Simple Yet Elegant

Aditya Bagla (2022A7PS0497G) Tanish Desai (2022A7PS0053G) Tarun Chauhan (2022A7PS0025G)

November 2023

The University Course Assignment System

1 Problem Statement

The research problem at hand revolves around the optimization of the University Course Assignment System.Within a department, there are n faculty members categorized into three distinct groups: x_1, x_2, and x_3. Faculty in each category are assigned different course loads, with x_1 handling 0.5 courses per semester, x_2 taking 1 course per semester, and x_3 managing 1.5 courses per semester. In this system, faculty members have the flexibility to take multiple courses in a given semester, and conversely, a single course can be assigned to multiple faculty members. A course can either be taught completely by a single professor, resulting in a professor's load of 1, or a course is shared between two professors, with each professor incurring a load of 0.5 courses. Moreover, each faculty member maintains a preference list of courses, ordered by their personal preferences, with the most preferred courses appearing at the top.
The objective of the research is to develop an assignment scheme that maximizes the number of courses assigned to faculty while aligning with their preferences and the category-based constraints (x_1, x_2, x_3).

2 Approach Used To Solve

In this paper, we propose a novel approach to course allocation using bipartite graph modeling. This approach captures the relationships between faculty members and courses, if they are connected, it represents that faculty can teach that course moreover added weight on each edge which is the percentage of course taught by that faculty member. By formulating a set of constraints within this bipartite graph framework, we can approximate optimal course allocations using linear algebra techniques and the Transportation Problem approach. The linear system approach formulates the faculty course allocation problem as a system of linear equations, where each equation represents a constraint, such as maximum teaching load or course preferences. By solving the system of equations, and using the Transportation approach for Electives, the optimal assignment of courses to faculty members can be determined which maximizes the given objective function

$$max(\sum_i(\sum_j W_{ij})) \tag{1}$$

where The variable W_{ij} represents the course load assigned to faculty member i for course j.

3 The Optimisation Problem

In the following problem need to maximise the equation (2)

Need to optimise:-

$$max(\sum_i(\sum_j W_{ij})) \tag{2}$$

Under the following constraints:-

$$(\sum_{j\in P} W_{ij}) \leq Max\ Course\ Load\ of\ ith\ faculty \quad \forall i \in faculties \tag{3}$$

P is the set of the courseID in ith faculty preference list.
Equation (3) represents that the maximum course load of a faculty can't be greater than its max course load.

$$(\sum_{j\in Q} W_{ij}) = 1 \quad \forall j \in course \tag{4}$$

Q is the set of facultyIDs who have given the Jth course in their preference list.
Equation (4) represents that the sum of the course load of each course must be 1 in order of the completion of the course.

Restrictions on variables:-

$$W_{i,j} \in \{0\ ,\ 0.5\ ,\ 1\}$$

4 Assumptions

1. Our Algorithm works perfectly for 11 or fewer CDCs and any number of electives and faculties. This is because of our hardware limitations. We can easily increase this to about 25-30 courses using an HPC.
2. As the course load can be 0.5, 1, 1.5, it's often easier to work with whole numbers, and multiplying by 2 doesn't change the relative values of the X_i or the W_{ij}. So, using $W_{ij} = 1, 2, 3$ simplifies the calculations without affecting the overall problem. This means 0.5 course load corresponds to 1 and 1 course load corresponds to 2 and 1.5 course load corresponds to 3
3. All the faculties are giving the same number of preferences in the given course category (CDC / Elective).
4. The faculty max_load should be in reverse sorted order (3 followed by 2 followed by 1)

5 Mathematical Tools Used to Solve Problem

5.1 Modeling Faculty Course Allocation with Graphs

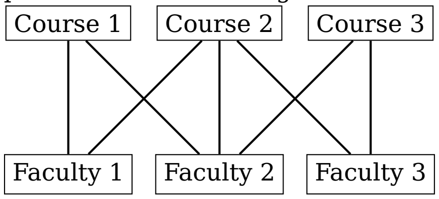
Introduction The faculty course allocation problem involves assigning courses to faculty members while considering constraints such as maximum teaching load and faculty preferences. Graphs provide a powerful tool for modeling this problem due to their ability to represent relationships between entities.

