Scheduling Summer Math Courses

Linni Cai *, Yanmeng (Anny) Kong †

November 19, 2018

Abstract

Based on our interview with Sarah Garner, the director of the Math Department's Advising office, we were shown the threshold for opening a section, having at least 10 out of 35 student enrollments. The approach Mrs. Garner applied previously, was not very effective. With the limited resources, we will model the case and solve it with the linear programming, simulation and statistical analysis. Then we will be ready to provide appropriate schedules of sections for 9 non-major service courses based on historical data.

*Undergraduate student at University of Washington, cail4@uw.edu
†Undergraduate student at University of Washington, yk57@uw.edu

1 Problem description

Recently, the Math Department of the University of Washington found that during the summer quarters, the number of sections offered for math courses is not appropriate sometimes. Specifically, some sections were offered with fewer students than expected, while some others were offered with more students than expected.

Based on our interview with Sarah Garner, the director of the Math Department's Advising office, students' tuition during the summer quarter goes completely into the salary of the summer quarter instructors. Therefore there will be a threshold for opening a section, the student enrollment of the section, which is 10 students out of a 35 limit. In other words, if there are only 9 students in a section with 35 capacity, this section will be canceled by the Math Department because the cost of an instructor will not be capable to cover by tuition. Consequently, the instructor might be forced to teach another class section due to the contract in the spring quarter or might lose teaching chance during that quarter without expected salaries.

The director of the Math Department, Sarah Garner met the problem. The approach Mrs. Garner applied previously, was to make some guesses based on previous years' data. She would close a section once its enrollment goes under 10 students; she would open a section once its enrollment goes over max limit and would schedule it an hour after that. With the limited resources, she would like to seek a new optimization method to reduce such cases while still providing as many appropriate numbers of sections for 9 non-major service courses.

1.1 Goal:

For this project, our goal is to *find appropriate schedules of sections for the following 9 non-major math courses,*

MATH 111/120/124/125/126/307/308/309/324,

such that each section has at $57\%^1 - 80\%^2$ *enrollment for at least* $95\%^3$ *of the time based on historical data.*

1.2 Impacts:

As requested, we would like to find appropriate schedules for the 2019 math summer courses, including the number of sections and time slots of sections.

And as requested, we will only consider non-major courses:

MATH 111/120/124/125/126/307/308/309/324.

Mrs. Garner is currently planning on making a summer schedule for non-major courses. And previously, she made guesses based on previous quarters. We will help Mrs. Garner with the scheduling with our knowledge of linear programming and mathematical modeling to find one or more convincing schedules.

For the end of the project, we will be able to answer the following questions for Mrs. Garner:

- What's the recommended # sections and time slots for 9 non-major math courses? (*)
- If we want to open/close a section, which section would we recommend to open/close? (**)
- If we want to open/close a course, which course would we recommend to open/close? (***)

[1]The number 57% comes from 10/35 as 10 out of 35 enrollment is a requirement.
[2]The number 80% comes from a preferred range between $75\% - 80\%$ given by Mrs. Garner.
[3]The number 95% gives the confidence that most schedules will be within the preferred range.

2 Simplification

In order to translate the problem into something simpler so that we can do some mathematical modeling with the technique of linear programming, and then some analysis with statistical visualizations, we have the following simplifications.

2.1 Assumptions:

1. Focus on 9 non-major math courses.

   After a discussion with Mrs. Garner, we reached an agreement on focusing only on 9 non-major math courses: MATH 111/120/124/125/126/307/308/309/324.

2. No limitations caused by the supply of professors.

   We are only considering non-major math courses. According to what Mrs. Garner said, all math professors at UW will be able to teach any of the courses. Also, she mentioned, for the case where a section had to be canceled due to the shortage of professors, it will be very less likely to have any influence on our decisions since it had never happened before. Therefore, in this project, we assume there will be no impacts from the professors' side.

3. The enrollment of each math course will not affect each other.

   To be specific, for 124/125/126 or 307/308/309 series, we do not consider any enrollments' dependencies.

4. No influence from class locations.

   There will not be a lot of possible location choices available there and most of the math classrooms will be around the Math department based on Mrs. Garner's past experience. In general, there is a lot of construction work going on during the summer on UW campus, so there will be a really limited set of choices for Math classroom locations.

5. The distribution of the given enrollment is Poisson distribution

   We define the event is enrollment, for time slot k, some course i, we're predicting

the number of enrollment in year 2019. We use Poisson distribution to describe the probability of this event.

6. Scale all section capacity with 40

   Due various section capacities, we re-scale each section capacity with to 40, however, percentage enrollment will not be changed, $pe_{i,j,k}$, predicted enrollment is based on capacity of 40 with unchanged percentage enrollment.

2.2 Hard Constraints (Requirements):

1. There will be at least 1 section open for each of the 9 courses during the summer.

   Mrs. Garner told us that these are service courses where most of them are pre-requisites for students to get into majors, so the Math department wishes to be as supportive as it can and will provide all of them during the summer. In other words, each course will have ≥ 1 section(s).

2. All sections need at least 10 out of 35 enrollments.

   This constructs the threshold lower bound: $10/35 \approx 28.6\%$

3. All section time slots have to be assigned to one of the time slots in the following table, as required by the Office of the University Registrar:

8:30 am	9:40 am	10:50 am	12:00 am	1:10 pm	2:20 pm
3:30 pm	4:40 pm	5:50 pm	7:00 pm	8:00 pm	9:00 pm

4. All sections will have a duration of one hour exactly.

   During the summer quarter, all sections for the 9 courses are required to have a duration of one hour by the Office of the University Registrar.

5. All lecture sections will be scheduled at MWF.

   All TTH slots are for TA sections while MWF ones are for lecture sections.

