EC504- Final Project Documentation

Course Scheduler

By
Esteban Hernandez and Luke Atlas

Description of the Problem

The problem many students face every semester is finding the schedule that is best suited for their studying habits, sleep habits, and overall mental health. Our “Course Scheduler” project aims to help students who want to have an easy way to check all their possible schedules in an easy to use program. We believe that having more options for course schedules will promote students’ academic success, utilization of campus space, maximize course schedule flexibility, and enable better overall mental health.

Project Goals

By using our program, students will have an easier time picking courses by:

- Filtering out the overlapping courses and making sure that all Lectures, Discussions, and Labs are included, so you don’t have to worry about missing needed sections
- The Graphical GUI shows the students their schedule on a clean graph styled schedule that is easy to comprehend
- Students will also have more schedule options to pick from because our program compares all the selected courses’ possible combinations.

Overall, students will get to see all the possible ways they can create their own course schedule, possibly leading to improved mental health across a semester in which they could discover a schedule that best suits their sleeping habits.

Background Information and Relevant References

Michael A. Gallo and Michael Odu conducted a study to find the relationship between student scheduling and student achievement in College Algebra.[1] The study examined 116 Florida community colleges and controlled the faculty and students' ability by having consistent course material and work. The course (College Algebra) was offered in 3-course options: one day/week, two days/week, and three days/week. The study found that there was a significant association between course schedules and academic success. It found that students who were taking the course for one day per week consistently scored worse than students taking the course that met two days per week and three days per week.[2]

This study shows evidence of the spacing theory which suggests that students comprehend more when spreading out the learning in shorter but more frequent classes. This way students are repeatedly being exposed to the course material making them connect to prior material more easily. This is especially true when the course is taught in sequence (new material relies on prior material learned) rather than in isolation (new material doesn't rely on prior material). This is because sequence courses rely more on prior knowledge and students taking a course one day per week will not have the proper time to connect the new material to prior material learned.

In another study conducted by Linda G. Carrington at Sam Houston State University, Linda studied the association between course schedules and success rates of students taking

---

[1] Gallo, M. A., & Odu, M. (2009). Examining the Relationship Between Class Scheduling and Student Achievement in College Algebra. Community College Review, 36(4), 299–325.
[2] Gallo, M. A., & Odu, M. (2009). Examining the Relationship Between Class Scheduling and Student Achievement in College Algebra. Community College Review, 36(4), 299–325.

Accounting and Business courses.[3] The study was conducted by examining the grades of students taking Intermediate Accounting I&II over the course of 3 years. The course was offered in 4 different schedule options one day/week, two days/week, three days/week, and five days/week (summer course). The study found that there is a significant association between course schedules and academic success. The study found that students taking the course three days/week failed or dropped the course consistently more than students taking the course one day/week, two days/week, and five days/week (summer course).[4]

Both studies show evidence that there is a significant correlation between course schedules and academic success. Although, both studies show contradictory evidence on what the best course schedule option is, however, it does suggest that the best course schedule varies for each student. So our project's aim is to provide students the option to personalize their academic experience to tailor to their individual study habits and schedule needs by providing them a convenient way to check all their course schedule options in a simple and practical program.

