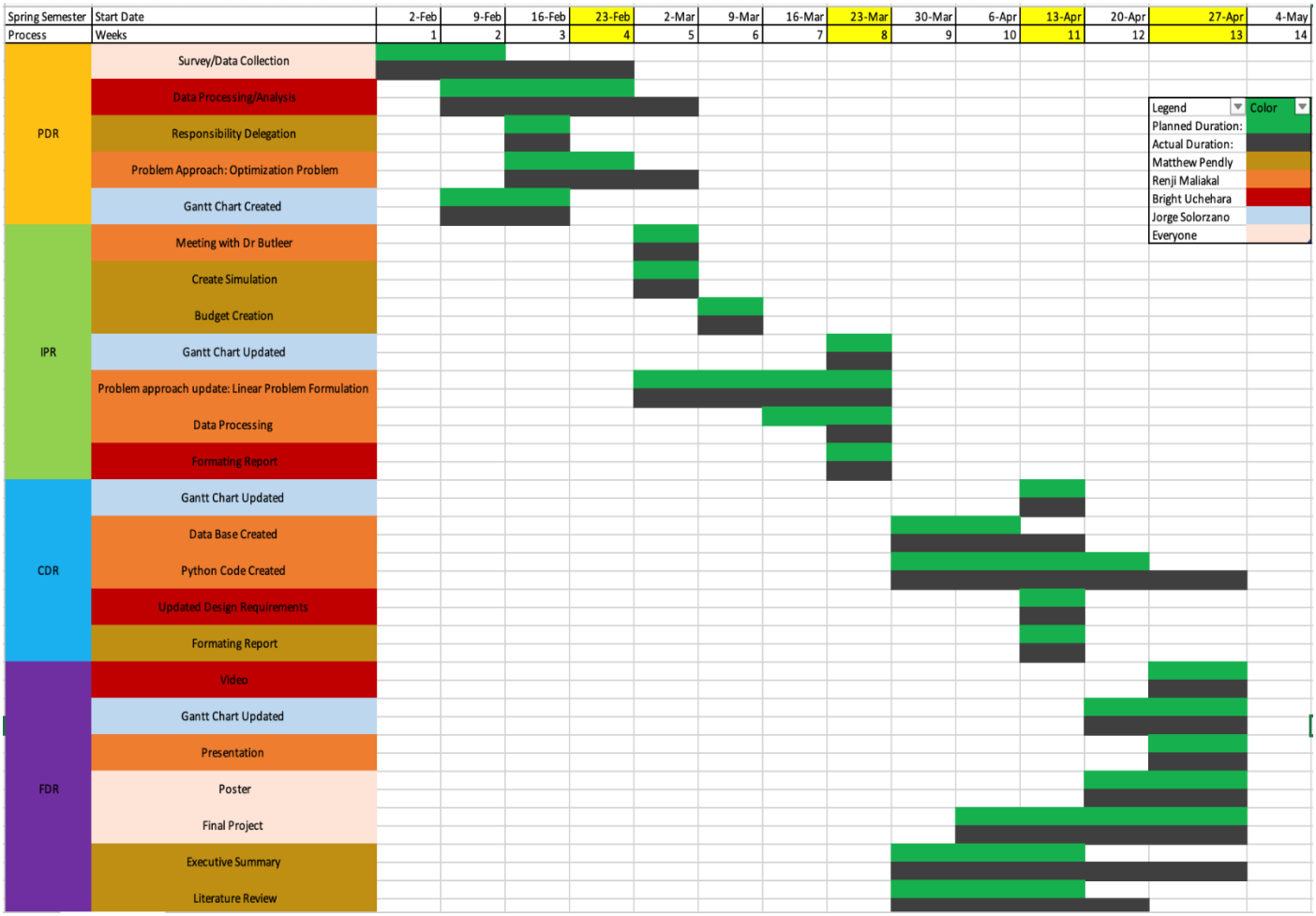



*Figure 2 - Gantt Chart of Major Developments*

Above is the Gantt chart that reveals the scheduling for this project. This chart is a visual representation of when the work for the project will be completed. This version of the Gantt chart is subject to change as the project progresses and responsibilities are designated. It may occur that some assignments take longer than expected. Also, as is stated in the chart previously mentioned, it is stated how the job load will be distributed among the team members. It is important to mention that there will be tasks that will be performed by everyone

## 3.4 Flow Chart

The flow chart below represented in Figure 2 is a visual depiction of the steps a student takes after arriving on campus for a class. This chart has been simplified to avoid any potential complications in the course arranging process.

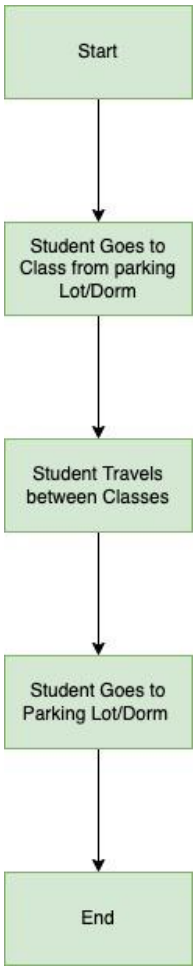



*Figure 3 - Flow Chart of Student Interaction on Campus*

## 3.5 Project Block Diagram

The flow chart below is the project block diagram for the solution method. First, the students will measure the times to and from each building. The time in between buildings will be remeasured once new infrastructure changes have been implemented, as well as the class rearrangements and parking reassignment.

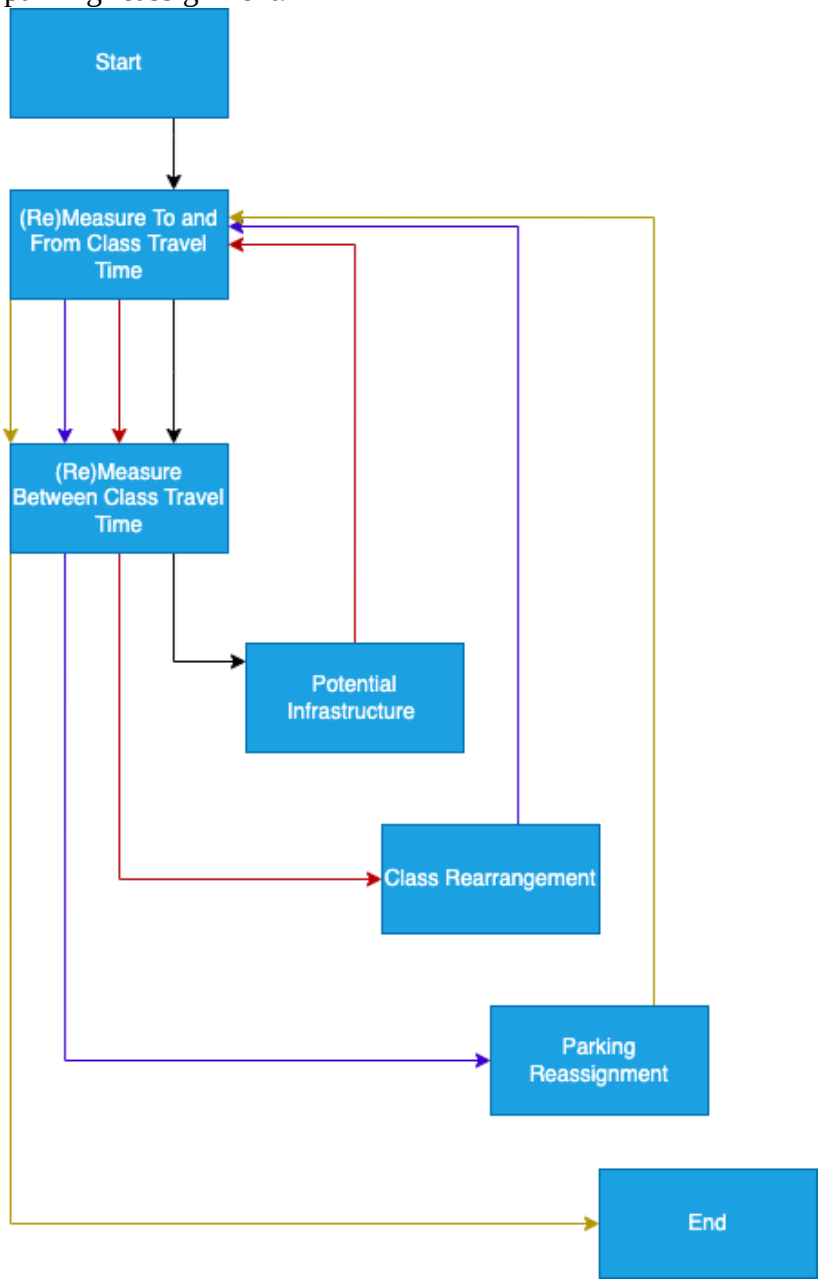



*Figure 4 - Process Block Diagram for Problem-Solving Strategy*

## 3.6 Project Management

To complete the project by the scheduled due date, the team will need to meet online for several days out of the week for several weeks until the bulk of the computation and analysis is complete. Once data collection is complete, the majority of time can be directed on discussing potential solutions and conducting independent research. Considerations about who to contact will be discussed, such as professors or board members.

