Optimizing Course Planning for MIT Students FireRoad053

Ashhad Alam, Ashar Farooq, Hesham Nawaz, McKinley Polen
MIT

May 13th, 2021

Abstract

Previous services on creating course schedules at MIT exist. These services take into account time constraints and allow an end-user, a student in this case, to view a schedule on a calendar-like interface for a specific semester. We improve these services by creating a more modular, dynamic, and valuable service designed to integrate different end-user preferences and the power of optimality. In particular, we use the foundational structure of decision variables, objective function, and constraints, alternatively known as a model. Our model, entitled FireRoad053, improves upon other existing models for class scheduling at MIT by creating time-compatible current and future semester roadmaps, taking into account the number of hours of each class in addition to the ratings of each class. The model ensures that the student receives an optimized schedule in order to sustain their academic career at MIT, specifically for Course 6-14 in the Department of Electrical Engineering and Computer Science and Course 2 in the Department of Mechanical Engineering, in addition to a Japanese minor housed in MIT Global Languages Department. Our results suggest that some semesters may have more technical requirements while other semesters are heavier in terms of other humanities courses. In particular, classes that are needed for future classes are prioritized for as early as feasibly allowed by our model.

Contents

1 Acknowledgements

We would like to recognize Gurobi for allowing us to utilize their computationally efficient mathematical optimization solver. In addition, we would like to recognize Mr. Guido van Rossum for de facto allowing us to use the data-processing power of Python and Mr. Alan Edelman for de facto allowing us to construct and implement our model for this important issue.

Finally, we would also like to acknowledge Mr. Daniel León Jiménez, MIT, for his insights in the modeling and data-parsing elements of this project in addition to Professor James Orlin for his educational materials on optimization methods in business analytics.

2 Problem Statement

A common problem for MIT students is to determine a schedule that works decently for them. With many requirements, prerequisites, and other Institute and departmental policies, it can be hard to fit everything in such a limited amount of time. Namely, there are only so many classes one can take in order to satisfy many objectives of majors, minors, concentrations, and other intellectual programs. To this effect, the FireRoad053 solution to this problem will optimally determine their courses for the upcoming semester while also planning for all subsequent semesters. This is important as usually long-term planning of courses hinders overall efficacy and productivity of an MIT experience. The solution to this problem will be an idealized system that can be viewed as a software that integrates external third-party services Firehose with CourseRoad and has optimization features that the application FireRoad does not have.

3 Project Goals

The prevalent goal of our project was to construct a working and efficient model that can be used to solve the issue of course roadmaps for other MIT students during their academic career. This goal involved several different components. First, the model should be able to garner accurate and large enough samples of classes in order to use in the model. This meant using an efficient approach on parsing, refactoring, and managing data in order to make it compatible with our eventual model. This would need to be implemented by researching external Application Programming Interfaces (APIs) that would provide authentic and reliable data from MIT-based and/or MIT-influenced sources. Second, the successful implementation of the model would require more fine-tuning and analysis of the model specifications to be done as this will help implement an efficient solution. The authors wanted to perform further analysis on the objective function parameters and different weights to place on different parts of the objective function, such as utility from workload or ratings of different classes, etc. In addition, this subgoal should involve analysis of the best optimal tradeoff between many of these different choices as a solution that combines many different choices might be socially optimal. This would need further analysis and discussion of the values of what it means to have a good schedule and a good utility from this said schedule. Using more analysis of what an optimal schedule will look like, perhaps through mock schedules and analysis of historical and

institutional precedent, the goal would be to construct a reasonable objective function in addition to decision variables on when a student should or should not take a certain class.

Third, the model construction would also require formulation of several different constraints in order to accurately output a roadmap that is feasible for a student to follow in an optimal manner. This would require different baseline standards that would need to be defined or assumed. With these three components of decision variables, objective function, and constraints, the goal of constructing an efficient model should be realized.

Furthermore, other project goals included creating an open-source repository of the model in order to maintain and increase not only transparency, but also accountability. Understanding the perspective of the user, a student, was also a goal of the project. This is a major topic in the realm of User Interface / User Experience fields in addition to media studies, content-management systems, and modular systems.

Finally, the specific classes, majors, timings, and data structures for the mathematical model also were to be determined strategically as part of the project. This is an important methodology in default choice architecture, as referenced in the works of behavioral economist Richard Thaler. The default classes, prerequisites, timings, designations of classes, and other model specifications were to be formulated with thought.

