# Quelling Class Conflicts: Course Schedule Optimization

Rhys O'Higgins and Lola Vescovo

March 12, 2023

## 1 Introduction

The stress of course registration is an established factor in the life of every Macalester student. Deciding what courses to take, when to take them, and how to time it all so that you can graduate on time is difficult. This struggle is compounded when temporal overlaps between courses rule out potentially ideal schedules, forcing students to delay taking certain courses, or simply hope the courses they need aren't scheduled at the same time.

Our project aims to improve the way courses are scheduled at Macalester through the use of linear programming. We want to minimize the number and severity of time conflicts between classes while also taking into account what and when professors are willing to teach. We limit our focus to the Macalester Math department, and make a handful of other simplifying choices, to arrive at course schedules that are nearly devoid of conflicts, while meeting most of the other requirements necessary to be a valid course schedule.

## 2 Background and Methods

At Macalester College, courses are taught in a number of pre-defined time slots.[1] Multiple sections of a course may be taught in a given semester, and some courses may require weekly labs in addition to course meeting time. Each course section is assigned to one professor (and occasionally one or more student preceptors), one classroom, and one time slot. Courses are numbered from one hundred to four hundred based on difficulty level.

For simplicity, we consider only math courses (which do not have labs), exclude cross-listed courses[2], and do not consider assignment to preceptor nor classroom.

Much of the existing literature on the topic either dealt with academic settings completely foreign to Macalester, made assumptions incompatible with ours, or focused on aspects of course scheduling (like classroom assignment) that didn't interest us. As such, our work, while loosely inspired by previous scholarship in the area, does not directly engage nor build upon other papers/scheduling programs.

Before beginning work on the problem, we conducted an electronic survey of all Mathematics professors to better understand the constraints a good schedule would meet. In addition to providing feedback about what properties a schedule should have, professors also designated which courses they could teach, and in what length time slot they would do so. They then designated when they would be willing to teach, allowing us construct all possible valid combinations of professor, course, and time slot.

We also had to decide how to weight class conflicts prior to constructing the program itself. This was the most subjective part of the project; we drew on our experience as students to decide which conflicts were "worse" than others, and thus more important to avoid. We decided that the most efficient way to weight course conflicts was by the level (100 to 400) of the conflicting courses. We gave two 100 level courses a conflict of 0, because the only two options for those courses are Calculus I and II, which a student cannot take simultaneously. Similarly, it is very unlikely that a student would be taking a lower level class (a 100 or 200)

[1] There are 7 one hour time slots on Mondays, Wednesdays, Fridays: 8:30-9:30, 9:40-10:40, 10:50-11:50, 12pm-1pm, 1:10-2:10, 2:20-3:20, and 3:30-4:40. There is one 1.5 hour time slot on Mondays and Wednesdays: 8am-9:30 and four 1.5 hour time slots on Tuesdays and Thursdays: 8am-9:30, 9:40-11:10, 1:20-2:50 and 3pm-4:30. There is also a three hour time slot on either Monday or Wednesday nights from 7pm-10pm, but no math professor elected to teach in this spot, so we did not include it in our analysis. In our program, we numbered the slots sequentially, with 0 corresponding to MWF 8:30-9:30, and 11 corresponding to 3:00-4:30 TR.

[2] With the exception of Computational Linear Algebra, which is solely taught by math professors

and an upper level class (a 300 or a 400) at the same time so we gave those lower weights. However, an upperclassman is very likely to be taking multiple upper level courses at the same time, so we gave those a higher weight. Below is the weighting complete matrix:

| | 100 | 200 | 300 | 400 |
|---|---|---|---|---|
| 100 | 0 | 2 | 0 | 0 |
| 200 | 2 | 3 | 3 | 1 |
| 300 | 0 | 3 | 4 | 4 |
| 400 | 0 | 1 | 4 | 3 |

The only conflict this weighting does not handle well is that between two sections of the same course. Conflicts between sections of the same course are the most serious – because spreading our sections greatly increases schedule accessibility for a student trying to take that course – but would be weighted lower for the courses most likely to offer multiple sections (i.e., 100 and 200 level courses). Therefore this is handled separately: any two sections of the same course are defined to have a weight of 5.

