Technical Report

Scheduling 2021 Fall Courses for the Department of Entomology at Mavericks University

Isidore Sossa and Chase Scott

TABLE OF CONTENTS

1. INTRODUCTION

This project is designed to optimize the class schedule for Fall 2021 for the Entomology Department at Mavericks University. We were given the necessary data for this task in the form of an excel file. This excel file contained information that included the course names and credit hours, the classrooms available, the professors and their workload for the coming semester, and much more. Included within the excel was also a preference list that we needed to use to generate a list of preferences for each professor according to their qualifications. We as a group was also tasked with the development and implementation of two unique rules that would shape the outcome. All data manipulation was performed through Python with no changes to the original data. This project was completed with the idea that any department would be able to use the code to solve their assignment problems so long as they provide their data in the same structure.

The remainder of the report is as follows. Section 2 describes the information relevant to the class schedule of Fall 2021 for the Entomology Department at Mavericks University. Section 3 discusses the relevant assumptions made while addressing the problem. Section 4 explains the mathematical model created to solve this scheduling problem. Section 5 provides the computer implementation as well as a detailed explanation of our implementation. Section 6 discusses the results of the implementation. Finally, Section 7 concludes the report, provides a recommendation and presents the topics for potential work.

2. PROJECT DESCRIPTION

This section presents the relevant information of the project. By analyzing the data provided within the excel file, we can find worksheets labeled: Colleges and Departments, Course Catalog, Professors, Qualification, Courses Offered Fall 2021, and Classrooms. The worksheet labeled Colleges and Departments contains the college, college code, department, and department code. The next worksheet contained all relevant information regarding the courses in the College of Agriculture. It contained the course number, course name, and credit hours per course. Credit hours are important because it lists how often a course is presented a week. For reference, one credit hour is worth fifty minutes.

The next worksheet, Professors, showcases the faculty member's names as well as the workload in credit hours for each respective member. It is important to note that some faculty members have a workload of zero, therefore are not teaching. Qualification is the next worksheet and is important because it lists each professor's qualification for every course available in the college of agriculture this semester. Every professor is given a qualification of either one (meaning they are qualified) or zero (meaning they are unqualified). This worksheet is used to complete the task of creating a preference list. The following worksheet showcases similar information for the Colleges and Departments section except that it only provides the courses offered in the Fall 2021 semester. This worksheet also shows how many sessions of each course are offered. The last worksheet displays a list of classrooms available for use as well as their respective capacity.

3. ASSUMPTIONS

For this project, we assume that any classroom could hold any number of courses. We also assumed that other than professors' preference, a professor can teach courses back to back as long as this solution maximizes the total preference score. Moreover, for courses having multiple sessions, we considered these as been independent, that is, unique courses. Last but not least, we also assume that a professor is available to teach for the entirety of his/her workload. For example, if a professor's workload is nine hours, s/he can teach up to those nine hours.

4. MATHEMATICAL MODEL

Our mathematical model has four main sections. These are the parameters (input to the model), a decision variable, the constraints (problem specific-requirements), and the objective function. The parameters are part or all of the information listed within the Excel file. The decision variable describes the quantities that the decision-makers would like to determine. These can be referred to as the unknowns of a mathematical model. Constraints are inequalities or equalities defining the limitations on decisions. Lastly, the objective function evaluates some quantitative criterion of immediate importance such as minimize cost, maximize profit, or yield.

Only one decision variable was created for this project. This decision variable, denoted by $\boldsymbol{x_{fcsrp}}$, is a binary variable that determines whether who (faculty), when (time) and where (classroom) a course is offered. **Table 1** presents the list of mathematical notations used in the report.

Decision Variable

$$x_{fcsrp} = \begin{cases} 1 & \text{if faculty member } f \text{ teaches session } s \text{ of course } c \text{ in room } r \text{ at period } p \\ 0 & \text{otherwise} \end{cases}$$

Feasibility Constraints

The next portion of the model is the constraints. There are nine constraints in total, seven of which directly impact the feasibility of the model and two of which the team has created per the project requirements. Not all the department's faculty are available for teaching this semester. Therefore, these faculty members are excluded from the domain of possible choices.