4 Data Source and Management

This project uses all relevant classes data from the FireRoad server and corresponding Application Programming Interface (API) at this link. The read-only API is utilized in order to perform a GET request in order to acquire the open-access data. This information is extracted for all relevant classes in the project. Each class's data has information regarding the total number of units, subject ratings, lecture times, recitation/additional class component times, designation of a General Institute Requirement (GIR), designation of semester offered, pre-requisites, and some additional data and metadata. The data is from the prior year. In total we collected data for 54 classes that fulfilled our given majors and minors.

In order to get the data from the FireRoad API into a format usable in our model we used Python scripts to parse through the provided data and place it into a CSV. The first set of data we had to manipulate was the prerequisites. Each set of prerequisites was given as a string of subject ids, which we transformed into a list of 54 binary variables for each class defined as follows:

$$\text{prerequisite}_{i,j} = \begin{cases} 1, & \text{if jth class is a prerequisite for class i} \\ 0, & \text{else} \end{cases} \tag{1}$$

The next collection of data to manage was for the time slots of each class. First we simplified the schedules of each class to just have one configuration of lecture, recitation, and lab. Then from this, we parsed the strings given from FireRoad and converted it to a data structure where for each week we account for Monday to Wednesday 9am-5pm. We split that timeframe into 30 minute blocks,

defined by:

$$\text{schedule}_{i,t} = \begin{cases} 1, & \text{if class i occurs at timeslot t} \\ 0, & \text{else} \end{cases} \tag{2}$$

Therefore for each class we were left with a length 80 array of binary variables to denote each class' schedule.

The designation of a General Institute Requirement (GIR) and whether a class is offered in Spring or Fall was also transformed to binary variables for use in our model. The remaining metrics were given to our Julia model as found in the FireRoad API.

5 Model Specifications

The model was designed to meet all the criteria outlined by the project goals: Create a valid schedule for two different cases a student in course X, and a student in Course Y and minoring in Z. To meet all the requirements outlined in the project we must first define the various aspects of the courses included in our model. In particular, the first major, X is defined as Course 6-14, also known as Computer Science, Economics, and Data Science. This Course is housed in the Department of Electrical Engineering and Computer Science at MIT. Similarly, the second major, Y is defined as Course 2, known as Mechanical Engineering. The minor, Z, is defined as Japanese, which is housed within the MIT Global Languages Department. Majors X, Y and Minor Z all have a large selection of unique required courses that must be completed to earn a degree. Due to the limited nature of this project, we have taken a small subset of 54 courses that allow for both combinations of X and YZ to be completed. We will now define binary parameters for each of the requirements for a valid FireRoad053 schedule.

In this project, subject ratings are defined to be a measure of how good a class is as the value of this numerical measure will be between 1 to 7, inclusive, and which is regarded as a relatively good indicator of the utility that can be gained from any given course. It is stored in the variable $rating_i$, which gives the rating of the i-th course in our limited catalog.

General Institute Requirement (GIR) are classes mandated by MIT to be taken by all enrolled undergraduate students. These are a specific set consisting of 6 classes including two (2) calculus courses, two (2) physics courses, one (1) biology course, and one (1) chemistry course. In this project, we will explicitly define the set of GIR classes to be $\{18.01, 18.02, 8.01, 8.02, 5.111, 7.012\}$, and a class' status as a GIR is stored in the variable is_gir_i which will be 1 if the i-th course in our catalog is a GIR.

Next, we need to accommodate MIT's HASS requirement; a student's feasible roadmap must include at least 8 HASS classes. To accomplish this, we define a variable is_hass_i which is 1 if the i-th course in our catalog is a HASS. In total our 54 course list contains 18 HASS subjects with a majority being within the Japanese and 14 departments.

The total number of units are defined to be roughly similar to the total number of hours a student is expected to spend on that class in 1 week. For example, a 12-unit class is a full-time class, where

students should expect to spend approximately 12 hours on all components of the class every week. We note that Institute requirements demand that a student take at least 216 units beyond the GIR classes that have been defined earlier on. To account for both the units per class and estimated time, we define units_i which denotes the number of units for the ith class in our catalog, and hours_i which denotes the number of hours for the i-th class.

For a class, x, a prerequisite, y, is defined as another class that must be taken before taking class x. There are many pre-requisites for different classes in the model, which are obtained from the FireRoad API. We will use the variable defined above, $\text{prerequisite}_{i,j}$, to calculate the prerequisites for a given class.