# 3 Linear Program

## 3.1 Introduction to the IP

In order to make the ideal class schedule, we want to manage two things. First, we want to minimize the number of overlaps between classes that students might want to take in the same semester. Second, we want to keep in mind professor workload and create a program that will have them teach as little as possible and only what they are willing to teach. We designate constraints for each requirement a schedule should meet and use an integer program with mainly binary variables throughout.[3]

[3]This makes sense with the nature of the problem: a class is either happening or not, with no in between, and partial professors are no good for teaching, etc.

## 3.2 Variables

We make use of the following variables in our program:

- $a \in A$, the multi-set of courses to be scheduled (sections of the same course are represented by identical entries in $A$)
- $a \in A_{UL}$, the multi-set of upper-level (300 and 400 level) courses
- $s \in \mathbb{R}$; the number of groups of repeated elements in $A$; the number of courses with more than one section offered
- $a \in A_k \ \forall k \in [1, 2, ...s]$; subsets of $A$ each containing all of one grouping of identical elements from $A$; that is, subsets each containing all sections of a course that has multiple sections
- $p \in P$, the set of professors
- $i \in I$, the set of time slots (intervals)
- $x_{pai} \in \{0, 1\}$; does professor $p$ teach course $a$ in interval $i$?
- $\varepsilon_{ab} \in \{0, 1\}$; is there a conflict between course $a$ and $b$?
- $z_p \in \{0, 1\}$; does professor $p$ teach exactly three classes?
- $v_{pai} \in \{0, 1\}$; can professor $p$ teach class $a$ in time slot $i$?
- $d_{pabi} \in \{0, 1\}$; does professor $p$ teach courses $a$ and $b$ back-to-back in time slots $i$ and $i + 1$?
- $w_{ab} \in \{0, 5\}$; the weight of conflict between classes $a$ and $b$

## 3.3 Constraints

### 3.3.1 Each class taught by one professor at one time

We require that each class is taught once and only once by exactly one professor in exactly one time slot. This includes courses that have multiple sections, because we consider each section to be a different course.

$$\sum_{p} \sum_{i} x_{pai} = 1 \ \ \forall a \in A \tag{1}$$

We set the sum of $x_{pai}$ over all professors and class intervals equal to 1 to make sure that a given class $a \in A$ is taught exactly once by exactly one professor. We do that for all $a \in A$ to make sure every course is included in the schedule.

### 3.3.2 Professors teach one class at a time

A professor cannot be in more than one place at a time. Therefore, each professor must teach at most once in each time slot; for each possible pair of professor and time slot, the sum of $x_{pai}$ across all courses must be at most 1.

$$\sum_{a} x_{pai} \leq 1 \ \ \forall i \in I, p \in P \tag{2}$$

### 3.3.3 Professors teach what they are willing to teach

This constraint accesses the viability matrix to make sure that each professor only teaches what they'd be willing to teach.

$$v_{pai} - x_{pai} \geq 0 \ \ \forall p \in P, a \in A, i \in I \tag{3}$$

Both $v_{pai}$ and $x_{pai}$ are binary variables, so we have cases:

- $x_{pai} = v_{pai} = 0$: class $a$ may NOT be taught by $p$ in interval $i$, and it's not; the constraint is fulfilled.
- $x_{pai} = 0$, $v_{pai} = 1$: class $a$ may be taught by $p$ in interval $i$, but it's not; the constraint is fulfilled.
- $x_{pai} = 1$, $v_{pai} = 1$: class $a$ may be taught by $p$ in interval $i$, and it is; the constraint is fulfilled.
- $x_{pai} = 1$, $v_{pai} = 0$: class $a$ may NOT be taught by $p$ in interval $i$, and it is; the constraint is violated.