Table 1: Notation for Class Scheduling Model

Sets	
F	Faculty members, indexed by $f = 1, \dots, \lvert F \rvert$
C	Courses offered, indexed by $c = 1, \dots, \lvert C \rvert$
S	Sessions of a course, indexed by $s = 1, \dots, \lvert S \rvert$
R	Classrooms, indexed by $r = 1, \dots, \lvert R \rvert$
P	Sequence of periods per day, indexed by $p = 1, \dots, \lvert P \rvert$
Parameters	
Preference_f	Course preference of each faculty member randomly generated
Workload_f	The total workload of each faculty member randomly generated
D	Duration in minutes of one credit hour
E	Duration in minutes of one period
d_c	Number of credits of class c
m_{cs}	Number of times a class meeting each week
n_{cs}	Number of periods covered by a course
K_r	Capacity of classroom r

The first constraint sets these faculty members' binary variables to zero. The mathematical equation is as follows:

$$x_{fcsrp} = 0, \ \forall f \in F', \forall c \in C, \ \forall s \in S, \ \forall r \in R, \forall p \in P,$$

$$\text{where } F' = \left\{ f \in F \,\middle|\, \text{Workload}_f = 0 \right\}$$

Data manipulation revealed that most of the courses offered this semester by the Department of Entomology at Mavericks University cover either two or three 30-minutes periods. Therefore, the second constraint establishes the rule that a faculty member only teaches at most one course in one room within a given time frame. $\boldsymbol{d_c}$ represents the number of credits of class $\boldsymbol{c}$ and $\boldsymbol{D}$, a constant represents the duration in minutes of one credit hour. In this project, $D = 50$ minutes. $\boldsymbol{m_{cs}}$ represents the number of times class $\boldsymbol{c}$ meets each week, and $\boldsymbol{E}$, a constant represents the duration of each period. $E = 30$ minutes. The mathematical equation is as follows:

$$\sum_{c\in C}\sum_{s\in S}\sum_{r\in R}\sum_{p\in[a,b)} x_{fcsrp} \leq 1, \forall f \in F$$

where

$$I \in P = \{[a,b) \ \forall a,b \ \in \ P^2 | \ b = a + \ n_{cs} \ \}$$

where

$$n_{cs} \ = \ \left\lceil \frac{d_c * D}{m_{cs} \ * \ E} \right\rceil$$

The third constraint states that a room can host at most one course at a given time. Similar to the previous constraint, the number of periods covered by a course is also taken into account. This is implemented to stop multiple courses from being taught in the same place within the same period. The mathematical equation is as follows:

$$\sum_{c\in C}\sum_{s\in S}\sum_{f\in F}\sum_{p\in[a,b)} x_{fcsrp} \leq 1, \forall r \in R, \forall \ [a,b) \in I$$

where

$$I = \{[a,b), \forall \ a,b \in P^2 | b = a + \ n_{cs}\}$$

The fourth constraint establishes the rule that each course is assigned to one professor in one room. This is done to ensure that a course is guaranteed to be taught in one room by one professor. The mathematical equation is as follows:

$$\sum_{f\in F}\sum_{r\in R}\sum_{p\in I} x_{fcsrp} = 1, \forall c \in C$$

where

$$I = \{i \in P | i + n_{cs} \in P\}$$

The fifth constraint creates the rule that each course is taught at the same time in the same room by the same faculty member on the days it is offered. The mathematical equation is as follows:

$$x_{fcsru_i} = x_{fcsrv_i}, \forall f \in F, \forall c \in C, \forall s \in S, \forall r \in R, \forall (u_i, v_i) \in U_i \times V_i$$

where

$$U_i \subseteq P \text{ and } V_i \subseteq P, \qquad i = 1 \dots 20$$

The sixth constraint ensures that a course is taught in a classroom that has enough capacity. $\boldsymbol{K_r}$ represents the room capacity. The mathematical equation is as follows:

$$x_{fcsrp} * \text{MaxEnrollment}_{cs} \leq K_r, \forall f \in F, \forall c \in C, \forall s \in S, \forall r \in R, \forall p \in P$$

The seventh constraint enables faculty to teach up to their declared workload. The mathematical equation is as follows:

$$\sum_{c \in C} \sum_{s \in S} \sum_{r \in R} \sum_{p \in I} \frac{d_c}{m_{cs}} * x_{fcsrp} \leq Workload_f, \forall f \in F$$

Custom Constraints

Our first custom rule states that senior courses, which are the course numbers 800 and above, start at 13:00 (or 1 P.M.). This was created with the thought that seniors are studying late into the night so a later start in the day would help them rest. The mathematical equation is as follows:

$$x_{fcsrp} = 0, \forall f \in F, \forall c \in C, \forall s \in S, \forall r \in R, \forall p \in \mathcal{P}$$

where $\mathcal{P} \subset P$ represents the periods before 1 PM.

The second custom constraint states that no classes are offered on Friday after 16:00 or 4 P.M. This was done because Fridays generally are days with low productivity because of the soon-to-be weekend. By setting this constraint, the school can give something back to the students, potentially raising morale in the process. The mathematical equation is as follows:

$$x_{fcsrp} = 0, \forall f \in F, \forall c \in C, \forall s \in S, \forall r \in R, \forall p \in \mathcal{P}$$

where $\mathcal{P} \subset P$ represents the periods beginning at 4 PM on Friday.

Objective Function

The last portion of the model is the objective function. The ultimate goal of the model is to maximize the total preference score of all faculty members.