In our model, a conflict can only exist, and occur, in the Spring 2021 semester. We define classes to be conflicting if 2 or more classes have a lecture, lab or recitation scheduled for the same half-hour block. Since we want to avoid conflicts in our schedule, no two classes in our final schedule for Spring 2021 can overlap at the same time. We will use the variable $schedule_{i,t}$ defined in the data section to account for this overlap.

As was stated in the project description, student preferences need to be accounted for within our model. As such, FireRoad053 presents it's users with the option to give their preferences in three different sections: number of hours in schedule, busy time slots, and preference for taking a certain class in a certain semester. Students can enter their own desired maximum number of hours they want to work per week in any of the 8 given semesters at MIT.

The second student preference to be their own judgement call on what they consider to be least-desired time slots for any class in a respective semester. Say, for instance, a student likes to have lunch at noon. With FireRoad053, they can list 12 PM as an undesired timeslot, and the model will make sure to accommodate for this, if at all possible.

The third and final input they can make into the model is to give a preference for taking a certain class in a certain semester, meaning that our model will constrain their schedule to guarantee that class is taken during their desired semester at MIT.

Specifically, our model is as follows:

Objective Function:

Maximizes student utility based on: Average Rating for all subjects taken (Defaults to Subject Evaluations but user can enter their own utility for each subject)

$$\max \sum_{i=1}^{54} \sum_{j=1}^{80} x_{i,j} * \text{rating}_i \tag{3}$$

Decision Variables:

In our case, we define a 54 by 8 array of binary decision variables for the 54 subjects offered in our limited FireRoad053 and the 8 semesters a typical student has at MIT:

$$X_{i,j} = \begin{cases} 1, & \text{if class } i \text{ is taken during semester } j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Constraints:

Each student takes (or gets credit for) all of the 6 Science/Math GIRs, where is_gir_i is 1 if the i-th course is a GIR.

$$\sum_{i=1}^{54} \sum_{j=1}^{8} x_{i,j} * \text{is\_gir}_i \geq 6 \tag{5}$$

Each student takes 8 HASS subjects, where is_hass_i is 1 if the i-th course is a HASS.

$$\sum_{i=1}^{54} \sum_{j=1}^{8} x_{i,j} * \text{is\_hass}_i \geq 8 \tag{6}$$

Each students satisfies the requirements of their major or minor.

For each required course:

$$\sum_{j=1}^{8} x_{i,j} \geq 1 \tag{7}$$

Which ensures that the i-th course is taken at least once during the student's 8 semesters at MIT. We modeled each required course as its own constraint with the i-th course being specific to courses 2, 6-14, and Japanese.

No subject is taken prior to or at the same time as its prerequisites. From our prerequisites data structure defined above, where $prerequisite_{i,j}$ is 1 if the j-th course is a prerequisite to the i-th course we define the constraint as follows:

$$x_{i,j} \leq \frac{\sum_{l=1}^{54} \sum_{m=1}^{j-1} x_{l,m} * \text{prerequisites}_{i,l}}{\sum_{l=1}^{54} \text{prerequisites}_{i,l}}, \forall i \in \{1, 2...54\}, \forall j \in \{1, 2...8\} \tag{8}$$

Since the right-hand side is only 1 when all pre-requisites are satisfied, this constraint ensures that we will not take a class before we satisfy all of it's pre-requisites.

Each student takes 216 units beyond GIRs. To do this, we sum over all $x(s,t) * units$ for subject s, restricting on s not being a GIR, and make sure that the sum is ≥ 216. We will use our previously defined parameter, is_gir to accomplish this.

$$\sum_{i=1}^{54} \sum_{j=1}^{8} x_{i,j} * (1 - \text{is\_gir}_i) * \text{units}_i \geq 216 \tag{9}$$

No two subjects taken in Spring 2021 should overlap in time (future schedules conflicts can be done). Here c is defined to be the student's semester number, for Spring 2021. As defined above, $\text{schedule}_{i,t}$ is 1 if the i-th class occurs during timeslot t.

$$\sum_{i=1}^{54} x_{i,c} * \text{schedule}_{i,t} \leq 1, \forall t \in \{1, 2...80\} \tag{10}$$

Students cannot take the same subject twice:

$$\sum_{j=1}^{8} x_{i,j} \leq 1, \forall i \in \{1, 2...54\} \tag{11}$$

Students can only take the same subjects offered in that current semester(Spring/Fall). This constraint uses data from FireRoad, stored in two variables offered_fall_i and offered_spring_i, where the i-th element is 1 if that course is offered in that respective term.

