Class Schedule

Gavin McCauley
Dzmitry Siaukovich
Marion Smith
Bowei Li
Siyu Lin
Karl Giesing

Project Agreement

Introduction

The software simulates creating an optimal class schedule based on matching courses to rooms and professors, such that every course has a room with the proper capacity, and the amount of time professors spend on campus is minimized.

Components

1. Room
   - Each room has its capacity and an ID in the format "building-floor-number" (as with UMass rooms.)
2. Professor
   - Each professor has a full name, and an ID consisting of eight numeric digits.
3. Course
   - Course includes the name of course (e.g. "Probability and Statistics"), the course ID (e.g. "MAT-351-01"), the Professor ID for this course, how many students are enrolled in the class, and a list of other courses that cannot be taught in the same time period ("conflicting"courses).
   - Each course section will be treated as an entirely separate course. If a course has a section number, it must be the final part of the course ID, as above.
4. Schedule
   - A schedule is a list of daily time blocks, per room. Each room's time block will contain a reference to the course that is being taught in that room in that time block. The program's output will be a representation of a particular schedule, optimized as described.
5. Scheduler
   - A scheduler is an object that generates a schedule, given the list of courses (including conflicting courses) and a list of rooms. Different scheduler modules may use different algorithms to produce optimized (or near-optimized) schedules.

Input

The input shall be the following information. It shall be supplied in three separate DSV (delimiter-separated value) files. [The exact format shall be specified at a later meeting.] The file names shall be passed to the program on the command line, in the specified order. Each file shall contain the information below, in the specified order.

1. Room information file: room ID, room capacity
2. Professor information file: professor ID, professor name
3. Course information file: course ID, course name, Professor ID, number of enrolled students (as an integer), number of time periods required by the course (as an integer), list of conflicting course ID's (if no conflicts exist, this section shall be empty)

Output

The output shall consist of this information:

1. The time period (including day of the week)
2. The room ID
3. The ID of the course scheduled for that room
4. The name of the course scheduled for that room

All output shall be sent to the console.

Constraints

- Each time period shall be a block of time 1.5 [?] hours long, with the start times beginning at 8:00am, and ending at 8:00pm. Time periods begin on Monday and end on Friday.
- A room must not be scheduled for more than one course in the same time period.
- Courses must only be scheduled in rooms that have a capacity greater than or equal to the number of students enrolled in that course.
- Conflicting courses must not be scheduled in the same time period.
- Each professor must teach no greater than three courses.
- Different courses taught by the same professor must not be scheduled in the same time period.
- Courses must require 2 time periods; or optionally 1, 2, or 3 time periods, if we have time to code this feature.
    - (Optional) If a course requires 3 time periods, that course must be scheduled on Monday, Wednesday, and Friday.
    - If a course requires 2 time periods, that course must be scheduled on Monday and Wednesday, or on Tuesday and Thursday.
    - (Optional) If a course requires 1 time period, it may be scheduled on any day of the week (but must be scheduled on exactly one day of the week).

Other Requirements and Optimizations

- The schedule shall be optimized such that each professor spends the minimum amount of time on campus (subject to the Constraints section, above).
- The software will halt execution in ~~a~~ ~~reasonable~~ ~~time~~ under one minute. [Alternative: "The program's algorithms should run in polynomial time." Something that takes under a minute on your desktop may take five minutes on my laptop. If we want to specify one minute, we need to also specify the number of inputs and the computer's processor speed. Or, we need to keep track of absolute (system) time somehow.
  By the way, the **general** scheduling algorithm is known to be NP-complete, so we'll have some work to do. - Karl]
- If the optimal schedule has not been found within that time, it will output the most optimized schedule that it has found at that point.