USC Marshall
School of Business

DSO 570: The Analytics Edge: Data, Models, and Effective Decisions
Final Project Report

Dhwani Kapadia

Angela Ma

Sharath Polu

Ziqing (Juno) Wen

Emma Yang

May 12, 2020

1 Executive Summary

The purpose of this project is to investigate the possibility of applying optimization to improve the current classroom scheduling system for the University of Southern California, Marshall School of Business (referred to as Marshall). After analyzing the current process and identifying the main inefficiency, an optimization tool was designed to accelerate the scheduling process and reduce manual workload, while satisfying the preferences and needs of the students and faculty members as much as possible.

Shannon Faris, the Assistant Dean of Institutional Research and Academic Administration at Marshall, and her colleague, Hal Warning, are in charge of coordinating classroom scheduling for the entire school. The current scheduling process is time consuming as it is completely manual. The scheduling process can be roughly divided into two phases. In Phase I, Shannon's team allocates to department coordinators classrooms and time slots based on historical records and the coordinators try their best to schedule classes within their department. Whatever unfinished job is left to Shannon in Phase II, during which scheduling becomes much more difficult. Having been trying for years to maximize the percentage of courses scheduled by the end of Phase I, Shannon hired our team to develop a solution.

Our team identified the large amount of manual work, as well as the unsatisfactory schedule rate in Phase I as the two major sources of inefficiencies that should be improved. Therefore, we developed a tool that automates Phase I scheduling. It takes a list of class sections as requests, and time and classroom resources that are available. It then attempts to assign each section to a classroom at a given time under a set of prespecified constraints, with the goal of maximizing the number of sections successfully assigned, weighted by the time preference of professors. The program takes in one Excel file as input and outputs one Excel file. Two sets of sample inputs and outputs are provided for demonstration and testing.

The tool was tested on semi-real input generated from Spring 2019 data. The results showed that the tool is able to process 515 sections and successfully schedule over 90% of them within 30 minutes. If this tool is adopted in Phase I to replace the current manual scheduling process, the amount of administrative effort required for both Shannon's team and the department coordinators, as well as the process time needed, can be significantly reduced. In addition to administrative personnel, professors and students can also benefit from this tool because it not only introduces preference scores to systematically accommodate the preferences and needs of the professors, but also tries to allow more flexibility for students to choose classes at their convenience and per their preference.

Considering all the potential gains, we strongly recommend that Shannon's Team consider the possibility of adopting the automated optimization tool in Phase I and implement a transition plan we drafted for the client to gradually move away from the current process and eventually fully adopt the tool a few semesters ahead.

2 Opportunity for improvement

2.1 Current Scheduling Process

In the current process, the initial scheduling of courses and classrooms begins almost one year before the semester begins. In Phase I, Shannon's team allocates a set of classrooms and time slots to each department based on what classrooms and slots it has used historically. The department coordinators will then try to put each class the department offers into a classroom at a certain time. Whatever classes the department coordinators cannot schedule become Shannon’s responsibility. Her team will try to assign the remaining classes to the remaining classrooms and time slots in Phase II. The difficulty of assigning classrooms in Phase II is tremendously high as a lot of slots are already taken. Shannon’s team may need to find potential course time swaps in order to find allocations that satisfy all courses.

In the past years, many improvements have been made to increase the percentage of courses scheduled by the end of Phase I. However, according to Hal, there are still around 25% classes that are not scheduled in Phase I. To validate and quantify the issue, we compared the initial allocation for Spring 2017 and the final schedule. We found that the total number of hours assigned, distribution of the time slots of different lengths, and the department match rate changed significantly from the beginning to the end.

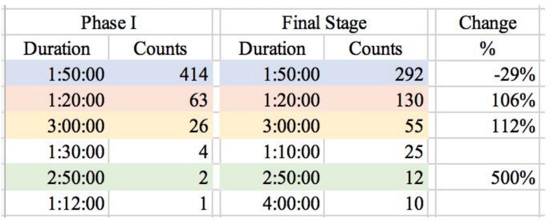
2.2 Difference of Total Scheduled Hours

During initial allocation in 2017, the total time slots assigned amounted to 1724 hours while in the final schedule, there were 1844 hours for all time slots assigned, which was 7% more than initial allocation. Here, the calculation was based on time slots. In other words, if 10:00-11:50 am on Mondays and Wednesdays were assigned to one course section, it was counted as 3.67 hours. This showed that initial allocation did not account for as many sections as the final schedule.

2.3 Variety of Time Slots Length

During Phase I initial allocation, there were only 6 types of class durations. However, in the final schedule, there were 22 types of time slot lengths. This shows that the initial allocation did not do a good job in meeting the department needs for various course durations. More specifically, during Phase I, 414 time slots of 1 hour and 50 minutes were assigned. However, only 292 of them were needed according to the final schedule. For the 1 hour and 20 minutes slots, only 63 slots were assigned during Phase I, but 130 slots were scheduled eventually, which means only half of the needs were met during initial allocation. Table 2.3.1 below shows the count and percentage of sections with different length and the percentage change is calculated in the rightmost column.

Table 2.3.1 Distribution of Top 5 Sections of Different Length



Phase I		Final Stage		Change
Duration	Counts	Duration	Counts	%
1:50:00	414	1:50:00	292	-29%
1:20:00	63	1:20:00	130	106%
3:00:00	26	3:00:00	55	112%
1:30:00	4	1:10:00	25	
2:50:00	2	2:50:00	12	500%
1:12:00	1	4:00:00	10	

2.4 Percentage of Matched Slots Over Total Final Slots

Data Cleaning

We looked at the initial and final schedule for 1 hour and 20 minute, 1 hour and 50 minute, and 3 hour classes separately to simplify the problem since they made up 98.6% of the initial slots and 84.7% of the final schedule slots. There are only six different time lengths for the initial stage while there are 22 different lengths at the final stage. Therefore many of the unconventional types of time slots cannot be directly compared across stages. For example, there are only ten classes scheduled for 4 hours in the final schedule while there aren't any in the initial allocation.

