Application of linear programming: An algorithm to optimize course selection

Fiona Feng, Zizhen Guo

November 2021

1 Introduction

A problem that every UBC student will encounter at the beginning of semester is how to choose courses. Research has found that a well-fitted course schedule can benefit students' academic performance (Dills & Hernández-Julián, 2008). This raised our interest at arranging a most suitable class schedule within various types of limits. Because there are often many restrictions on course selection, for example, a maximum of five courses can be selected in a semester, conflicts in the timings and pre-requisites. Meanwhile, students also hope that they can choose some courses they are interested in with the compulsory courses required by the major. As a result, there is an application of linear programming, how to get the optimal solution for course selection and course cheduling from hundreds of courses.

The aim of the project is to find the optimal solution for each student's course selection by setting all the limitations as constraints. For different majors and interests, as well as different personal life schedules, every student

will have different requirements for course selection. Then, linear programming can directly help students draw up a reasonable course.

2 Method

The methodologies of this study aims to effectively perform an linear programming algorithm that actively selects a combination of courses that gives the highest expected outcome from the "pool" of pre-set courses. We will utilize computational tools (packages in Python and R) to explore course scheduling problem. Our methods should ensure reproducibility to maintain reliable performance across different contexts.

Our main focus for this project is to construct the LP optimization problem that models UBC students' course selection with `pulp` package and other helpful contexts that drive our program closer to the reality. Meantime, we also make use of the knowledge pieces in MATH 340 as the framework to structure the mathematical model based on linear programming.

2.1 Dependency

The study makes use of the linear programming package `pulp` and data science package `pandas` from Python to explore the course scheduling problem. R is used for the data cleaning and wrangling library `tidyverse` and the visualizing component `ggplot2`. R is bridged to a Python Jupyter Notebook with package `rpy2`.

2.2 Data and resource

For this project, the priminary resource includes the Github repository ubc-pair-grade-data, course details in UBC's registration system and program

requirements in UBC academic calendar.

The Github repository shows overall average, standard deviation, upper bound, lower bound and number of students in each grade interval for most past UBC courses. Since it is not an official repository, the accuracy and quality of data cannot be guarenteed.

UBC's registration system and academic calendar provides a comprehensive description of course requisites and program requirements. Since UBC has not yet provided any interface for the access of course resource, without web scrapping tools that could potentially bring legal issues, information must be humanly readed and recorded. Therefore, due to the time limits and cost-effectiveness, the study only operates a subset of all courses and program requirements.

2.3 A mathematical model

Over the population of all courses in UBC, we take mathematics (MATH) as a representative for convenience. We purposefully selects popular 200-level, 300-level, and 400-level courses to form a linear program. Fundamental first-year courses are excluded since they are fixed requirements and insignificant to the choice of future schedules. For the same reason, we neglect pre-requisites with 100-level courses.

And the program requirement is 4 course for 200-level and 6 course for 300 and 400 level.

There are many components that make up the LP model for course selection. In this study specifically, three substantial factors are considered. The very first to encounter is the average score of one course. The mean grade also stands for the expected outcome for one course. Assuming the grade is

a random variable X, we have:

$$E(X) = \bar{X}$$

Maximizing the expected grades for the timetable is the dream for every college student. In reality, each student chooses a collection of courses from a larger collection of available courses, and one needs to decide whether to take one course or another. A binary variable x fits to this scenario with the former equality:

$$E(X)x = \bar{X}x$$

$$x = \begin{cases} 1 & \text{taking the course} \\ 0 & \text{otherwise} \end{cases}$$

The case where $x = 1$ can be intepreted as one course being planned, and the expected final grades for the students is just the average. The case $x = 0$ implies that the course is never taken, so the students do not have an expected grade. Moreover, the binary variable x constitutes the constraints of the linear programming algorithm. For a manditory requirement to a program, the constraint is always $x = 1$. Conversely, an unaccessible graduate course has the constraint $x = 0$ for a undergraduate student.

