Optimizing The Scheduling of Classes

Operations Research II, 21-393, Final Project Report

Brian Brazon, Ryan Chandler, Egan McClave, Christian Schmidt

December 19th, 2017

Table of Contents

1 Abstract

For incoming and older students alike, navigating through the list of courses that are offered to create a manageable schedule can be an extremely daunting task. The process of scheduling becomes especially difficult once you consider all the abstract graduation requirements students must take in order to receive their degree. Since students might not even be 100% committed to the education path they originally propose they might have to add several other courses in their schedule to diversify their options. In addition, not all students have an advisor that can guide their educational experience and provide insightful feedback on potential schedules. Thus, over the course of this paper we will explore how to optimize the allocation of all necessary classes to graduate for a student aiming to earn a Bachelor of Science from the Carnegie Mellon University (CMU) Mellon College of Science for Mathematical Sciences with a Concentration in Operations Research and Statistics. This optimized model would primarily be minimized in terms of overall difficulty of the schedule. The rest of the paper will describe the problem at hand, our technical assumptions, our approach to the problem, our results, and finally future work that could be done.

2 Background of the Problem

The academic workload provided by CMU is known to be extremely time-consuming, due to the rigor of its classes. It is not difficult for students to find themselves overwhelmed by the number of hours per week they are spending on their coursework. The current degree requirements for the Operations Research and Statistics concentration demand a student take a wide breadth of classes in addition to particular, required classes that define the concentration. The base of this degree is ten mathematical science courses, five statistics courses, and five

courses in economics, business, and computer science, an introductory writing course, a general computing course, a freshman seminar, and five ENGAGE courses that all CMU students must take. In addition to these required courses, for students in MCS, other degree requirements include the following electives: five Depth; one Life Science; one Physical Sciences; one Mathematics, Statistics and Computer Science; one STEM based; one Cultural and Global Understanding; and four Humanities and Social Sciences. For each elective group, a list of approved classes is provided for a student to select his or her choices. Finally, three hundred sixty units are necessary to receive a degree. To complete our project we decided to code all of our work in Microsoft Excel and R.

3 Technical Assumptions

There were several assumptions that were made to simplify the necessary calculations and to simplify our model. The first assumption was that the difficulty of any one class should be considered consistent across semesters. Our next assumption was to use an average of the Faculty Course Evaluation (FCE) data of each class when predicting the amount of time spent on a class. We further assumed that if the amount of time spent on a class does not appear in the FCE data, then the unit amount of the class will be a viable alternative. Our next assumption was that we would not consider the grade requirements that are detailed in each course description when deciding on what classes to put in the schedule. Probably our biggest assumption was that all the classes one needs to take are well documented on the course website and that our own interpretation of the requirements are correct. Our next assumption was that no classes could double count and meet multiple requirements. The next two assumptions were that we would assume that a student would always be able to register for their desired classes, and that all the classes we suggested to take had no conflicts with one another within the week. Finally, the last few assumptions that we used were about the specific labeling of classes used for requirements. We assumed that the list of all classes offered for the History and Social Sciences requirement

would also meet the Cultural and Global Understanding requirement. We also assumed that the list for STEM and Free elective requirements just primarily include all the courses listed on the MCS website and not every class offered by CMU which is how it is currently.

There were also a few simplications that needed to be made in order to make the calculations in such a short period of time. Firstly, we had to not consider classes that were listed as "offered intermittently" from the list of classes that could be taken in a semester. This would make each class have a designated time of the year for when it is offered and simplify the types of schedules that could be created by the algorithm. The next simplification was that we would not consider taking prerequisites classes as potential co-requisites. This would reduce the number of optimal schedules that could be created but it would be easier for the program to handle the calculations.

4 Data

To run our algorithm we needed several pieces of data about classes offered by CMU. Specifically, we needed information on the requirement the classes meet (label), the number of units of the classes, the prerequisite classes that must be taken prior, the semester the classes are offered, the hours per week the average student spends working for each class and finally the total number of students who filled out the course evaluation. Most of the required information had to be manually inputted into an Excel spreadsheet as there was no easy way to grab information from the Mellon College of Science course catalog. The last two pieces of information were easily retrievable from the CMU FCE website.

The information that was collected was presented in R as two dataframes. The "Courses" dataframe represented all the important information from the course catalog about the specific concentration. The "Prereq" column was coded so it could be evaluated with boolean logic and simple regex/string search when figuring out if the prerequisite classes have been met. The

“Time” column of the dataframe was coded so “0” would represent the class only being offered in the Fall Semester, “1” for classes only in the Spring, and “2” for classes that were offered year-round.