High-level Description of the Implementation

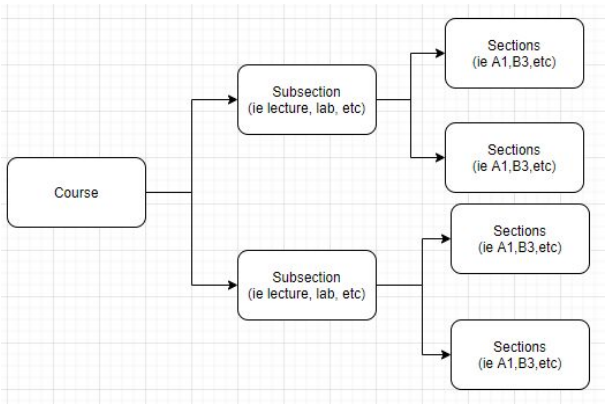
Data Structure

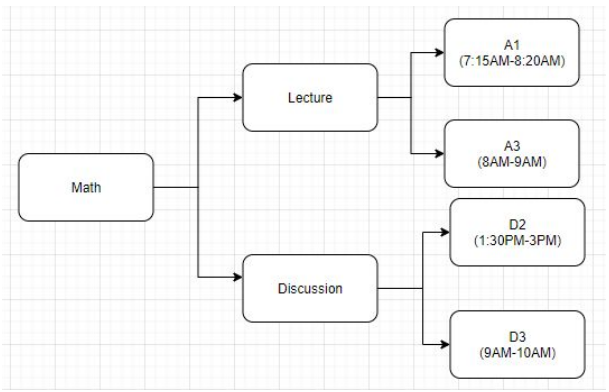
Our project's core structure utilizes class objects that contain course information. Each course object contains a vector of objects named "subs" containing the details of what subsections (i.e. lecture, lab, discussion, etc) a course requires. Then each subsection contains a second nested vector of objects named "sects" that contains all of the available sections (i.e. different times or professors for a given subsection).

---

[3] Carrington, L. (2010). The Impact Of Course Scheduling On Student Success In Intermediate Accounting.

[4] Carrington, L. (2010). The Impact Of Course Scheduling On Student Success In Intermediate Accounting.





The diagram above shows a great depiction of how the objects are nested. All of the children of any “node” in our “tree” is contained within a vector of objects in its parent object/node. The sections (the ‘bottom’/far-right of our ‘tree’) shown above each contain a vector of vectors of doubles (e.g. <<1,8.30,9.15>,<3,12.20,13.05>>). The vectors of doubles contain 3 elements. The first element is a numeric indication as to which day of the week the section meets (Monday=1,Tuesday=2,Wednesday=3,etc). The second and third elements represent the start and end time of a section. The formatting we found most easy to implement was to represent time in military hours (24 hour clock) and use the decimal like colon would normally be used in a digital clock (i.e. 8.15=8:15AM, 14.29=2:29PM, etc).

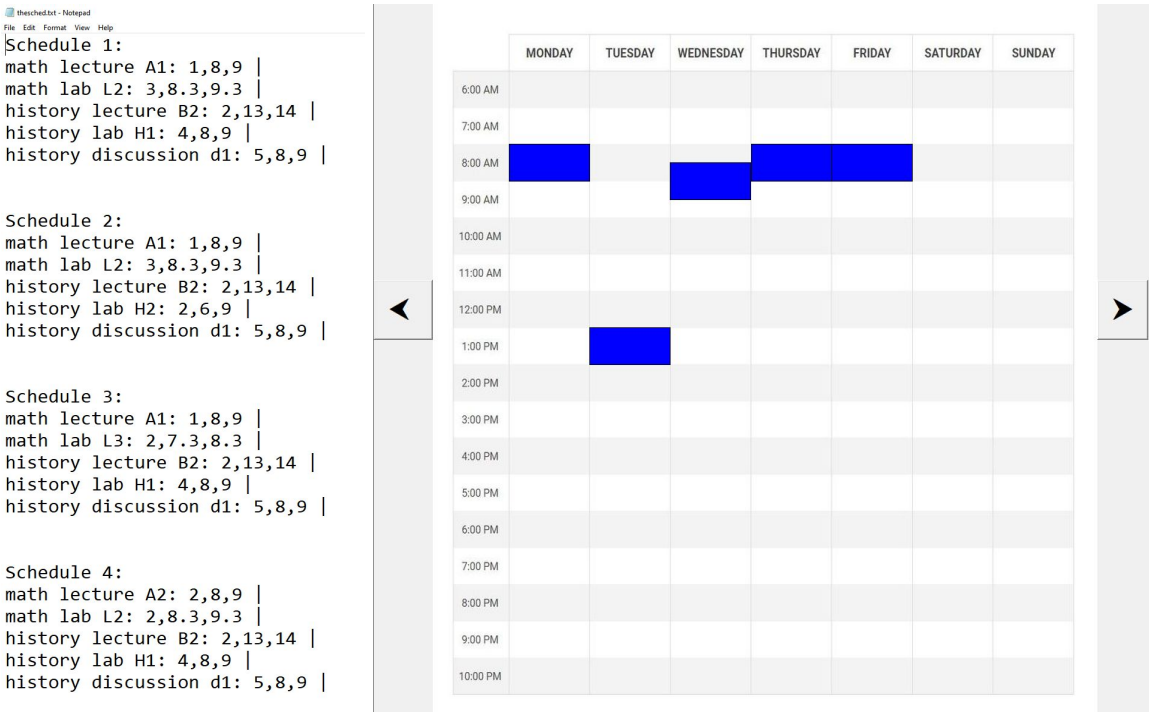
Algorithm

To compute all the possible combinations of all the sections of all the subsections of all the courses could take an incredibly long time. This is represented by the equation:

