Optimizing Math Department Scheduling

Annabelle Piotrowski and Katherine McCallum
Faculty Mentor: Dr. Abra Brisbin
University of Wisconsin-Eau Claire

Introduction

Scheduling classes is a challenging and time-consuming task. To make this process less time-consuming and ensure an optimal result, we are using Python and linear algebra. We are using the Cplex library to build a model consisting of an objective function and a system of constraints. The models are built in Jupyter Notebooks using Python version 3.7.9.

Optimization:
In the long run this will be a more optimal solution for both the scheduler and the professors being scheduled, because this will make a schedule that is more ideal for all involved and be simpler to create and adjust as well, once a model is mostly completed. Starting in the Summer of 2023, this process has aided in scheduling classes for the subsequent semesters, including Spring and Fall of 2024. While there is still much work remaining, our code reduced the time required to assign sections of Calculus and Statistics in the math department.

Definitions

Model: A combination of a goal (or objective) function and constraints, defined using variables.

Optimization: Finding the maximum value of the goal function, while satisfying all constraints.

DataFrame: An object in Python, in the pandas library, to create a table with column and row functions.

Linear Programming: Optimizing the objective function while satisfying the constraints. With linear constraints, once a local optimal solution is found, that is the global optimum for the whole model.

Branch and Bound: An approach to solving linear programming problems in which variables must be integers. When a solution such as $x = 0.5$ is found, the problem is "branched" into two parts by adding the constraints $x \leq 0$ or $x \geq 1$.

Binary Variables

Each variable is represented by $X_{c,p,t}$ with one professor (p), class (c), and time (t). A value of 1 means that professor will teach that class at that time, and a value of 0 means they won't. The goal function now includes professors' preferences for courses and times, on an integer scale from 0 (not wanted) to 5 (really want that time/course).

We included the constraint that a professor can only teach one class at one time.

$$\sum_{c} X_{c,p,t} \leq 1$$

Another constraint is that the number of sections of each class must be between a minimum and maximum number.

$$min\ \#\ sections \leq \sum_{p}\sum_{t} X_{c,p,t} \leq max\ \#\ sections$$

An additional constraint is the minimum and maximum number of credits a single professor can have assigned to them in a week.

$$min\ \#\ credits\ \leq \sum_{c} H_c X_{c,p,t} \leq max\ \#\ credits$$

Where H_c= the number of credit hours for class c.

Meeting Patterns

In 2022-23, each binary variable represented a course and professor at one time and day (e.g., Monday at 9 am). This year, we changed our model to represent meeting patterns instead (e.g., MWF at 9 am). This simplifies many previous constraints. For example, we no longer need constraints to ensure that if a course meets multiple days per week, then all the class meetings occur at the same time. The meeting pattern approach also simplifies scheduling courses with unusual combinations of credit hours and meeting patterns, such as a 4-credit lab that meets three days a week for an hour, and a fourth day for two hours, e.g., at 10:00AM to 10:50 AM MTW, and 10:00AM to 11:50AM on R.

Additional examples:
- A meeting group would be classes that meet for four times a week for an hour, e.g., MTWF at noon to 12:50 PM
- A class that meets two days a week for two hours, e.g., MW at 1:00PM to 2:50PM
- An asynchronous class that carries credits but has no scheduled meeting time

Future Work

- Add classroom assignments to the model
- Allow preferences regarding number of different preps per professor
- Finding a more efficient way to input preferences for professors
  - Focusing on how to input preferences for time intervals to avoid manually entering the same preference for multiple times

References and Acknowledgments

Jupyter Notebook
Python Programming Language
Pandas Database
UWEC Office of Research and Sponsored Programs
UWEC Learning and Technology Services
UWEC Printing Services

Overlapping and Back-to-Back Classes

A professor cannot teach multiple classes at the same time, so it is necessary to find which time slots overlap. Figure 1 shows how a code cell works to identify overlapping class times.

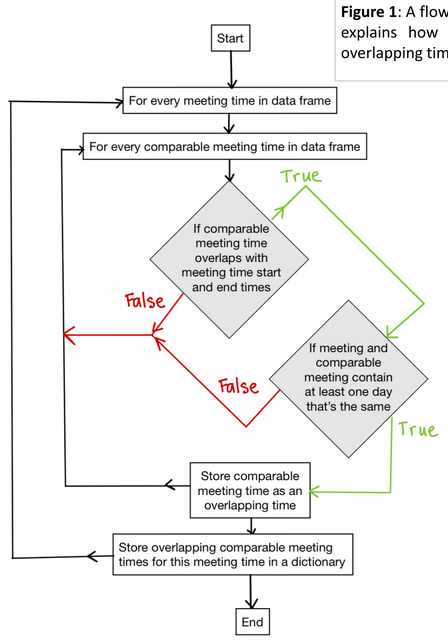

Figure 1: A flow chart that explains how to handle overlapping times.