Thus, the constraint is only fulfilled when the professor is teaching a class they are willing to teach when they are willing to teach it.

### 3.3.4 Professors teach at most three courses

A professor cannot teach more than three courses in a semester. Ideally, each professor would teach at most two courses, leaving them ample time for their research and other academic duties. Unfortunately, the math department offers too many courses for this to be possible and some professors must teach three courses. [4]

For a given professor, the sum across all intervals and courses of $x_{pai}$ is equal to the total number of courses that professor is teaching. Here, we introduce the binary variable $z_p$, and note that if professor $p$ is teaching three courses then $z_p = 1$. $z_p$ will appear again in the objective function.

$$\sum_i \sum_a x_{pai} - z_p \leq 2 \ \ \forall p \in P \tag{4}$$

### 3.3.5 How many upper level courses a professor can teach

We use $z_p$ again to note that when a professor is teaching three courses, at most one of them is an upper level course (where upper level means either a 300 or 400 level course). This also means that we have to create another set: the set of only upper level courses. Call this $A_{UL}$

$$\sum_i \sum_{a \in A_{UL}} x_{pai} + z_p \leq 2 \ \ \forall p \in P \tag{5}$$

We have two cases: when $z_p = 0$ and when $z_p = 1$:

- $z_p = 0$: Professor $p$ is only teaching two courses. In this case, both of them can be upper level courses. So, when summing over all upper level courses taught in all time intervals, $x_{pai}$ can be equal to 2 or less, meaning that professor $p$ is teaching at most two upper level courses.
- $z_p = 1$: Summing over all upper level courses in all time slots, $x_{pai}$ can only be equal to 1 or 0 in order for the constraint to hold, meaning that if a professor is teaching three courses, at most one of them is an upper level course.

[4] In our case, one professor, David Ehren, was only able to teach at most one course per semester. To handle this irregularity, we include an additional constraint analagous to this one, limiting Prof. Ehren to one course per semester: $\sum_i \sum_a x_{\text{David}ai} \leq 1$

### 3.3.6 If a professor is teaching three courses, two are the same section

In order to keep overall class-prep time for each professor manageable, it is required that if a professor is teaching three courses, two of them should be two different sections of the same course. Note that in our model, different sections of the same course are represented as distinct courses with identical course numbers (repeat entries) in the overall course multi-set. Our constraint is:

$$(\sum_i x_{pai} + \sum_i x_{pbi} + \sum_i x_{pci}) m_{abc} \leq 2.5 m_{abc} \;\; \forall p \in P \;\; \forall a, b, c \in A \tag{6}$$

$$m_{abc} = |(b-a)(c-a)(c-b)| \;\; \forall a, b, c \in A$$

Note $m_{abc} = 0$ if and only if at least two of $a, b$ and $c$ have the same course number, and will be some positive integer otherwise. For a given professor $p$ and combination of three courses $a, b$ and $c$, we have three cases:

1. If $p$ teaches 0, 1, or 2 of the courses the equation is fulfilled regardless of the value of $m$
2. If $p$ teaches all 3 courses and all courses have distinct course numbers, we have $3m \leq 2.5m$, and the constraint is violated.
3. If $p$ teaches all three courses and at least two of them share a course number (i.e., are sections of the same course), both sides of the inequality go to zero, and it is satisfied.

Thus, this constraint ensures that if a professor is teaching three courses, two of them must be sections of the same course.

### 3.3.7 Course Sections Taught by the Same Prof. Must Occur Sequentially

In the current creation of teaching assignments, the math department assures that if a professor is teaching two sections of the same course, those sections can back-to-back (that is, in adjacent intervals). This reduces the amount of travel time/dead time in a professor's day, and is generally preferred by teachers in the department.