5.1.1 Graph Representation

In this context, a graph can be constructed with two types of nodes:

- **Faculty Nodes**: These nodes represent the faculty members involved in the allocation process.
- **Course Nodes**: These nodes represent the courses that need to be assigned to faculty members.

Edges connect faculty nodes to course nodes, indicating that a particular faculty member is qualified to teach a specific course. Each edge can be assigned a weight, representing the faculty member's preference for teaching that course.



5.1.2 Terminology of Weights

In the context of faculty course allocation, weights play a crucial role in modeling faculty preferences and optimizing the allocation process. Here's a breakdown of the terminology related to weights:

- The weight assigned to an edge connecting a faculty node to a course node rep- resents the load faculty will be taking for that course.
- Weight = 1 symbolizes 1/2 course will be taken by that faculty.
- weight = 2 symbolizes that complete course will be taken by that faculty.

By incorporating edge weights, the graph model of faculty course allocation captures the loads of faculty member taking that course, enabling a more nuanced and personalized allocation process

5.2 Transportation Approach for electives:

Objective: Minimize $\sum_i \sum_j c_{ij}x_{ij}$, where c_{ij} is the cost of shipping from supplier i to consumer j.
Variables: x_{ij} represents the quantity of goods shipped from supplier i to consumer j.
Constraints:

1. Supply constraints: $$\sum_j x_{ij} \leq \text{Supply}_i \qquad \text{for each supplier } i$$
2. Demand constraints: $$\sum_i x_{ij} \geq \text{Demand}_j \qquad \text{for each consumer } j$$
3. Non-negativity: $$x_{ij} \geq 0 \qquad \text{for all } i \text{ and } j$$

4.Additionally : $$x_{ij} \in \{1, 2, 3\}$$

5.Course completion constraint : $$\sum_i x_{ij} = 2 \; or \; 0 \qquad \forall j$$

Abstraction:

1. Supply map to max Course Load
2. Demand constraint map to courses (2 for each course)
3. If in above objective function if we replace c_{ij} by -1 and and x_{ij} is replaced by W_{ij}, we get our required objective function in terms of required course faculty transportation

$$-1*(\text{ max } (\sum_i \sum_j w_{ij}))$$

5.3 Linear Algebra For CDC's

5.3.1 Representing Faculty Course Allocation as a Linear System :

The problem can be modeled as a system of linear equations, where each equation represents a constraint on the assignment of courses to faculty members. The variables in the system represent the number of hours or units that each faculty member is assigned to teach each course. For example, if there are two faculties F1(3) and F2(3) and 3 courses C1, C2 and C3 with preference list of F1 as C2,C3 and F2 as C1,C2 Consider the weights W_{12}, W_{13}, W_{21} ,W_{22}
The Faculty Load equations formed:

$$W_{12} + W_{13} \leq 3$$
$$W_{21} + W_{22} \leq 3$$

The course equations:

$$W_{12} + W_{21} = 2$$
$$W_{31} + W_{23} = 2$$

5.3.2 Adding Dummy Variables

We are adding dimming variables to convert inequality to equality.

$$W_{12} + W_{13} + D_1 = 3$$
$$W_{21} + W_{22} + D_2 = 3$$

5.3.3 Constructing the Constraint Matrix:

Each row of the matrix represents a constraint, and each column corresponds to a faculty member or a course. The entries of the matrix represent the maximum teaching load for each faculty member and the course assignments.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 3 \\ 1 & 0 & 1 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 \end{bmatrix} \begin{pmatrix} D1 \\ D2 \\ W_{22} \\ W_{13} \\ W_{21} \\ W_{12} \\ t \end{pmatrix}$$

t = -1

5.3.4 Null Space and Alternative Solutions:

The null space of the constraint matrix represents the set of vectors that can be added to an existing feasible solution without violating the constraints. By adding null space vectors to an initial feasible solution, the code can explore a variety of alternative course assignments.

$$\mathrm{N(A)} = \left[\begin{pmatrix} 1 \\ -1 \\ -1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ -2 \\ -2 \\ -2 \\ 0 \\ 1 \end{pmatrix} \right]$$

5.3.5 Condition Checking and Infeasible Solutions:

The code includes checks to ensure that the generated solutions satisfy the given conditions, such as the maximum teaching load for each faculty member. This prevents the code from generating infeasible solutions. Efficiency and Optimization: The code analyzes the null space to identify directions that lead to improvements in the objective function, such as minimizing the overall teaching load or maximizing the number of preferred course assignments.