In reality, students choose a collection of courses from a larger collection. Our algorithm aims to locate the optimality under restrictions. Therefore, the objective function is defined by:

$$\text{maximize} \quad \sum_i (\bar{X}_i)x_i$$

Pre-requisites courses for more advanced courses is the next element to handle. In short, pre-requisites are courses that must be taken before registering for one course. Co-requisites are extension of pre-requisites that can be taken the same term as the course being registered. However, distributing courses among terms is outside the boundary of our scheduling problem.

Table 1: Possible outcomes of constraints

	Register course	Not register course
At least one pre-requisite	$x' = 1, P \geq 1, x' - P \leq 0$	$x' = 0, P \geq 1, x' - P < 0$
No pre-requisite	$x' = 1, P = 0, x' - P = 1$	$x' = 0, P = 0, x' - P = 0$

Therefore, we decided to make both concepts indifferent. We realized the importance of pre-requisitions in restraining students taking courses for their own will. Hence, pre-requisites show up as constraints in the LP model, which fits perfectly with the binary variables defined previously. Let x' be the course for registration, we have $P = \sum_k x_k$ representing the sum of every binary variable x_i from the pre-requisites. If the course x' requires at least one of the k courses as the pre-requisite, the constraint can be formalized as:

$$\text{constraint:} \quad x' - P \leq 0 \iff x' - \sum_{i=1}^{k} x_i \leq 0$$

The exhaustion of possibilities for constraints with respect to course pre-requisites on Table 1 indicates that if the algorithm attemps to register a course without pre-requisites, a resulting dictionary will lie outside the feasible region of the LP problem. This behavior ensures the validity of our scheduling program.

However, a single constraint only solves the "or" problem taking one class in a list of pre-requisites. There are "and" problems that require two or more conditions to be satisfied, and we simply multiply the target variable by the amount of "and" requirements. For example, if the target course x' requires one course from a list of pre-requisites represented by P_1, and another course from P_2, the constraints can be formalized as:

$$\text{constraint:} \quad 2x' - P_1 - P_2 \leq 0$$

Program requirements often limit a small range of courses allowing student

to choose from. These restrictions are inflexible and substantial for one student's graduation. Different from the pre-requisite concerning the course taken or not previously, program requirements check if the accumulate credits meet a certain requirement. Define $x_{1...n}$ be the binary variable of n courses restrainted by one program requirement, $t_{1...n}$ be the corresponding credits for each course, and t' be the credit requirement, the constraint can be formulazed as:

$$\text{constraint:} \quad \sum_{i=1}^{n} c_i x_i \geq t'$$

In UBC's context, every MATH course has 3 credits. So the constriant can be simplified as:

$$\text{constraint:} \quad 3\sum_{i=1}^{n} x_i \geq t'$$

For this study, we manually sampled 17 representative MATH courses from ubc-pair-grade-data of the 2020 winter term. Table 2 indicates the details for the sample. The dot plots below demonstrates the distribution of average grade in our sample with respect to the population of all MATH courses in the dataset.