Course # <chr>	Labels <chr>	Units <dbl>	Prereq <chr>	Time <dbl>
21-260	DQ	9	(21-122)	2
21-261	DQ	10	(21-122)	1
33-231	DQ	10	(21-122)&(33-132|33-107|33-122|33-152|33-142|33-112)	0
21-292	REQ	9	(21-122)&(21-241|21-242)	1
21-369	REQ	9	(15-110|15-112)&(21-259|21-269|21-268)&(21-240|21-241|21-242)&(21-260|33-231|21-261)	2
21-393	REQ	9	(15-251|21-228)&(21-292)	0

Figure 1: Snapshot of “Courses” dataframe

Labels were based on choices that students are required to make. For instance, if a class is required for graduation and there are no substitutions for the class, then we assign it the label “REQ”. Any requirement where students have a choice in selecting classes are called alternative electives. Alternative electives make up a large percentage of the groupings. For instance, there is a differential equations requirement that must be completed but there is a choice of 3 classes students can choose from. So we have assigned to these classes the grouping “DQ” (differential equations). The remaining labels include: "DM" (Discrete Math), "MAT" (Matrices), "C3" (Calc 3), "PR" (Probability), "CS" (Computer Science), "EC" (Economics), "DE" (Depth Electives), "LS" (Life Sciences), "PS" (Physical Sciences), "CG" (Cultural and Global Understanding), "MSC" (Mathematics Statistics and Computer Science).

The second data frame, “FCE”, represented key information collected from the Faculty Course Evaluations. The values in the “Difficulty” column was the average of all the hours students felt they spent on the coursework for the class from the previous three years. We also included the total number of responses as a measure of accuracy of the calculated difficulty. For example, it might not make much sense to consider the difficulty of a class if the total number of responses across three years is less than twenty as the calculated value may not be truly representative of the level of work done for the class. Within our “FCE” dataframe there are a

total of 2524 classes, out of 4338 total, with less than twenty responses, and almost half of the 2524 classes had at most 5 responses.

Course # <chr>	Difficulty <dbl>	# of Responses <dbl>
90-739	1.005	192
94-739	1.074	68
90-740	1.470	117
03-301	2.000	25
03-401	2.000	11

Figure 2: Snapshot of "FCE" dataframe

5 Implementation

We start our algorithm by inputting the "suggested schedule" from the Math Department website. All alternative electives were listed as "XX-XXX", which we left in as a placeholder to be replaced by future courses.

Course # <chr>	Labels <chr>	Semester <dbl>
21-120	REQ	1
21-127	REQ	1
38-101	REQ	1
76-101	REQ	1
99-101	REQ	1
XX-XXX	LS	1
XX-XXX	CS	2
21-122	REQ	2
XX-XXX	MAT	2
XX-XXX	PS	2

Figure 3: Snapshot of "suggested schedule" prior to Part I

Part I of our algorithm then sought to take this (incomplete) suggested schedule and flesh it out into a complete, feasible schedule. All placeholder classes in the suggested schedule were to be matched with classes of similar labelings. For each alternative elective, we formed a subset of all courses that (a) matched the correct labeling, (b) had not yet been placed inside of the suggested schedule, and (c) had placed all the prerequisite classes in the semesters prior to the semester of the current alternative elective placeholder. We then took the class from this subset

with the lowest difficulty and placed this course number for the placeholder course number in the suggested schedule. This resulted in a low-difficulty schedule that satisfied all of the requirements of the major.

Course # <chr>	Labels <chr>	Semester <dbl>
21-120	REQ	1
21-127	REQ	1
38-101	REQ	1
76-101	REQ	1
99-101	REQ	1
03-115	LS	1
15-110	CS	2
21-122	REQ	2
21-241	MAT	2
33-121	PS	2

Figure 4: Snapshot of "suggested schedule" after Part I

We then sent this new schedule into Part II of our algorithm, with the goal of evening out the difficulties between semesters. While the output from Part I did give us a schedule that would allow a student to graduate in 4 years, the main problem was that some semesters would be vastly more difficult than other schedules. To help fix this, for each semester we determined was too difficult (in comparison to other semesters), we would try to take a single class out and put it into an "easier" semester. However, we had to make sure that such a change could only happen if it did not violate the unit cap for a semester, and if all prerequisites continued to be met. We would then continue Part II until we reached a stable point, where we could no longer make a change that would improve the difficulty disparities. Below is a simplified example showing a subset of a schedule before Part II and the same subset after Part II. As we can see originally, Semester 3 has a far larger total difficulty in comparison to Semester 7. Thus our algorithm attempts to switch classes from Semester 3 to Semester 7. It will first choose the classes with the highest difficulty and attempt to switch it into the other semester (Class E goes into Semester 7). What it then realizes is that this invalidates the prerequisite requirements for another class in the schedule so this is an invalid move. Then it attempts to move the next

highest difficulty class into Semester 7 (Class D goes into Semester 7). This is maintains the feasibility of the schedule and the algorithm continues to balance other semesters.