Also, we only kept the slots from the seven major departments, Business Communication (BUCO), Data Sciences and Operations (DSO), Finance and Business Economics (FBE), Leventhal School of Accounting (ACCT), Management and Organization (MOR), Marketing (MKT), and Lloyd Greif Center for Entrepreneurial Studies (BAEP), since there are other departments that appear only in either Phase I or the final schedule. There are some ambiguous department names such as 'TUTORING' and 'OFFLINE', so we decided to eliminate those departments and were left with seven major departments across the three conventional class lengths we chose. Spelling discrepancies in classrooms exist between the initial allocation and the final schedule (i.e some used white space and some did not). We performed necessary data cleaning to make sure the naming conventions are consistent and the count of matches are therefore valid.

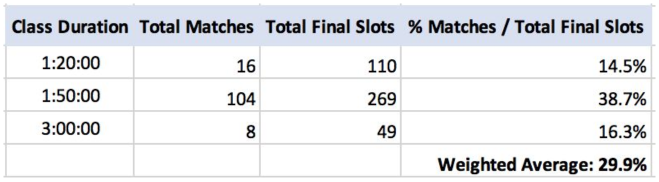
A slot in the initial allocation and a slot in the final stage are considered matched if their classroom, department, days, and start time are exactly the same. In order to find such matches, we created a unique identifier, concatenating classroom, department, days, and class start time, for each initial and final stage section. Duration is not included since we already separated the data by duration. For all three types of class durations, there is no duplicate value in the initial stage with the same classroom, department, days, and class start time. However, there are duplicate values in the final stage, mainly because there are first-half and second-half semester classes, with different section codes but the same classroom, department, days and start time.

There is a small portion of the duplicates that are all semester long classes with consecutive section codes, which we observed from our current registration system that they are different sections open for different departments at the time of registration for the purpose of reserving enough spaces for a specific department. For example, there are DSO classes that are mainly reserved for the MSBA program but are also open to other graduate programs. In this scenario, the school creates a section for only the MSBA program while having another section with different section code for other graduate programs, and these two sections would have the same classroom, days, and start time. After careful consideration, we decided to combine these duplicates and only count once in the final schedule. We then merged the identifiers created in the Phase I and final schedule together and got the minimum number of unique identifier counts for Phase I and the final schedule to as the number of matches.

Key Findings

Table 2.4.1 below shows the percentages of matched slots between the initial and final schedule over the total number of final slots across the three different class durations. The weighted average of match rate across all length types is as low as 29.9%. This match rate serves as the upper bound estimate of the percentage of courses that can possibly be scheduled by the end of Phase I, which is an important performance metric for evaluating the efficiency of the current system. The reason why we could only calculate an upper bound rather than a direct number is because no end of Phase I data was provided. The only data available regarding Phase I is the initial allocation.

Table 2.4.1: The Percentage of Matched Slot Over Total Final Slots



Class Duration	Total Matches	Total Final Slots	% Matches / Total Final Slots
1:20:00	16	110	14.5%
1:50:00	104	269	38.7%
3:00:00	8	49	16.3%
			Weighted Average: 29.9%

Note that the comparison analysis we performed is based on the Spring 2017 data, which is quite outdated. Diligent improvement has been made since then and the match rate should be expected to be much higher now. We chose Spring 2017 data because this is the only semester of which both initial allocation and final schedule is available to allow a comparison. Regardless of the limitations, the match rates indicate that most changes happened outside of what was planned in Phase I, and this is where we saw the opportunity for improvement. The purpose of our optimization software is to help Shannon's team better take in more requests in Phase I that can be fit in the final schedule with changes as little as possible to make the scheduling process easier for the team.

3 Optimization Methodology

In order to improve the current classroom scheduling process after understanding the major issues, an optimization tool was developed to automate Phase I scheduling and replace the majority of human work while achieving similar or higher performance. The most important performance metric is the number of sections successfully scheduled by the end of Phase I. Other KPIs like faculty satisfaction, student satisfaction, classroom utilization and adaptability are also weighted.

3.1 New Process

With the successful deployment of the tool, Shannon's team and the department coordinators will be freed from manual scheduling completely in Phase I. In the new process, each department will collect its own class information and submit it to Shannon's team. Such information includes the numeric code of the section, name of the course, length of the section, beginning and end dates of the section (aka the session type), how many times the section meets every week, the instructor of the section, the number of seats offered, and the instructor's preferences towards different teaching times. None of these categories exceed the scope of information departments would normally collect for class scheduling except the professor preferences, which will be discussed in detail in Section 3.2.

In the meantime, Shannon's team will decide which classrooms they want to use and how they would like to break down the available times in a week. To a large extent, these decisions will follow conventions and will not change too much from semester to semester. For example, two-hour slots would always need to be provided to accommodate undergrad classes and most likely will be arranged to start at even hours like they always have been. Paired slots will probably still be set on either Monday and Wednesday or Tuesday and Thursday to host classes that meet twice a week. Classrooms to be considered would always include and mostly consist of Marshall classrooms.

The major difference is that now Shannon does not need to differentiate undergrad classrooms and graduate classrooms. Instead of having a set of two-hour time slots only eligible in 30 classrooms and a set of one-and-a-half-hour slots only eligible in 8 classrooms (this is what the current initial allocation does), slots of different lengths can be mixed up in the same room as long as overlapping slots are not used at the same time. All that Shannon needs to specify is the standard slots they would like to allow, defined by the day of the week, the starting time, and the length. The tool will figure out which combination of slots to use in each classroom that maximizes time usage without conflict.

Upon receiving class information from the departments, Shannon's team will compare the offered courses with historical data and group the one that students are oftentimes taking together. Then all the classes information, list of classrooms and their capacity, and the list of standard time slots should be combined into one master Excel file. This file can be fed into the optimization tool by running the following command in the command line:

```
python scheduling.py input.xlsx output.xlsx
```

If everything is in the correct format, within 30 minutes, another Excel file will be outputted to the same directory where the optimization tool and the input file live. The output file will contain a sheet of all the successfully scheduled sections, with their assigned classroom, meeting days, and the begin and end times. The instructors' preference score towards the scheduled time and the capacity of the scheduled classroom will also be outputted for easy analysis. All the unassigned sections can be found in the second sheet. From the third sheet on, each sheet is a Monday to Friday, 8am to 9:30pm timetable designated to one classroom, visually showing which sections are scheduled in this classroom and at what times. And this wraps Phase I.

