Extension to the Project for Future Improvements

There are several topics we learned later in this course that we could apply to our model to further improve it, as well as some things that can be improved that weren't due to time constraints for the project.

Time slot conflict done more robustly

The time slot constraint we used is limited in its usage. In our model, a time slot conflict only occurs when two courses have the same start and end time. This means that 2 courses in the optimal schedule could theoretically have a conflicting (but different) time slot, such as a double-length class conflicting with a single-length one starting at the same time.

To improve our model, we can first run through all the time slots and find a list of all possible times in which a course begins or ends. From there, we create a time slot for each section of time. When we create our time conflict constraints, we include a course in a time conflict constraint if that constraint's corresponding time slot is a subset of the course's allotted time. This would increase the amount of time constraints, but not by a significant amount, as it does not multiply with the number of courses that exist.

Dantzig-Wolfe decomposition.

We can use Dantzig-Wolfe decomposition to solve this problem more efficiently. This is because we have many constraints that use variables that are used only in that constraint (save for a few coupling constraints).

One way to apply Dantzig-Wolfe is to have the course section constraints (the sum of the number of course sections for the same course should be 1) each be a decoupled equation, and the coupling constraint would be the remainder of the equations. So if there are x unique courses (each with several sections), then we can frame the problem as x separate constraint sets (each with one constraint: the sum of all course sections is 1) and the rest of the constraints being the coupling constraints (the time slot and credit count constraints). If we do this, we can utilize Danzig-Wolfe decomposition to save on memory.

We can also do a similar decomposition by having one constraint per time slot, and having the rest of the constraints (course sections and credit count) to be the coupling constraints. Both of these methods take one of the 3 types of constraints and use those types of constraints to create the subproblems.

Administrator's Perspective

The entire course scheduling problem as solved in this project was done from the perspective of a student trying to build a schedule off of an already existing master schedule. The alternative way of looking at the problem is to try to build a master schedule by looking at student demand. We assume that we are given which professor is teaching which course and also a list of proposed course schedules for each student. We try to minimize course conflicts for students subject to there being no course conflicts for professors, along with constraints about how many times per week each course must run a lecture, constraints about two days in a row with a lecture, and so on. This would be an entirely different optimization problem from what was done in this project, but still achievable with linear optimization and integer linear programming methods used in this project and in class.

Benders Decomposition

Using the administrator's perspective leads to the possibility of turning the problem into a stochastic linear programming problem and using Benders decomposition to solve it. Assuming we don't have student data about what students want to take what class for the next semester, we can instead use previous class roster sizes (and waitlist sizes) to generate a probability distribution for how many students will want to take each class. Since we use discrete probability for stochastic linear programming, we can make a probability for every integer number of students for every single class, and from there we can generate probabilities for each scenario of student demand for courses. From there, we apply Benders decomposition as shown in section 6.5 of the textbook for this stochastic linear programming problem, using it to solve a problem like the one mentioned in the previous section.