Teaching Scheduling for Utah Valley University in Computer Science Department

Dishank K. Inani

Department of Computer Science, Utah Valley University

CS 6150: Advanced Algorithms

Dr. Larry Zeng

October 07, 2023

Teaching Scheduling for Utah Valley University in Computer Science Department

Abstract

Teaching load allocation is a critical task in academic management and assigning the courses according to the faculty preference for time, their experience and expertise. The conventional manual allocation, reliant on MS Excel proves to be time consuming and prone to bias. This paper brings forth an approach using the Pulp library in Python. Primary data was gathered from lecturers about their preferences to teach for the Fall 2023 using the Google Forms and applied to courses offered in Fall. It was gathered in the excel sheet then transformed into a csv file. The data was then processed and optimized using the linear programming approach, adhering to the departmental policies.

Introduction

Teaching load allocation has been well recognized as a major contributing factor to the teaching quality (McClure and Wells, 1987; Partovi and Arinze, 1995; Shin and Jung, 2013, Qu and Xiaobo, 2014). Teaching load allocation, that is, which teaching staff teaches which subject, is an essential task that is carried out in all teaching and teaching/research institutions every year or every semester (McClure and Wells, 1985).

In teaching class allocation, three constraints should be kept in mind. First, all the subjects should be allocated. Second a teaching staff should be allocated a higher teaching load then what is appropriate for their designated role. Third, teaching staff should be assigned

subjects that align with their proficiency. This ensures they can effectively manage teaching tasks and schedules while also accommodating other activities, events, and responsibilities.

Literature Review

The ultimate goal of all educational endeavors is to improve learning. Although the existing literature proposes optimization models to adjust teaching load distribution, the main focus is on faculty preferences. For example, Breslav (1976) developed a linear programming solution focusing on faculty priorities, while Schniederjans and Kim (1987) introduced an objective programming model to increase departmental priorities in learning activities Badri (1996) used a two- stage multiobjective scheduling model We were established . In addition to good teaching, it is equally important to ensure that the work is evenly distributed according to the role of each academic staff member. Burgess (1996) conducted a comparative study of different methods of allocating workload to university students. Verdi (2009) examined the impact of different models of workload allocation on academic satisfaction and work life. Bentley and Kivic (2012) conducted a comprehensive cross-national study of academic performance. Furthermore, Horta et al. (2012) emphasized the importance of assessing learning environment along with learning effort. Several researchers have delved into the complex relationship between research and teaching for academic staff (e.g., Simons & Allen, 2007; Halsey et al., 2007; Taylor, 2007; Shin et al., 2011; Malcolm, 2013). Together, these studies highlight the important role of equity in increasing academic satisfaction. Unfortunately, the existing optimization models for teaching load allocation often fall short in adequately addressing the equity issue.

Data Sources

The teaching load is prepared by the director of the department before each semester. Several factors come into play like the number of teaching staff, the number of courses offered, the number of sections offered based on the student enrollment for that subject.

The director of the department must decide in such a way that courses distributed amongst the teaching staff are optimized with some measure of output criteria decided by the organization and bring satisfaction among the faculty and the organization. Usually, this assignment is done manually based on the teaching preferences but sometimes these preferences are not considered due to various constraints like more courses offered, lack of available teaching staff for course, and many more.

Since each lecturer has different expertise, preferences and experience towards a particular course. This data was gathered by circulating a google form which had three levels to be filled for the subject preference. There were 20 CS Faculties who participated in this activity and 81 courses to assign within them. The UVU department restricts 3 hours of contact hours for each course.

The faculties fill the google form with the details about the year of experience, their designation, their preferred courses in in order 1, 2 and 3 where preference 1 is the highest preferred course to teach.

Data Section

I have made a preference table in excel where the first column is the professor's name, next is year of experience, third column is the designation and the rest other columns are course numbers concatenated with M or T for classes on Monday and Wednesday or Tuesday and Thursday respectively accompanied by M for morning time, A for afternoon and E for evening classes. Say if there are two classes for Morning then it is given a number.

I have given 3 for 1st preference course, 2 for 2nd preferred course and 1 for 3rd preferred course. Then added 1 to that number if the class time is according to their preferred course. I have added 0.2 for each Lecturer/ Assistant Professor, added 0.4 for each associate professor and 0.6 for each Professor. Then added 0.001 for each year of experience they have. Each Lecturer/ Assistant must teach at 3 courses per semester. Each Associate Professor must teach at 4 courses per semester. Each Professor has to teach 5 courses per semester.