To make this constraint, we use of an indicator variable $d_{pabi}$. We let $d_{pabi} = 1$ exactly when sections $a$ and $b$ are taught back-to-back by professor $p$, in time-slots $i$ and $i+1$, and

$d_{pabi} = 0$. This is achieved through the constraint below:

$$0 \leq x_{pai} + x_{pa(i+1)} + x_{pbi} + x_{pb(i+1)} - 2d_{pabi} \leq 1 \ \ \forall p \in P \ \ \forall C(a,b) \in A \ \ \forall i \neq n \in I$$

Where $C(a,b)$ denotes all non-ordered combinations of two courses $a$ and $b$ in the multi-set of courses, and $i = n$ is the last time slot (3-4:30 Tuesday/Thursday in our model). Note that if $x_{pai}$ can equal 1 if and only if $x_{pa(i+1)} = 0$, so we have three cases:

- Neither $a$ nor $b$ are taught by professor $p$ in the pair of time slots being considered: $x_{pai} + x_{pa(i+1)} + x_{pbi} + x_{pb(i+1)} = 0$, so $d_{pabi}$ must be 0.
- One of $a$ or $b$ is taught by professor $p$ in the pair of time slots being considered: $x_{pai} + x_{pa(i+1)} + x_{pbi} + x_{pb(i+1)} = 1$, so $d_{pabi}$ must be 0.
- Both $a$ and $b$ are taught by professor $p$ in the pair of time slots being considered (the classes are back-to-back): $x_{pai} + x_{pa(i+1)} + x_{pbi} + x_{pb(i+1)} = 2$, so $d_{pabi}$ must be 1.

Therefore $d_{pabi} = 1$ if and only if courses $a$ and $b$ are taught sequentially by professor $p$ in the pair of time slots starting at $i$.

With this in place, we implement the constraint itself – that sections of the same course taught by one professor must be back-to-back – as follows:

$$\sum_{C(a,b) \in A_k} \sum_{i} d_{pabi} \geq \sum_{a \in A_k} \sum_{i} x_{pai} - 1 \ \ \forall p \ \ \forall k \in [1, 2, ...s] \tag{7}$$

Recall that $A_k$ is the subset of $A$ containing all sections of some course that have more than one section (that is, one group of identical elements in the multi-set of courses), and $s$ is the number of such subsets – that is, the number of courses with more than one section. Here, as above, $C(a,b)$ denotes all non-ordered combinations of courses from the given subset.

Note that the sum of $d_{pabi}$ over all intervals (for a given professor and course combination), will be 1 if those courses are taught back-to-back by that professor in any interval, and zero otherwise. Summing this result across all combinations of the courses (sections) in a section subset, gives the number of times the professor teaches *any* two sections in the subset sequentially. In a subset of three (identical) courses all taught by the same professor, for example, this result will be either 0, if none are sequential, 1, if a pair of them are sequential, or 2, if all three are sequential in any order.

The right-hand side of the inequality counts the number of courses in the given section subset that a professor teaches, minus 1. Again, we have cases:

- If a professor teaches none of the courses in $A_k$, the right-hand side is -1, and the inequality is fulfilled vacuously.
- If a professor teaches only one of the courses in $A_k$, the right-hand side is 0, and the left-hand side can again take on any value (however, it will by definition be zero).
- If a professor teaches two of the courses in $A_k$, the right-hand side is $> 0$, and the inequality holds *only* if they are taught back to back.
- If a professor teaches $n > 2$ of the courses in $A_k$, the right-hand side is $> n - 1$, and $n - 1$ courses must be taught adjacently, assuring they are all back-to-back.

#### 3.3.8 Counting Course Conflicts

This constraint counts class conflicts, which will eventually be what we try to minimize in the objective. We introduce a new binary variable $\varepsilon_{ab}$ for two classes $a, b \in A$ where $\varepsilon_{ab} = 1$ if a and b are taught at the same time and 0 if they are not.

$$\sum_p x_{pai} + \sum_p x_{pbi} - \varepsilon_{ab} \leq 1 \forall i \in I, \forall a, b \in C \tag{8}$$

### 3.4 Objective

Our objective has two main parts: minimizing the number of class conflicts and minimizing professor workload. In order to minimize class conflicts, we minimize $\varepsilon_{ab} w_{ab}$ for all $a, b \in A$. This means that whenever $a, b$ happen at the same time, meaning $\varepsilon_{ab} = 1$, the corresponding weight of those two courses, $w_{ab}$ will be added to the objective.