2.3 Soft Constraints:

1. The average enrollment of students will be around: $20/35 \approx 57\%$.

   (This is the **lower bound** of our goal)

2. The preferred enrollment of students will be around: $75 - 80\%$.

   (This is the **upper bound** of our goal)

3. Mrs. Garner prefers more time varieties over repeated sections at the same time slot.

4. Mrs. Garner would like to appropriately maximize the # sections with limited resources.

3 Mathematical Model

With the $2008 - 2018$ summer course end-of-quarter schedules provided by Mrs. Garner, we will be able to analyze the raw data with some manipulations in Python and some visualizations in R.

To solve three questions(*)(**)(**) mentioned in our impacts, we have two stages:

1. Use linear programming to get the appropriate number of sections for each course at year 2019;

2. Use the Poisson simulation to find the specific time slot we want to select for each course at year 2019.

3.1 Stage I

With the technique of linear programming, we start by declaring some variables for the model. Then we will give our objective function and list our constraints.

3.1.1 Variables:

Symbols including sets, indices, variables and boolean functions required for the model are listed below.

Symbols	**Definitions**
Sets	
C	Set of Course numbers, where $C = \{111, 120, 124, 125, 126, 307, 308, 309, 324\}$.
Y	Set of years, where $Y = \{2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,$ $2016, 2017, 2018, 2019\}$.
T	Set of time slots, where $T =$ {8:30 am, 9:40 am, 10:50 am, 12:00 am, 1:10 pm, 2:20 pm, 3:30 pm, 4:40 pm, 5:50 pm, 7:00 pm, 8:00 pm, 9:00 pm}.
W	Set of weights, where $W = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 1.0, 1.2, 1.4, 1.6\}$, and $\sum_j w_j = 8.1$ for $j \in Y$.
W'	Set of weights for years, where $W' = \{0.1/8.1, 0.2/8.1, 0.3/8.1, ..., 1.0/8.1, 1.2/8.1, 1.4/8.1, 1.6/8.1\}$, and $\sum_j w'_j = 1$ for $j \in Y$.
Indices	
i	Course number, $i \in C$.
j	Year, $j \in Y$.
k	Time slot, $k \in T$.
Variables	
$s_{i,j,k}$	The # sections of course i in year j with time slot k.
$s'_{i,j,k}$	Appropriate # sections of course i in year j with time slot k.
$S_{i,j}$	Total # sections for course i at year j.
$S'_{i,j}$	Total # appropriate sections for course i in year j.
$S_{i,2019}$	Predicted total # sections for course i in year 2019.

Continued on next page

Table 1 – *Continued from previous page*

Symbols	**Definitions**
$S'_{i,2019}$	Predicted total # appropriate sections for course i in year 2019.
$p_{i,j,k}$	Percentage of enrollment of course i in year j with time k.
w_j	Weight of year j, sum to 8.1.
w'_j	Weight of year j, sum to 1.
Boolean functions	
$\texttt{selected}_{i,j,k}$ /$\texttt{selected}(i,j,k)$	1 if the section of course i in year j with time slot k is selected; 0 otherwise.
$\texttt{satisfy}_{i,j,k}$ /$\texttt{satisfy}(i,j,k)$	1 if $\texttt{selected}_{i,j,k}$ with the percentage of enrollment between 57% and 80% in year j; 0 otherwise.
$\texttt{overHigh}_{i,j,k}$ /$\texttt{overHigh}(i,j,k)$	1 if $\texttt{selected}_{i,j,k}$ with the percentage of enrollment between 80%(including) and 100%(not including) in year j ; 0 otherwise.
$\texttt{overHigh2}_{i,j,k}$ /$\texttt{overHigh2}(i,j,k)$	1 if $\texttt{selected}_{i,j,k}$ with the percentage of enrollment over 100%(including) in year j ; 0 otherwise.
$\texttt{below}_{i,j,k}$ /$\texttt{below}(i,j,k)$	1 if $\texttt{selected}_{i,j,k}$ with the percentage of enrollment between 29%(including) and 57%(not including) in year j ; 0 otherwise.
$\texttt{below2}_{i,j,k}$ /$\texttt{below2}(i,j,k)$	1 if $\texttt{selected}_{i,j,k}$ with the percentage of enrollment below 29%(not including) in year j ; 0 otherwise.
$\texttt{increment}_{i,j}$ /$\texttt{increment}(i,j)$	1 if one more section should be added to $S_{i,j}$ to construct appropriate number of sections, $S'_{i,j}$. 0 otherwise.
$\texttt{decrement}_{i,j}$ /$\texttt{decrement}(i,j)$	1 if one more section should be deducted from $S_{i,j}$ to construct appropriate number of sections, $S'_{i,j}$. 0 otherwise.
Supplementary variables	
$pe_(i,j,k)$	in year j, for course i, predicted enrollment of section slot k.

Continued on next page

Table 1 – *Continued from previous page*

Symbols	Definitions

3.1.2 Objective function:

- To maximize the total number of sections:

$$max \sum_{i} S_{i,2019} \tag{1}$$

3.1.3 Constraints

To find an appropriate number of sections for each course in 2019, we analyzed the previous data with tables and visualizations, and finally decided to use the following constraints.

1. In 2019, or each course i, the number of sections assigned has to be ≥ 1:

$$S'_{i,2019} \geq 1 \text{ for all } i \text{ (2.2.1)}. \tag{2}$$

2. In 2019, for each course i, for each time slot k, the percentage of enrollment has to be $\geq 29\%$:

$$p_{i,2019,k} \geq 29\% \text{ for all } i \in C, k \in T \text{ (2.2.2)}. \tag{3}$$

3. In 2019, for each course i, the total # sections should be less than the total # appropriate sections. (where "appropriate # sections" will be discussed below):

$$\sum_{i} S_{i,2019} \leq \sum_{i} S'_{i,2019} \text{ for all } i \in C \tag{4}$$

4. In 2019, for each course i, the total # sections should be less than the total # appropriate sections. (where "appropriate # sections" will be discussed below):

$$S'_{i,2019} \leq \sum_{j} w'_j \cdot S'_{i,j} \ \forall i \in C, j \in Y \tag{5}$$