The objective function equation is as follows:

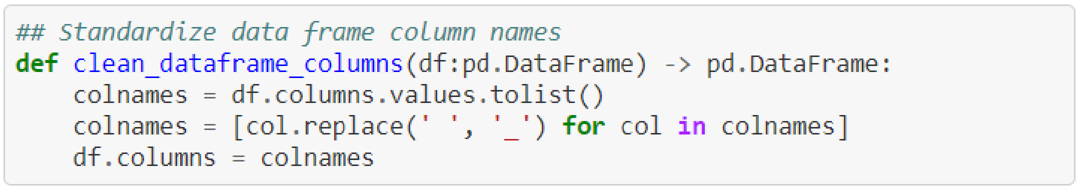
$$\text{maximize}\left(\sum_{f\in F}\sum_{c\in C}\sum_{s\in S}\sum_{r\in R}\sum_{p\in P}\text{Preference}_f\cdot x_{fcsrp}\right)$$

5. COMPUTER IMPLEMENTATION

The first step when starting the computer implementation was to import the many packages we needed to accomplish the task. These included: pandas, DOCplex, math, and random.

After standardizing the column names to leverage pandas' internal implementation, we manipulated the input data to build two main data frames, preferences_df and courses_offered. We also added a data frame that represents the periods from 1 to 100, where each consecutive 20 periods are that of one day, starting on Monday.

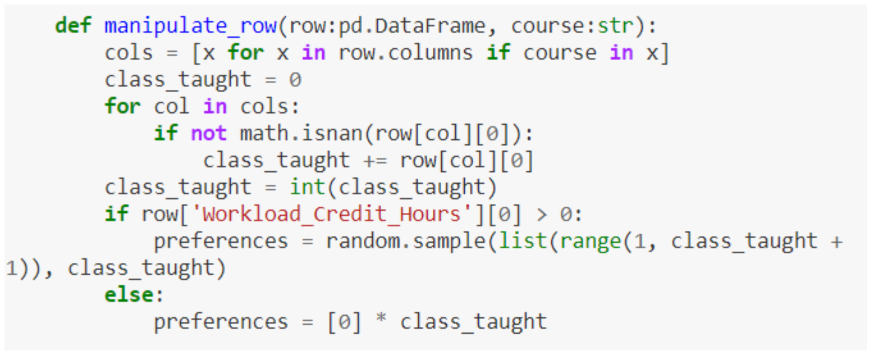
To organize our code, we wrote different functions. Each function is used to accomplish one specific task. For example, the code used to standardize data frame columns is shown below.



```python
## Standardize data frame column names
def clean_dataframe_columns(df:pd.DataFrame) -> pd.DataFrame:
    colnames = df.columns.values.tolist()
    colnames = [col.replace(' ', '_') for col in colnames]
    df.columns = colnames
```

Figure 1 Snapshot of the function used to standardize data frame columns

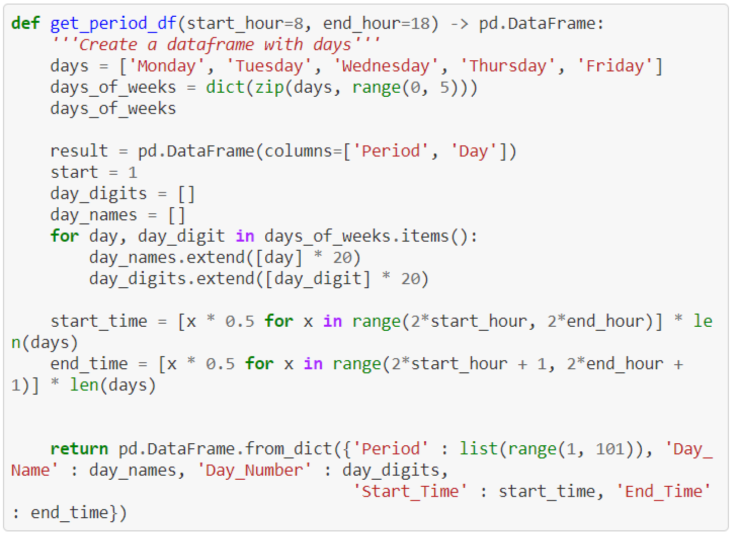
Following the standardization of the column names was the task of duplicating the courses with multiple sessions. This ensures that different sessions of the same course are counted as different classes. Once the courses with multiple sessions were duplicated, the work to create the preferences list started. One of the tasks specifically listed in the project description was to create a randomly generated list of preferences for each professor according to their qualification. A portion of the code can be seen below in figure 2.



```python
    def manipulate_row(row:pd.DataFrame, course:str):
        cols = [x for x in row.columns if course in x]
        class_taught = 0
        for col in cols:
            if not math.isnan(row[col][0]):
                class_taught += row[col][0]
        class_taught = int(class_taught)
        if row['Workload_Credit_Hours'][0] > 0:
            preferences = random.sample(list(range(1, class_taught +
1)), class_taught)
        else:
            preferences = [0] * class_taught
```