Two sets of sample input and output files are provided for demonstration and testing purposes. The user can refer to them to get an idea of what kind of preparation is needed and what result can be expected. In addition, a documentation of the optimization tool is attached which provides extensive details about how to run the tool, prepare the input, and interpret the output.

It is expected that the amount of human labor work for both the department coordinators and Shannon's team in Phase Ican be significantly reduced if this new process and tool is adopted. The remaining human work mainly focuses on collecting and preparing input data, most of which are things the departments or Shannon would already be collecting. Only three pieces of information are fairly new to the users, either in the concept or in the format, and they are discussed in detail in the following section.

3.2 New Data

All the three novel data inputs are either optional or semi optional. The tool will run perfectly fine if the user does not input professor preference scores (help increase faculty satisfaction) or the together courses list (help increase student satisfaction). It just makes certain constraints of the optimization problem less strict and therefore more classes can potentially be scheduled. The list of standard slots must be provided, but in the case that the user is still figuring out how to generate this list, the sample input (large) offers a pretty comprehensive example that the user can take and use directly or with minimum modification.

Professor Preference Scores

In the past, individual professors may reach out to the department coordinator to express preference for teaching times, but it has never been done in a systematic way. The optimization tool we designed makes it possible. The professor preference scores we propose measures a professor's willingness to teach at a given time. It takes three values, with 2 meaning preferred, 1 meaning acceptable, and 0 meaning unavailable. Any section that a professor teaches should never be assigned to a time that the professor has assigned 0 to. The score should be detailed to every 30-minute window on every weekday, divided into first half and second half semesters. This data can be collected through surveys.

Standard Time Slots

Shannon's team has long experience dealing with time slots, but perhaps not in the specific format our tool requires. A standard Monday 8:00am to 9:50am slot needs to be written as "M-08:00:00-4", with the first part indicating the day (or days) of the week, second part indicating the start time, and the third part indicating the length of the slot, rounded up to the closest times of half hours and measured in units of half hours. The concatenation (patching strings of text together) and the text (adding leading 0 to the start time to maintain consistent length) functions in Excel will come in handy when trying to create slots in large batches. The list of standard time slots should cover all the common slots Marshall classes are usually scheduled at, including at least one-day and two-day (MW or TH) 90-minute, 120-minute, and 180-minute slots. Slots of other lengths, or with atypical starting time, can also be included with the expectation that the more slots there are, the longer it takes to compute a solution.

Together Courses

The reason we care about the courses that students usually take together is because we want to schedule them in a way that new students will continue to be able to take them all and thus are more satisfied. A group of together courses are usually core courses of an academic program plus a couple of popular electives.

There are two ways Shannon's team can extract this information. The first way is to use each program's study plan that recommends a set of core courses a student should take in each semester. The second way is to look at historical data. The client has been tracking similar information for years in the "student_course_selection_2015to2019.xlsx" file. Now they just need to go one step further to extend the pair of courses students usually take together into groups of together courses.

3.3 Optimization Setup

The optimization tool solves the scheduling problem in two steps. It first pairs up half semester sections and combines them into full-semester sections in Part I, and then performs the actual classroom allocation in Part II. Any half-semester sections that are not paired in Part I will be treated as full-semester sections in Part II.

Part I: Pair Up Half-Semester Sections

Decision Variables: Whether to pair up a first-half-semester section with a second-half-semester section

Objective: Maximize the total eligible slots of all successfully paired sections

For example, consider that we have sections A, B and C. A is a 2-hour first-half-semester section, B and C are 2-hour second-half-semester sections. Suppose there are only two slots available: 8am and 10am. The instructors of section A and C are available at both 8am and 10am, while the instructor of section B is only available at 8am. If sections A and B are paired up, the resulting combined section will only have the 8am slot eligible for consideration. If sections A and C are paired up, the combined section will be eligible at both MW 8am and 10am. Therefore, the optimization will choose to pair up sections A and C.

Constraints:

1. Only sections with the same duration and meeting frequency can be paired.
2. Sections can only be paired when the resulting combined section has at least one eligible slot that works for the instructors of both sections.
3. Each section can only be paired once.

Part II: Assign Sections to Classrooms and Time Slots

Decision Variables: Whether to assign a section to a given classroom and a given time slot

Objective: Maximize the total professor preference scores of all assigned sections

Constraints:

1. Sections belonging to the same course cannot be scheduled to the same time slot or to overlapping time slots.
2. For each classroom, each time slot can only be assigned no more than once and no two overlapping slots can be used together. For example, if a section is scheduled for Monday 2pm, the Monday-Wednesday paired 2pm slot can no longer be used. Additionally, if a classroom is occupied from 2pm to 4pm, it cannot be assigned to a slot starting at 3:30pm.
3. Every section needs to be put into a time slot that 1) works for the professor who teaches the section, 2) has the shortest possible length that can fit the section in, and 3) has the same meeting frequency. For example, if there are three types of standard slots: 90-min, 120-min and 180-min, a 130-minute section that meets twice a week must be assigned to a 180-min two-day time slot during which the professor is available.
4. Sections must be assigned to classrooms big enough.

5. Every section can be assigned no more than once.
6. Sections taught by the same professor cannot be assigned to the same time slot or to overlapping time slots.
7. For any set of courses that students usually take together, at least one combination of their sections must satisfy that no two sections in the combination are assigned to the same time slot or to overlapping time slots.

4 Optimization Results

The proposed optimization tool was tested on a semi-real dataset generated from the actual schedule of Spring 2019. A total of 515 sections with real class related information were inputted. A total of 134 standard time slots were provided, covering the most common slot types (90/120/180/240 minute slots on M-F/MW/TH). All 38 Marshall classrooms were allowed. Two hundred and eight professors were involved, among which preference scores of 189 were provided. It is designed to not provide preferences for those 19 professors to mimic a potential real life situation of partially missing data.