Then, we also sum over all $z_p$ so that whenever a professor is teaching three classes, a 1 is added to the objective.

$$\min \sum_{a,b} \varepsilon_{ab} w_{ab} + \sum_p z_p \tag{9}$$

Overall we minimize these two sums so that we have the least number of class conflicts and least number of professors teaching three classes.

### 3.5 The Whole IP

$$
\begin{cases}
\min & \sum_{a,b} \varepsilon_{ab} w_{ab} + \sum_p z_p & \\
\text{s.t} & \sum_p \sum_i x_{pai} = 1 & \forall a \in A \\
& \sum_a x_{pai} \leq 1 & \forall i \in I, p \in P \\
& v_{pai} - x_{pai} \geq 0 & \forall p \in P, a \in A, i \in I \\
& \sum_i \sum_a x_{pai} - z_p \leq 2 & \forall p \in P \\
& \sum_i \sum_{a \in C_{UL}} x_{pai} + z_p \leq 2 & \forall p \in P \\
& (\sum_i x_{pai} + \sum_i x_{pbi} + \sum_i x_{pci}) m_{abc} \leq 2.5 m_{abc} & \forall p \in P \quad \forall a, b, c \in A \\
& m_{abc} = |(b-a)(c-a)(c-b)| & \forall a, b, c \in A \\
& 0 \leq x_{pai} + x_{pa(i+1)} + x_{pbi} + x_{pb(i+1)} - 2 d_{pabi} \leq 1 & \forall p \in P \ \forall C(a,b) \in A \ \forall i \neq n \in I \\
& \sum_{C(a,b) \in A_k} \sum_i d_{pabi} \geq \sum_{a \in A_k} \sum_i x_{pai} - 1 & \forall p \ \forall k \in [1, 2, ...s] \\
& \sum_p x_{aip} + \sum_p x_{bip} - \varepsilon_{ab} \leq 1 & \forall i, a, b \\
& \sum_i \sum_a x_{\text{David}ai} \leq 1 & \\
& v, x, z, \varepsilon \in \{0, 1\} & \\
& m \geq 0 \in \mathbb{Z} &
\end{cases}
$$

# 4 Code

We chose to implement the IP in Python, using the PuLP 2.70 LP solving package. Our goal was a fully generalized program that could take in any sublist of math courses and professors from our complete list (as well as read in the viability matrix $V$ containing all values $v_{pai}$ from professor preference .csv files), and return the ideal schedule, detailing when and by whom each course should be taught, as well as how many conflicts and workload violations occurred.

The run time of the program proved significantly variable, solving in a under a minute for the Fall '22 course and professor list, while not terminating even after multiple hours for the Spring '22 courses and professors. We elected to use the PULP_CBC_CMD solver, which allows the user to set a time limit, and produced all included results with a time limit of 3000 seconds. We observed stability in the overall best answer after only around 600 seconds, and ended with a small solution gap in every case.

Code for the project may be accessed here.

# 5 Results

## 5.1 Fall 2022

Our proposed schedule, built using the above IP, significantly outperformed the actual schedule in terms of conflicts and professor workload. The original schedule for Fall 2022 has an objective value of 22.0: four weight 3 conflicts and three weight 2 conflicts, plus 4 points from professor workload. In contrast, our proposed fall schedule had an objective value of only 4.0, all of which came from professor workload (since there are more classes than twice the number of professors, some professors must teach three classes). The table below sums up these results, and Figures 1 and 2 compare the two schedules, with courses in red constituting a meaningful (non-zero weight) conflict.