Figure 2 Snapshot of the function used to create random preferences

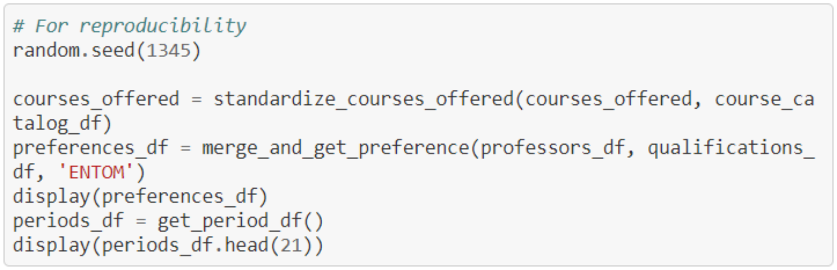
After the first task listed in the project was completed, it was time to define periods. For this project, units of 30 minutes were created called periods. These exist from 08:00 to 18:00 every day to assist in the scheduling of classes. A period covers 30 minutes. For instance, period 1 begins at 08:00 and ends at 8:30, period 2 begins at 08:30 and ends at 9:00 on Monday. Period 21 begins at 8:00 and ends at 8:30 on Tuesday, and so on. This trend continued through Friday. The code that accomplished this task is shown below in figure 3.



```
def get_period_df(start_hour=8, end_hour=18) -> pd.DataFrame:
    '''Create a dataframe with days'''
    days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
    days_of_weeks = dict(zip(days, range(0, 5)))
    days_of_weeks

    result = pd.DataFrame(columns=['Period', 'Day'])
    start = 1
    day_digits = []
    day_names = []
    for day, day_digit in days_of_weeks.items():
        day_names.extend([day] * 20)
        day_digits.extend([day_digit] * 20)

    start_time = [x * 0.5 for x in range(2*start_hour, 2*end_hour)] * le
n(days)
    end_time = [x * 0.5 for x in range(2*start_hour + 1, 2*end_hour +
1)] * len(days)


    return pd.DataFrame.from_dict({'Period' : list(range(1, 101)), 'Day_
Name' : day_names, 'Day_Number' : day_digits,
                                   'Start_Time' : start_time, 'End_Time'
: end_time})
```

Figure 3 Snapshot of the function used to create a data frame of periods

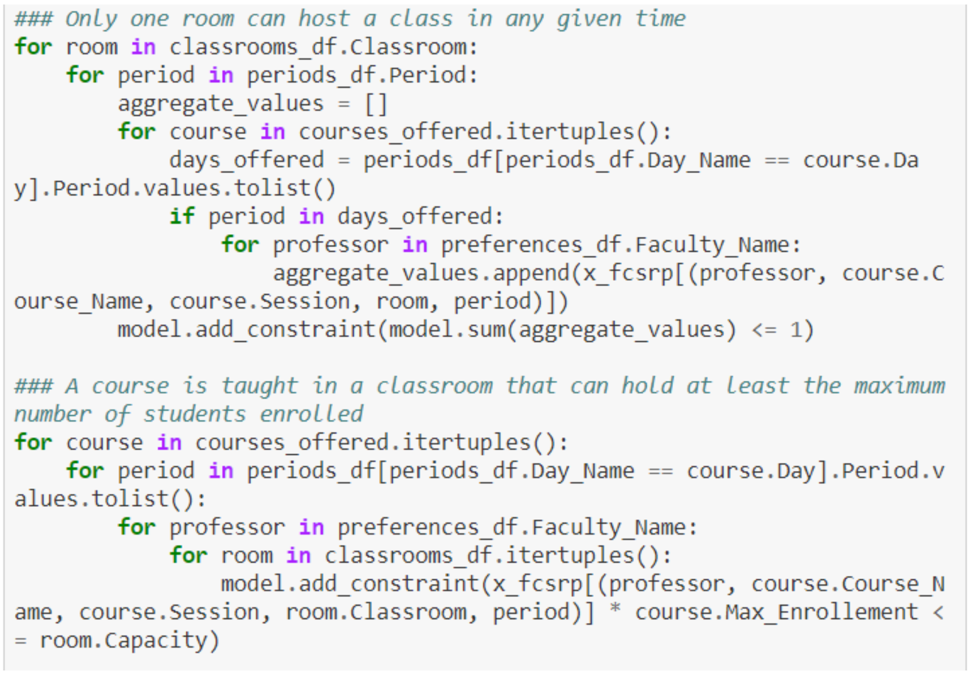
As listed in the project description, the code should be written in a way that another department can easily replicate the assignment process. Therefore, the department code is a parameter to the function that generates professors' preferences. For example, if the Department of Mathematics were to schedule classes using our mathematical model, aside from inputting the correct data, it would only need to change 'ENTOM' to 'MATH'. The code can be seen below.



```
# For reproducibility
random.seed(1345)

courses_offered = standardize_courses_offered(courses_offered, course_ca
talog_df)
preferences_df = merge_and_get_preference(professors_df, qualifications_
df, 'ENTOM')
display(preferences_df)
periods_df = get_period_df()
display(periods_df.head(21))
```