As project manager, Matthew will be discussing potential infrastructure changes with campus executives. According to the survey, many students are claiming the need for additional crosswalks and stop signs across campus. These additions will need to be discussed and approved by the university; however, similar additions and improvements are made by other colleges around the country.

These executives are the decision makers for the problem, but the students at KSU's Marietta campus are the main influencers and end-users of this project. It will be their decisions and support that will allow the proposed solutions to come to fruition. Surveys were conducted not only to gain insight on student issues with campus scheduling and locations, but also to ensure that they are the top priority for this project.

## 3.7 Responsibilities

As project manager, Matthew will be in charge of coordinating meetings and keeping team members up to date with responsibilities and understanding. Matthew will conduct the majority of the literature review and research. In addition, Matthew will assist in data collection, report formatting, and resource management

As data analyst, Renji will oversee the problem formulation, contact facilities department for data, physical data collection, spreadsheet synthesis, information processing, data analysis, coordinate and attend meeting with Dr. Renee Butler, and create the solution algorithm using Python. Also included are the draft of the schedule and process flow diagrams, as well as the facilitation of project documents, PowerPoints, and resources.

As project engineer, Jorge will work on designing the visuals and presentations for the project. The Gantt chart will be assigned to Jorge, in addition to the sourcing for applications (Arena, Lingo, etc.).

As engineering coordinator, Bright will facilitate the data collection with Renji. Bright will also discuss with potential leads on campus for more information. The budget will be discussed with Matthew to determine what resources can be reused or resold.

## 3.8 Budget

*Table 1 - Budget Table for OwlGo and Research Team*

| | OwlGo Budget | | | |
|---|---|---|---|---|
| | | Forecasted | Actual (Predictions) | Difference |
| Software Development | Python Software | | | $0.00 |
| | Research & Analysis | $1,800.00 | $2,100.00 | -$300.00 |
| | Formatting and development | $4,000.00 | $1,050.00 | $2,950.00 |
| | Trial runs | $1,350.00 | $1,500.00 | -$150.00 |
| | Additional Data Analysis | $0.00 | $300.00 | -$300.00 |
| | Application Marketing | $0.00 | $7,000.00 | -$7,000.00 |
| | *$150/hr rate for Software Developer* | | | |
| Construction | Additional Stairs including Signage | $120,000.00 | $99,200.00 | $20,800.00 |
| | Total: | $127,150.00 | $111,150.00 | $16,000.00 |

## 3.9 Material Required/Used

Below is a list of materials that will be needed for executing any campus changes (staircase and stop sign additions)

- Concrete stairs
- Tools
- Metal tubing (for rail)

## 3.10 Resources Available

Below is a list of resources that were utilized for this project thus far. More may be added to this list as the project progresses.

- Excel
- Google Maps
- Microsoft Teams
- Microsoft Word
- Microsoft PowerPoint
- SQL Database
- Python

# Chapter 4 Measure

## 4.1 Data Required

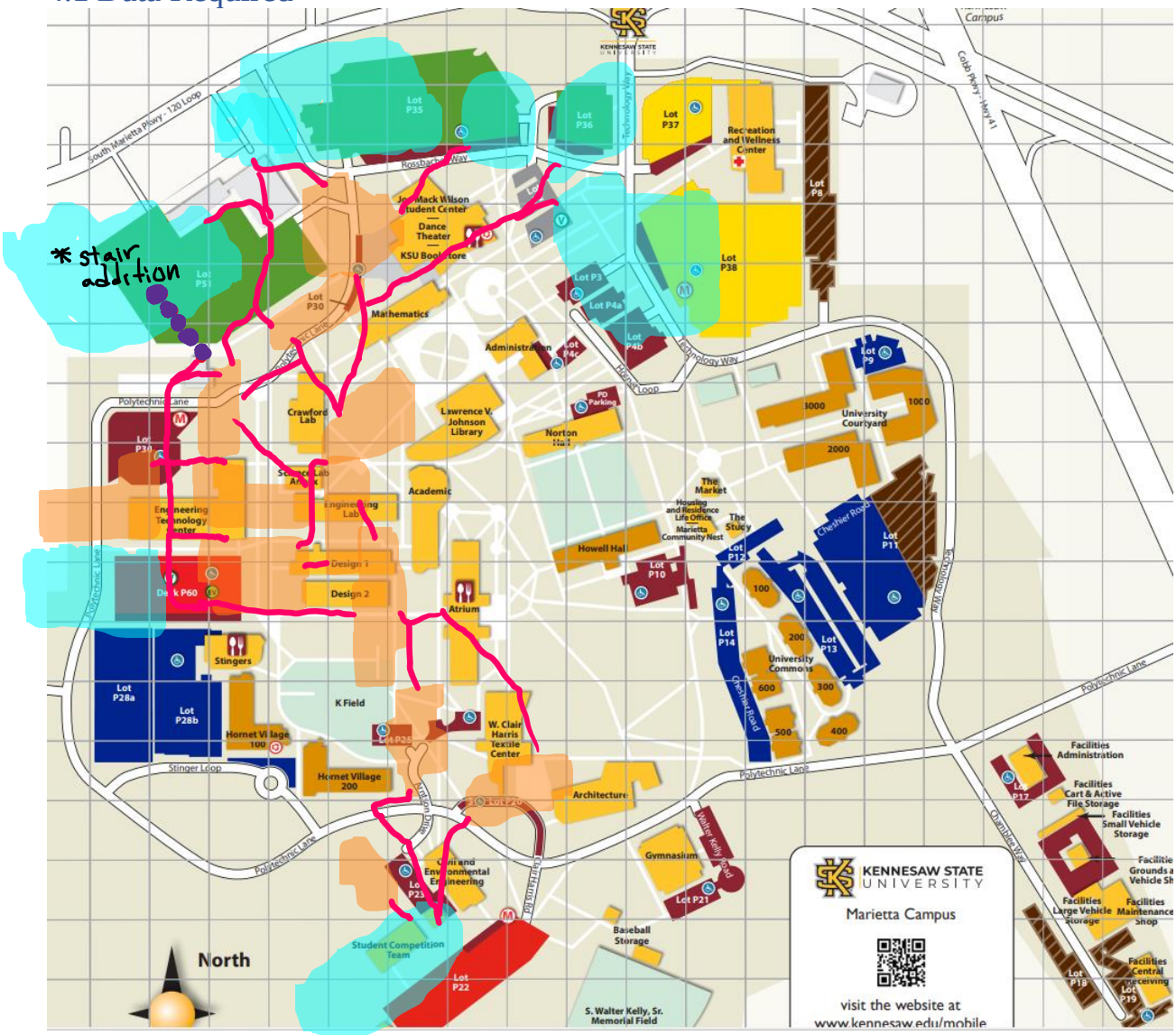


Figure 5 - Map of Recorded Walking Paths

Figure 5 is a visual representation of the paths that were timed while walking. On the map, the parking lots are highlighted blue, the lecture buildings are highlighted in orange and the quickest walking paths/hypothetical routes are marked with a pink line. The initial staircase addition to lot 51 has also been labeled as purple dots with a note. Figure 5 will help OwlGo establish the variables, constraints, and all possible combinations a student will have to walk while selecting their classes. In addition, this shows all the routes that were evaluated by the researchers and to be subject to the formulation.

While the researchers had started this process by manually timing each route between lots and buildings, Google Maps was used to verify the manual times but also determine the optimal

paths between each location. Once the researchers learned that Google Maps could list each step/turn as feet, it was used for the remainder of the distances.

### 4.1.1 Supply Data

Building-to-Building Array

The five key variables that are needed in terms of supply include the parking lots, department buildings, classrooms, class time slots, days. This data will be transposed from the create a supply array that will consist of:

# Possible Combinations = #lots * #buildings * #classrooms * # time slots * # days/week

OwlGo shall reduce the travel time between parking lots and department classes for each, and OwlGo shall also ensure that building-to-building times are within two minutes.