| Original Fall 2022 | Our Fall 2022 |
|---|---|
| Objective: 22.0 | Objective: 4.0 |
| Loss from conflicts: 18.0 | Loss from conflicts: 0.0 |
| Loss from prof. workload: 4.0 | Loss from prof. workload: 4.0 |

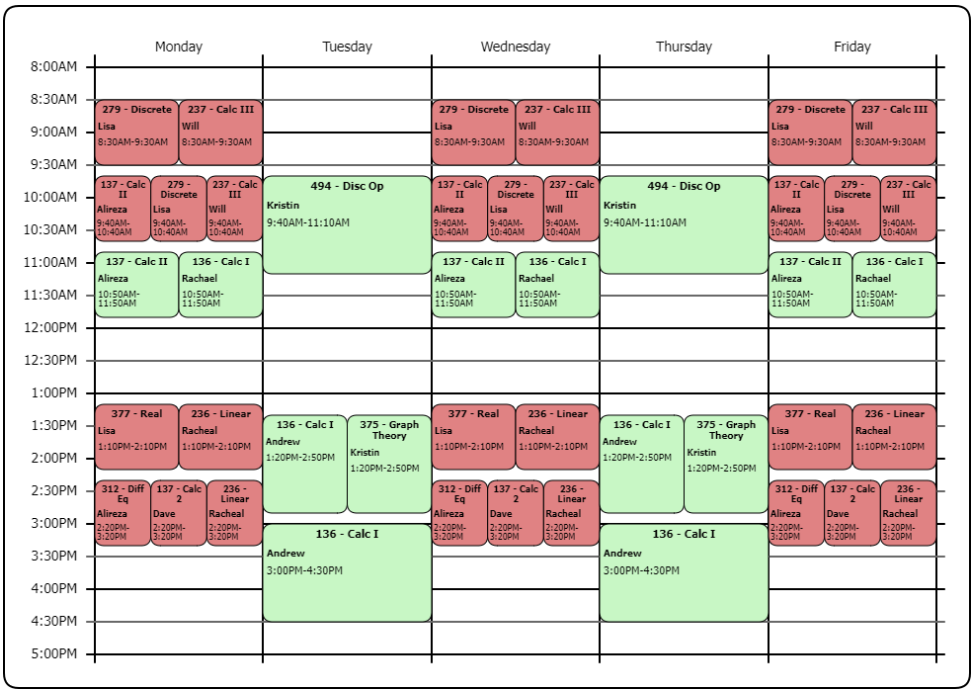


Figure 1: Real Course Schedule Fall 22'