I : Index of all courses, i

J: Index of all Faculties, i

X_{ij} : The number of teaching hours for course i by the lecturer j

W_{ij} : The weight preference for course i, by the lecturer j

C_i : The number of classes for course i

t_i (=3): contact hours for course i

The Linear Programming model can be formulated as follows:

$$max \sum_{i=I} \sum_{j=J} (WijXij)$$

subject to:

$$\sum_{i=I} (Xij = 3)\ for\ j = 9$$

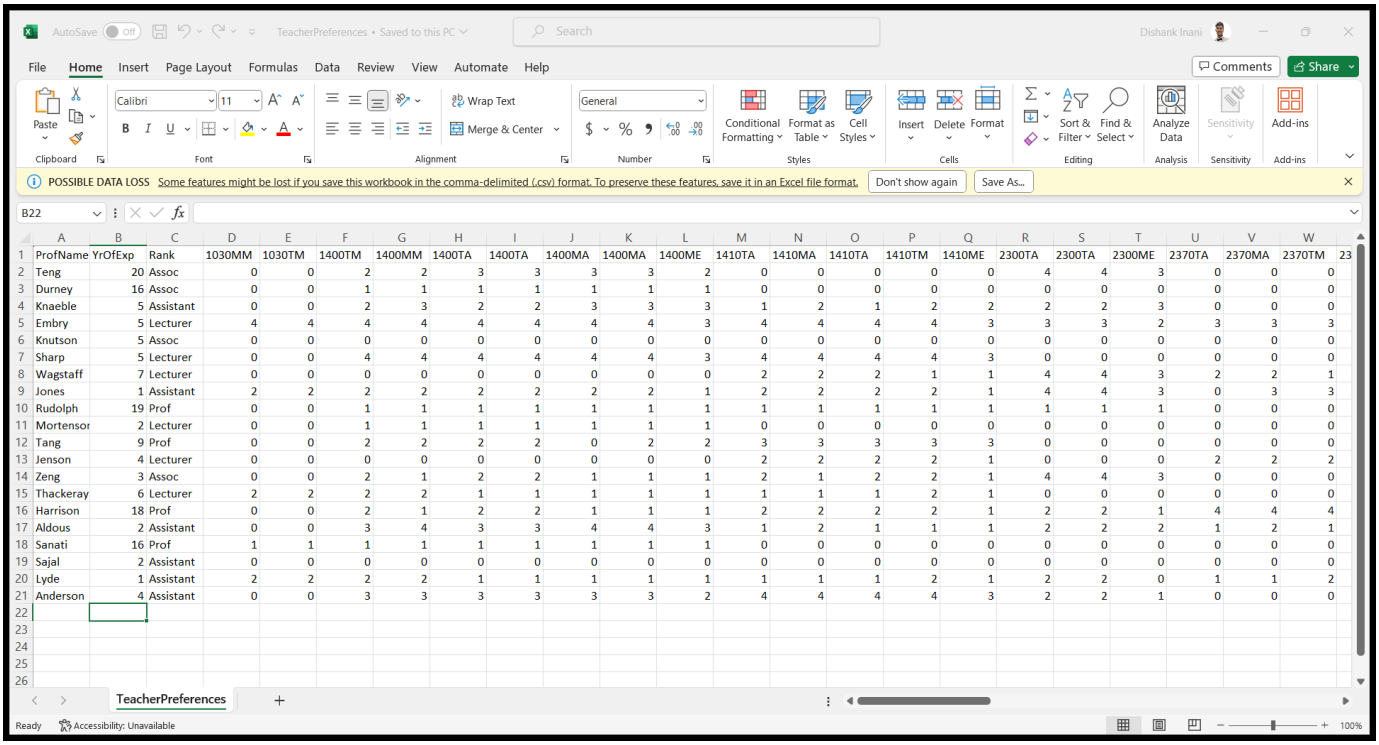
$$\sum_{i=I} (Xij = 4)\ for\ j = 12$$

$$\sum_{i=I} (Xij = 5)\ for\ j = 15$$

$$\sum_{j=J} (Xij = 1)\ for\ i = I$$

$$Xij \in 0, 1, for\ i \in I and\ j \in J$$

Preference Table:



ProfName	YrOfExp	Rank	1030MM	1030TM	1400TM	1400MM	1400TA	1400TA	1400MA	1400MA	1400ME	1410TA	1410MA	1410TA	1410TM	1410ME	2300TA	2300TA	2300ME	2370TA	2370MA	2370TM
Teng	20	Assoc	0	0	2	2	3	3	3	3	2	0	0	0	0	0	4	4	3	0	0	0
Durney	16	Assoc	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Knaeble	5	Assistant	0	0	2	3	2	2	3	3	3	1	2	1	2	2	2	2	3	0	0	0
Embry	5	Lecturer	4	4	4	4	4	4	4	4	3	4	4	4	4	3	3	3	2	3	3	3
Knutson	5	Assoc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sharp	5	Lecturer	0	0	4	4	4	4	4	4	3	4	4	4	4	3	0	0	0	0	0	0
Wagstaff	7	Lecturer	0	0	0	0	0	0	0	0	0	2	2	2	1	1	4	4	3	2	2	1
Jones	1	Assistant	2	2	2	2	2	2	2	2	1	2	2	2	2	1	4	4	3	0	3	3
Rudolph	19	Prof	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
Mortensor	2	Lecturer	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Tang	9	Prof	0	0	2	2	2	2	0	2	2	3	3	3	3	3	0	0	0	0	0	0
Jenson	4	Lecturer	0	0	0	0	0	0	0	0	0	2	2	2	2	1	0	0	0	2	2	2
Zeng	3	Assoc	0	0	2	1	2	2	1	1	1	2	1	2	2	1	4	4	3	0	0	0
Thackeray	6	Lecturer	2	2	2	2	1	1	1	1	1	1	1	1	2	1	0	0	0	0	0	0
Harrison	18	Prof	0	0	2	1	2	2	1	1	1	2	2	2	2	1	2	2	1	4	4	4
Aldous	2	Assistant	0	0	3	4	3	3	4	4	3	1	2	1	1	1	2	2	2	1	2	1
Sanati	16	Prof	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Sajal	2	Assistant	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lyde	1	Assistant	2	2	2	2	1	1	1	1	1	1	1	1	2	1	2	2	0	1	1	2
Anderson	4	Assistant	0	0	3	3	3	3	3	3	2	4	4	4	4	3	2	2	1	0	0	0

Algorithm

Input: Dataset (professors, years of experience, rank, and weights of preferences)

Output: Assignment of professors to courses

1. Calculate updated preference values for each professor.

2. Initialize Linear Programming problem (prob) for course assignment optimization.

3. Define Decision Variables:

   - Define x[i, j] representing professor i assigned to course j.

4. Define Objective Function:

   - Maximize the sum of preferences.

5. Define Constraints:

- For each professor i:

  - If rank[i] is 'Lecturer' or 'Assistant', set constraint on the number of courses <= 3.

  - If rank[i] is 'Assoc', set constraint on the number of courses <= 4.

  - If rank[i] is 'Prof', set constraint on the number of courses <= 5.

- Ensure each course is assigned to exactly one professor.

6. Solve the Linear Programming problem (prob).

7. Extract and Store Solution:

- Store the assignment of professors to courses.

8. Display Results:

- Print the assignments.