Semester 3

Classes	Difficulty
Class A	5
Class B	7
Class C	5
Class D	10
Class E	13
Class F	3
Total	43

Semester 7

Classes	Difficulty
Class G	6
Class H	7
Class I	9
Total	22

Figure 5: Subset of simplified schedule before Part II

Semester 3

Classes	Difficulty
Class A	5
Class B	7
Class C	5
Class E	13
Class F	3
Total	33

Semester 7

Classes	Difficulty
Class G	6
Class H	7
Class I	9
Class D	10
Total	32

Figure 6: Same subset as Figure 5, now after Part II

6 Results

Our completed algorithm provided the following schedule, with the sum of difficulties followed by the total units for each semester in the parentheses:

Freshman Fall (32, 35)
Calc I
Concepts
EUREKA
C@CM
Phage Genomics Research I

Freshman Spring (46, 52)
Interp
15-110
Calc II
21-241
Physics I for Sci Students
ENGAGE (Service)

Sophomore Fall (35, 39)
Discrete Math
21-268
Principles of Micro
Evolution
ENGAGE (Arts)

Sophomore Spring (44, 52)
Managing Across Cultures
Intro to ODEs
OR I
Intro to Accounting
Global Histories
Phage Genomics Research II
ENGAGE (Wellness I)

Junior Fall (38, 46)
Numerical Methods
36-225
Principles of Macro
Math Models for Consulting
World Music
ENGAGE (Wellness II)

Junior Spring (46, 51)
36-226
36-410
Operations Management
Intermediate Micro
Intro to Gender Studies
PROPEL

Senior Fall (36, 46)
OR II
36-401
ENGAGE (Wellness III)
Supply Chain Management
Survey of Western Music History
Stars, Galaxies, and the Universe

Senior Spring (44, 45)
36-402
Intro to Math Finance
Algebraic Structures
Intro to Philosophy
Engineering Stat and Quality Control

This new suggested schedule is very different from the suggested schedule on the CMU website. Regarding the total units distributed over the 8 different semesters, the product of the algorithm has distributed them more on the Spring semesters while the original suggested schedule has it evenly distributed at around 45 units every semester. What this leads to is that

some Spring semesters of the new schedule will have a slightly higher difficulty rating but every Fall semester of the new schedule will always be easier than originally suggested. Additionally, on the CMU website the suggested schedule does not take into account the additional courses required by CMU like ENGAGE or PROPEL. Overall, this new suggested schedule provides an answer as to what classes provide the least workload. As an added benefit, it also has a lower difficulty if we were to compare the ordering of the classes suggested by the school to our completed schedule.

7 Potential Errors

Because of the assumptions that we've made and how we interpreted the problem there might be errors associated with our approach. Specifically, the schedule that we have generated might not be the best because information on the course catalog websites are not clear and not well documented. For instance, not every classes listed on the course catalog has the correct course prerequisites. Additionally, we know that it might be possible to get a better schedule by discussing with an advisor to have some classes double count or have classes meet different requirements than listed. Also, because we ignored the idea of co-requisites there might exist a whole set of schedules that have the same minimum difficulty. Finally, our current solution to the problem should be considered more of a local minimum as we start with a feasible schedule and try to develop a better schedule while staying within the constraints of the problem.

8 Further Improvements

If there was more time to work on the project it might have been of interest to add several other capabilities into our algorithm. For example, extending the application of our algorithm to other majors not only within CMU but also across different schools. It might also be of interest for some students to create a suggested schedule based on other metrics instead of doing amount

of time spent on coursework. For example, a student might want to get a schedule with the highest overall teaching score, which is another metric given by the FCE's. Additionally, having the ability for students to interact with the program through a GUI to select courses they have already taken in high school or non-major classes they want to add into their schedule. It would also be very important to communicate with the school officials so they can update their websites to have better documentation on their classes. Finally, it would be worthwhile to develop a method that would determine the absolute minimum schedule according to difficulty instead of our current approach.

9 Conclusions

Altogether, the final schedule this method produced is one that would offer one of the least, if not the least, stressful paths offered at CMU. If we were to able to resolve some of our assumptions it would be possible to find a more realistic version of a feasible and minimum difficulty based schedule. With some adjustments, the code used for this problem could be used to find similar schedules for other majors at possibly any college. Further focusing of the problem would result in an explicitly optimal schedule, which then in turn could be used for more broad applications.

10 References

http://coursecatalog.web.cmu.edu/melloncollegeofscience/#generaleducationrequirementstext

http://coursecatalog.web.cmu.edu/melloncollegeofscience/departmentofmathematicalsciences/#curriculatext

https://wwws2.smartevals.com/reporting/SurveyResults.aspx?srfall=Y&MenuItemID=148