We must check for Spring offerings during even terms and Fall offerings during odd terms:

$$x_{i,j} \leq \text{offered\_spring}_i, \forall i \in 1, 2...54, \forall j \in 2, 4, 6, 8 \tag{12}$$

$$x_{i,j} \leq \text{offered\_fall}_i, \forall i \in 1, 2...54, \forall j \in 1, 3, 5, 7 \tag{13}$$

6 Assumptions

There are numerous assumptions that we are utilizing in this project. In order for the computational complexity of the model to be reasonable, we will assume that the default Courses that the model is valid for are Course 6-14 (Computer Science, Economics, and Data Science) and Course 2 (Mechanical Engineering). The model will be accurate for any authentic majors and minors if that is a valid input to the model, just not all majors and minors are assumed to be in this particular optimization model. We will assume that FireRoad053 model does not integrate other more complicated aspects of the MIT curriculum, such as CI-H/HW subjects — Communication Intensive in the Humanities, Arts, and Social Sciences, Communication Requirement, First-Year Essay Evaluation (FEE), Physical Education & Wellness (PE&W) Requirement, Swim Test, three HASS Distribution subjects of HASS - A (Arts), HASS - H (Humanities), HASS - S (Social Sciences), subjects taken during the Independent Activities Period (IAP), subjects taken during the summer term, credits from the Undergraduate Research Opportunities Program (UROP), Restricted Electives in Science and Technology (REST) Requirement, and any laboratory requirements.

Another important assumption we make is to ignore co-requisites in our model. These are defined to be classes that can be taken at the same time or earlier than the class we are considering, although their value has been historically volatile, thus for simplicity and the natural flow of classes, co-requisites are not integrated into FireRoad053.

In addition, our model does not allow pre-requisites to be ignored based on petitions or on prior instructor approval. Petitioning is a very human-driven process that involves applications to departmental staff in addition to other instructors. This is a mechanism that cannot be reasonably optimized as decisions are based on red-tape bureaucracy measures and very student-specific, which can be hard for a model designed for producing general optimized roadmaps. This is a similar reasoning for gaining prior instructor approval in order to satisfy a requirement, which is not always generalizable.

We also assume that students generally want to maximize their happiness, thus they want to maximize the ratings of the classes, as that is assumed to be correlated with a better overall class experience.

Lastly, we assume that students generally want to minimize their number of hours worked as that is correlated with overall better health, and a better social life.

7 Results and Accomplishments

We were able to create a functional model that follows the specifications that are laid out in the project instructions. In effect, we have replicated a course scheduling service while also optimizing for certain objective values. What this means is that for the majors and minors that we have specified, we are able to generate a valid course plan for them to meet their requirements while also taking into account their individual preferences across certain factors.

We were able to also process and parse relevant accurate data from an external data source. In

particular, we were able to create new data structures that were efficient at storing and manipulating our relevant class data, etc. that was then utilized in the model itself.

We were also able to make the model reasonable by conducting various trials on mock schedules as inputs to the model, even basing it off the real-life scenario of the author's schedules themselves. These tests and analysis proved to be valuable validation of the model.

Furthermore, we were able to create an open-source repository to maintain the impressive codebase, fulfilling many functions, that can be found at this link in addition to the user interface open-source repository located at this link.
With this user interface and analysis on what a user wants, we were able to take into the perspective of the user in order to have a reasonable experience for them when they are utilizing FireRoad053. An explanation of the current and potential future user interface can be found at this YouTube link.

Ultimately, we were also able to acquire optimized roadmaps for different individuals with different user parameters as to be followed in this section.

7.1 Course 6-14 major

7.1.1 Freshman:

First, we ran our algorithm on a freshman, planning to major in $6-14$ who took the following classes:

- **Freshman Fall:** 6.00, 14.01, 18.01, 8.01

Figure 1 below shows the FireRoad our code generates for this freshman.

Figure 1: FireRoad for the a 6-14 Freshman

7.1.2 Sophomore:

Next, we ran our algorithm on a sophomore, planning to major in $6-14$ who took the following classes:

- **Freshman Fall:** 6.00, 14.01, 18.01, 8.01
- **Freshman Spring:** 6.042, 18.02, 5.111
- **Sophomore Fall:** 21G.503, 6.009, 14.30

Figure 2 below shows the FireRoad our code generates for this sophomore.