Figure 4 Snapshot of the code that generates the data frames of courses offered and professors' preference

Once the reproducibility was created, the team began working on shaping the model. The first step was to create the model environment and decision variable. Once this was completed, the constraint begins. Each constraint was coded after the other. A portion of the code is shown below.

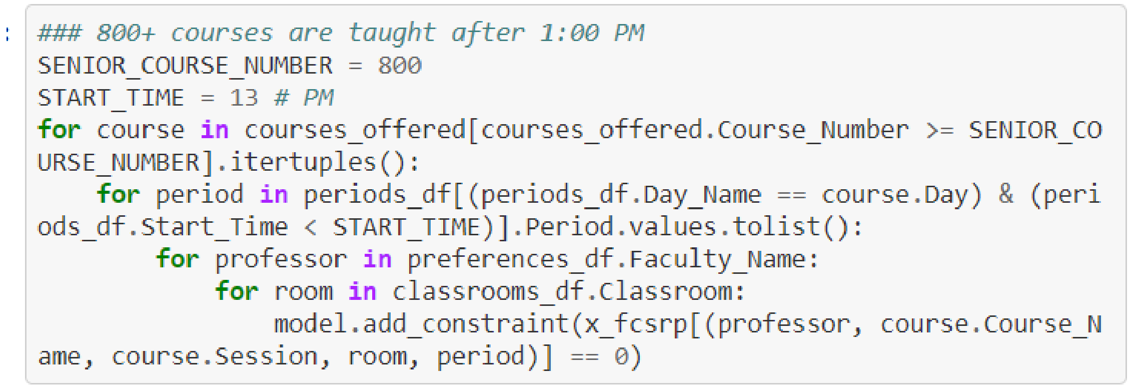


```
### Only one room can host a class in any given time
for room in classrooms_df.Classroom:
    for period in periods_df.Period:
        aggregate_values = []
        for course in courses_offered.itertuples():
            days_offered = periods_df[periods_df.Day_Name == course.Da
y].Period.values.tolist()
            if period in days_offered:
                for professor in preferences_df.Faculty_Name:
                    aggregate_values.append(x_fcsrp[(professor, course.C
ourse_Name, course.Session, room, period)])
        model.add_constraint(model.sum(aggregate_values) <= 1)

### A course is taught in a classroom that can hold at least the maximum
number of students enrolled
for course in courses_offered.itertuples():
    for period in periods_df[periods_df.Day_Name == course.Day].Period.v
alues.tolist():
        for professor in preferences_df.Faculty_Name:
            for room in classrooms_df.itertuples():
                model.add_constraint(x_fcsrp[(professor, course.Course_N
ame, course.Session, room.Classroom, period)] * course.Max_Enrollement <
= room.Capacity)
```

Figure 5 Snapshot of constraints written in Python

Once all of the required constraints have been written, the team began coding the two special rules as listed in the project description. The two rules that were created were that senior courses started at 13:00 and no classes were taught on Friday after 16:00. The code for both of these constraints is shown in Fig. 6 and 7.

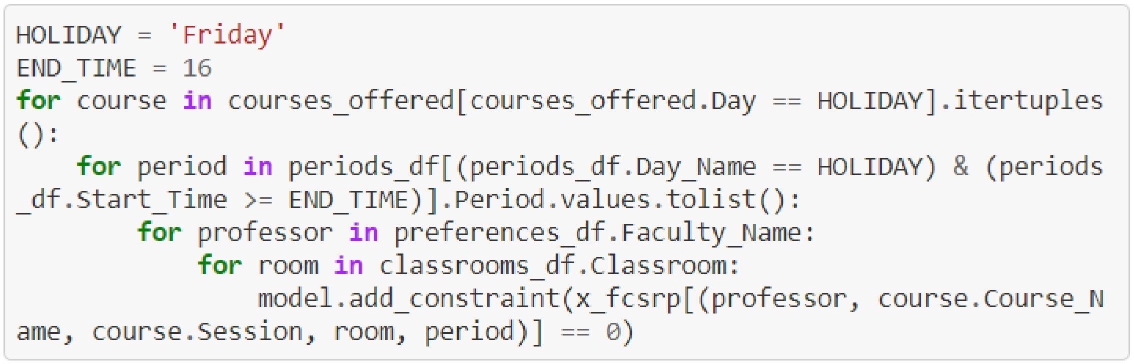
1. Seniors courses start at 1:00 PM



```
### 800+ courses are taught after 1:00 PM
SENIOR_COURSE_NUMBER = 800
START_TIME = 13 # PM
for course in courses_offered[courses_offered.Course_Number >= SENIOR_CO
URSE_NUMBER].itertuples():
    for period in periods_df[(periods_df.Day_Name == course.Day) & (peri
ods_df.Start_Time < START_TIME)].Period.values.tolist():
        for professor in preferences_df.Faculty_Name:
            for room in classrooms_df.Classroom:
                model.add_constraint(x_fcsrp[(professor, course.Course_N
ame, course.Session, room, period)] == 0)
```