Professor preference scores were generated in a way that 1) sets all actual scheduled times to available and preferred (2), 2) sets any time period between two scheduled classes on the same day to available and preferred, 3) sets the three hours before and three hours after the preferred times that's between 8am and 9:30pm to available and acceptable (1), and 4) adds additional availability (set to 1) if a professor only has availability on one day (except if that day is Friday).

The performance of the optimization tool is evaluated based on six client-desired goals.

4.1 Efficiency

The tool is able to assign 90.7% (467) sections within 30 minutes. Note that the test input only selected 515 out of the 582 total scheduled classes in Spring 2019, with the exclusion being mainly sections of unconventional session types that cannot be read by the tool directly. Even with a conservative estimate that none of those excluded sections can be transformed into a readable format, the percentage of classes successfully scheduled by the end of Phase I using our tool is still above 80% (467/582). This is a remarkable increase from the 29.9% upper bound in 2017 (see Section 2), and a plausible improvement from the 75% in current days quoted by Hal. Most importantly, the tool is automated. If adopted, it can significantly reduce the amount of manual work required for both Shannon's team and the department coordinators in Phase I.

4.2 Student Satisfaction

The tool tries to achieve a high level of student satisfaction by forcing sections under the same course to be scheduled at different times and mandating together courses to have at least one non-conflicting section combination. Every course in the 13 groups of together courses has at least one section successfully scheduled. Eight out of the 13 groups have sections all scheduled at different times, while the average percentage of non-conflicted combinations across groups is 83.8%. Even in the most challenging group that contains 12 courses that need to be considered together, the tool still manages to secure 15 different combinations of non-conflicting sections a student can possibly use. In short, the tool is doing a good job providing flexibility for students so that they are more likely to be able to register the courses they most desired.

One thing we found from the test result is that among the 48 unassigned sections, 22 of them belong to the same course WRIT-340. This may be an unwanted result of forcing sections under the same course to be all scheduled to different and non-overlapping times, interacting with only allowing the smallest possible slots to be used. More discussion will be covered in Section 5.1.4.

4.3 Faculty Satisfaction

The tool successfully implemented a systematic way of accommodating faculty preferences through the preference scores. As the test result reveals, 70.4% of assigned sections have a preference score of 2, indicating that they are scheduled to a time slot that the professor prefers. Average professor preference score across all sections is around 1.76. No section has a preference score below 1, meaning that every 30 minute period within the scheduled time is available for the instructor.

Moreover, our tool does a fairly good job at minimizing the number of days a professor needs to be on campus to teach. On average, professors are on campus 2.1 days every week, with nearly 80% of the professors teach for at most 2 days in a week. As Table 4.3.1 below shows, the percentage of professors who need to be on campus for 1, 2, 3, 4 and 5 days in the test result are very close to the actual schedule of Spring 2019. This provides some guarantee that even without explicitly trying to minimize the number of days on campus, our tool does at least an equally good job as the current process does.

Table 4.3.1: Professor Breakdown by Number of Days Required on Campus

	Spring 2019 Actual		**Spring 2019 Test**	
	Professor Count	**% Total**	**Professor Count**	**% Total**
Days on Campus				
1	43	19.4%	50	24.8%
2	130	58.6%	111	55.0%
3	22	9.9%	15	7.4%
4	23	10.4%	21	10.4%
5	4	1.8%	5	2.5%
Total	222	100.0%	202	100.0%

We also acknowledged that the more availability professors provide, the more likely more classes can be successfully scheduled. And the more 2s a professor gives, the more likely his final assigned section has a higher preference score. The two metrics below are used to measure the preference and availability leeway professors provide.

- <u>Availability to Demand Ratio:</u> The total weekly time a professor marks as available (scored 1 or 2, averaged across the semester) divided by the total time he or she needs to teach every week (half semester classes are weighted in half)

- Preferred Percentage: The total weekly time a professor marks as preferred (scored 2, averaged across the semester) divided by the time marked as available

Among the 189 professors whose preference scores are provided, the average availability to demand ratio is around 3, while 13.2% of the professors have a ratio below 2. The lowest ratio is about 1.5. Such narrow availability leeway should be avoided in a real world scenario because it increases the difficulty of optimization. On the other hand, the average preferred percentage is 40%, and around 2.1% professors set more than 70% of their availability to prefered. The largest preferred percentage is 88.9%. It is suggested to have the preferred percentage capped at a threshold so that no professor can take advantage of the system by assigning a huge proportion of their availability to preferred.

4.4 Transparency and Fairness

Our tool is transparent because how the preferences and needs of students and faculty are weighted is clearly set up and fairly quantifiable. It is not very balanced, however, considering that the tool leans more towards prioritizing the faculty satisfaction. But this is more because of the fact that the classroom scheduling process, by its nature, requires more involvement from the faculty than from the students.

4.5 Administrative Effort

The input files are either readily available or can be generated by the administration within a relatively short amount of time. Besides, the tool completely automates the actual scheduling happening in Phase I.

4.6 Adaptability

The tool has good adaptability as a prototype. It is able to parse any length of sections and any shape of standard slots as long the input does not go beyond the allowed range. It is able to read professor preferences measured in half-hour windows and translate it into ratings towards standard time slots. The tool also smartly detects if two slots are overlapping.

5 Discussion

5.1 Appropriateness of Methodology

Our optimization methodology is aimed at solving the inefficiency problem that has been one of the most critical issues that lingered for years, thus the proposal is relevant. Our approach takes into account all the major stakeholders involved in the classroom scheduling process and tries to make improvements for Shannon's team, department coordinators and teaching professors, while maintaining a similar level of student satisfaction. Thus the solution is well thought, well balanced, and does not have fatal flaws. Even with simplification, the optimization tool we propose is able to directly process nearly 90% of the scheduling requests (515 out of 582 when testing with Spring 2019 data), and if slight tweaks with input are considered, more can be done.