5. In year j, for course i, the appropriate # sections should be have at most 1 section difference from the previous actual # sections. Therefore, we have

$$S'_{i,j} = S_{i,j} + increment_{i,j} \; or \; S_{i,j} \; or \; S_{i,j} + decrement_{i,j}$$

or in mathematical words:

$$|S'_{i,j} - S_{i,j}| \leq 1 \text{ for } i \in C, j \in Y. \tag{6}$$

6. For previous section whose percentage of enrollment between 80%(including) and 100%(not including),

   - for every two that kind of sections, add one section to the appropriate # sections:

$$\sum_{allk} s_{i,j,k} >= 2 \text{ s.t. } \texttt{overHigh}_{i,j,k} = 1 \implies \texttt{increment}(i,j) = 1$$

   Therefore, it gives

$$S'_{i,j} = S_{i,j} + \frac{\sum\limits_{k} s_{i,j,k}}{2} \tag{7}$$

   for $s_{i,j,k}$ being the number of sections with its percentage of enrollment within this bound.

7. For previous section whose percentage of enrollment between 29%(including) and 57%(not including),

   - for every two that kind of sections, reduce one section to the appropriate # sections:

$$\sum_{allk} s_{i,j,k} >= 2 \text{ s.t. } \texttt{below}_{i,j,k} = 1 \implies \texttt{decrement}(i,j) = 1$$

   Therefore, it gives

$$S'_{i,j} = S_{i,j} - \frac{\sum\limits_{k} s_{i,j,k}}{2} \tag{8}$$

for $s_{i,j,k}$ being the number of sections with its percentage of enrollment within this bound.

8. For previous section whose percentage of enrollment over 100%(including),

   - for every such section, add one section to the appropriate # sections:

   $$\sum_{allk} s_{i,j,k} >= 1 \text{ s.t. } \texttt{overHigh2}_{i,j,k} = 1 \implies \texttt{increment}(i,j) = 1$$

   Therefore, it gives

   $$S'_{i,j} = S_{i,j} + \sum_{k} s_{i,j,k} \tag{9}$$

   for $s_{i,j,k}$ being the number of sections with its percentage of enrollment within this bound.

9. For previous section whose percentage of enrollment below 29%(not including),

   - for every such section, reduce one section to the appropriate # sections:

   $$\sum_{allk} s_{i,j,k} >= 1 \text{ s.t. } \texttt{below2}_{i,j,k} = 1 \implies \texttt{decrement}(i,j) = 1$$

   Therefore, it gives

   $$S'_{i,j} = S_{i,j} - \sum_{k} s_{i,j,k} \tag{10}$$

   for $s_{i,j,k}$ being the number of sections with its percentage of enrollment within this bound.

10. Summarizing constraints 6-9, we finalize the appropriate number of sections as:

$$S'_{i,j} = S_{i,j} + \frac{\sum_{k} s_{i,j,k}^{80\% \le p_{i,j,k} < 100\%}}{2} - \frac{\sum_{k} s_{i,j,k}^{29\% \le p_{i,j,k} < 57\%}}{2} + \sum_{k} s_{i,j,k}^{p_{i,j,k} \ge 100\%} - \sum_{k} s_{i,j,k}^{80\% \ge p_{i,j,k} < 29\%} \tag{11}$$

   As we also satisfy constraint 5, we as well have the difference will not be greater than one. As we believe a reasonable adjustment of the previous schedules should not involve more than one sections to ensure a high level of confidence.

11. A special case to consider is when the previous quarter, for course i, only one section was provided. For the one section case, we will add one section to its appropriate # sections once it has one section satisfying the bound discussed in 6.

3.2 Stage II

we calculate weighted average as following:

- With fixed i and k

$$\frac{\sum_j pe_{i,j,k} * S_{i,j}}{\sum_j S_{i,j}} \tag{12}$$

For example:

- We have calculated the weighted average for MATH 111 in time slot 10:50 am.

$$\frac{pe_{i,k,j=2008} * S_{i,j=2008} + pe_{i,k,j=2009} * S_{i,j=2009} + ... + pe_{i,k,j=2018} * S_{i,j=2018}}{S_{i,j=2008} + S_{i,j=2009} + ... + S_{i,j=2018}} \tag{13}$$

4 Solution of the mathematical problem

4.1 Stage I

We are going to use Python for solving using the Linear Programming techniques. The algorithm follows the following steps.

4.1.1 Filter raw data

We wrote the following script for filtering raw data to remove redundant and unrelated information, and to get clean and concise data in csv format.

```
# Import time schedule in excel
import pandas as pd

df = pd.read_excel('project/MATHSummerEnrollments.xlsx',
    sheet_name='Time_Schedule_Information')

# Filter data
# 1) Filter Current Enrlmnt == 0
```

```
df = df[df['Current Enrlmnt'] != 0]
# print(df)

# 2) Filter Days of Week == ' T R  '
df = df[df['Days of Week'] == 'M W F ']
# print(df)

# 3) add Percentageof Enrollment
df['Percentageof Enrollment'] =
df["Current Enrlmnt"] * 100.0 / df["Limit/Est Enrlmnt"]
# print(df)

cols = df.columns.tolist()
n = int(cols.index('Percentageof Enrollment'))
n2 = int(cols.index('Current Enrlmnt'))
cols = cols[:10] + [cols[n]] + [cols[n2]] + cols[10+1:n-1]
df = df[cols]
# print(df)

cols = df.columns.tolist()
cols = [cols[1]] + cols[6:8] + [cols[11]]
+ [cols[10]] + [cols[13]] +[cols[18]]
df = df[cols]
df = df.sort_values(['Crs No', 'Yr'])
print(df)

with open("data_filtered.csv", "w+") as f:
    df.to_csv(f, sep=',', index=False)
```

4.1.2 Translate data into tables

With the data we filtered from the last step, we started to evaluate the percentages of the student enrollment of each section, i.e. the percentage of student enrollment in Math 111 section A in 2008. This gives us a clear layout of where do most percentages locate and what is the proportion located among 57% and 80%. Based on the information, we generalize the algorithm for solving the # sections in 2019 for each course.