Figure 6 Snapshot of custom constraint 1

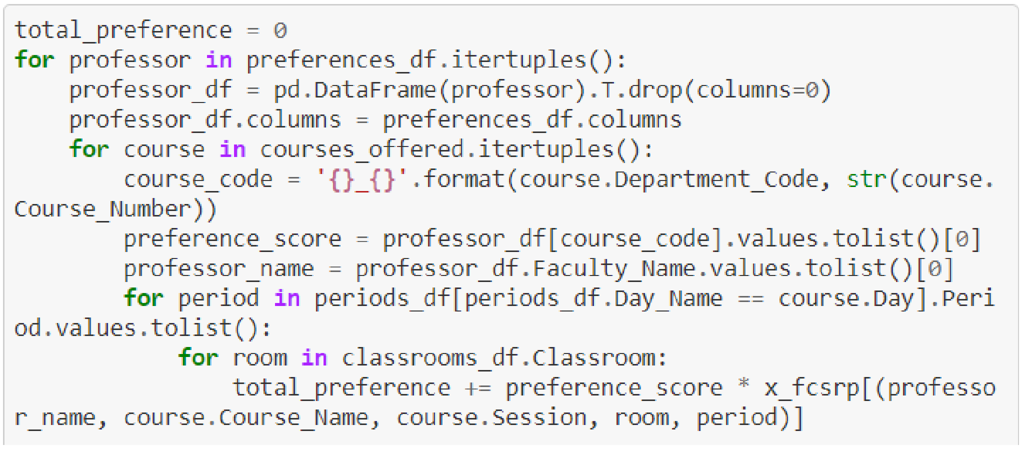
1. No classes are taught on Friday after 4 PM



```
HOLIDAY = 'Friday'
END_TIME = 16
for course in courses_offered[courses_offered.Day == HOLIDAY].itertuples
():
    for period in periods_df[(periods_df.Day_Name == HOLIDAY) & (periods
_df.Start_Time >= END_TIME)].Period.values.tolist():
        for professor in preferences_df.Faculty_Name:
            for room in classrooms_df.Classroom:
                model.add_constraint(x_fcsrp[(professor, course.Course_N
ame, course.Session, room, period)] == 0)
```

Figure 7 Snapshot of custom constraint 2

Once the feasibility constraints and special rules were written, the next step was to create the objective function. The purpose of this project was to maximize the total preferences of the professors. The Python code is shown in Fig. 8.



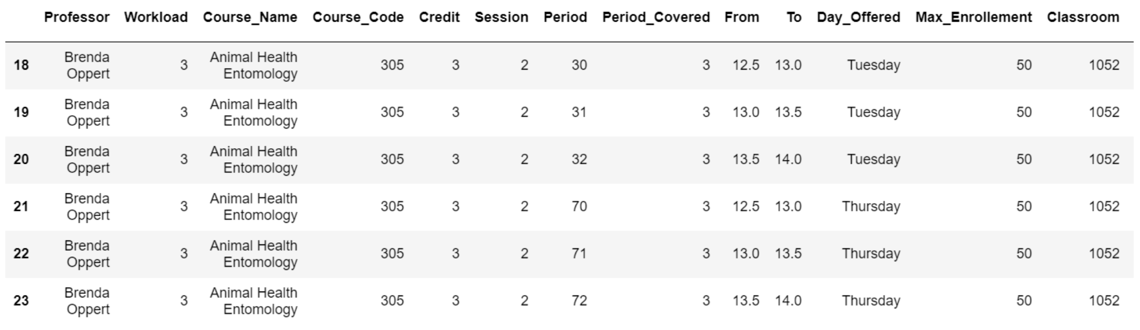
```
total_preference = 0
for professor in preferences_df.itertuples():
    professor_df = pd.DataFrame(professor).T.drop(columns=0)
    professor_df.columns = preferences_df.columns
    for course in courses_offered.itertuples():
        course_code = '{}_{}'.format(course.Department_Code, str(course.
Course_Number))
        preference_score = professor_df[course_code].values.tolist()[0]
        professor_name = professor_df.Faculty_Name.values.tolist()[0]
        for period in periods_df[periods_df.Day_Name == course.Day].Peri
od.values.tolist():
            for room in classrooms_df.Classroom:
                total_preference += preference_score * x_fcsrp[(professo
r_name, course.Course_Name, course.Session, room, period)]
```

Figure 8 Objective function written in Python

When the objective function was completed, all that was left was to solve the model and conduct post-processing. Post-processing was done so that the result would be cleaner and easier to visualize and understand.