Figure 2: FireRoad for a 6-14 Sophomore

7.1.3 Junior:

Here are the results for running it on a 6-14 junior who took the following classes:

- **Freshman Fall:** 6.00, 14.01, 18.01, 8.01
- **Freshman Spring:** 6.042, 18.02, 5.111
- **Sophomore Fall:** 21G.503, 6.009, 14.30
- **Sophomore Spring:** 21G.504, 18.06, 21H.354, 14.32
- **Junior Fall:** 21G.593, 21G.505, 14.41, 14.19

Figure 3 on the next page shows the FireRoad our code generates for this junior.

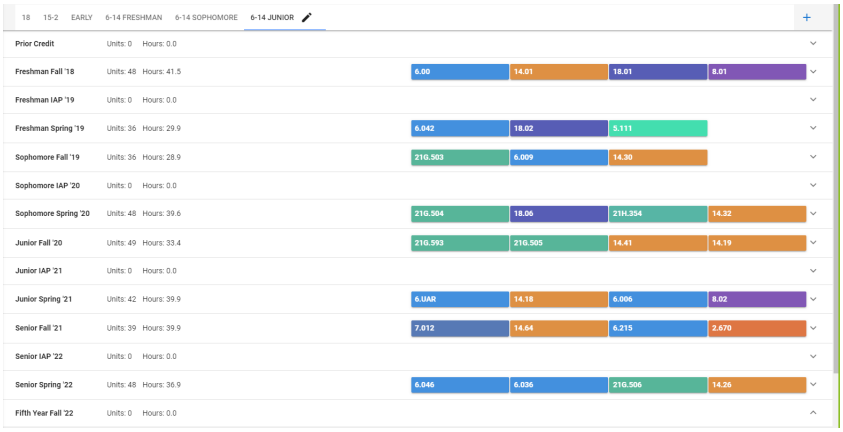


Figure 3: FireRoad for a 6-14 Junior

7.2 Course 2 Major with a Japanese Minor

7.2.1 Freshman:

We ran the same algorithm on a freshman aiming for a course 2 major with a Japanese minor, who took the following classes:

- **Freshman Fall:** 21G.503, 18.01, 8.01

Figure 4 below shows the FireRoad our code generates for this freshman.

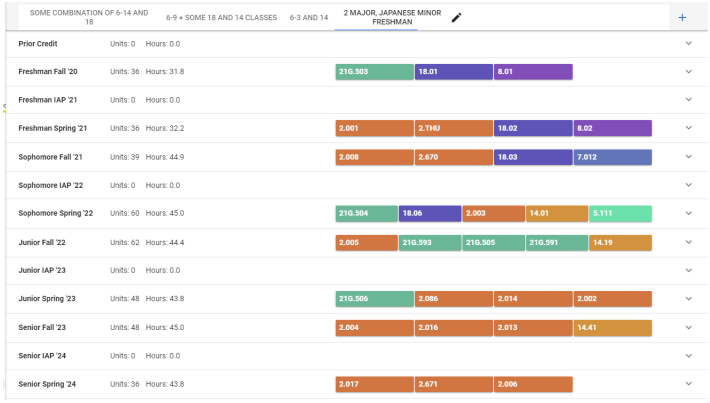


Figure 4: FireRoad for the a 2 Major, Japanese Minor Freshman

7.2.2 Sophomore:

Next, we ran our algorithm on a sophomore, planning to major in Course 2 with a Japanese minor who took the following classes:

- **Freshman Fall:** 21G.503, 18.01, 8.01
- **Freshman Spring:** 2.001, 2.THU, 8.02, 18.02
- **Sophomore Fall:** 2.008, 2.670, 18.06, 5.111

Figure 5 below shows the FireRoad our code generates for this sophomore.

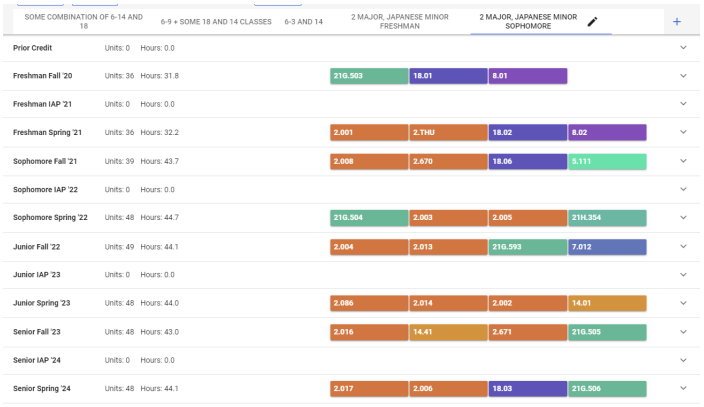


Figure 5: FireRoad for a 2 Major, Japanese Minor Sophomore

7.2.3 Junior:

Here are the results for running it on a 2 major, Japanese minor junior who took the following classes:

- **Freshman Fall:** 21G.503, 18.01, 8.01
- **Freshman Spring:** 2.001, 2.THU, 8.02, 18.02
- **Sophomore Fall:** 2.008, 2.670, 18.06, 5.111
- **Sophomore Spring:** 21G.504, 2.003, 2.005, 21H.354
- **Junior Fall:** 2.004, 2.013, 21G.593, 7.012

Figure 6 one the next page shows the FireRoad our code generates for this junior.

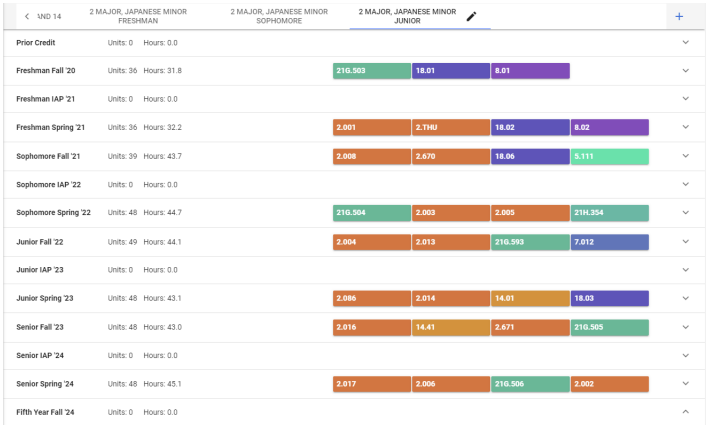


Figure 6: FireRoad for a 2 major, Japanese minor Junior

8 Additional Analysis

8.1 Variations in models

We were able to consider different variations in models in order to select the most reasonable, complete yet sustainable, and efficient version to be utilized for this project. We considered analysis on what other variations of constraints and objective functions we can use. For instance, we discussed the tradeoffs of adding new constraints of the co-requisites, etc. We also formulated different variations of what constitutes utility instead of just the average class rating. Other measures of utility could come from taking classes with certain instructors as their teaching methodologies might be better suited for certain students. In addition, utility can also come from taking classes with other people that you know or the different structure of the course. Many of these variations can likely result in a better optimized roadmap, but is likely not computationally efficient or feasible to model. Optimization models in general are almost always likely to be imprecise in modeling all of the different intricacies of daily life; however, these were considered and analysed by us in order to choose a good balance between simplicity and reality.

One variation that we implemented was by changing the objective function of our model. In our variation we used a linear weighting between two objectives: Maximizing utility(either by average rating or inputted per class by user) and minimizing the average hours per semester. By allowing the user to input a lambda value with lambda = 1 meaning all weighting being on minimizing average hours per semester and lambda = 0 meaning all weighting on maximizing utility per year. The objective function was formulated to maximize utility and the negative of hours. Initially, when minimizing just total hours summed over all semesters, we found that a for lambda values of ≤ 0.85, the model could not discern any difference between total hours vs total utility. At all values of lambda $\leq .85$ it returned the same configuration of classes and with the same objective value, and only for lambda values greater than this did it begin to give a weighting to . To mitigate this issue we decided to minimize the average hours per semester, and maximize utility per year. This made it so that the importance of 1 utility point vs 1 extra hour was more evenly weighted in the model. Using this formulation, we found that hours and utility would change as lambda changed over the range (0,1).

lambda	Average Hours	Total Utility
0	44.265	215.28
0.1	44.265	215.28
0.2	44.265	215.28
0.3	44.265	215.28
0.4	43.965	214.51
0.5	42.765	209.78
0.6	34.85375	173.14
0.7	29.475	138.05
0.8	29.475	138.05
0.9	29.4375	137.39
1	29.4375	137.39

