INDE 597 Final Project

The goal of this project is to create an interactive, practical program that accomplishes a given task. You may work in groups or alone, but I suggest you work in groups and distribute the work in such a way that group members work on separate things in parallel. Each group will give a presentation on the last day of class showing the results of their project. You will also turn in your code, which should include comments and be easy to read. In the class presentation, provide details about the theory and implementation of your project. Be prepared to answer questions from the audience at the end. You will be graded on how well you solve the given task, on the functionality of the graphical user interface, the readability of your code, and on how clear your presentation is. The projects are somewhat open-ended, and you are free to take certain creative choices with the implementation, make reasonable assumptions, etc.

Task: A department chair has to assign courses to the faculty of the department. Different faculty members have different preferences, priorities, and requests. Your task is to design a versatile program which creates an optimal course schedule. The program should accept as input the following data:

1. An entry for each course, including:
    a. Time-of-day/day-of-the-week restrictions for the class (e.g. Calc 1 must be offered on MWF sometime between 9am and 1pm)
    b. Number of days-per-week/hours-per-day the class meets (e.g. Calc 1 meets 3 days a week, 50min each day)
    c. Number of credits the class counts for (e.g. Calc 1 is 4 credits)
    d. How many sections of the class there are (e.g. there are five sections of Calc 1 and one section of Functional Analysis)
2. An entry for each faculty member, including:
    a. Preferred times and days of the week that the faculty wants to teach (e.g. Alice doesn't want to teach earlier than 10:00am; Bob can teach anytime except Monday 12:00-2:30pm and Thursday 2:00-5:00pm)
    b. Teaching load of the faculty (e.g. Bob has to teach 15 credits this semester)
    c. Number of *distinct* courses that the faculty member wants to teach (e.g. Clifford has to teach 4 classes this semester but he wants no more than 2 distinct classes; so, he might get assigned three sections of Calc 1 and one section of Calc 2)
    d. A measure of the faculty's preference for teaching each course. For example, you may use a designation like "wants to teach course X", "doesn't want to teach course X", "is indifferent about teaching course X" or a point-based scale like in the table below:

	Calc 1	Calc 2	Calc 3	Lin. Alg.	Diff Eq.	Graph Th.	Stats
Alice	90	80	70	100	20	20	50
Bob	0	0	0	100	0	100	0
Clifford	50	50	50	50	40	20	100

3. How many classes can be taught simultaneously each day of the week and each hour of the day (e.g. at most 4 classes can be placed in the 9:00am-10:00 slot on MWF because there are only 4 classrooms available at that time)
4. Pairs of classes that cannot be taught at the same time (e.g. Calc 1 and Linear Algebra are both core courses and should not overlap because many students need to take both)

With this in mind:

- Your program must output an assignment of classes to faculty, in which all classes are covered.
- The sum of the credits of the classes that each faculty member teaches must be reasonably close to the faculty member's teaching load (e.g. within 1-2 credits).
- There must be a mechanism by which faculty members' preferences and priorities are taken into account. Their preferences and priorities should be taken as input. For example, faculty member i could be given p_i points to distribute to different preferences; the more points they give to some preference, the more likely it should be that the assignment produced by your program is compatible with that preference (e.g., if Alice has 75 points to distribute, she might allocate 60 points toward not teaching before 10:00am on any day, 15 points toward not teaching more than two distinct classes, and 0 points toward which specific courses she teaches).
- The assignment found by the program should be optimal with respect to the faculty preferences. You may have different options (to be taken as input) about what exactly the program tries to optimize (e.g. maximize the number of faculty who have all their preferences met, minimize the number of violated preferences over all faculty, maximize the weighted sum of everyone's preferences, etc.).
- You may add other features, options, and more complex constraints as you see fit (e.g., Clifford wants at least one day every week where he doesn't teach, but doesn't care which day; the program outputs a set of 5 different near-optimal schedule assignments instead of just the optimal one; the chair has an overwrite option that assigns a course to a specific time slot and to a specific person, regardless of preferences).
- You may make certain reasonable assumptions on the input (e.g., the total number of credits for the courses offered is roughly equal to the total teaching load of all the faculty, there are a limited number of classrooms available at any time, the number of classes that have to be offered is smaller than the number of time slots multiplied by the number of classrooms available in each time slot, there aren't too many pairs of classes that can't be offered at the same time, etc.).
- The user interface for your program should be intuitive, easy-to-use, and, if necessary, include instructions on how to be used. One option is to have the interface be a website made with HTML/CSS/Javascript and have textboxes, etc, for entering data; then, either call an IP solver or have the Javascript generate code based on the input which can be copy-pasted into an IP solver like Gurobi, which would then generate a solution. If you choose to make your interface using HTML/CSS/Javascript, I may be able to help you with creating the interface.
- Be creative and have fun!