5.1.1 Input and Output Data

Overall, the input and output data are selected and designed to be as user friendly as possible. The majority of the input data are readily available and those that require extra work are made optional. The submission of the mandatory data alone is sufficient for a successful basic schedule to be produced. As the user collects and inputs more data like the preference scores and together courses list, the tool becomes more powerful. Although the actual optimization problem requires data input of much more complicated form, only the raw form of input is asked from the users to make it easy for them to use the tool. For example, we do not ask the user to specify which standard time slots are conflicting and cannot be used together. Instead we programmed the tool so that it reads the time slot input and automatically detects conflicts.

Similarly with the output data, for any assigned section, we output the scheduled times and classrooms in the format that is consistent with the client's own writing so that data consolidation is seamless. Professor preference scores and classroom capacities are also exported to allow quick analysis. In addition, we demonstrate the scheduling results visually in timetables for each classroom because it is much easier to spot an available spot by looking at a timetable than by looking at rows of text entries.

5.1.2 Optimization Objectives and Constraints

Part I: Pair Up Half-Semester Sections

Our solution is built on two optimization problems. In Part I, half-semester sections are paired up and combined into full-semester types. Then Part II does the actual classroom assignment. We break the optimization problem into two steps so that a seemingly unsolvable problem is made possible. Originally when trying to run the scheduling optimization directly with three types of semesters, too many decision variables were created and way too many constraints were set up. The resulting optimization problem is outrageously too large for a personal computer to solve.

At this stage, the number one goal of the classroom scheduling is to get as many classes scheduled as possible. With this in mind, the objective of the optimization Part I is set to be maximizing the total eligible slots of the resulting combined slots. In other words, we want the preference scores table of the newly formed instructor team to have as many non zero values as possible, so that the constraints of the optimization Part II will be as loose as possible. The three constraints are there to make sure the sections paired up are indeed meaningful and are at least possible to be scheduled in the next state.

Part II: Assign Sections to Classrooms and Time Slots

Optimization Part II, the scheduling optimization, is designed to closely reflect the thought process of how a human would perform classroom scheduling. The objective is the sum of total classes assigned weighted by the preference score the professor of each class gives to the time slot it is assigned to. By incorporating the weight, we introduced a systematic way to accommodate professors' preferences and needs and improve professor satisfaction. More importantly, now the level of satisfaction can be quantified.

Constraint #1: Sections belonging to the same course cannot be scheduled to the same time slot or to overlapping time slots. This provides the students with more choices to choose from depending on their convenience, and thus increases student satisfaction.

Constraint #2: For the same classroom, each time slot can only be assigned no more than once and no two overlapping time slots can be used together. This makes sure the assignment is meaningful. Assigning a class to start before another class has ended is not a valid assignment. Assigning a class to a Monday slot while the slot is already used by a Monday-Wednesday paired slot is also not a successful assignment.

Constraint #3: Every section needs to be put into a time slot that works for the professor who teaches the section and the slot must have matching meeting frequency and is the smallest that can fit the section. This constraint is necessary to make sure that the professor is available to teach at the scheduled time and the allocated time window is long enough for the class.

The constraint to consider only the smallest possible slot is put in place to maintain a certain level of consistency in class times. Historically the start time of classes of different lengths follows a pattern and therefore the end time of a class is relatively predictable given the start time. For example, 120-minute classes usually start at even hours like 8am, 10am, or 12pm, etc. 90-minute classes usually start at 8am and every 90 minutes after. Suppose that the tool is considering whether to put a 120-minute class in a classroom large enough to fit. In the circumstance where the classroom is scheduled before 12:30pm and after 3:30pm, even if the professor of the section is available between 12:30pm and 3:30pm, the section will not be scheduled because the 120-minute slots only start at 2pm and 4pm.

Constraint #4: Sections must be put in classrooms big enough. Regulations require that the number of students registered should not be greater than the classroom size.

Constraint #5: Every session can be assigned no more than once. Every section is unique and is only designed to be offered once each semester.

Constraint #6: Sections taught by the same professor cannot be assigned to the same or overlapping times. A professor simply cannot be at two different places at the same time.

Constraint #7: For any set of courses that students usually take together, at least one combination of their sections must satisfy that no two sections in the combination are assigned to the same time slot or to overlapping time slots. Students usually take certain classes together (core courses and popular elective courses) and a smart class schedule should allow it. Part of it is a hard requirement because all students must take core courses usually in certain semesters in order to graduate. Not allowing them to do so will result in inconvenience for both the students and the department. The other part of it is a soft requirement to make students happy as they should be able to register for the popular electives.

These constraints are quite comprehensive and take care of the requirements of many stakeholders. With that said, a few assumptions are made to rule out the uncommon corner cases for simplification and there are limitations that come as a result of these assumptions, as well as the result of design choices. Section 5.1.3 below summarizes the assumptions and 5.1.4 acknowledges the major limitations.

5.1.3 Assumptions

1. Sections can only be held between 8:00 am and 9:30 pm Monday to Friday. They can meet either once a week or twice a week on Mondays and Wednesdays or Tuesdays and Thursdays.
2. Only 6 session types are allowed, which are full semester sessions 1 and 33, first half semester sections 411 and 431, and second half semester sections 415 and 442. In most cases, users can tweak the sections that don't directly satisfy this assumption into formats that do. For example, sections of session type 406 start slightly earlier than a standard full-semester section of session type 1 or 33 and end slightly later. But it can be treated as a full semester section. Temporarily changing the session code to 1 or 33 will make sections of this type processable by the optimization tool.
3. Available time slots are universal for all classrooms.
4. Standard slots always start or end at whole hours and their lengths are always multiples of 30 minutes.
5. Sections are allowed to be scheduled back to back without a break in the same classroom. For example, if a three-hour section is scheduled on Monday at 3:30 pm in JKP 212, it is allowed to have another section scheduled to start at 6:30 in that same classroom.
6. Each section only has one instructor. If a section involves multiple instructors, a work-around is to consider them as a team. The availability of the team will be the intersection of the availability of all the members in the team, and the preference score will be the average score across all the members.