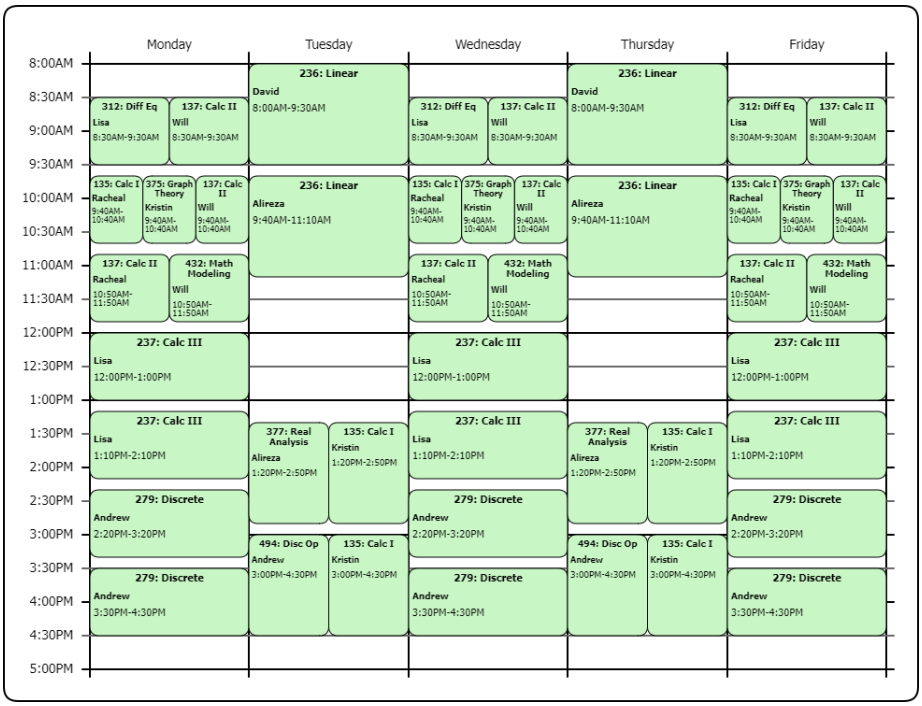


Figure 2: Proposed Course Schedule Fall 22'

## 5.2 Spring 2023

For spring 2023, our IP-built schedule again far out performed the actual schedule. The original spring 2023 schedule had an objective value 30.0: six weight 3 conflicts, four weight 2 conflicts and one weight 1 conflict, and 3 points from professor workload. On the other hand, our proposed spring schedule has an objective value of 7.0: one weight 3 conflict and one weight 1 conflict, along with the same professor workload of 3. Our objective is higher than it was for the fall because more courses are offered in the spring, so some conflicts are inevitable. These results are reflected in the table below, as well as Figures 3 and 4.

| Original Spring 2023 | Our Spring 2023 |
|---|---|
| Objective: 30.0 | Objective: 7.0 |
| Loss from conflicts: 27.0 | Loss from conflicts: 4.0 |
| Loss from prof. workload: 3.0 | Loss from prof. workload: 3.0 |

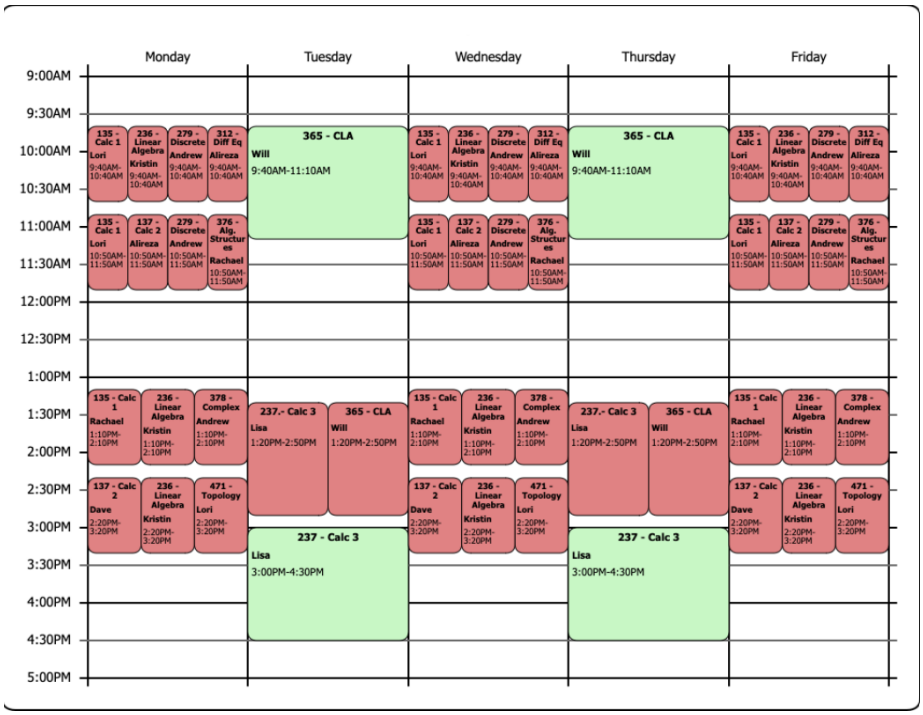


Figure 3: Real Course Schedule Spring 23'