### 4.1.2 Demand Data

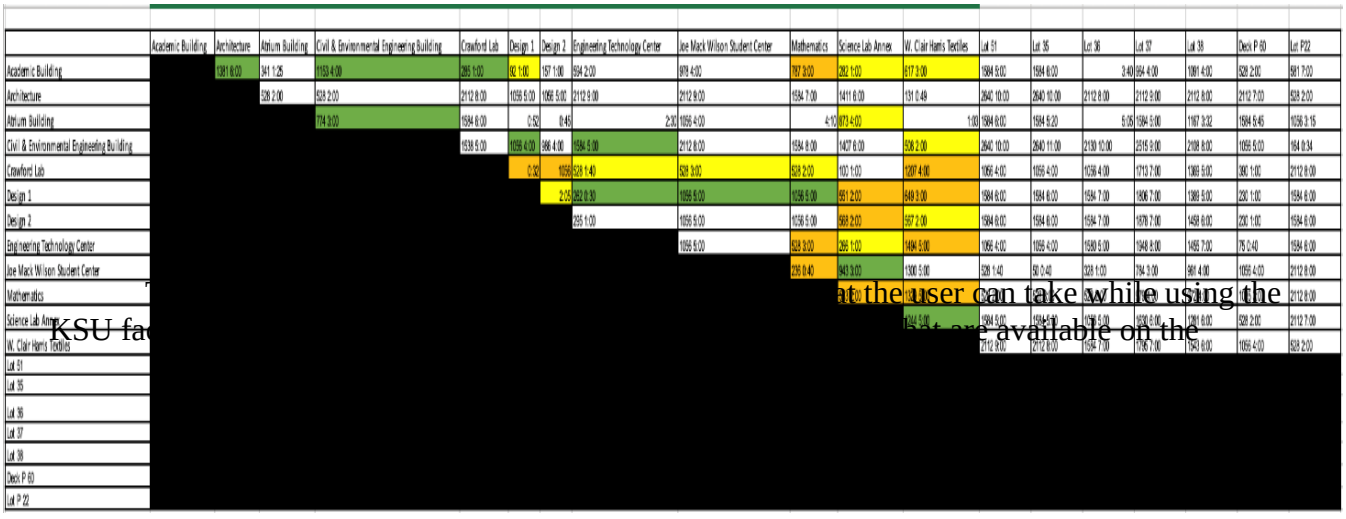
The demand will be determined with the use of an Excel sheet that is uploaded into the Python program. The data will consist of basic classroom information such as

- CRN
- Department
- Class Code
- Weekly Frequency
- Class Start
- Class Duration
- Class Size

The time values will be used to determine the time slots that will be populated from the supply array for each class's demand. The weekly frequency will be used to populate the supply array days for each class's demand. The class size will be used to ensure that class size >= ½ * classroom capacity. Special use and lab classroom information is also needed to ensure the specified classes have the required infrastructure and equipment to conduct lessons.

## 4.2 Data Collection

*Table 2 - Distance and Time between Buildings and Parking Lots (Marietta Campus)*



KSU fa at the user can take while using the t are available on the

premises. On each cell, there are two values, the first one represents the distance in feet between each point and the second one represents the time that it takes to arrive from one place to another. These are the values that were used in creating the functional database in Python.

*Table 3 - Transit Times Converted to Binary Values*

| | Academic | Arch. | Atrium | Civil | Crawford | Design 1 | Design 2 | ENGR Tech | Joe Mack | Math | Lab Annex | Textiles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Academic | 2 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| Arch. | 0 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Atrium | 1 | 1 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| Civil | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Crawford | 1 | 0 | 0 | 0 | 2 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| Design 1 | 1 | 0 | 1 | 0 | 1 | 2 | 1 | 1 | 0 | 0 | 1 | 0 |
| Design 2 | 1 | 0 | 1 | 0 | 1 | 1 | 2 | 1 | 0 | 0 | 1 | 0 |
| ENGR Tech | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 0 | 0 | 1 | 0 |
| Joe Mack | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 |
| Math | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 0 |
| Lab Annex | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 0 |
| Textiles | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |

| | Academic | Arch. | Atrium | Civil | Crawford | Design 1 | Design 2 | ENGR Tech | Joe Mack | Math | Lab Annex | Textiles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lot 51 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Lot 35 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| Lot 36 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Lot 37 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Lot 38 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| Deck P 60 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Lot P22 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

The time data was converted into two-dimensional arrays as shown above. The arrays created are building-to-building and parking-to-building. As Table 3 shows, there are either 0, 1, or 2 in the cell. These values represent whether the transit between the locations would satisfy the distance requirements. “2” means that the class is within the same building and a “1” means that the next class would in fact be within the 2-minute walking limit. A value of “0” would indicate that there were no possible routes to and from the buildings that would satisfy the distance limit. For the Lot data, a 1 indicated the building could be reached from that parking lot within 5 minutes, whereas a 0 indicated that it was beyond the 5-minute threshold.

### 4.2.1 Variable Data

The different variables data for the project were gathered using the Spring 2022 Dynamic Schedule. The data was transferred to an Excel spreadsheet and was transformed into a workable table. The data was then collected. Each classroom and the maximum capacity for each was found by grouping the rooms and taking the maximum value of capacity for each.

The data is in the hands of Kennesaw State’s facilities department, but they are reluctant to release the information for some reason or regulation. Gathering the most data helps to increase the margin for classes to comfortably fit within the schedule. However, given the information in the Spring 2022 Dynamic Schedule, a feasible solution should be possible.

## 4.3 Problem-Solving Method

OwlGo primarily utilizes the formulation of a multidimensional assignment problem as our solving method. Given this problem, we need to gather data regarding each of the variable types. We initially followed the procedures shown in Figure 3. After going through and collecting the data, we found that we can increase the range of potential solutions by adding a

staircase from the underutilized end of Parking Lot 51 towards the Q-building. After this, we had the total list of possible solutions. The solution for each class consists of five different solution variables: a designated parking lot, a department building based on major, the classroom assignment, the time slot, and the day of the week. Of these variables, we assigned a coefficient for each combination of parking lot, department building, and classroom building. This represents whether the relative locations meet the criteria of two minutes from department building to classroom, five minutes from lot to building, and five minutes from lot to classroom building. This results in a list of all the coefficients for the potential class locations which can be expressed below.

$$\sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} \sum_{m=1}^{n} \sum_{p=1}^{n} C_{jklmp} x_{jklmp}$$

Next, we need to create a set of constraints for each individual class so that we have a list of all the feasible solutions in relation to each class. This results in the cost array for the assignment problem. The constraints are determined by extracting information from the user inputs of class capacity, time slots, and days. A separate array is needed for each in relation to class with the same length as its respective array dimension. This is needing so the values can be multiplied across the array for each of the possible values to create the resultant assignment problem.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} \sum_{m=1}^{n} \sum_{p=1}^{n} C_{ijklmp} x_{ijklmp}$$

Ideally, this problem will result in the solving of our solution. However, python code does not typically allow for multidimensional assignment problems to be assigned using the “ortools” import. Due to this the five-dimensional cost matrix needs to be converted into a two-dimensional array that will be run by the program and results in an optimal parking lot, department building, and classroom location for each class. Further constraints can be utilized to ensure that department courses are placed together.

### *4.3.1 Initial Staircase Addition*

We have analyzed the benefits of adding a staircase and crosswalk to the parking lot that is on the Marietta campus and have concluded that it will benefit our overall goal. The location of this staircase will be on polytechnic lane ascending from lot 51. The exact location of this staircase is 33°56'21.8"N 84°31'19.3"W, which can be seen in Figure 6 below. Following the basic community designs, we see that the sidewalks that border lot 51 and the crosswalks that allow for safe passage to the buildings surrounding it are a promising idea. For our project specifically, there is an unnecessary delay when walking to either the engineering technology building area or the mathematics building area from lot 51. To eliminate this delay, we have decided to provide the area with a staircase. This will allow for fluid pedestrian traffic from lot 51 to the north-facing area of the Marietta campus. Students will be able to traffic to and from lot 51 and optimize the time that it will take them to travel to classes on the north side of campus.