We used Table 2 for recording all data for Math 111/120/124/125/126/307/308/309/324. (Tables with specific data for reference in Appendix)

Year	section A	section B	section C	...
2008				
2009				
2010				
2011				
2012				
2013				
2014				
2015				
2016				
2017				
2018				

Table 2: The percentages of the student enrollment of each section for Math 111

```
from tabulate import tabulate

for num in mydict.keys():
    secDictionary = mydict[num]
    line = []
    maxHeader = 1
    for year in sorted(secDictionary.keys()):
        row = []
        row.append(year)
        for enroll in secDictionary[year]:
            newEnroll = "%.2f" % float(enroll)
            if float(enroll) >= 57 and float(enroll) <= 80:
                newEnroll += "(\u2713)"
            else:
                newEnroll += "(\u2717)"
            row.append(newEnroll)
        line.append(row)
        if len(secDictionary[year]) > maxHeader:
            maxHeader = len(secDictionary[year])

    headers = []
    headers.append("year")
    for i in range(maxHeader):
        headers.append("Percent. Enrollment sec %d" % (i+1))

    print("Math " + num)
    print (tabulate(line, headers=headers))
    print()
```

4.1.3 Translate raw data into dictionaries

Constructing a dictionary of dictionaries, `mydict`, mapping from course -¿ year -¿ time -¿ percentage of enrollment.

For example, the dictionary `count` is a result of constructed dictionary using the following script.

```
# For num of sections
count = {}
# course -> year -> # sections

# num -> year -> time -> percentage of enrollment
for i in mydict2.keys():
    years = mydict2[i]
    # year -> time -> percentage of enrollment
    for y in years.keys():
        times = years[y]
```

```
        secCount = 0
        # time -> percentage of enrollment
        for t in times.keys():
            enrolls = times[t]
            secCount += len(enrolls)
        yCount = {}
        if i in count:
            yCount = count[i]
        yCount[y] = secCount
        count[i] = yCount
```

4.1.4 Compute linear programming I

By applying constraints, we calculated the appropriate number of sections for each course each year.

```
# please run only once
# reinitialize count again o/w
import math
s = {}
# course -> year -> appro. # sections

# count is dict
# course -> year -> # sections

# course i -> year j -> appropriate # sections
for i in count.keys():
    years = count[i]
    years2 = mydict[i]
    temp = {}
    for y in years:
        nSec = years[y]
        enrolls = years2[y]
        c2Up = 0
        c2Lo = 0
        c1Up = 0
        c1Lo = 0
        for enroll in enrolls:
            enroll = float(enroll)
            if enroll >= 80 and enroll < 100:
                if len(enrolls) == 1:
                    c1Up += 1
                else:
                    c2Up += 1
            elif enroll >= 29 and enroll < 57:
                c2Lo += 1
            elif enroll >= 100:
                c1Up += 1
            elif enroll < 29:
```

```
                c1Lo += 1
        if i in count:
            temp = count[i]
        temp[y] = nSec + 1 * float(c2Up) / 2.0
            - 1 * float(c2Lo) / 2.0 + 1 * c1Up - 1 * c1Lo
        temp[y] = math.ceil(temp[y])
        if temp[y] - nSec > 1:
            temp[y] = nSec + 1
        elif nSec - temp[y] > 1:
            temp[y] = nSec - 1
        if temp[y] < 1:
            temp[y] = 1
    s[i] = temp

for i in s:
    headers = ['Year', 'Appr. # sections']
    data = sorted([(k,v) for k,v in s[i].items()])
    print("Math" + i)
    print(tabulate(data, headers=headers))
    print()
```

4.1.5 Compute linear programming II

By adding weights and round down for the final result, the following script as well outputs the final # sections for each course in 2019.

```
# weight
weigh = [float(x)/10 for x in range(1, 7)]
weigh.extend([0.8, 1.0, 1.2, 1.4, 1.6])

w = [float(x) / 10 / sum(weigh) for x in range(1, 7)]
w.extend([0.8 /sum(weigh), 1.0 / sum(weigh),
    1.2 / sum(weigh), 1.4 / sum(weigh), 1.6 /sum(weigh)])

# s = {}
# course -> year -> appro. # sections

S = {}
# course -> appro.# sections in 2019

for i in s:
    years = s[i]
    appSec = 0
    j = 0
    for y in sorted(years):
        appYSec = years[y]
        appSec += float(w[j]) * float(appYSec)
        j += 1
    S[i] = "%.2f" % appSec
```

```
# Construct as table
headers = ['Couse No.', 'Appr. # sections', '2018 # sections']
data = sorted([[k,v + " (" + u"\u2248" + str(int(float(v)))
    + ")"] for k,v in S.items()])

added = False
j = 0
for i in y2018:
    for k in data:
        if not added:
            added = True
            data[j].append(y2018[i])
            j += 1
    added = False

# Output result in terminal as a table
print(tabulate(data, headers=headers, numalign="left"))
print()

# Output result as csv
import csv
headers2 = ['Course', '# Sections']
data2 = sorted([[k,str(int(float(v)))] for k,v in S.items()])
with open("table_csv.csv", "w") as f:
    f.write(tabulate(data2, headers=headers2, numalign="Left"))
```

4.2 Stage II

In this stage, we use Monte Carlo Simulation to obtain a large group of approximate data as our prediction of year 2019, from the weighted average of the given $pe_(i, j, k)$.

Then we're 95% confident that the population average is inside the interval with upper bound and lower bound. We use branch and bound to obtain the number of sections which satisfied some condition, such as above 80%, which is the upper bound of satisfied region; inside 57% 80%, which is the interval range of satisfied region; and below 57%, which is the lower bound of satisfied region.