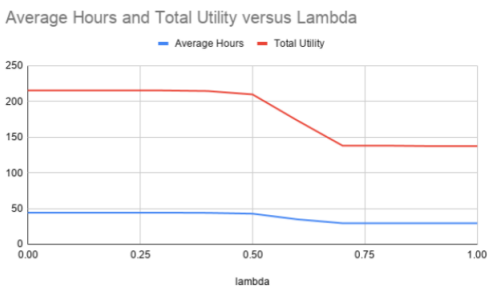


Figure 7: Results of our model variation with varying lambda

As seen in the figure above of a 6-14 freshman with a 45 hour limit per semester, varying lambda in our modified model causes it to optimize more towards either utility or minimum hours respectively. Here you can see that even in the case that it optimizes towards strictly maximizing utility per year, our model is still relatively constrained by the student preference for maximum hours per semester(which is 45 in this case) to still complete their major. We found that if the maximum hours cap was increased, the number of hours would increase dramatically for lambda values near 0, and would be the result of a student taking virtually every class in our catalog.

Another interesting trend we saw was that when maximizing utility was weighted more than minimizing hours, the number of HASS' tended to be higher. In the above case, for lambda near 0, the student takes 15 of our catalog's 16 HASS classes, and then for lambda near 1 the student only takes 9. This introduces an interesting bit of information that HASS classes tend to have a high rating, but perhaps a lower utility to hours ratio than different classes in our catalog.

8.2 Sensitivity Analysis

We were able to perform a sensitivity analysis on the Right-Hand Side(RHS) of a constraint. Namely, we were able to conduct analysis on the maximum hours per semester that a student prefers. The results strongly suggest that increasing the maximum number of hours per semester by one increases the objective value of the model. The results can be summarized below:

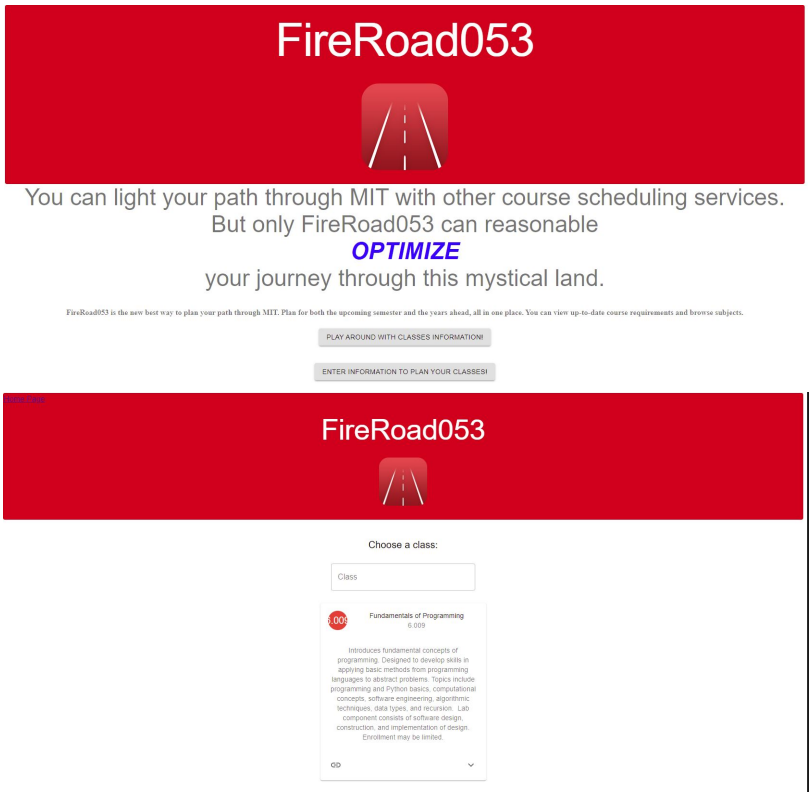
RHS Value	Objective Function Value
45	215.28
46	218.39
47	221.78
48	226.03
49	228.37
50	232.57
51	235.92
52	238.37
53	242.12
54	243.81
55	248.5200128