Results

The Pulp Library returns back the assignment of professor to the courses using the constraints passed in.

```
Professor Teng assigned to Course 2300TA
Professor Teng assigned to Course 2450TA
Professor Teng assigned to Course 2810TA
Professor Teng assigned to Course 3320MA
Professor Durney assigned to Course 3270TA
Professor Durney assigned to Course 339RTE
Professor Durney assigned to Course 3540MA
Professor Durney assigned to Course 3680TM
Professor Knaeble assigned to Course 2420MA
Professor Knaeble assigned to Course 3310MA
Professor Embry assigned to Course 1030MM
Professor Embry assigned to Course 1030TM
Professor Embry assigned to Course 1400TM
Professor Embry assigned to Course 1400MM
Professor Embry assigned to Course 1400TA.1
Professor Knutson assigned to Course 305GMA
Professor Knutson assigned to Course 305GTE
Professor Knutson assigned to Course 305GME
Professor Knutson assigned to Course 3450TA
Professor Sharp assigned to Course 1400TA
Professor Sharp assigned to Course 2550TM
Professor Sharp assigned to Course 2550ME
Professor Wagstaff assigned to Course 3380TA
Professor Wagstaff assigned to Course 4660MA
Professor Wagstaff assigned to Course 4690MA
Professor Jones assigned to Course 2420TM
Professor Jones assigned to Course 2420TA
Professor Jones assigned to Course 2810TM
Professor Rudolph assigned to Course 2600TE
Professor Rudolph assigned to Course 3250ME
Professor Rudolph assigned to Course 4470MA
Professor Rudolph assigned to Course 6150ME
Professor Rudolph assigned to Course 6470TE
Professor Mortenson assigned to Course 3310TM
Professor Mortenson assigned to Course 4380MA
Professor Mortenson assigned to Course 4380TA
Professor Tang assigned to Course 1410ME
Professor Tang assigned to Course 3060TM
Professor Tang assigned to Course 3060TE
Professor Tang assigned to Course 4230TE
Professor Jenson assigned to Course 2550TA
Professor Jenson assigned to Course 3410TA
Professor Jenson assigned to Course 3660MA
Professor Zeng assigned to Course 3240TA
Professor Thackeray assigned to Course 2450TM
Professor Thackeray assigned to Course 3260TA
Professor Thackeray assigned to Course 3450MA
Professor Thackeray assigned to Course 6300TA
Professor Harrison assigned to Course 2370TA
Professor Harrison assigned to Course 2370MA
Professor Harrison assigned to Course 2370TM
Professor Harrison assigned to Course 2450MA
Professor Harrison assigned to Course 3370TA
Professor Harrison assigned to Course 4400MA
Professor Sanati assigned to Course 3520TA
Professor Sanati assigned to Course 3520TA.1
Professor Sanati assigned to Course 3530ME
Professor Sanati assigned to Course 4700MA
```

Conclusion

The optimization and assignment techniques inculcated in this paper has proven to be effective and judicious way to assign the instructors to courses. This techniques can be used

by the director of other programs which will provide similar advantages and make the timetable management smooth and easy. To further refine their can be additional constraints be added such as allowing more classes for lecturer and assistant, adding time preference, adding extra teaching weight if he is a subject coordinator, and many more which could enhance the teaching assignment.

Appendix

```
import pandas as pd

pip install pulp

import pulp

df=pd.read_csv("TeacherPreferences.csv")

df.column

df.shape[1]

def calculate_updated_preference(row):
    years_of_experience = row['YrOfExp']
    rank = row['Rank']
    if rank == 'Assoc':
        increase_value = 0.4
    elif rank in ['Assistant', 'Lecturer']:
```

```
        increase_value = 0.2
    elif rank == 'Prof':
        increase_value = 0.6

    for col in df.columns[3:]:
        row[col] = row[col] + 0.001 * years_of_experience + increase_value

    return row

# Apply the function to each row
df = df.apply(calculate_updated_preference, axis=1)
df.head(15)
prob = pulp.LpProblem("Professor_Class_Assignment", pulp.LpMaximize)

# Define decision variables
num_faculty = len(df)
```

```
num_courses = len(df.columns) - 3  # Exclude the first 3 columns (ProfName, YrOfExp, Rank)


# Create a binary variable for each combination of professor and course
x = pulp.LpVariable.dicts("prof_course", ((i, j) for i in range(num_faculty) for j in range(num_courses)), cat='Binary')


# Define the objective function (maximize the sum of preferences)
objective = pulp.lpSum(df.iloc[i, j+3] * x[(i, j)] for i in range(num_faculty) for j in range(num_courses))
prob += objective


# Define constraints
for i in range(num_faculty):
    if df.iloc[i, 2] in ['Lecturer', 'Assistant']:
        prob += pulp.lpSum(x[(i, j)] for j in range(num_courses)) <= 3
    elif df.iloc[i, 2] == 'Assoc':
        prob += pulp.lpSum(x[(i, j)] for j in range(num_courses)) <= 4
```

```
    elif df.iloc[i, 2] == 'Prof':
        prob += pulp.lpSum(x[(i, j)] for j in range(num_courses)) <= 5

for j in range(num_courses):
    prob += pulp.lpSum(x[(i, j)] for i in range(num_faculty)) == 1

# Solve the problem
prob.solve()

# Extract the solution
solution = [[pulp.value(x[(i, j)]) for j in range(num_courses)] for i in range(num_faculty)]

# Print the results
for i in range(num_faculty):
    for j in range(num_courses):
        if solution[i][j] == 1:
            print(f"Professor {df.iloc[i, 0]} assigned to Course {df.columns[j+3]}")
```