4.2.1 Monte Carlo Simulations

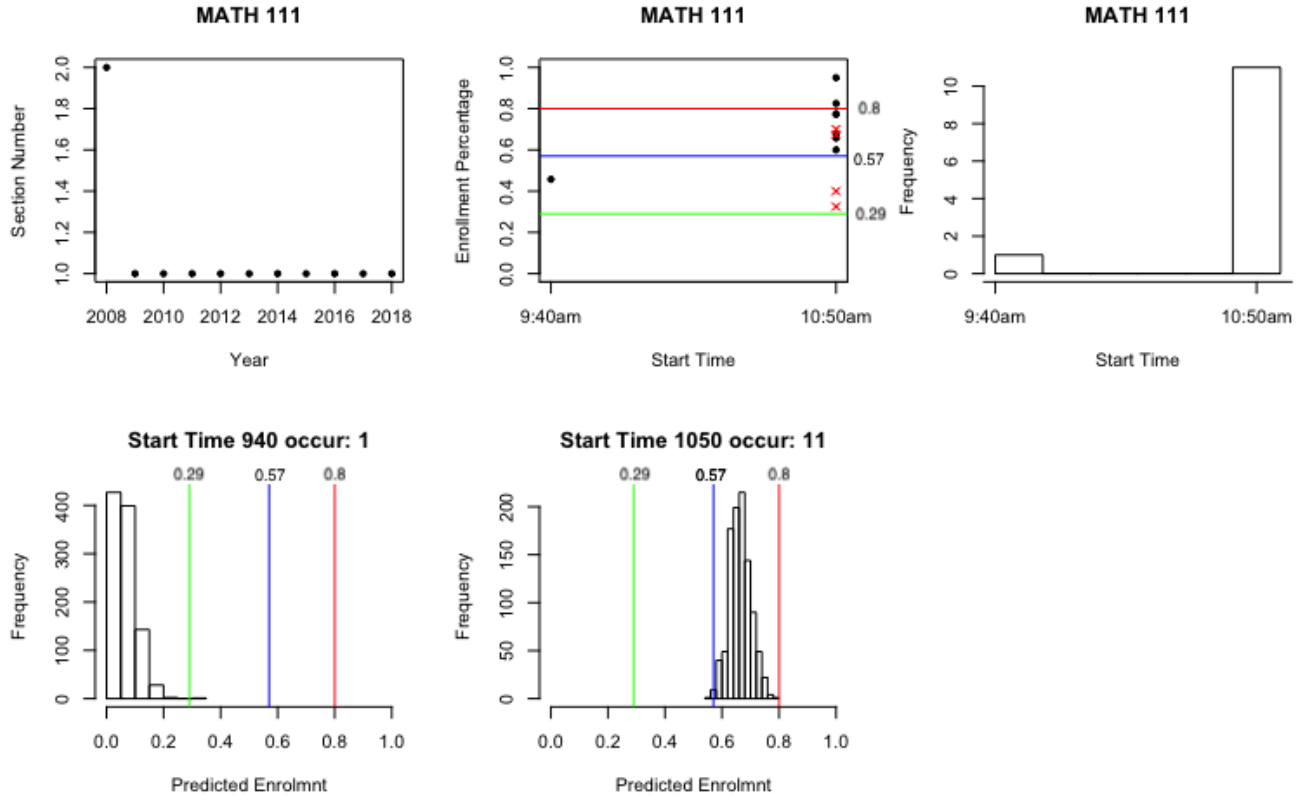
Recall Monte Carlo Simulations, people use it to simulation some process, based on some distribution, run and produce random variables to obtain a large group of samples, approximately population. Then put some condition and count the number of cases satisfy

the given condition and solve the real problem.

We assume this data have a Poisson distribution, then we use *rpois*, which is a built-in function of R, randomly produce some data of Poisson distribution, based on the given numbers n as the size of a group of random variables, λ as the number of occurrences of event. In our case, the size of a group of random variables is $S_{i,j}$, the number of occurrences of event is weighted average of $pe_{i,j,k}$. Then we run it for thousands of times, and obtain a large sample group, which is approximate population. We use this approximate population to predict year 2019 enrollment for each section. We create branch and bounds to count the number of sections satisfy some conditions. Then we set up 95% each confidence interval for Poisson distribution with corresponding count.

4.2.2 Priority Queue

After obtaining confidence interval with lower bound and upper bound, we use the idea of priority queue to obtain specific section time slot for each course. Recall priority queue is that a FIFO queue, priority queue has the most advantage section at the first place, and people pick the first object to optimize real problem. In our case, we sort all sections in descending order by comparing their percentage enrollment with various year weights and morning or afternoon weights. After sorting, we pick the most advantage section and re-scale the weight of this section, due to variety service offering. Since time slot 2:20 pm has good performance based on known data, we scale its weight up when there is no section time slot has higher lower bound of satisfied region. Here we define good is that the percentage enrollment is above 57%. For section time slot 2:20 pm, its percentage enrollment is stable above 80%, which is above the upper bound of satisfied region. Even though it is an afternoon section with lower afternoon weight compared to morning weight, its performance adds some weight and scale this section bounds upward. We sort again and repeat previous process until the number of picked sections are the number of approximate number of sections calculated by Stage I by Python programming. Finally, we output all the sections with corresponding number of the section.



4.2.3 Graph Explanation

To verify and provide visualization of our solution, we make graph for each course and its known section performance. For the first graph, we represent the distribution between year and the number of sections. The tendency shows that the number of sections is stable around 1 for a relatively long period. We might maintain or scale up weight on opening one section for MATH 111.

For the second graph, we represent the distribution between section time slot and percentage enrollment. The cross symbols represent recent four years' data, and concrete circle symbols represent year from 2008 to 2014. Combined the first two graphs, we can obtain a fact that the percentage enrollment of MATH 111 is decreasing recently. It also shows that there is low probability to open an additional section. Note that in this case, we cannot cancel the only one section, due to required university public service, the number of section for these nine courses is at least one.

For the third graph, we represent the histogram of section time slot with corresponding

frequencies. It clearly shows that time slot 10:50 am has more occurrences than 9:40 am, which means that its weight will be higher than 9:40 am.