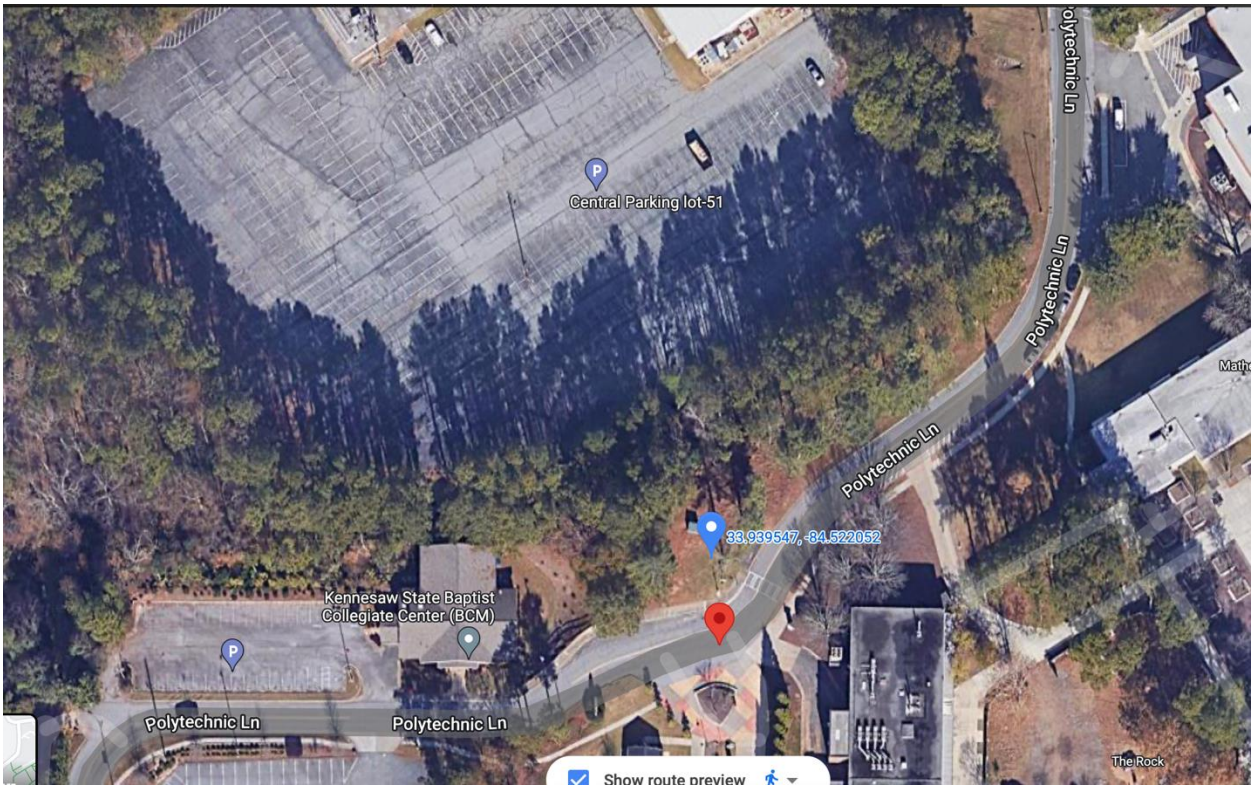



*Figure 6 - Initial Staircase Addition to Lot 51*

### 4.3.2 Creation of Supply Data Array

The two-dimensional array was converted into an array within Python by utilizing the building-to-building coefficient data [Code E.1, Appendix E]. The shape of this array is [12,12]. Each line will need to be converted into a [12, x] array where x is the number of classes per building. In this matrix, values are assigned to each location within the array. These values are assigned whether the building-to-building locations fit the criterion for being located within a two-minute walk from building to building.

Each building has a different number of available classrooms (found using Spring 2022 Dynamic Schedule.) Each one-dimensional building array is multiplied by a ones array with shape [1, x] where x is equal to the number of classrooms for the building the ones array is being multiplied [Code E.2, Appendix E]. The result is 12 arrays with differing dimensions [12, x]. The arrays are then stacked to add up all the x values to represent the total number of classrooms. The Marietta campus is utilizing 145 of its classrooms. The resultant array represents the Department Building-to-Classroom array.

The data from the two-dimensional parking-to-building data is broken down line by line for each parking lot [Code E.3, Appendix E]. Each one-dimensional parking-to-building array of

size [12,1] is then multiplied across the Department Building-to-Class Room (size [12,145]. The values are multiplied row by row. This results in an array that communicates which classrooms are within two minutes of the department building and which parking lots are within each building. The resultant array is three-dimensional with size [12,145,7]. The time array and days array are found within the code by creating a ones array for the length equal to the number of possibilities for the respective variable [Code E.4, Appendix E]. The resultant array's shape is [12,145,7,199, 7].

1. 12 = # Department Buildings
2. 145 = Total # of different classrooms available
3. 7 = # Parking Lots
4. 199 = # Time slots from 6:15AM to 10:45PM in increments of 5 minutes
5. 7 = # Days/Week (Sunday through Saturday)

The array represents all the possible classroom combinations (12*145*7*199*7 = 16,966,740 combinations. All the basic coefficients defined based on each location's proximity to its surroundings is given. The resultant array is five-dimensional and can be used to find specific values within the supply array. A search can be conducted by stating "print (Coefficient[a,b,c,d,e])," where "a," "b," "c," "d," and "e" represent the index positions for a specific value or location.

### 4.3.3 Demand Array through Excel

A user filled Excel file is uploaded into the Python code. This creates a readable table with the following columns: CRN, Department, Code, Credit Hour, Lab (Boolean), Weekly Day Schedule, Class Start Time, Duration, and Class Size. From this data we will extract data and setup constraints. The python code is utilized to extract the time values from the user filled Excel sheet [Figure E.1, Appendix E]. A "for" loop is created stating that for each row (class required) from the Excel sheet a function will be run. In this case, a nested "for" loop is used. The nested loop takes the starting time and using the number of consecutive time slots as the number of loops produces an array of all the time slots required for the specified class (row).

The capacities of each class are calculated as shown in Appendix E Code E.6 to determine whether they fall within the constraint of class size >= ½ * class capacity. This is done by creating a loop that checks each row's capacity. Another for loop is created, for each building's classes. It is run to check the class size against each of the class capacities for the building. At the end, the complete list of potential classrooms is formatted in a way where it coincides with the format of the classrooms within the array.

Once the demand arrays are found we can multiply each array for demand x class for all the workable solutions for each class to further constrain our array. This is done by assigning 0's for values we do not want and 1's for what we want to keep.

### 4.3.4 Python Assignment Problem Using "ORTools"

The assignment problem examples in Python are kept within two dimensions. Going outside of the two dimensions causes Index errors within the code. To counteract this, an array is

created that consists of the class by all the possible combination of parking lot, department building, classroom, time, and date. This array is expressed in Appendix E Code E.7; the array gives all the possible outcomes and creates a two-dimensional array from a multidimensional array and allows for the “ortools” assignment problem to be solved.

# Chapter 5 Analyze

The simplified code expressed, [Code E.8, Appendix E], shows the python formulation for the assignment problem. However, there are some issues with the code. Firstly, it is designed to handle smaller two-dimensional arrays; large arrays will run indefinitely. Secondly, the rearrangement of the five-dimensional array to be formatted into a two-dimensional array caused some problems in terms of formulation. The lengths of each element within a dimension must be the same. This posed difficulty when converting the information into a two-dimensional array. The size of the array also becomes an issue. The formatting for the constraints is not clear and is tricky to work with using the "ortools" import. Although the code does not list the classroom for where to park, it does suggest a parking lot for each class. This is due to the lack of a constraint based on department of each class. The mathematical formulation for the problem works consistently with the assignment formula OwlGo uses. The problem and total number of variables is a large problem to solve. The scale of the problem for the KSU – Marietta Campus had to be reduced for the problem to be run without the Python program crashing. The solution for our problem does determine a parking lot for a student to reach his/her intended classroom within five minutes.