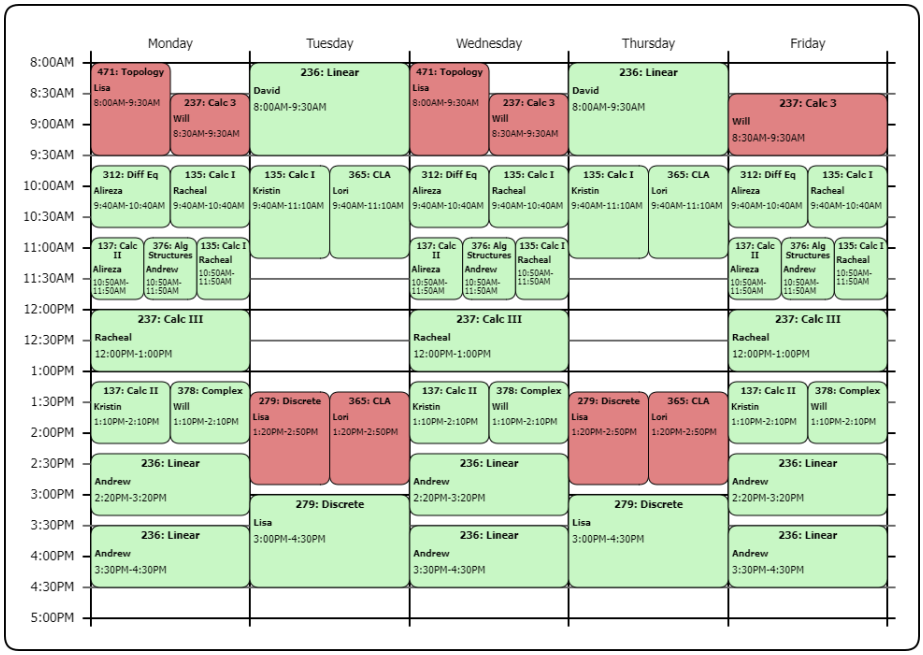


Figure 4: Proposed Course Schedule Spring 23'

# 6 Conclusion

While very successful at optimizing the metrics it was built on within the constraints given, there are some things the model fails to account for, and ways that it could be improved with more time.

For example, we didn't take into account professor preference. When actually creating course schedules at Macalester, Math professors select whether they'd be "excited" to teach a course or just "willing" to teach it. We only asked professors what classes they'd be willing to teach, so they might not be as excited about our proposed schedule as the current one, even if it is more conflict-free.

We didn't include a constraint that some professors requested: that if they are teaching two distinct courses they should *not* be back-to-back.

Given more time, we would also want to make sure to implement a 2-3 or 3-2 workload, meaning that a professor is only teaching five courses over the school year, either two in the fall and three in the spring or visa versa. Since our program only does one semester at a time, we were unable to create a constraint that would work over two semesters, but that is definitely something to consider in the future.

Another area to improve was our weighting scheme. As previously mentioned, it was mostly subjective and built on only our opinion about how "bad" any given conflicts were. If there was a way to make a more unbiased way to judge class conflicts, that might make the schedule more useful to a wider range of departments.

Along those lines, our integer program only used data from the math department so that we wouldn't have too much to work with, but in the future if we could scale up our work so that it could be used across other department (or perhaps other schools) it would be much more applicable.

Overall, while our integer program definitely has ways in which it can improve, it did create a more conflict-free class schedule than the original, so we did achieve our goal.

# References

[1] E. A. Akkoyunlu. "A Linear Algorithm for Computing the Optimum University Timetable". In: *The Computer Journal* (1972).

[2] Stuart Anthony Mitchell Jean-Sebastien Roy. *PuLP*.

[3] Stefan Helber Katja Schimmelpfeng. "Application of a Real-World University-Course Timetabling Model Solved by Integer Programming". In: *Or Spectrum* (2007).

[4] E. Housos S. Daskalaki T. Birbas. "A Linear Algorithm for Computing the Optimum University Timetable". In: *European Journal of Operational Research* (2004).