For the fourth graph, we represent the histogram of simulated predicted enrollment for time slot 9:40 am with corresponding frequencies. Specifically, we can find that the most of data is below the satisfied region lower bound and cancellation bound, which means that in the calculation of priority queue, its weight will be lower. Since it breaks two soft constraint, one is below the satisfied lower bound, another is below the cancellation bound.

For the fifth graph, we represent the histogram of simulated predicted enrollment for time slot 10:50 am with corresponding frequencies. Compared to section time slot 9:40 am, this section has better performance, and its major data is inside the satisfied region. Our calculation of 95% confidence interval of its average will fall into satisfied region shows [0.9350971, 1.059828104], which means this section is more stable to choose instead of picking 9:40 am.

We don't show other sections' histogram with their frequencies, since we don't have known data to predict other sections which don't open before year 2019. However, we can generalize all the data and give predictions on these unknown sections. This part should be done in improvement.

5 Results

5.1 Stage I

By running the Python algorithm, we have the resulting # sections as below.

Course	# sections
111	1
120	2
124	3
125	4
126	6
307	6
308	6
309	3
324	4

Table 3: The number of sections for each course in 2019

In our interpretation, for course Math 111, we will schedule 1 section for 2019 summer; for course Math 124, we will schedule 3 sections for 2019 summer; for course Math 126, we will schedule 6 sections for 2019 summer. We have validated by manually reasoning with histograms and scatter plots. Since there are some special cases, our very first result was not accurate and output Math 120 as 1 because for recent 4 years, it only opened one section per year. Which means, our algorithm gives only 1.5 appropriate sections for three out of four years, resulting in a weighted average of slightly over 1. As this is a upper bound in our constraints, we could only take 1 for the # sections in 2019.

To deal with the problem, we added a special case to our algorithm to make sure it also works for the case when there was only one section open and to accurately calculate the appropriate number of sections for that year.

5.2 Stage II

After obtaining the number of appropriate number of sections for each math courses in year 2019, we create the following detailed time schedule for each math courses in year 2019. For MATH 111, this represent that we suggest to open one section at 10:50 am.

Time Slot	Number of Sections
8:30 am	0
9:40 am	0
10:50 am	1
12:00 am	0
1:10 pm	0
2:20 pm	0

Table 4: Section Schedule for Math 111 in 2019 (1 section)

Time Slot	Number of Sections
9:40 am	0
10:50 am	0
12:00 am	1
1:10 pm	0
2:20 pm	1

Table 5: Section Schedule for Math 120 in 2019 (2 sections)

For MATH 120, this represent that we suggest to open two sections, one at 12:00 am, one at 2:20 pm.

Time Slot	Number of Sections
9:40 am	1
10:50 am	1
12:00 am	1
1:10 pm	0
2:20 pm	0

Table 6: Section Schedule for Math 124 in 2019 (3 sections)

For MATH 124, this represent that we suggest to open three sections, one at 9:40 am, one at 10:50 am, one at 12:00 am.

Time Slot	Number of Sections
9:40 am	0
10:50 am	1
12:00 am	2
1:10 pm	0
2:20 pm	1

Table 7: Section Schedule for Math 125 in 2019 (4 sections)

For MATH 125, this represent that we suggest to open four sections, one at 10:50 am, two sections at 12:00 am, one at 2:20 pm.

Time Slot	Number of Sections
9:40 am	1
10:50 am	1
12:00 am	2
1:10 pm	1
2:20 pm	1

Table 8: Section Schedule for Math 126 in 2019 (6 sections)

For MATH 126, this represent that we suggest to open six sections, one at 8:30 am, one at 9:40 am, one at 10:50 am, two sections at 12:00 am, one at 1:10 pm, one at 2:20 pm.

Time Slot	Number of Sections
9:40 am	1
10:50 am	1
12:00 am	2
1:10 pm	1
2:20 pm	1

Table 9: Section Schedule for Math 307 in 2019 (6 sections)

For MATH 307, this represent that we suggest to open six sections, two at 12:00 am, otherwise one.

Time Slot	Number of Sections
9:40 am	1
10:50 am	2
12:00 am	2
1:10 pm	0
2:20 pm	1

Table 10: Section Schedule for Math 308 in 2019 (6 sections)

For MATH 308, this represent that we suggest to open six sections, two at 10:50 am, two at 12:00 am, one at 2:20 pm, one at 9:40 am.

Time Slot	Number of Sections
9:40 am	1
10:50 am	0
12:00 am	1
1:10 pm	0
2:20 pm	1

Table 11: Section Schedule for Math 309 in 2019 (3 sections)

For MATH 309, this represent that we suggest to open three sections, one at 9:40 am, one at 12:00 am, one at 2:20 pm.

Time Slot	Number of Sections
9:40 am	1
10:50 am	0
12:00 am	2
1:10 pm	1
2:20 pm	0

Table 12: Section Schedule for Math 324 in 2019 (4 sections)

For MATH 324, this represent that we suggest to open four sections, one at 9:40 am, two at 12:00 am, one at 1:10 pm.

6 Improvements

We made the assumption to ignore the effects by dependency graph below.

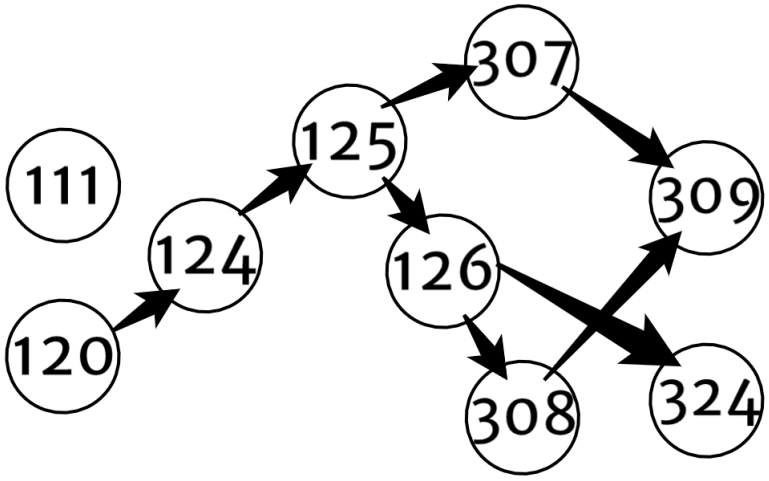


Figure : Dependency graph of 9 math courses

For future projects, we may take the effect by staff as well as the dependency graph into account. It would make the data to be more efficient and practical.