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | CRN | DEPARTM | Code | Credit Ho | LAB | Weekly Fi | START | DURATION | SIZE |
| 2 | 13804 | ISYE | 3125 | 3 | 0 | T | 15:30 | 1:15 | 28 |
| 3 | 10472 | ISYE | 4500 | 3 | 0 | T | 14:00 | 1:15 | 28 |
| 4 | 10442 | ISYE | 4900 | 3 | 0 | W | 9:30 | 2:45 | 30 |
| 5 | 10445 | ISYE | 2600 | 3 | 0 | MW | 18:30 | 1:15 | 36 |
| 6 | 10455 | ISYE | 3200 | 3 | 0 | M | 15:30 | 1:15 | 36 |
| 7 | 10467 | ISYE | 3407 | 3 | 0 | T | 9:30 | 1:15 | 36 |
| 8 | 10468 | ISYE | 3600 | 3 | 0 | TR | 8:00 | 1:15 | 36 |
| 9 | 10469 | ISYE | 4200 | 3 | 0 | TR | 12:30 | 1:15 | 36 |
| 10 | 10470 | ISYE | 4250 | 3 | 0 | T | 17:00 | 1:15 | 36 |
| 11 | 10471 | ISYE | 4425 | 3 | 0 | R | 15:30 | 1:15 | 36 |
| 12 | 10454 | ISYE | 3150 | 3 | 0 | R | 18:30 | 1:15 | 40 |
| 13 | 10456 | ISYE | 3350 | 3 | 0 | M | 17:00 | 1:15 | 40 |
| 14 | 10466 | ISYE | 3400 | 3 | 0 | TR | 11:00 | 1:15 | 40 |
| 15 | 11885 | ME | 1311 | 3 | 0 | TR | 17:00 | 1:15 | 23 |

*Figure 7 - List of courses to be assigned from simplified model*

# Chapter 6 Results

OwlGo works in finding defining a parking lot for a student to park. This location ensures that the student is able to get to their class within five minutes from this location. This is an improvement from the up to twelve-minute walk. In terms of the formulation OwlGo is a success, but the problem OwlGo runs into is the scope of the problem. The size of the problem does not allow for all possibilities of the problem to be run through the code. Even after reducing the problem using less values, the conversion of the five-dimensional array into a two-dimensional array is difficult to follow. OwlGo produces results that need to be decoded to determine which parking lot, department building, and classroom to utilize. Some setbacks in the time and coding efficiencies resulted in the code lacking a vital constraint that could be added to ensure that the two minutes between classes are ensured. This code would have made each assignment dependent upon the other assignments. Currently the values are running independent of one another, which is why OwlGo works. Setting a constraint code that would make each assignment dependent on one another could have helped to better solve the problem and ensure the two-minute in-between class time problem is solved.

```
#Print the solution
if status == pywraplp.Solver.OPTIMAL or status == pywraplp.Solver.FEASIBLE:
    print(f'Total cost = {solver.Objective().Value()}\n')
    for i in range(num_classes):
        for j in range(num_classrooms):
            # Test if x[i,j] is 1 (with tolerance for floating point arithmetic).
            if x[i, j].solution_value() > 0.5:
                Answer_Array = [i, j]
                print(f'Class {i} assigned to classroom {j}.' +
                      f' Cost: {New_Array[i][j]}')
else:
    print('No solution found.')
```

*Figure 8 - Final Assignment Code Block*

# Chapter 7 Conclusion

Although the initial assignment problem that was proposed can be solved in theory with the multidimensional formulation, it is limited in terms of the current programs that are readily available. The amount of information regarding coding for a multidimensional assignment problem is also lacking. Although OwlGo intends to improve the campus by making classes more easily accessible, the problem should have been broken down into a smaller problem that can compare two or more independent sets rather than multiple dependent figures. Even though OwlGo fell short of the original requirements, the theory and models prove that given more time or better technology, an optimized course scheduling process can be conducted to make KSU's Marietta campus as efficient as the Kennesaw campus.

David Broz captures the importance of the course scheduling problems by stating, "Campus designs embody institutional values, and students, faculty, alumni, and partners want to align themselves with institutions whose values mirror their own. Buildings have an enormous impact on the environment, and institutions can leverage real estate to reflect their core values" [15]. It is imperative that KSU place the students first, for they would not be the university they are today without them. Rather than making the students continuously suffer through inefficient course locations, the OwlGo team would like for the university to consider an approach similar to the one presented here. Although complex, solving the course assignment problem based on optimal student travel would not only pay dividends to the faculty, but also bring about more satisfaction among those students. The researchers would like for the KSU to be remembered not by the challenges faced, but instead by the effort that the university put in place to please and enable each students to succeed.

# Chapter 8 References

[1] J. Nakasuwan, P. Srithip, and S. Komolavanij, “Class Scheduling Optimization,” Thammasat Int. J. Sc. Tech, vol. 4, no. 2, 1999, Accessed: Apr. 17, 2022. [Online]. Available: https://thaiscience.info/Journals/Article/TSTJ/10480612.pdf

[2] Dahiya, Siddharth. “COURSE SCHEDULING WITH PREFERENCE OPTIMIZATION,” 2015. Accessed: Apr. 17, 2022. [Online]. Available: https://etda.libraries.psu.edu/files/final_submissions/10828

[3] Zhang, S. and Frimpong Boamah, E. (2021), "Managing campus parking demand through course scheduling – an approach to campus sustainability", International Journal of Sustainability in Higher Education, Vol. 22 No. 4, pp. 909-930, Accessed: Apr. 17, 2022. [Online]. Available: https://doi.org/10.1108/IJSHE-11-2020-0461

[4] H. Barreto, “Why Excel?,” The Journal of Economic Education, vol. 46, no. 3, pp. 300–309, Jun. 2015, doi: 10.1080/00220485.2015.1029177.

[5] L. Dunbar, “The Other Part of the Job: Rapid Data Analysis with Excel and Sheets,” General Music Today, vol. 33, no. 2, pp. 83–86, Oct. 2019, doi: 10.1177/1048371319880873.

[6] B. Naderi, “Modeling and Scheduling University Course Timetabling Problems,” International Journal of Research in Industrial Engineering, vol. 5, no. 1–4, pp. 1–15, 2016, doi: 10.22105/riej.2017.49167.

[7] A. Azlan and N. Mohd. Hussin, “Implementing graph coloring heuristic in construction phase of curriculum-based course timetabling problem,” IEEE Xplore, Apr. 01, 2013. https://ieeexplore.ieee.org/document/6612369 (accessed Apr. 26, 2022).

[8] A. Ozbilge, A. Ulucan, and K. B. Atici, “Elective course assignment problem: a revenue management based approach,” INFOR: Information Systems and Operational Research, pp. 1–24, Jul. 2020, doi: 10.1080/03155986.2020.1796064.

[9] O. Solarte Pabón and L. Machuca Villegas, “Fostering Motivation and Improving Student Performance in an introductory programming course: An Integrated TeachingApproach,” Revista EIA, vol. 16, no. 31, p. 65, Jan. 2019, doi: 10.24050/reia.v16i31.1230.

[10] R. Natu, “Assignment Problem in Operations Research Using Python,” Analytics Vidhya, Nov. 26, 2020. https://medium.com/analytics-vidhya/assignment-probleminoperations-research-using-python-3fa48ac2d342 (accessed Apr. 26, 2022).

[11] R. J. Eggert, Engineering design. Meridian, Idaho: High Peak Press, 2010.

[12] L. Woolfson, “Schedule Optimisation using Linear Programming in Python,” Medium, Jun. 18, 2021. https://towardsdatascience.com/schedule-optimisation-using-linear-programming-in-python-9b3e1bc241e1 (accessed Apr. 26, 2022).

[13] “Solving an Assignment Problem | OR-Tools,” Google Developers. https://developers.google.com/optimization/assignment/assignment_example (accessed Apr. 26, 2022).

[14] F. S. Hillier and G. J. Lieberman, Introduction to operations research. New York, Ny: Mcgraw-Hill, 2015.

[15] “10 ideas for tomorrow’s campus | Building Design + Construction,” www.bdcnetwork.com, Jun. 30, 2016. https://www.bdcnetwork.com/blog/10-ideas-tomorrows-campus (accessed Apr. 26, 2022).

[16] “Python Syntax with Examples,” Python Geeks, May 26, 2021. https://pythongeeks.org/python-syntax/

[17] A. Esasky, M. Iltsenko, S. Jones, and M. Tharakan, “Delivery Route Optimization,” Senior Design Project For Engineers, Apr. 2021, Accessed: Apr. 26, 2022. [Online]. Available: https://digitalcommons.kennesaw.edu/egr_srdsn/51/

[18] J. Bertram, J. Britt, B. Ngo, and M. Diesing, “Project Scrappie (Clear Constellation) Project Scrappie (Clear Constellation).” Accessed: Apr. 26, 2022. [Online]. Available: https://digitalcommons.kennesaw.edu/cgi/viewcontent.cgi?article=1067&context=egr_srdsn

[19] “Southern Polytechnic State University (SPSU) Introduction and Academics - Marietta, GA,” www.stateuniversity.com. https://www.stateuniversity.com/universities/GA/Southern_Polytechnic_State_University.html (accessed Apr. 26, 2022).

[20] “Campus Maps,” www.kennesaw.edu. https://www.kennesaw.edu/maps/index.php (accessed Apr. 26, 2022).