6. MODEL SOLUTION

The objective of this class scheduling problem was to maximize faculty's preference scores for a list of courses offered by the Department of Entomology in the Fall semester. To do so, we developed a mathematical model bounded by different constraints. Fig. 9 shows a snapshot of the class schedule. It outlines the faculty teaching a course as well the room and period when that course is taught. Additional information such as maximum seats available and course credit as well as faculty's workload is also displayed.

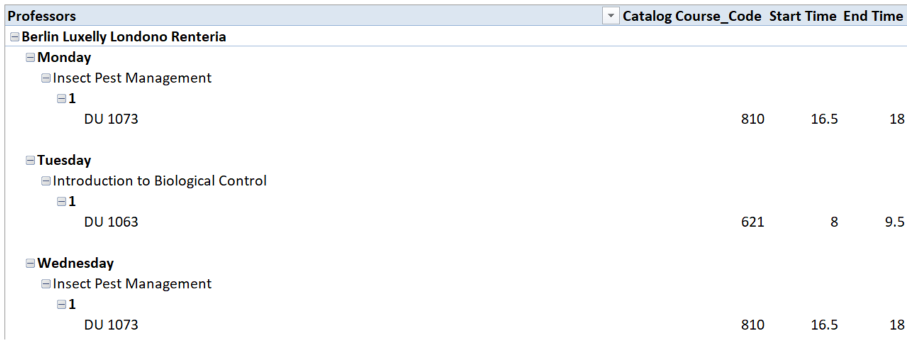


	Professor	Workload	Course_Name	Course_Code	Credit	Session	Period	Period_Covered	From	To	Day_Offered	Max_Enrollement	Classroom
18	Brenda Oppert	3	Animal Health Entomology	305	3	2	30	3	12.5	13.0	Tuesday	50	1052
19	Brenda Oppert	3	Animal Health Entomology	305	3	2	31	3	13.0	13.5	Tuesday	50	1052
20	Brenda Oppert	3	Animal Health Entomology	305	3	2	32	3	13.5	14.0	Tuesday	50	1052
21	Brenda Oppert	3	Animal Health Entomology	305	3	2	70	3	12.5	13.0	Thursday	50	1052
22	Brenda Oppert	3	Animal Health Entomology	305	3	2	71	3	13.0	13.5	Thursday	50	1052
23	Brenda Oppert	3	Animal Health Entomology	305	3	2	72	3	13.5	14.0	Thursday	50	1052

Figure 9 Class Schedule

Joint to this report is an Excel file that helps visualize the class schedule. We provided four separate visualization tools to analyze the final schedule. Each view seeks to ascertain that the schedule followed all the problem constraints. A full table is of course assignments is also contained in this Excel file.

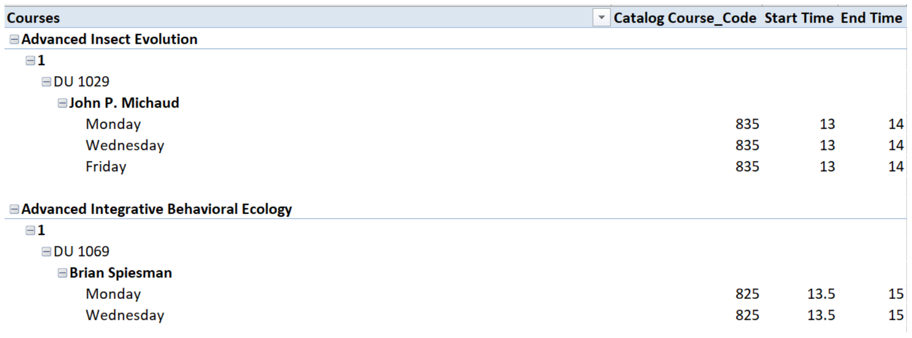
The first view is the professors' view. This view is meant to show when and where a professor teaches a course. This can be distributed to each faculty member via email communications.