7 Conclusions

By analyzing constraints carefully and looking into the visualizations as well as specific data in tables, we come up some solutions for the summer schedules. In Stage I, we use Python to do linear programming and optimization to achieve the goal to maximize the total number of sections for each Math courses. In Stage II, we use Monte Carlo simulation to simulate data based on known data and Poisson distribution, then we set up confidence interval for several conditions, such as satisfied region and cancelled region. We build up various weights for different soft constraints, such as morning and afternoon weights, recent and previous years weights. After creating priority queue for section time slots, we use the number of appropriate sections for each course and obtain the corresponding number of each section time slots. The final solution is the combination of course, section time slots and corresponding number to open.

8 Acknowledgments

8.1 Our community contact:

Sarah Garner, Department of Mathematics, 206-543-0388, sterrs@uw.edu

References

[1] Cai, Lini, and Yanmeng (Anny) Kong. "MathSummer-Project/MathSummerScheduling." *GitHub*, 14 Nov. 2018, `github.com/MathSummerProject/MathSummerScheduling`.

[2] Cocchi, Guido, et al. "Scheduling the Italian National Volleyball Tournament." *Interfaces*, vol. 48, no. 3, 2018, pp. 271–284., doi:10.1287/inte.2017.0932.

[3] coin-or. "Coin-or/Pulp." *GitHub*, 24 Oct. 2018, `github.com/coin-or/pulp`.

[4] "COLLEGE OF ARTS SCIENCES MATHEMATICS." *University of Washington*, `www.washington.edu/students/crscat/math.html`.

[5] "Department of Mathematics." *Sarah Garner | Department of Mathematics | University of Washington*, `math.washington.edu/people/sarah-garner`.

[6] Garner, Sarah. "2008-2018 Math Summer Enrollments." `Https://Drive.google.com/File/d/1riRba01zYr4EtKn3hpP62hHETIFBDvdV/View?Usp=Sharing` 4 Nov. 2018.

[7] Kong, Yanmeng (Anny), et al. "Math Summer Course Scheduling Interview." `https://drive.google.com/file/d/1V98-hL3KeOjKKFHi52uePkMsEZ0vSAfI/view?usp=sharing` 31 Oct. 2018.

[8] "Summer Start Times." *Summer Start Times*, Office of the University Registrar, 29 July 2015, `registrar.washington.edu/classrooms/summer-start-times/`.

[9] "Summer Quarter." *Summer Quarter - University of Washington*, `www.summer.uw.edu/registration-costs/tuition-fees/`.

[10] Haight, Frank A. Handbook of the Poisson Distribution. Wiley, 1967.

[11] Hu, Xiao, et al. "Monte Carlo Simulation." Springer Handbook of Metrology and Testing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 1117–1157.

[12] "Integrating Monte Carlo Simulation, Momentum-Based Impact Modeling, and Restitution Data to Analyze Crash Severity." Edited by Nathan A. Rose and International Body Engineering Conference Exhibition Automotive Transportation Technology Congress -09 Paris, France, Integrating Monte Carlo Simulation, Momentum-Based Impact Modeling, and Restitution Data to Analyze Crash Severity, SAE International, 2001.

9 Appendix

For the following tables (Year v.s. sections),

- a check mark means it is within the appropriate range between 57 percent and 80 percent;
- the number within the bracket represents the time slot for sections.

Math 111

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2
2008	45.71(940)	60.00(1050)
2009	65.71(1050)	
2010	82.50(1050)	
2011	77.14(1050)	
2012	95.00(1050)	
2013	67.50(1050)	
2014	77.50(1050)	
2015	70.00(1050)	
2016	67.50(1050)	
2017	40.00(1050)	
2018	32.50(1050)	

Math 120

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3
2008	57.14(1050)	71.43(940)	77.14(1200)
2009	72.00(1050)	88.00(940)	104.00(1200)
2010	60.00(1200)	74.29(1050)	
2011	85.71(1050)	97.14(1200)	
2012	42.86(1050)	74.29(1200)	80.00(940)
2013	62.86(1050)	63.33(1200)	
2014	28.57(1050)	74.29(1200)	
2015	91.43(1200)		
2016	85.71(1200)		
2017	88.57(1200)		
2018	100.00(1200)		

Math 124

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3	Percent. Enrollment sec 4
2008	51.43(110)	62.86(940)	77.14(1200)	
2009	40.00(110)	62.86(940)	71.43(1200)	85.71(1050)
2010	57.14(1050)	71.43(940)	74.29(1200)	94.29(110)
2011	53.33(110)	70.00(940)	76.67(1050)	80.00(1200)
2012	77.14(110)	85.71(1050)	91.43(1200)	100.00(940)
2013	25.71(110)	54.29(1050)	57.14(1200)	71.43(940)
2014	34.29(940)	48.57(1050)	65.71(110)	88.57(1200)
2015	46.67(110)	73.33(940)	76.67(1050)	76.67(1200)
2016	80.00(1050)	85.71(110)	88.57(940)	
2017	45.71(110)	82.86(1050)	85.71(940)	
2018	51.43(1200)	57.14(940)	57.14(1050)	