[21] “Dynamic Schedule,” owlexpress.kennesaw.edu. https://owlexpress.kennesaw.edu/prodban/bwckschd.p_disp_dyn_sched (accessed Apr. 26, 2022).

# Appendix A: Acknowledgements

Tackling a project that will utilize every bit of information learned throughout our Collegiate career is only possible by our faculty resources. The researchers would like to thank Professor Renee Butler for assisting with the initial data processing. After this meeting, the project direction began to narrow drastically. The student feedback that was collected during the preliminary stages of the project were very appreciated, for they indicated the need for change and how severe the class locations were mislocated. A special thanks goes out to Dr. Adeel Khalid, as it was his feedback and instruction that brought about the most improvement with the project requirements and report specifications. Finally, the researchers would like to thank Kennesaw State University for giving the researchers access to the dynamic schedule and other campus resources.

## Appendix B: Contact Information

| Name | Title | Email | Phone # |
|---|---|---|---|
| Renji Maliakal | Data Analyst | rmaliaka@students.kennesaw.edu | 678-517-5968 |
| Matthew Pendley | Project Manager | mpendle4@students.kennesaw.edu | 678-833-4177 |
| Jorge Solorzano | Project Engineer | jsolorz2@students.kennesaw.edu | 771-163-1089 |
| Bright Uchehara | Engineering Coordinator | bucheha1@students.kennesaw.edu | 404-966-3319 |
| Dr. Adeel Khalid | KSU Professor – Project Advisor | akhalid2@kennesaw.edu | 470-578-7241 |

## Appendix C: Reflections

**Renji Maliakal** - The project was a huge headache. We first started with the simple goal of making the campus more efficient by grouping class locations together. However, we were not aware of what we were in for. At first, we wanted to stay away from coding as we are ISYE students. However, we later found that sometimes as an engineer, we need to use and learn to use the tools that are available to us. The multidimensional aspect of the assignment forced our hand to code the problem. As I spoke with other coding experts, I found the importance of having a plan ready. In our case, it was the mathematical model for our problem. This helped us to focus our scope. However, dealing with the coding and how it is extremely difficult to create a multidimensional assignment problem using code given my own experience/skill level. Coding for this problem was a LOT of trial and error. I made many mistakes, went down many rabbit holes, and spent countless days and sleepless nights trying to formulate a solution for our problem. Although we feel short of our intended solution, given the skills we learned in ISYE, I felt we were able to do what we were taught and instructed to do in the last 4+ years of college. After all the work that I put in, at this point I am satisfied with how far we have come.

**Matthew Pendley** – This project was a big challenge for our team. By the end of the semester, we all learned the value of communication and proper planning. After performing the literature review section, I became aware of how big the University Course Timetabling Problem is and how many universities face this issue each year. There is a lot to be learned through good, credible research, but even more to be learned through the implementation of individual experiences and the skillsets we have acquired as engineering students over the years at KSU. One of the major setbacks of the project was determining the solution method that was to be run in the Python algorithm. Since none of the members were familiar with this program, it took weeks and experimentation to determine a functional solution method. If I were to redo this project, I would suggest evaluating the possible solution methods that were within our current capabilities as Industrial Engineering students at KSU. I believe that this project has given me insight as to what I might see as an engineer in the future, especially having to overcome the challenges and obstacles along the way. I would specifically like to thank my dad for believing that I could get this project done and succeed in this course. In addition, a big thanks goes out to all the teachers that have prepared me for this course at KSU. I look forward to the next chapter of my professional career and feel prepared for whatever challenges I might face along the way.

**Jorge Solorzano** - From the moment I heard that I must take this course, I was concerned about my performance in the class, because I struggle in courses that require talking in public, so I was a little scared, but it was something I eventually needed to face. At the beginning of the course, we were not for a good start, because we ended up forming our team a day before the first deadline, so we missed it. It was hard to overcome this issue, but in the end, communication was what took us to the next level, and helped us to meet future deadlines. This project took a lot of hours to complete, and it was a big challenge for all of us because it required us to put into practice everything that has been taught to us in all our years of college at KSU; so, most of the time we had to look back and open old books to see how it could help us figure out a proper solution to our problem. It was a remarkably interesting sneak peek at how our life as engineers is going to be, and the importance of facing the problem in diverse ways to find the most feasible solution. If I had to give some advice to future students taking this course, I would say that don’t

procrastinate on each assignment and keep in constant communication with the advisor in charge of the course. I want to thank every teacher involved in my education because thanks to them I made it this far; but I want to give a special shout out to my team members: Renji, Bright, and Matthew. I know that you all have many other responsibilities besides being a student, and you all took the time and effort to work on this project, thanks for letting me be part of this team. I wish you all luck and success in every goal you have in mind and congratulations, you are all going to be great engineers.

**Bright Uchehara** - From the beginning of this project, I knew that we had our work cut out for us. When trying to create a better system for a user such as yourself, the student, you become overly attached to the goal at hand. One of the biggest issues that we faced was the Python algorithm that we chose to use to make our system functional. At first glance, this was not an issue, but we grossly underestimated the task. Being industrial engineering students, my group members and I lacked formal training in coding. This energy-killing problem caused us to miss our personal deadlines. If I could begin this project again, I would create a Gantt chart with timelines, goals, and solutions that are feasible; given our time and knowledge constraints. My advice to the students coming behind me to take this course includes proper planning and communication. Recalling the beginning of this course, I assumed that the action items needed to get to each goal were extremely far away in dates. I would like to thank all my group members for their sacrifice and dedication, and I would also like to thank my family that continually encouraged me to keep going in my academic endeavors.