5.1.4 Major Limitations

The optimization tool treats all sections the same, no matter whether they are core or elective, or undergrad or graduate level. This may not be desired because in most cases there is a priority. For example, there need to be enough sections of a core course scheduled to allow all the students who are required to take the course take it, while in the worst-case scenario, the school can give up on an elective if the time doesn't work out. Potential ways to take the rank of priority into account can be found in Additional Limitation 2 in Appendix A2.

The program is not designed to partition time in the day itself based on the scheduling request and resources, but rather requires the user to input formed slots. In addition, sections are required to be scheduled to the shortest possible slots that fit and longer slots are never considered. Inevitably, the usage of time and space cannot truly be optimized. However, this might be desired for administrative concerns so that classes start and end at consistent times.

Another issue with the time slots is that currently only certain days of the week and only certain combinations of days are allowed as input. Trying to input values outside of allowed ranges can cause the program to crash or produce invalid output. See Additional Limitation 6 in Appendix A2 for suggestions on how to lift this limit on scalability.

Moreover, the program does not allow a certain time period to be blocked out for a single classroom without affecting all classrooms. In reality, the flexibility of saving a classroom for a certain time period for purposes like remodeling or conferences might be important to the client. A relatively easy amendment can be made to the program to allow such flexibility (see Assumption 3 in Appendix A2 for details).

Although it is often desired that sections under the same course be scheduled at different times so that students can have more choice, it is not required when a course has sufficiently enough section options. The current setting makes it a strict constraint that could result in courses like this systematically not being assigned, which is particularly problematic because these courses are usually core courses for multiple academic programs. Potential workarounds or adjustments to the constraint are discussed in Additional Limitation 3 in Appendix A2.

5.2 Recommendations

The proposed optimization tool has demonstrated excellent efficiency in the ability to process at least 90% of requests that the school receives in a semester and successfully schedule over 90% of them within 30 minutes. If this tool is adopted in Phase I to replace the current manual scheduling process, the amount of administrative effort required for both Shannon's team and the department coordinators can be significantly reduced as well as the process time needed. In addition, a moderate increase in the percentage of classes scheduled by the end of Phase I from 75% to 80% can be achieved.

The administrative personnels are not the only ones who can benefit from this tool. For the first time ever, a new approach is introduced into the classroom scheduling process that systematically accommodates the preferences and needs of the professors. With the help of the preference scores, the optimization receives a concrete objective to increase the faculty satisfaction and the level of satisfaction is now more quantifiable. Moreover, certain constraints are imposed to allow more flexibility for students to choose classes at their convenience and per their preference.

Considering all the potential gains, we strongly recommend that Shannon's Team considers the possibility of adopting the automated optimization tool in Phase I. We drafted a transition plan for the client to gradually move away from the current process and eventually fully adopt thetool in a few semesters ahead. The transition plan should include the following major steps:

1. Pilot the optimization tool alongside the current system for the first couple of semesters. Calculate key performance metrics for both systems to quantify the improvement and identify any issues unrecognized during the test stage. KPIs to be considered are: percentage of courses successfully scheduled by the end of Phase I, average professor preference score, percentage of professors who are assigned to preferred time slots, etc.

2. Implement the tool in a small scope in the initial trial. For example, at first it may not be possible to decide on the requirements of preference score inputs or figure out the logistics to pass the

survey to all professors, but the survey process can be tested within one department. With high adaptability, the tool can still run if only the preference scores of some professors are submitted.

3. The client should try to collect feedback from all parties involved in any aspect of the new process. For example, feedback from the department coordinators and professors can be used to evaluate if the preference survey is feasible to be performed every semester, what's the most efficient way to carry it out, and if it's not feasible, what alternatives can be proposed. One way to get student feedback is sending out surveys to students with the candidate schedule produced by the optimization tool attached and asking them which schedule they like better.

4. On the technical side, all kinds of sensitivity analysis should be performed to test stricter or looser versions of constraints as well as input. The parameters of the tool and the requirements of the input can be adjusted accordingly. For example, based on the results of sensitivity analysis and the feedback from professors, the client can adjust the availability to demand ratio and the preferred percentage that professor must satisfy when providing preference score input.

5. Potential ways to lift current limitations are discussed extensively in appendix A2. The client should try to gradually make these adjustments to unleash the full power of the tool.

Based on all the real time comparisons of the two systems and the feedback collected from all stakeholders, Shannon can decide whether or not to eventually launch the new system.

A1. Mathematical Formulation

Optimization Part I: Pair Up Half-Semester Classes

Data:

- I': set of first-half-semester sections
- J': set of second-half-semester sections
- s_{ij}: the number of eligible slots that work for both the professors of the two sections for each pair of sections $i \in I'$ and $j \in J'$. If sections i and j have different length or meet at different frequency, $s_{ij} = 0$.

Decision Variables:

- x'_{ij}: whether to pair up a first-half-semester section i with a second-half-semester section j (binary)

Objective and Constraints:

$$\begin{array}{rll}
\text{Maximize:} & \displaystyle\sum_{i \in I} \sum_{j \in J} s_{ij} x'_{ij} & \\
\text{subject to:} & & \\
\text{(Eligibility)} & x'_{ij} = 0 & \text{for every pair of sections } (i, j) \text{ such that } s_{ij} = 0 \\
\text{(No double pair)} & \displaystyle\sum_{i \in I'} x'_{ij} \leq 1 & \text{for each section } j \in J' \\
 & \displaystyle\sum_{j \in J'} x'_{ij} \leq 1 & \text{for each section } i \in I'
\end{array}$$

Optimization Part II: Classroom Scheduling

Data:

- I: set of sections
- J: set of times
- K: set of classroms
- a_{ij}: the preference score of the professor of section i towards time j, 2 means favorable, 1 means okay, and 0 means unavailable.
- S: set of section groups that belong to the same courses. For example, course DSO-570 have two sections 16298 and 16301 , therefore {16298, 16301} $\in S$.
- U: set of time slot js that do not overlap with any other time slots
- D: set of time slot pairs (j, j') that overlap with each other
- A: set of section-time pairs (i, j) such that either time slot j does not work for the professor of section i, or section i and time slot j have different length or meeting frequency
- H: set of section-classroom pairs (i, k) such that section i cannot fit in classroom k
- P: set of section groups that are taught by the same professor
- G: set of course groups that students usually take together. For example, DSO-530, DSO-570, DSO-562, DSO-553, and DSO-599 is a group of courses students usually take together, therefore {DSO-530, DSO-570, DSO-562, DSO-553, DSO-599}$\in G$.
- C_g: all the section combinations of course group $g \in G$. For example, course group 1 contains courses MKT-556 (section 16538) and MKT-566 (sections 16542, 16544, and 16546), therefore set $C_1 = \{(16538, 16542), (16538, 16544), (16538, 16546)\}$

Decision Variables:

- x_{ijk}: whether to put session i in classroom k at time j (binary)
- y_{combo}: whether any two sectinos in combination $combo \in C_g$ are assigned to the same time slot or to overlapping time slots (binary)

Objective and Constraints:

$$\text{Maximize:} \quad \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} a_{ij} x_{ijk}$$

subject to:		
(1.Same course conflict)	$\sum_{i \in secs} \sum_{j \in jpair} \sum_{k \in K} x_{ijk} \leq 1$	for each time pair $jpair \in D$ and each section group $secs \in S$
	$\sum_{i \in secs} \sum_{k \in K} x_{ijk} \leq 1$	for each time $j \in U$ and each $secs \in S$
(2.Same classroom conflict)	$\sum_{i \in I} \sum_{j \in jpair} x_{ijk} \leq 1$	for each classroom $k \in K$ and each $jpair \in D$
	$\sum_{i \in I} x_{ijk} \leq 1$	for each $j \in U$ and $k \in K$
(3.Ineligible slots)	$x_{ijk} = 0$	for each $k \in K$ and every section time pair $(i, j) \in A$
(4.Ineligible classrooms)	$x_{ijk} = 0$	for each time $j \in J$ and every section classroom pair $(i, k) \in H$
(5.Same section conflict)	$\sum_{j \in J} \sum_{k \in K} x_{ijk} \leq 1$	for each $i \in I$
(6.Same professor conflict)	$\sum_{i \in psecs} \sum_{j \in jpair} \sum_{k \in K} x_{ijk} \leq 1$	for each pair $jpair \in D$ and each section group $psecs \in P$
	$\sum_{i \in psecs} \sum_{k \in K} x_{ijk} \leq 1$	for each $j \in U$ and each $psecs \in P$
(7.Together course conflict)	$\sum_{i \in combo} \sum_{j \in jpair} \sum_{k \in K} x_{ijk} \leq 1 + \lvert combo \rvert y_{combo}$	for each $jpair \in D$, combination $combo \in C_g$, and group $g \in G$
	$\sum_{k \in K} \sum_{i \in combo} x_{ijk} \leq 1 + \lvert combo \rvert y_{combo}$	for each $j \in U$, $combo \in C_g$, $g \in G$
	$\sum_{combo \in C_g} y_{combo} \leq \lvert C_g \rvert - 1$	for each $g \in G$

A2. Discussion of Technical Details

At its core, the proposed classroom scheduling optimization tool harnesses the power of Mixed Integer Programing with a linear objective function and linear constraints. The tool is built using the gurobipy python library (a python API for the Gurobi Optimizer). When translating the business problem into precise mathematical formulation, certain decisions and assumptions are made to simplify the task so that 1) the problem fits in the scope of linear programming, 2) a feasible solution can be calculated within reasonable time and with reasonable computer resources, and 3) input data can be collected and prepared by the user within reasonable time and efforts. With this said, there are five major assumptions underlying the solution, as well as an important decision to combine half-semester sections before assigning classrooms. The reasoning behind the decision and each assumption, the resulting limitation, and potential ways to lift the limitation are discussed below.

1) Optimization Part I

Optimization Part I (pair up half-semester sections) is introduced to reduce the number of standard time slots in optimization Part II and scale down the task. Originally, every standard time slot has half semester and full semester versions. For example, a Monday 8am two-hour slot will have three variations for first-half semester, second-half semester, and full semester respectively. The advantage is that each section is allowed to try out all possible combinations of time and classroom. The disadvantage is that it takes too much time and resources to compute a solution.

As shown in the mathematical formulation of Part II (Appendix A1), the number of decision variables *x* is decided by the number of sections that need to be scheduled, times the number of classrooms available, times the number of time slots provided. As the number of time slots triples, the number of decision variables triples. More importantly, when half-semester slots are introduced, there are many more ways a time slot can overlap with another slot. As a result, the set of time slot pairs that overlap with each other (set *D*) grows exponentially and constraints that require iterating through each pair in the set increase rampantly (see constraints #1, #2, #6, and #7 Appendix A1). When tested with around 500 sections, which is less than what Marshall typically receives for a spring or fall semester, the module takes up a memory space much larger than what a personal computer would have and does not output a result after 7 hours. It is infeasible, or at least extremely inconvenient, for Shannon's team to deploy a tool like this.

2) Assumptions

Assumptions 1 & 2: Sections can only be held between 8:00 am and 9:30 pm Monday to Friday. They can meet either once a week, or twice a week on Mondays and Wednesdays or Tuesdays and Thursdays. Only 6 session types are allowed, which are full semester sessions 1 and 33, first half semester sections 411 and 431, and second half semester sections 415 and 442.

These assumptions restraint the complexity of the data that the user can input and reduce the number of cases the python module needs to consider when parsing the input file. With the limited time, we are not able to program the tool to take care of all possible slot types and session types that have occured in the history of Marshall, but have to focus on the most common types. Taking Spring 2019 as an example, there are in total 582 sections scheduled in that semester, none of which starts before 8am and only 11 (1.9%) ends after 9:30pm. The vast majority of sections meet either once a week during weekdays, or twice a week on Mondays and Wednesday or Tuesdays and Thursdays, with only 17 sections (2.9%) meeting on Saturdays or more than twice a week. Around 9.8% (57) of the sections are not of one of the six conventional session types. We believe that assumptions 1 and 2 cover around 90% of the common scenarios and are reasonable simplifications for a prototype solution.