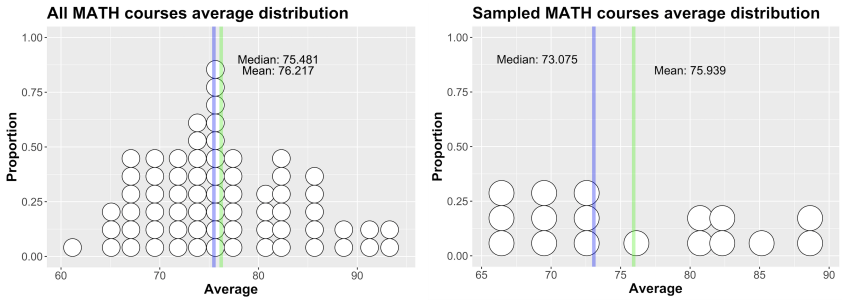

Table 2: An overview of the MATH course sample

Binary variable	Code	Name	Pre-requisites	Average	Result
1	MATH 200	Calculus III		70.178	1
2	MATH 215	Elementary Differential Equations I	One of MATH 221, MATH 223 and one of MATH 200, MATH226	76.136	1
3	MATH 220	Mathematical Proof		67.198	1
4	MATH 221	Matrix Algebra		72.075	0
5	MATH 223	Linear Algebra		88.258	1
6	MATH 226	Advanced Calculus I	One of MATH 221, MATH 223	89.000	0
7	MATH 302	Introduction to Probability	MATH 200	79.833	0
8	MATH 303	Introduction to Stochastic Processes	MATH 302	69.628	0
9	MATH 307	Applied Linear Algebra	MATH 200 and one of MATH 221, MATH 223	72.029	0
10	MATH 320	Real Variables I	One of MATH 226, MATH 200 and MATH 220	68.817	1
11	MATH 321	Real Variables II	MATH 320	72.583	1
12	MATH 322	Introduction to Group Theory	One of MATH 221, MATH 223 and MATH 220	65.595	1
13	MATH 340	Introduction to Linear Programming	One of MATH 221, MATH 223	67.267	0
14	MATH 418	Probability	MATH 321	81.600	1
15	MATH 426	Introduction to Topology	MATH 321 and MATH 322	82.000	1
16	MATH 427	Topics in Topology	MATH 426	85.143	1
17	MATH 441	Mathematical Modelling: Discrete Optimization Problems	MATH 340	82.625	0

We consider two program requirements related to MATH courses for Combined Major (1135): Mathematics and Economics stated in UBC science's academic calendar:

1. MATH 200 (or 226), 215, 220, 221 (or 223) — 12 credits. This requirement needs to be re-structured to sub-constraints to fit in our model:

   (1.1) One of MATH 200 or 226

   (1.2) One of MATH 221 or 223

   (1.3) At least 12 credits among the chosen courses

2. MATH courses numbered 300 or higher — 18 credits

Therefore, we set x_i as the binary decision variable for $i = 1, 2, \ldots, 17$ with one-to-one correspondence to the MATH courses in Table 2. For $\bar{X}_i$ being the average score of the i^{th} course, we can thereby construct a LP problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{17} \bar{X}_i x_i \\ \text{subject to (program requirement 1)} \quad & x_1 + x_3 \geq 1 \\ & x_4 + x_5 \geq 1 \\ & 3\sum_{i=1}^{6} x_i \geq 12 \\ \text{(program requirement 2)} \quad & 3\sum_{i=7}^{17} x_i \geq 18 \end{aligned}$$

$$
\begin{aligned}
&\text{cont.} \\
\text{(pre-requisites)} \quad & x_2 - x_4 \leq 0 \\
& x_7 - x_1 \leq 0 \\
& x_8 - x_7 \leq 0 \\
& 2x_9 - x_1 - x_4 \leq 0 \\
& 2x_{10} - x_3 - x_6 \leq 0 \\
& x_{11} - x_{10} \leq 0 \\
& 2x_{12} - x_3 - x_5 \leq 0 \\
& x_{13} - x_4 \leq 0 \\
& x_{14} - x_{11} \leq 0 \\
& 2x_{15} - x_{11} - x_{12} \leq 0 \\
& x_{16} - x_{15} \leq 0 \\
& x_{17} - x_{13} \leq 0 \\
& x = 1 \text{ or } 0
\end{aligned}
$$

For constraints above, we start with the requirements for the program. And the following represents pre-requisites for all of course within the sample. By solving this linear program, we can get the optimal course selection.

2.4 Design

An object-oriented design pattern is introduced for the sake of cohesion and abstraction to our problem. We designed the Python object `Course` with class-level variables `specialization, code, year, pre_reqs, credit` and `average` representing different properties of one course. For the sample with purely 3-credit MATH course, `Course.specialization` and `Course.credit` are not decisive factors to our algorithm. However, a broader course-selection

program with increasing complexity would require our algorithm to deal with courses with various specialization and different credits.