Professors	Catalog Course_Code	Start Time	End Time
⊟ Berlin Luxelly Londono Renteria			
⊟ Monday			
⊟ Insect Pest Management			
⊟ 1			
DU 1073	810	16.5	18
⊟ Tuesday			
⊟ Introduction to Biological Control			
⊟ 1			
DU 1063	621	8	9.5
⊟ Wednesday			
⊟ Insect Pest Management			
⊟ 1			
DU 1073	810	16.5	18

Figure 1 0 Pivot table showing classes taught by each professor

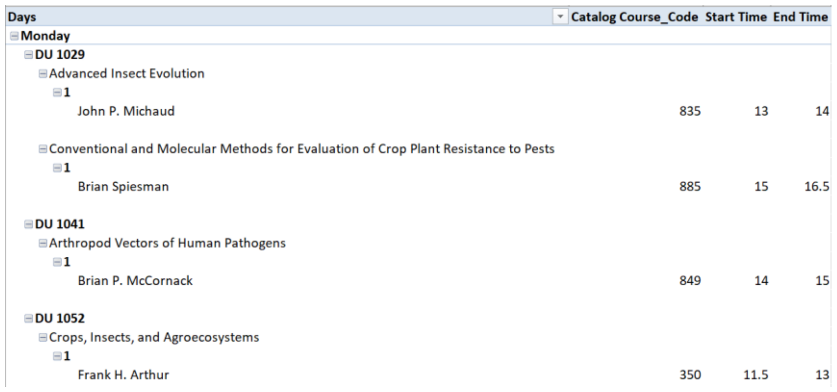
The second view is the course view. It presents the schedule from a course/session perspective. This view helps to answer questions such as: who teaches a class? When does a faculty teach a class? Where is a course held on a given day of the week? It is particularly relevant for classes with different sessions.



Courses	Catalog Course_Code	Start Time	End Time
⊟Advanced Insect Evolution			
⊟1			
⊟DU 1029			
⊟John P. Michaud			
Monday	835	13	14
Wednesday	835	13	14
Friday	835	13	14
⊟Advanced Integrative Behavioral Ecology			
⊟1			
⊟DU 1069			
⊟Brian Spiesman			
Monday	825	13.5	15
Wednesday	825	13.5	15

Figure 1 1 Pivot table showing classroom and professor teaching each course

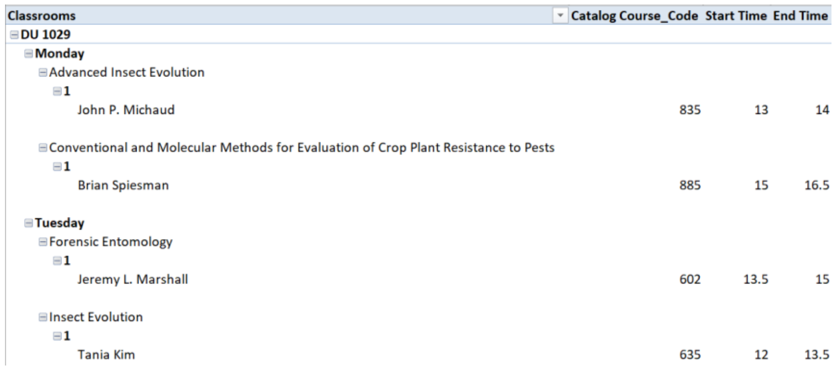
The third view presents the schedule from a week-day perspective. The purpose of this goal is to visualize courses when, where, and who teaches a course. It also provides a quick way to view courses taught in parallel.



Days	Catalog Course_Code	Start Time	End Time
⊟Monday			
⊟DU 1029			
⊟Advanced Insect Evolution			
⊟1			
John P. Michaud	835	13	14
⊟Conventional and Molecular Methods for Evaluation of Crop Plant Resistance to Pests			
⊟1			
Brian Spiesman	885	15	16.5
⊟DU 1041			
⊟Arthropod Vectors of Human Pathogens			
⊟1			
Brian P. McCornack	849	14	15
⊟DU 1052			
⊟Crops, Insects, and Agroecosystems			
⊟1			
Frank H. Arthur	350	11.5	13

Figure 1 2 Pivot table showing what courses are taught in each classroom each weekday

The four and final view presents the schedule from a classroom perspective. It emphasizes when a classroom is not available during the day.



Classrooms	Catalog Course_Code	Start Time	End Time
⊟DU 1029			
⊟Monday			
⊟Advanced Insect Evolution			
⊟1			
John P. Michaud	835	13	14
⊟Conventional and Molecular Methods for Evaluation of Crop Plant Resistance to Pests			
⊟1			
Brian Spiesman	885	15	16.5
⊟Tuesday			
⊟Forensic Entomology			
⊟1			
Jeremy L. Marshall	602	13.5	15
⊟Insect Evolution			
⊟1			
Tania Kim	635	12	13.5

Figure 1 3 Pivot table showing courses taught in each classroom

7. CONCLUSION

This report showcased a feasible course semester schedule intended to optimize the preference list. The parameters consisted of the department and course information, professor qualifications, and credit hours. The report shows the need for the implementation of this model to easily solve the scheduling problems for the departments within Mavericks University. Regarding future work, this model can be used within the many departments of Maverick University so long as the data is presented in a similar structure. This model is intended to be used by multiple departments across multiple semesters.