Assumption 3: Available time slots are universal for all classrooms.

This assumption holds true for most of the time in real life because the school prioritizes the usage of classrooms towards academic classes and therefore is a reasonable simplification. However, it will be convenient for the user to allow a certain time period to be blocked out for a single classroom without affecting all classrooms. This feature can be realized by allowing the user to submit extra information through the input file that specifies all the blackout time for each classroom. The python module will then translate the blackout time into unavailable standard time slots and create a set of time-classroom pairs (*j*, *k*) such that time slot *j* cannot be used for classroom *k*. Let *B* denote this set, and the following constraint can be added to the Gurobi Optimizer.

$$x_{ijk} = 0 \qquad \text{for each session } \boldsymbol{i} \in \boldsymbol{I} \text{ and every time classroom pair } (j, k) \in \boldsymbol{B}$$

Assumption 4 & 5: Standard slots always start or end at whole hours and their lengths are always multiples of 30 minutes. Sections are allowed to be scheduled back to back without a break in the same classroom.

Historically, most classes start at whole hours or half hours. The most common lengths are 1 hour 20 minute2, 1 hour 50 minutes, and 2 hour 50 minutes and they together consist of 75.9% of all classes in Spring 2019. In rare cases, sections have lengths that are exact multiples of 30 minutes. Among the 582 sections scheduled in Spring 2019, two are one-and-a-half-hour long, one is two-hour long, and 48 are three-hour long. Except one three-hour section that is scheduled on Monday morning, all the others are scheduled at night and is the last class scheduled that day. It is possible that they are scheduled as the last class in the day because of their unusual length, but it is also possible that their lengths are recorded this way because they are the last classes in the day. For simplification, we assumed the second case and treated these special cases the same way as their slightly shorter counterparts.

If there is actual need for classes that last exactly for multiples of 30 minutes, two options can be considered. In the first option, the python module stays the same, but the user needs to tweak the length of these sections and create standard time slots of more length varieties to accommodate them. For example, to differentiate two-hour sections from one-hour-and-fifty-minute sections, the two-hour sections will be adjusted to two-hour-and-ten-minute long and thus will be scheduled to only slots longer than two-hour.

Another option requires modifying the python module and adding one more constraint to the optimization to make sure all the sections with a length that is exact multiples of 30 minutes can only be assigned to the last evening slot of the same duration in a day. The tool needs to first detect all the qualifying sections and slots and generate a set (let E denote the set) of section-time pairs (i, j) such that the section i cannot be scheduled at time slot j. Then the following constraint needs to be added:

$$x_{ijk} = 0 \qquad \text{for each classroom } k \in K \text{ and every session time pair } (i, j) \in E$$

Assumption 6: Each section only has one instructor. In the past, very few classes had multiple instructors (8.4% sections had a second instructor in Spring 2019). Having multiple instructors makes it harder for the user to prepare the professor preference input, but it does not challenge the foundation of the setup.

3) Additional Limitations

Additional limitations exist under the current setup of the optimization problem and the design of the python module. In general, choices are made to enable the timely delivery of a working tool that does the high-priority tasks, while leaving the low-priority requirements to future iterations. The following paragraphs examine the additional limitations.

Additional Limitation 1: The tool treats all sections the same, no matter whether they are core or elective, undergrad or graduate level. This can be improved in two potential ways. One approach is to assign different weights on different courses to prioritize some over others. Let w_i denote the weights of each section and the objective function can be rewritten as follows:

$$\text{Maximize:} \qquad \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} w_i a_{ij} x_{ijk}$$

Another approach is to add constraints that demand the successful scheduling of certain sections. Let M denote the set of sections that must be assigned, the following constraint will do the job:

$$\sum_{j \in J} \sum_{k \in K} x_{ijk} = 1 \qquad \text{for each section } i \in M$$

Be aware that adding such hard constraints may result in no feasible solutions, in which case none of the classes will be assigned.

Additional Limitation 2: The tool systematically fails to assign courses that have a lots of sections, potentially because of constraint #1 that requires sections under the same course to be scheduled at different and non-overlapping time slots.

Because of the time limit, we were unable to confirm the bonding constraint that caused this observation. So it is suggested to first perform a sensitivity analysis to identify the constraint or the interaction of constraints, that is causing the problem. Then the pinpointed constraint(s)can be gradually loosened to find the optimal balance. For example, instead of 1, the right hand side of constraint #1 can be made dynamic so that it eithers grows proportionally as the number of sections in the course grows, or changes to a larger value when the latter exceeds a threshold.

Additional Limitation 3: When preference score is not provided for a professor, the tool assumes that the professor is available at all time without any preference.

This default setting may be too generic. In future iterations, it can be adjusted to reflect the general teaching preferences towards different times in the day or days in the week.

Additional Limitation 4: The tool does very limited input validation or error handling and therefore is very fragile.

Currently, the tool highly relies on the user to submit input in the exact format as specified in the documentation. Any typo or invalid entries that are beyond the allowed range can either cause the program to crash, or even worse, produce incorrect output. For example, the module can only read two types of twice-a-week slot: MW or TH. The combination is hard coded to speed up the development of the tool, meaning that when creating the list of overlapping time slots, only these two combinations, if exist in the input, will be processed to generate conflicting pairs between themselves and their children slots (i.e. M and W slots are children slots of MW slots). If a user inputted MF slots, the module can still run, but will not prevent conflicting Monday slots and Monday-Friday slots being used at the same time.

In future iterations, any hard coded parameters should be replaced and all parameters should be made possible to be read through the input file for flexibility and scalability. The tool should also validate if the user input is legit so that the program works properly as intended. If any violation occurs, the program should stop and provide a proper error message for the user to debug, rather than either keep running or simply crash.