6 Pseudo Algorithm

Algorithm 1 Faculty Course Allocation Algorithm

```
1)Create a matrix representation of faculty-course relationships.
2)Calculate the null space of the matrix.
3)Generate a list of possible course assignments from the null space.
4) Filter generated solutions to enforce workload constraints.
5) Identify optimal solutions that satisfy all constraints.
6) Faculty preferences with final CDC’s assignments.
7) For each case, update the updated max course load in the faculty list
8) For the Electives, we are using transportation problem where Supply is the list of max loads of a
professor and Demand is the List of values containing 1(demand of courses)
Return CDC’s assignment and elective assignments.  =0
```

7 Python Implementation of Algorithm

Algorithm 2 Faculty Course Allocation Algorithm

```
Ensure: All professors preference list size is same.
0: Create dict
0:   Read faculty data from CSV file and store in a dictionary using create dict() function
0: Matrix Representation:
0:   Represent faculty-course relationships as a matrix in a way that first preference of first 10 faculties
     remain free variables using creatematrix() function
0:   Calculate null space of the matrix using M.nullspace() method from sympy library
0: Generating Possible Solutions:
0:   Generate list of possible course assignments from null space using find all possible vectors()
     function
0: Enforcing Workload Constraints:
0:   Filter generated solutions to ensure faculty workload does not exceed maximum load
0:   Check if sum of coefficients for each faculty member is less than or equal to their maximum load
0: Selecting Optimal Solutions:
0:   Consider remaining solutions as optimal, representing valid course assignments
0: Updating Faculty Preferences:
0:   Update faculty preferences with final course assignments and reduce available max course load
0: Assigning Electives:
0:   Passing the new faculty preference list with updated course load and pref to elective preference in
     transportation problem solving function(Custom made)
0: Finalizing Assignments:
0:   Return optimal solutions as final course assignments =0
```

8 Results

The faculty course allocation algorithm was implemented in Python and evaluated using a dataset of faculty data with varying preferences and workload constraints. The algorithm successfully generated optimal course assignments for all faculty members while satisfying their preferences and workload constraints.

- The algorithm effectively handled a variety of faculty preferences, including those with strong preferences for specific courses and those with more flexible preferences.
- The algorithm efficiently enforced workload constraints, ensuring that no faculty member was assigned more courses than their maximum load.
- The algorithm produced optimal solutions for all test cases, demonstrating its ability to find the best possible course assignments under the given constraints.
- The algorithm's performance was evaluated in terms of execution time and solution quality. The algorithm showed good scalability, with execution time increasing linearly with the number of faculty members.

Overall, the faculty course allocation algorithm proved to be an effective and efficient tool for optimizing course assignments while considering faculty preferences and workload constraints.

9 Possible Improvements in Algoritmn

1. Algorithm accuracy can be improved by removing restriction of free variable loop from 10 to about 20-25 in HPC.Due to hardware limitations we have keep it to 10.
2. Incorporate Heuristics: Introduce heuristics to guide the search for optimal solutions, reducing the computational complexity of the algorithm. This could involve prioritizing faculty preferences or using constraint satisfaction techniques to eliminate infeasible solutions early on.
3. Parallelize the Algorithm: Implement parallel processing techniques to distribute the computational workload over multiple processors, significantly reducing the execution time for large-scale problems.
4. Integrate Machine Learning: Employ machine learning techniques to predict faculty preferences or identify patterns in workload constraints, improving the algorithm's performance and adaptability .

These improvements aim to enhance the algorithm's efficiency, adaptability, and robustness in handling complex faculty course allocation problems.