$$n!/(r!(n-r)!)$$

where n is the amount of possible classes (including various times) and r is how many classes we need to be enrolled in. To help mitigate this, we first check each individual course to find out what individual sections for that course can be taken together to fulfill all the subsection requirements (needed lectures, labs, discussions, etc) without having any time conflicts. These groups of viable section combinations are stored into vectors. These viable combinations of

sections for each course are then compared to the viable section combinations of the other courses in a second filter. This filter will check that each course is only signed up for once and has no time conflicts with sections in the section groups of the other courses. This second filter leaves us with a vector containing the various combinations (stored in vectors) of sections from all the courses that fill a schedule without time conflicts. This schedule is then saved into a text file (see below) that can then be turned into a visual graph (see below) by our schedule visualizer code.



Features from the Project Proposal

We had two big features we committed to accomplishing in our project proposal. Firstly we agreed to create an algorithm that takes user inputted course information and outputs viable course schedules. We successfully accomplished this in the time allotted to us using the methods described above in the previous section of this document. In summary, our program finds all combinations of course sections and then checks them in a preliminary and secondary round of comparisons for time conflicts. The primary check narrows down the scope of

comparison required for the secondary check. The result of our two filters is a list of viable schedules that are saved into a text file.

The second goal we aspired to accomplish was to create a program that takes our outputted schedule text file and creates a visual graph from it. We succeeded in creating a basic version of this using Tkinter (python visual library) to add the courses (represented as rectangles) to a static image of a week long time table. However, with more time, we would have liked to distinguish all the courses on the graph with varying colors and add a legend. Additionally, we would like to have made the arrow buttons cycle unique schedule titles at the top of the graph while cycling through the various schedules. Overall we are quite pleased with the neatness and clarity of our graph given the time we were able to allot to it due to the complexity and time involved in getting the schedule maker algorithm functional and bug free. Our graph enables the user to see the spread of their courses to determine which days and time spans are ‘course-heavy’ and can use this knowledge to make a better educated decision as to which course they would like to commit to. Overall it was a really fun assignment that pushed us to learn a lot of new things along the way and were happy to have the opportunity to do so.

Work breakdown

Esteban Hernandez- Worked on the project idea, frontend coding, PowerPoint presentation, and documentation for the project.

- Project idea- Esteban suggested the idea of a graph styled visualizer of the course schedule running on python.
- Coding- Esteban worked on the frontend GUI side. He created a python program that takes the output of the course scheduler backend algorithm and creates a visual graph scheduler like the graph shown on the bu student link.
- Powerpoint presentation- Esteban worked on the Project Description, Algorithm visualization, and Conclusion slides.
- Documentation- Esteban worked on the Project Description, Project Goals, and Background research material for the documentation. He also worked on the work breakdown.

Luke Atlas- Worked on the project idea, backend coding, PowerPoint presentation, and documentation for the project.

- Project idea, Luke came up with the original project idea of a course scheduler program and proposed it to Prof. Brower. He also suggested the possibility of a database feature or a web-based course scheduler.
- For the coding, Luke worked on the backend C++ algorithm that outputs all the possible combinations of course schedules. He also created a readme file and added comments to the code.
- Powerpoint presentation- Luke worked on the Algorithm Description and Program outputs slides. He also made a project demo video that shows how our program runs.
- Documentation- Luke worked on the High-Level Description of the Implementation and the Features of the Project Proposal. He also worked on the work breakdown.

Citations

Gallo, M. A., & Odu, M. (2009). Examining the Relationship Between Class Scheduling and Student Achievement in College Algebra. Community College Review, 36(4), 299–325. Retrieved December 12, 2020, from https://doi.org/10.1177/0091552108330902

Carrington, L. (2010). The Impact Of Course Scheduling On Student Success In Intermediate Accounting. Retrieved December 12, 2020, from https://files.eric.ed.gov/fulltext/EJ1060320.pdf