Math 125

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3	Percent. Enrollment sec 4
2008	28.57(110)	40.00(940)	60.00(1050)	80.00(1200)
2009	68.57(110)	76.67(1050)	80.00(940)	96.00(1200)
2010	71.43(110)	82.86(940)	88.57(1050)	91.43(1200)
2011	66.67(110)	70.00(940) 96.67(940)	70.00(1200)	80.00(1050)
2012	51.43(940) 65.71(940)	77.14(110)	77.14(1050)	94.29(1200)
2013	60.00(940) 84.00(940)	76.67(1050)	80.00(110)	90.00(1200)
2014	71.43(110)	71.43(1200)	80.00(940)	91.43(1050)
2015	60.00(110)	63.33(1200)	83.33(1050)	97.14(940)
2016	22.86(110)	74.29(1200)	82.86(940)	86.67(1050)
2017	80.00(940)	94.29(1050)	94.29(1200)	
2018	82.86(1050)	91.43(940)	97.14(1200)	

Math 126

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3	Percent. Enrollment sec 4
2008	65.71(1200)	71.43(110)	80.00(940)	88.57(1050)
2009	71.11(110)	75.56(940)	84.44(1050)	86.67(1200)
2010	57.14(940) 84.44(940)	71.11(110)	75.56(1200)	82.22(1050)
2011	62.50(1050) 80.00(1050)	80.00(940) 95.00(940)	85.00(110)	95.00(1200)
2012	55.00(1050) 65.71(1050)	60.00(940) 92.50(940)	77.50(110)	90.00(1200)
2013	51.43(940) 77.50(940)	72.50(1050) 97.14(1050)	95.00(1200)	100.00(110)
2014	57.14(1050) 88.57(1050)	71.43(940) 96.67(940)	80.00(110)	97.14(1200)
2015	80.00(940) 93.33(940)	93.33(110)	93.33(1050) 96.67(1050)	93.33(1200)
2016	77.14(1050) 80.00(1050)	90.00(110)	92.50(1200)	97.50(940)
2017	27.50(1050) 92.50(1050)	55.00(110)	80.00(1200)	92.50(940)
2018	42.50(940) 65.00(940)	75.00(110)	75.00(1050)	92.50(1200)

Math 307

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3	Percent. Enrollment sec 4
2008	65.71(1200)	77.14(1050)	82.86(940)	
2009	76.67(940) 90.00(940)	83.33(1050)	86.67(1200)	
2010	72.00(940) 88.00(940)	74.29(1050)	82.86(1200)	
2011	74.29(1200)	74.29(940) 77.14(940)	77.14(1050)	85.71(110)
2012	51.43(940) 87.50(940)	82.86(110)	85.00(1050)	88.57(1200)
2013	77.14(1200)	77.14(940) 94.29(940)	80.00(1050)	82.86(110)
2014	68.57(940) 71.43(940)	88.57(110)	91.43(1050)	91.43(1200)
2015	56.67(940) 60.00(940)	83.33(110)	83.33(1050) 93.33(1050)	90.00(1200)
2016	37.14(940) 94.29(940)	77.14(1050)	80.00(110)	91.43(1200)
2017	45.71(1050) 91.43(1050)	57.14(940) 74.29(940)	82.86(110)	88.57(1200)
2018	45.71(1050) 91.43(1050)	68.57(940) 82.86(940)	77.14(110)	88.57(1200)

Math 308

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3	Percent. Enrollment sec 4	Percent. Enrollment sec 5
2008	37.14(110)	54.29(940)	62.86(220)	62.86(1050)	74.29(1200)
2009	59.26(110)	74.07(1200)	74.07(1050) 80.00(1050)	77.78(220)	88.89(940)
2010	54.29(1200)	65.71(940)	68.57(110)	74.29(220)	77.14(1050)
2011	60.00(1050) 86.67(1050)	70.00(940)	76.67(220)	76.67(110)	83.33(1200)
2012	51.43(940)	62.86(1050)	85.71(110)	95.00(1200)	
2013	50.00(1050) 91.43(1050)	60.00(940)	82.86(220)	94.29(110)	100.00(1200)
2014	45.71(940)	51.43(1050)	80.00(110)	85.71(220)	88.57(1200)
2015	50.00(1050) 70.00(1050)	73.33(940) 83.33(940)	90.00(1200)	93.33(220)	
2016	42.86(940) 71.43(940)	51.43(1050) 85.71(1050)	82.86(220)	92.50(1200)	
2017	40.00(1050) 80.00(1050)	45.71(940) 82.86(940)	94.29(220)	97.14(1200)	
2018	60.00(940) 80.00(940)	65.71(1050) 94.29(1050)	91.43(1200)	94.29(220)	

Math 309

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3
2008	68.57(940)		
2009	82.86(940)		
2010	57.14(1200)	60.00(940)	
2011	60.00(940)	65.71(1050)	94.29(1200)
2012	34.29(940)	65.71(1200) 71.43(1200)	80.00(1050)
2013	45.71(1200) 82.86(1200)	54.29(940)	80.00(1050)
2014	34.29(940)	48.57(1200) 68.57(1200)	71.43(1050)
2015	53.33(1200) 86.67(1200)	63.33(940)	
2016	40.00(940)	54.29(1050)	80.00(1200)
2017	37.14(1050)	62.86(940)	82.86(1200)
2018	28.57(1050)	74.29(940)	82.86(1200)

Math 324

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3	Percent. Enrollment sec 4
2008	34.29(1050)	34.29(110)	40.00(940)	71.43(1200)
2009	48.57(110)	60.00(1050)	65.71(940)	74.29(1200)
2010	54.29(940)	77.14(1050)	85.71(1200)	85.71(110)
2011	62.86(110)	71.43(940)	71.43(1050)	74.29(1200)
2012	57.14(940)	77.14(1200)	82.86(1050)	85.71(110)
2013	71.43(1050)	82.86(940)	85.71(1200)	94.29(110)
2014	63.33(110)	76.67(1200)	80.00(1050)	83.33(940)
2015	60.00(1050)	60.00(110)	70.00(940)	100.00(1200)
2016	22.86(1050)	65.71(940)	65.71(1200)	82.86(110)
2017	71.43(940)	71.43(1200)	82.86(110)	
2018	60.00(110)	65.71(1200)	82.86(940)	

10 Verification Statement

TODO verification message

11 Sharing Final Version

TODO sharing with Mrs. Garner