The pre-requisites of one course is represented by `pre_reqs` with a Python list containing multiple lists of courses. The inner list (a list of `Course`) corresponds to the requirement for at least one course from a given set, whereas the outer list (a list of list) indicates different requirements that must be satisfied simutaneously. The design pattern provides a guideline for our coding component to follow, and keeps our program self-contained and reproductive.

3 Result

By running the algorithm through `pulp` package in Python, we obtained the optimal solution regarding the program requirements. As shown in Table 2, the result suggests a schedule of MATH 200, MATH 215, MATH 220, MATH 223, MATH 320, MATH 321, MATH 322, MATH 418, MATH 426 and MATH 427. The total expected grades over 10 courses is 757.508, producing an expected optimized average score of 75.751.

Notice that our solution is a local optimality only subject to the given information set. We will conduct further discussion about the algorithm and its limitations.

4 Limitation

We chose mathematics and some of the courses as a sample to construct a linear program. The optimal course selection is obtained by considering the element like average score, program requirements and prerequisite courses.

For all other majors and courses, the optimal solution can be obtained by constructing such a linear program. But there are some limitations in the construction of this model.

First of all, we only considered objective factors including the average score, program requirements and pre-requisite courses, but ignored the subjective factors such as the choice of professors, students' interests and personal ability. Those factors will also influence the optimal solution for course selection. However, adding subjective factors to the mathematical model is difficult, hence we can only gain a limited degree of optimal solution after considering the objective factors. Also, the objective factors under our consideration is incomplete. Despite that the LP algorithm suggests a path starting from MATH 320, the fact that MATH 320 requires at least 80% for MATH 220 is neglected from this study because of the complexity of programming constraints to a percision of grade percentages.

Furthermore, the average score of courses referred by this study is from the most recent 2020W term, which could be potentially biased because of the online teaching environment due to the pandemic. With sufficient resource and time, computing out the overall average of courses throughout years could bring up the accuracy and percision of estimates. From the aspect of statistics, our choice of expected outcome for one course can be random variables other than means. In the case that mean values could be significantly influenced by outstanding high-scored students, the effectiveness of median may outweight averages provided in our dataset (Kristensen, 2014). Additionally, mean scores represent the averaged-out performance of the whole class, but might not be the best estimators for individual students. For example, with the assumption of normality, a top-class student could add-up 2 standard deviations to the expected outcome for approximately 95% higher

grades than other classmates, and vice versa for students that are struggling.

Meanwhile, another limitation is about the source. Since the dataset for average scores is not an official repository, the accuracy and reliability of data cannot be guarenteed. While UBC has not yet provided public access to the course information database with Python, applying our algorithm to courses of a wider range could be technically challenging.

5 Conclusion

In this paper, we constructed a linear programming model for course selection problem. The purpose of this research is to help students arrange an optimized course schedule with the highest expected grades. We experimented to verify the validity and effectiveness of the algorithm by operating on MATH major and a bundle of MATH courses. With Python, the program managed to suggest us the optimal solution for course selection. However, the solution is restricted by the sample, resources and computational tools we adapted throughout the research. The flaws and limitations therefore provide rooms for us to conduct further in-depth research and explorations to encounter course scheduling problem.

References

Dills, A. K., & Hernández-Julián, R. (2008, December). Course scheduling and academic performance. *Economics of Education Review*, *27*(6), 646–654. Retrieved 2021-11-29, from `https://www.sciencedirect.com/science/article/pii/S0272775707000957` doi: 10.1016/j.econedurev.2007.08.001

Kristensen, J. T. (2014, May). Factor-based forecasting in the presence of outliers: Are factors better selected and estimated by the median than by the mean? *Studies in Nonlinear Dynamics & Econometrics*, *18*(3), 309–338. Retrieved 2021-11-29, from https://search.ebscohost.com/login.aspx?direct=true&AuthType=shib&db=bsu&AN=95963430&site=ehost-live&scope=site&custid=s5672194 (Publisher: De Gruyter) doi: 10.1515/snde-2012-0049