9 User Interface

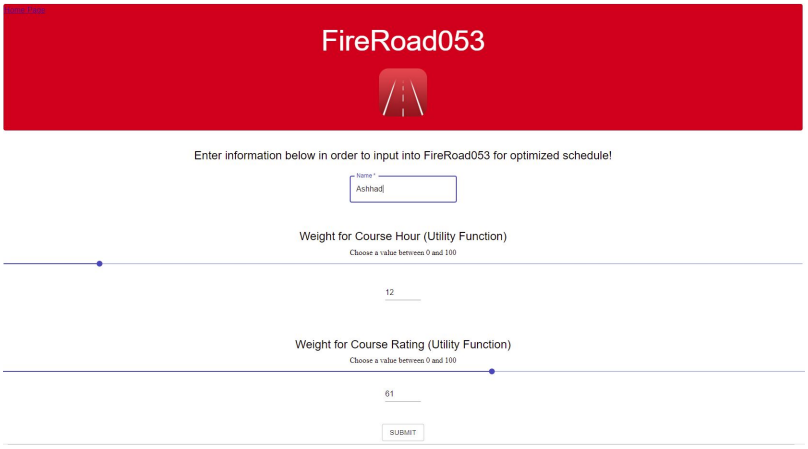
In order to create a reasonable way for users to enter various information, such as their classes taken, maximum hours they prefer per semester, and potentially other parameters, we have created simple variables in the Julia notebook that can be changed by the user. These are very dynamic as FireRoad053 can work with various inputs depending on what year a student is and other such adaptable attributes.

In addition, we have created a sample web application linked here, whose source code can be found at this link that can potentially serve as the complete front-end to a real service that can be used by many individuals. This tool was created with many packages, JavaScript, and the React framework. It allows different functionalities, such as the fact that the user can play around and see the data for various classes in a much nicer way. In addition, the input section of the application can also serve as the mechanism for collecting user information in the future once it is connected to a relational database like Cloud Firestore, etc. The output of the model can also be viewed in a calendar-like seamless format on the web application in the future as well. An explanation of the current and potential future user interface can be found at this link.

Here are some screenshots on what the Interface looks like:


FireRoad053
You can light your path through MIT with other course scheduling services.
But only FireRoad053 can reasonable
OPTIMIZE
your journey through this mystical land.
FireRoad053 is the new best way to plan your path through MIT. Plan for both the upcoming semester and the years ahead, all in one place. You can view up-to-date course requirements and browse subjects.
PLAY AROUND WITH CLASSES INFORMATION!
ENTER INFORMATION TO PLAN YOUR CLASSES!

FireRoad053
Choose a class:
Class
6.009
Fundamentals of Programming
6.009
Introduces fundamental concepts of programming. Designed to develop skills in applying basic methods from programming languages to abstract problems. Topics include programming and Python basics, computational concepts, software engineering, algorithmic techniques, data types, and recursion. Lab component consists of software design, construction, and implementation of design. Enrollment may be limited.


FireRoad053
Enter information below in order to input into FireRoad053 for optimized schedule!
Name *
Ashhad
Weight for Course Hour (Utility Function)
Choose a value between 0 and 100
12
Weight for Course Rating (Utility Function)
Choose a value between 0 and 100
61
SUBMIT

10 Limitations and Future Improvements

We made several simplifications to make the problem of finding an optimal schedule for students feasible. Given the wide range of majors, minors and concentrations at MIT, it would've been practically impossible for us to develop a comprehensive course planner within the timespan given. As a result, we focused specifically on just the two majors and the one minor mentioned above. We also further simplified these by removing some of the electives in the major since otherwise it would be very difficult to model the major requirements. We also had to simplify the prerequisites for classes because some classes have very idiosyncratic sets of prerequisites with multiple OR and AND conditions that are complicated to model. Lastly, we also simplified the timings of classes significantly, assuming that each class only has a single lecture session and a single recitation session that occur at the same times respectively in the week. Thus, our model is restricted by the above limitations and is still some way away from being a full FireRoad.

In terms of future improvements, we would like to address the limitations discussed in the previous paragraph and create a comprehensive model that accounts for all possibilities. We would also like to fully develop the user interface and connect it with our data sources in the backend so that it is actually usable. Lastly, another potential improvement could be more sophisticated ways to model student preferences with respect to classes, and perhaps develop an algorithm for identifying which classes are most suitable for which students based on their characteristics, which classes they've taken/enjoyed so far and what their goals for the future are. Perhaps machine learning could have some utility here.

11 Appendix

11.1 Data Preparation

This link contains the fireroading_python_data.ipynb file, which contains the Python code that extracts the data from the FireRoad API at this link and parses it appropriately to be used in the Julia model.

11.2 Julia Model

This link contains 1) fireroading_julia_lambda.ipynb file and 2)fireroading_julia_max_util.ipynb file. The first file contains the code for a variation of our original model. The second file contains the code for our main original model, utilizing Julia, JuMP and the Gurobi optimizer in order to implement the decision variables, constraints, and the objective function.