Overlapping class times are then used to build constraints, as described in the "Binary Variables" section. Back-to-back class times are identified in a similar way but are used to adjust the goal function.

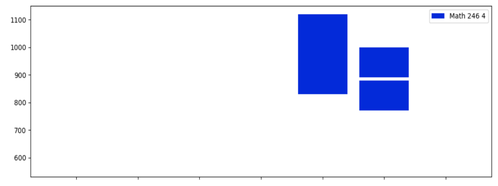

Figure 3A: Example classes for this professor A, who has a strong preference for back-to-back classes.

Professors' preferences about whether they teach back-to-back classes are included in the model through the goal function. This is done by adding the first time slot for a professor for a specific class multiplied by a back-to-back time slot for a professor for a specific course multiplied by that professor's preference for the same course being back-to-back. In one simulated example, Professor A wants courses back-to-back; the resulting schedule is shown in Figure 3A. However, Professor B does not want classes back-to-back, and thus has gaps between the hours of when she is teaching, shown in Figure 3B.

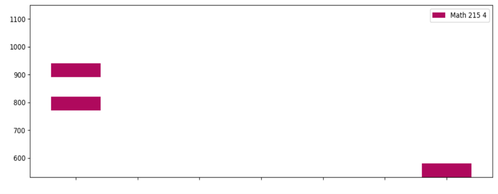

Figure 3B: Example classes for Professor B, who does not want back-to-back-classes.

Course Distribution Throughout Day

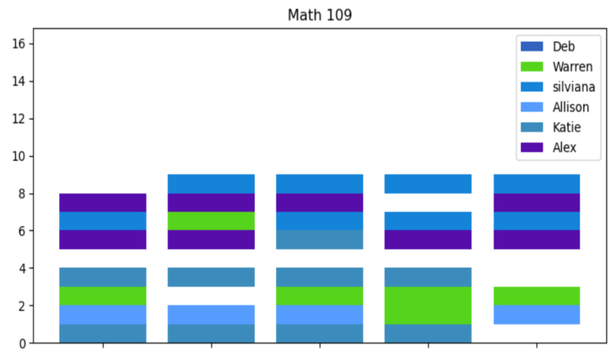

Figure 2a: A chart that shows when classes are happening throughout the week, with no constraints on distributing evenly throughout the day.

An optimal schedule should be convenient for both professors and students. Figures 2a and 2b show the schedules for one course in two different solutions of the model. The above model (Figure 2a) has no constraints on the distribution of sections throughout the day. This means that 14 sections take place in only 8 time slots, reducing options for students who may be taking other courses at those times.

The model below (Figure 2b) adds the constraint that no two sections of the same course may occur in the same time slot. (Note that time slots 0 to 8 correspond to one-hour classes and slots 9 to 16 correspond to two-hour classes.) This results in classes that are more spread out throughout the day, providing more options for students. We are continuing to work on ways to address the distribution of classes throughout the day to enhance our model's usefulness.

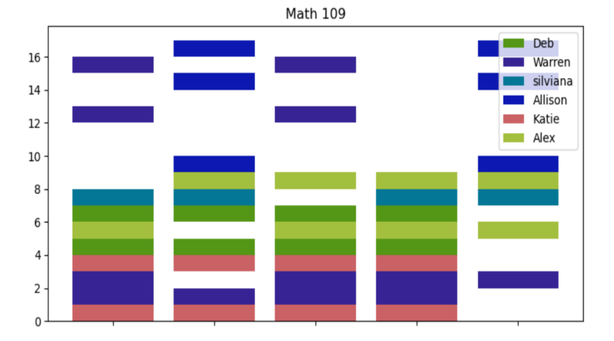

Figure 2b: A chart that shows when class are happening throughout the week, with the added constraint that no two sections should occur in the same time slot.

Visualizations

The visualizations are created using histogram plots in the matplot library, but with bars ranging from the start time to the end time rather than starting at 0, the bottom of the graph, as is standard in a histogram. The x axis of the histogram represents individual days or combinations of days. We have created visualizations based on courses and based on individual professors.