# Appendix D: Gantt Chart/Contributions

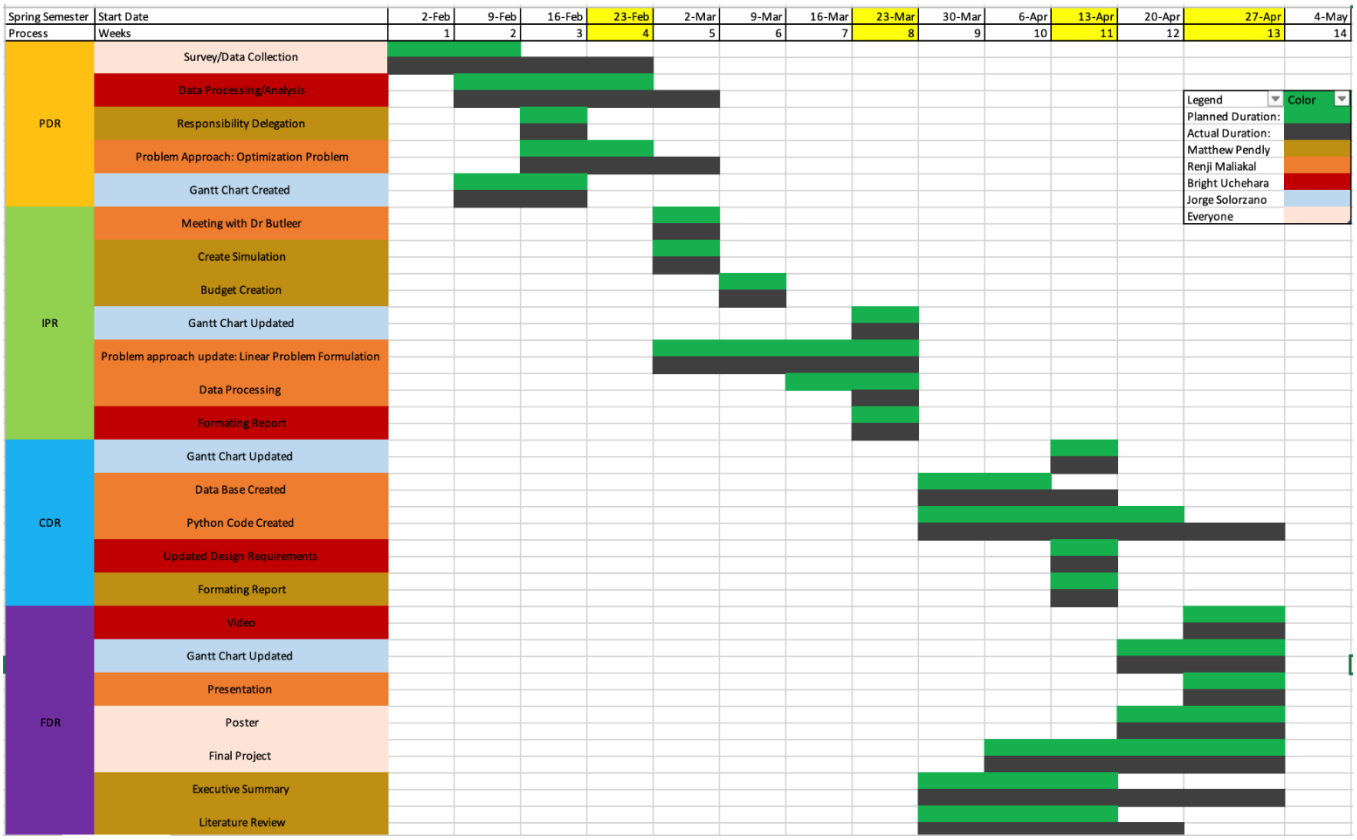



*Figure 5 - Gantt Chart of Major Developments*

| Name | Contributions |
|---|---|
| Renji Maliakal | **Chapters 1,** 3, 4, 5, 6, 7, Appendices<br><br>**Sections** 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 3.1, 3.2, 3.5, 3.7, 3.10, 4.1, 4.2, 4.3, E |
| Matthew Pendley | **Chapters**: 1, 2, 3, 4, 5, 6, 7 Appendices<br><br>**Sections**: 1.1, 1.2, 1.3, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 4.1, 4.2, 4.3, 4.5, 5, 6, 7 |
| Jorge Solorzano | **Chapters: 1,** 3, Appendices<br><br>**Sections**: 1.5, 1.6, 3.3, 3.4 |
| Bright Uchehara | **Chapters**: 3, 4, Appendices<br><br>**Sections**: 3.8, 4.2 |

# Appendix E: Project Resources

Initial Python Code

**Code E.1**

```
#Building to Building Data

a = 0 #further away than 2 minutes
s = 2 #same building
d = 1 #within 2 minutes

Academic = np.reshape(np.array([s, a, d, a, d, d, d, d, a, a, a, d, a]),(12,1))
Architecture = np.reshape(np.array([a, s, d, d, a, a, a, a, a, a, a, d]), (12,1))
Atrium = np.reshape(np.array([d, d, s, a, a, d, a, a, a, a, a, a, d]),(12,1))
Civil = np.reshape(np.array([a, d, a, s, a, a, a, a, a, a, a, d]),(12,1))
Crawford_Lab = np.reshape(np.array([d, a, a, a, s, d, d, d, a, d, d, a]),(12,1))
Design1 = np.reshape(np.array([d, a, d, a, d, s, d, d, a, a, d, a]),(12,1))
Design2 = np.reshape(np.array([d, a, d, a, d, d, s, d, a, a, d, a]),(12,1))
Engineering_Tech = np.reshape(np.array([a, a, a, a, d, d, d, s, a, a, d, a]),(12,1))
Joe_Mack = np.reshape(np.array([a, a, a, a, a, a, a, a, s, d, a, a]),(12,1))
Mathematics = np.reshape(np.array([a, a, a, a, d, a, a, a, d, s, d, a]),(12,1))
Science_Lab = np.reshape(np.array([d, a, a, a, d, d, d, d, a, d, s, a]),(12,1))
Textiles = np.reshape(np.array([a, d, d, d, a, a, a, a, a, a, a, s]),(12,1))

        #Class Room Capacities Arrays
ACAD = np.array([137, 128, 32, 32, 25, 25, 25, 25, 25, 25, 24, 24, 24])
ARCH = np.array([40, 40, 38, 36, 25, 20, 18, 4])
ATRI = np.array([72, 72, 70, 50, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 39, 36, 35, 32, 32, 32, 32,
32, 30, 30, 30, 25, 25, 24, 24, 24, 24, 23, 23, 22, 18, 18])
CIVI = np.array([40, 40, 40, 25, 20, 18, 18, 16])
CRAW = np.array([45, 40, 30, 0])
DES1 = np.array([30, 25, 20, 18, 17, 16, 16])
DES2 = np.array([18])
ENGI = np.array([165, 74, 60, 50, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 32, 25, 25,
25, 25, 24, 24, 23, 20, 20, 20, 20, 20, 20, 20, 20, 20, 16, 16, 16, 12, 10])
JMWS = np.array([12])
MATH = np.array([40, 40, 40, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 25, 25, 20, 18])
LABA = np.array([24, 24, 24, 22])
TEXT = np.array([65, 36, 20, 18, 17, 13, 10])

ACAD_Class = Academic*np.ones([1,len(ACAD)], dtype= int)
ARCH_Class = Architecture*np.ones([1,len(ARCH)], dtype= int)
ATRI_Class = Academic*np.ones([1,len(ATRI)], dtype= int)
CIVI_Class = Academic*np.ones([1,len(CIVI)], dtype= int)
CRAW_Class = Academic*np.ones([1,len(CRAW)], dtype= int)
DES1_Class = Academic*np.ones([1,len(DES1)], dtype= int)
DES2_Class = Academic*np.ones([1,len(DES2)], dtype= int)
```

```
ENGI_Class = Academic*np.ones([1,len(ENGI)], dtype= int)
JMWS_Class = Academic*np.ones([1,len(JMWS)], dtype= int)
MATH_Class = Academic*np.ones([1,len(MATH)], dtype= int)
LABA_Class = Academic*np.ones([1,len(LABA)], dtype= int)
TEXT_Class = Academic*np.ones([1,len(TEXT)], dtype= int)
Classes =
np.hstack((ACAD_Class,ARCH_Class,ATRI_Class,CIVI_Class,CRAW_Class,DES1_Class,DE
S2_Class,ENGI_Class,JMWS_Class,MATH_Class,LABA_Class,TEXT_Class))
```

**Code E.2**

```
#Parking to Building Data
z = 1 #further than 5 minutes
x = 1 #within 5 minutes

Lot51 = np.reshape((np.array([x, z, x, z, x, x, x, x, x, x, x, z])),(12,1))#[0]
Lot35 = np.reshape((np.array([x, z, z, z, x, z, z, x, x, x, z, z])),(12,1)) #[1]
Lot36 = np.reshape((np.array([x, z, z, z, x, z, z, z, x, x, z, z])),(12,1)) #[2]
Lot37 = np.reshape((np.array([x, z, z, z, z, z, z, z, x, x, z, z])),(12,1)) #[3]
Lot38 = np.reshape((np.array([x, z, x, z, z, x, x, z, x, x, z, z])),(12,1)) #[4]
Deck60 = np.reshape((np.array([x, z, x, x, x, x, x, x, z, x, x, x])),(12,1)) #[5]
Deck22 = np.reshape((np.array([z, x, x, x, z, z, z, z, z, z, z, x])),(12,1)) #[6]

Lot51_B = Classes*Lot51
Lot35_B = Classes*Lot35
Lot36_B = Classes*Lot36
Lot37_B = Classes*Lot37
Lot38_B = Classes*Lot38
Deck60_B = Classes*Deck60
Deck22_B = Classes*Deck22

Lot_B_B = np.dstack((Lot51_B, Lot35_B, Lot36_B, Lot37_B, Lot38_B, Deck60_B,
Deck22_B))
Lot_B_B = np.reshape(Lot_B_B, (12, 145, 7, 1))
```

**Code E.3**

```
#Time Array
Time_Array = np.reshape((np.ones([199], dtype= int)),(1,1,1,199))


#Location_Time Array
Location_Time = (Lot_B_B*Time_Array)
Location_Time = np.reshape(Location_Time, (12, 145, 7, 199, 1))
```

**Code E.4**

```
#Days Array
#Sunday = [0], Monday = [1], Tuesday = [2], Wednesday = [3], Thursday = [4], Friday = [5],
Saturday = [6]
Days = array([0, 1, 1, 1, 1, 1, 1])
Days = np.reshape(Days, (1, 1, 1, 1, 7))
```

**Code E.5**

```
#Optimization Coefficient Array
Coefficient = (Location_Time*Days)
        #Time
Time_Arrays = []
Time_Slots = []
for w in range(len(df)):
   Time_Arrays += [Time_Slots]
   Time_Slots = []
   for e in range(df[w,9]):
      Time_Slots += ([df[w,10]+e])
   #print(Time_Slots)
Time_Arrays += [Time_Slots]
```

