An Application of Graph Optimization-Course Project Report

Anwesh Das (2022A7PS1127G) ,
Prakhar Kumar Gupta (2022A7PS1145G) ,
Yash Agarwal (2022A7PS1104G)

30 November 2022

Abstract

In this paper we attempt to understand and put out a possible approach to a research problem that revolves around the optimization of a University Course Assignment System. This report recaps the problem presented , the challenges faced with different methods tried and refused and a feasible solution.

1 Introduction

The research problem at hand revolves around the optimization of the University Course Assignment System. Within a department, there are ”n” faculty members categorised into three distinct groups: “x1”, “x2”, and “x3”. Faculty in each category are assigned different course loads, with ”x1” handling 0.5 courses per semester, “x2” taking 1 course per semester, and ”x3” 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. When a course is shared between two professors, each professor's load is considered to be 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. Importantly, there is no prioritisation among faculty members within the same category. The primary objective of this problem is to develop an assignment scheme that maximises the number of courses offered. While aligning with the faculty's preferences and the category-based constraints (“x1,” “x2,” “x3”). The challenge lies in ensuring that a course can only be assigned to a faculty member if it is present in their preference list.

This problem is unique due to the flexibility it offers regarding the number of courses faculty members can take, distinct from typical assignment problem. Potential modifications may include adjusting the maximum number of courses “y” for each category of professors, instead of requiring exact adherence, or extending the number of professor categories beyond the existing three to devise a more generalised solution.

2 Initial Thoughts

We realise that this is an assignment problem where we need to do a matching between the two distinct sets:- courses and professors in which every edge either connects a vertex from courses to a vertex of Professors set or the other way around. Here the edges of the matching would signify the possible courses a professor can take in agreement to his preferred list of courses. Bipartite Graphs are well-suited for the task at hand. We then look for appropriate algorithms to maximize matching. A significant design choice was made to align Courses with Professors instead of the other way around. This decision was driven by the goal to maximize the assignment of courses to professors, thereby ensuring the maximum number of courses can be offered.

3 Max Bipartite Matching

We guessed after surfing through a number of Bipartite Matching graph algorithms that the Maximum Bipartite matching might be worth a shot. A maximum matching is a matching of maximum size (maximum number of edges). In a maximum matching, if any edge is added to it, it is no longer a matching. There can be more than one maximum matching for a given Bipartite Graph.

Maximum Bipartite Matching (MBP) problem can be solved by converting it into a flow network. We use Ford-Fulkerson algorithm to find the maximum flow in the flow network. But we see a very big issue with this model as we it uses non weighted edges in the matching which means that we don't do the matching in accordance to the priority list of the professors

4 Hungarian

A more promising alternate seems to be the Hungarian Algorithm which uses weighted edges in matching. The only sensible way this algorithm can implemented is by making two sets of courses and professors and running three cycles of allotting 0.5 of a course to its most suitable professor and removing a course when it has been allotted in two cycles and removing a professor when he has reached limit of 0.5 , 1 or 1.5 courses. However a few grave problems arise from this arrangement like:-

- After the assignment there could be instances where a course is allotted a professor with only 0.5 of the course load and the other 0.5 of the load is not assigned.
- Professors of class X2 and X3 may be allotted course loads of 0.5 and 0.5/1 only.
- No distinction between the different types of courses i.e. electives may be allotted before CDCs have a chance, thus resulting in a CDC not being offered even when it could have.

5 Our Solution

Our algorithm can be viewed as a tailored version of the Hungarian method, specifically adapted to address our unique problem. We construct a master map where courses serve as keys. Each course key is associated with a vector of professors, which includes every professor who has listed the course in their preference list, arranged according to their interest level in the course. Furthermore, the courses in the map are organized in descending order of their importance to the University, starting with FDCDCs, followed by HDCDCs, FDELs, and finally HDELs.

To understand what our algorithm does we take a small test case in which we have 3 professors (A,B,C) and three courses(1,2,3) where 1 is the most important course for the University and 3 is the least important.

Prof	Course Load
A	1
B	1.5
C	0.5

Table 1: Shows the max course load for a professor.

Prof	Course Priority
A	2
B	2 , 1 , 3
C	1 , 3

Table 2: Shows the course preferences for a professor.

Course	Professors in prospect
1	C , B
2	B , A
3	C , B

Table 3: Shows the master map

Our algorithm works by the help of roughly two functions:-assign and recursive. These functions work in tandem to optimize the course assignment process. The assign function traverses down the master map as shown in Table 3 and assigns the full course load of a given course 0.5 at a time to professor who is most interested in the course and hasn't fulfilled his load limit. If in case this is not possible, we call the recursive function to free the load of the most interested professor in the course (say x) by freeing up his load in another course (say

y) giving the load 0.5 at a time to another professor who is interested in the course y. If no such "another" professor can take the load, we call the recursive function on this professor. The cycle goes until the load can be shifted and course x can be allotted.

In the example in Table 1 and Table 2, we make the corresponding master map i.e. Table 3 and assign the professors to course 1 and 2 but as we move to course 3, we see that both C and B have exhausted their limit and we need to call the recursive function. We intend to transfer the 0.5 load of course 1 to B from C. But as B is exhausted we go to course 2 and transfer the 0.5 course load of B to A. We go back to transfer the 0.5 load of 1 from C to B. Now C is free and able to take the course 3. Thus we obtain the Table 4 as our final assignment solution.

Course	Professors assigned
1	B
2	A
3	B , C

Table 4: Shows the most optimal assignment

6 Major obstacles overcome by this algorithm

1. The foundation of this algorithm is that we give priority to allot the CDC courses over the Elective courses and make sure that the maximum number of courses that are mathematically possible are offered. We ensure that in the master map every course before the course we are currently assigning is either assigned and can be offered or cannot be mathematically offered.

2. We take care of the course priorities provided by the professors and due to the self correcting nature of the code, we give out the most optimal solution.

3. We have attempted to produce a more generalized solution as our algorithm is not affected by the number of classes in which we classify our professors i .e. x1,x2,x3, etc. If a professor is willing we can assign him/her any arbitrary course load as long it is a multiple of 0.5.