**Code E.6**

```
        #Class Capacity Constraint
ACADtrue = []
ACADrep = []
ARCHtrue = []
ARCHrep = []
ATRItrue = []
ATRIrep = []
CIVItrue = []
CIVIrep = []
CRAWtrue = []
CRAWrep = []
DES1true = []
DES1rep = []
DES2true = []
DES2rep = []
ENGItrue = []
ENGIrep = []
JMWStrue = []
JMWSrep = []
MATHtrue = []
```

```
MATHrep = []
LABAtrue = []
LABArep = []
TEXTtrue = []
TEXTrep = []

for w in range(len(df)):
    ACADtrue += [ACADrep]
    ARCHtrue += [ARCHrep]
    ATRItrue += [ATRIrep]
    CIVItrue += [CIVIrep]
    CRAWtrue += [CRAWrep]
    DES1true += [DES1rep]
    DES2true += [DES2rep]
    ENGItrue += [ENGIrep]
    JMWStrue += [JMWSrep]
    MATHtrue += [MATHrep]
    LABAtrue += [LABArep]
    TEXTtrue += [TEXTrep]

    ACADrep = []
    ARCHrep = []
    ATRIrep = []
    CIVIrep = []
    CRAWrep = []
    DES1rep = []
    DES2rep = []
    ENGIrep = []
    JMWSrep = []
    MATHrep = []
    LABArep = []
    TEXTrep = []

    for w1 in range(len(ACAD)):
        if df[w,8] <= ACAD[w1] and df[w,8] >= 1/2*ACAD[w1]:
            ACADrep += [w1]
    for w2 in range(len(ARCH)):
        if df[w, 8] <= ARCH[w2] and df[w, 8] >= 1 / 2 * ARCH[w2]:
            ARCHrep += [w2+(len(ACAD))]
    for w3 in range(len(ATRI)):
        if df[w, 8] <= ATRI[w3] and df[w, 8] >= 1 / 2 * ATRI[w3]:
            ATRIrep += [w3+(len(ACAD))+(len(ARCH))]
    for w4 in range(len(CIVI)):
        if df[w, 8] <= CIVI[w4] and df[w, 8] >= 1 / 2 * CIVI[w4]:
            CIVIrep += [w4+(len(ACAD))+(len(ARCH))+(len(ATRI))]
    for w5 in range(len(CRAW)):
```

```
        if df[w, 8] <= CRAW[w5] and df[w, 8] >= 1 / 2 * CRAW[w5]:
            CRAWrep += [w5+(len(ACAD))+(len(ARCH))+(len(ATRI))+(len(CIVI))]
    for w6 in range(len(DES1)):
        if df[w, 8] <= DES1[w6] and df[w, 8] >= 1 / 2 * DES1[w6]:
            DES1rep += [w6+(len(ACAD))+(len(ARCH))+(len(ATRI))+(len(CIVI))+(len(CRAW))]
    for w7 in range(len(DES2)):
        if df[w, 8] <= DES2[w7] and df[w, 8] >= 1 / 2 * DES2[w7]:
            DES2rep +=
[w7+(len(ACAD))+(len(ARCH))+(len(ATRI))+(len(CIVI))+(len(CRAW))+(len(DES1))]
    for w8 in range(len(ENGI)):
        if df[w, 8] <= ENGI[w8] and df[w, 8] >= 1 / 2 * ENGI[w8]:
            ENGIrep +=
[w8+(len(ACAD))+(len(ARCH))+(len(ATRI))+(len(CIVI))+(len(CRAW))+(len(DES1))+(len(D
ES2))]
    for w9 in range(len(JMWS)):
        if df[w, 8] <= JMWS[w9] and df[w, 8] >= 1 / 2 * JMWS[w9]:
            JMWSrep +=
[w9+(len(ACAD))+(len(ARCH))+(len(ATRI))+(len(CIVI))+(len(CRAW))+(len(DES1))+(len(D
ES2))+(len(ENGI))] #Capacity Array
Capacity_Array = []
for w in range(len(df)):
    Capacity_Array += [ACADtrue[w] + ARCHtrue[w] + CIVItrue[w] + MATHtrue[w] +
TEXTtrue[w]]
Capacity_Array = np.reshape(Capacity_Array, (13,1,53,1))
```

**Code E.7**

```
Cost = Capacity_Array * Coefficient
Cost = np.array(Cost)


New_Array = []
Array = []
for w in range(len(Cost)):
    New_Array += [Array]
    for w1 in range(5):
        for w2 in range(53):
            for w3 in range(7):
                Array += [Cost[w,w1,w2,w3]]
New_Array += [Array]
popped_element = New_Array.pop(0)
    for w10 in range(len(MATH)):
        if df[w, 8] <= MATH[w10] and df[w, 8] >= 1 / 2 * MATH[w10]:
            MATHrep +=
```

```
[w10+(len(ACAD))+(len(ARCH))+(len(ATRI))+(len(CIVI))+(len(CRAW))+(len(DES1))+(len(
DES2))+(len(ENGI))+(len(JMWS))]
   for w11 in range(len(LABA)):
      if df[w, 8] <= LABA[w11] and df[w, 8] >= 1 / 2 * LABA[w11]:
         LABArep +=
[w11+(len(ACAD))+(len(ARCH))+(len(ATRI))+(len(CIVI))+(len(CRAW))+(len(DES1))+(len(
DES2))+(len(ENGI))+(len(JMWS))+(len(MATH))]
   for w12 in range(len(TEXT)):
      if df[w, 8] <= TEXT[w12] and df[w, 8] >= 1 / 2 * TEXT[w12]:
         TEXTrep +=
[w12+(len(ACAD))+(len(ARCH))+(len(ATRI))+(len(CIVI))+(len(CRAW))+(len(DES1))+(len(
DES2))+(len(ENGI))+(len(JMWS))+(len(MATH))+(len(LABA))]
```

**Code E.8**

```
      #Create the data
num_classes = len(New_Array)
num_classrooms = len(New_Array[0])


#Declare the MIP Solver
solver = pywraplp.Solver.CreateSolver('SCIP')

#Create the Variables
x = {}
for i in range(num_classes):
   for j in range(num_classrooms):
      x[i, j] = solver.IntVar(0, 1, '')


#Create the constraints
for i in range(num_classes):
   solver.Add(solver.Sum([x[i, j] for j in range(num_classrooms)]) == 13)
for j in range(num_classrooms):
   solver.Add(solver.Sum([x[i, j] for i in range(num_classes)]) <= 1)

#Create the objective function
objective_terms = []
for i in range(num_classes):
   for j in range(num_classrooms):
      objective_terms.append(New_Array[i][j] * x[i, j])
solver.Minimize(solver.Sum(objective_terms))
#Invoke the solver
status = solver.Solve()

#Print the solution
```

```
if status == pywraplp.Solver.OPTIMAL or status == pywraplp.Solver.FEASIBLE:
    print(f'Total cost = {solver.Objective().Value()}\n')
    for i in range(num_classes):
        for j in range(num_classrooms):
            # Test if x[i,j] is 1 (with tolerance for floating point arithmetic).
            if x[i, j].solution_value() > 0.5:
                Answer_Array = [i, j]
                print(f'class {i} assigned to classroom {j}.' +
                      f' Cost: {New_Array[i][j]}')
else:
    print('No solution found.')
```

**Figure E.1**

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | CRN | DEPARTM | Code | Credit Ho | LAB | Weekly Fi | START | DURATION | SIZE |
| 2 | 13804 | ISYE | 3125 | 3 | 0 | T | 15:30 | 1:15 | 28 |
| 3 | 10472 | ISYE | 4500 | 3 | 0 | T | 14:00 | 1:15 | 28 |
| 4 | 10442 | ISYE | 4900 | 3 | 0 | W | 9:30 | 2:45 | 30 |
| 5 | 10445 | ISYE | 2600 | 3 | 0 | MW | 18:30 | 1:15 | 36 |
| 6 | 10455 | ISYE | 3200 | 3 | 0 | M | 15:30 | 1:15 | 36 |
| 7 | 10467 | ISYE | 3407 | 3 | 0 | T | 9:30 | 1:15 | 36 |
| 8 | 10468 | ISYE | 3600 | 3 | 0 | TR | 8:00 | 1:15 | 36 |
| 9 | 10469 | ISYE | 4200 | 3 | 0 | TR | 12:30 | 1:15 | 36 |
| 10 | 10470 | ISYE | 4250 | 3 | 0 | T | 17:00 | 1:15 | 36 |
| 11 | 10471 | ISYE | 4425 | 3 | 0 | R | 15:30 | 1:15 | 36 |
| 12 | 10454 | ISYE | 3150 | 3 | 0 | R | 18:30 | 1:15 | 40 |
| 13 | 10456 | ISYE | 3350 | 3 | 0 | M | 17:00 | 1:15 | 40 |
| 14 | 10466 | ISYE | 3400 | 3 | 0 | TR | 11:00 | 1:15 | 40 |
| 15 | 11885 | ME | 1311 | 3 | 0 | TR | 17:00 | 1:15 | 23 |