

## **DISCO Project Report**

### **Team Members:**

Tanishq Jain - 2021B3A72941G

Tanishka Sugandhi - 2021B3A72143G

Hursh Sethiya - 2021B3A71878G

### **Summary Of Project Topic:**

The research focuses on optimizing the University Course Assignment System, accommodating three faculty categories with different course loads. Faculty members can share courses, and each maintains a preference list. The goal is to develop an assignment scheme that maximizes course assignments while adhering to preferences and category-based constraints.

### **Our Approach:**

Our team went through many articles to understand the famous allocation problem. We found many solutions like the famous Hungarian Algorithm, which were suitable for the problem but we decided to follow the unconventional way of developing our algorithm and coding our approach ourselves to deal with the problem at hand.

Following are the **Rules/Constraints** that have been kept in mind throughout the procedure:

1. A hierarchical order among the different categories of professors was established. The order is as follows:

$x_2 > x_3 > x_1$

Elaborating further, this order means that the preference orders of  $x_2$  professors will first be satisfied, followed by  $x_3$  professors, and then at last, the  $x_1$  professors. Further the courses selected by  $x_1$  category of professors are shared with  $x_3$  category of professors randomly.

2. Courses can only be shared by  $x_1$  and  $x_3$  categories of professors. Thus, splitting the load of 1 in 0.5 & 0.5 is not allowed for  $x_2$  category of professors.

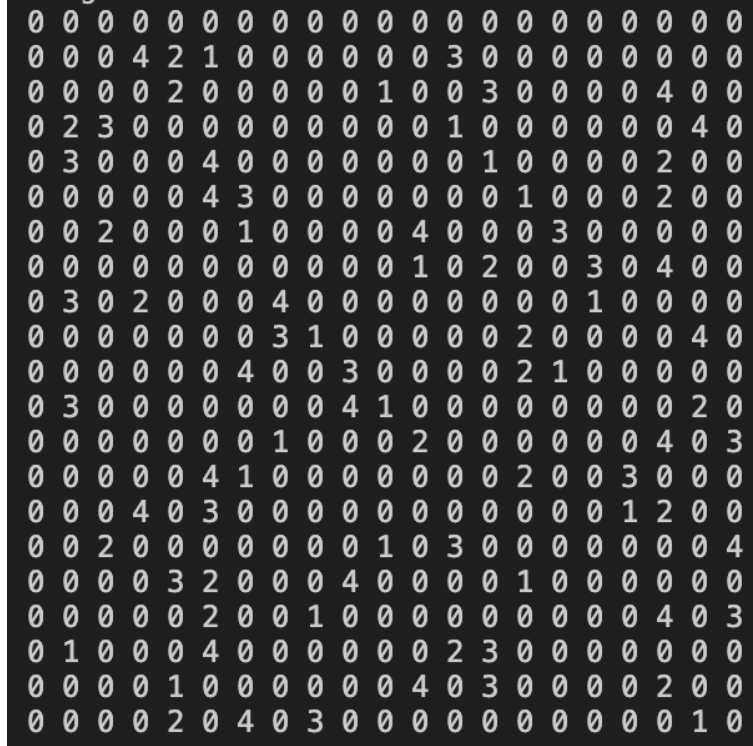
3. The assumption of 20 courses and 20 professors has been used. Courses numbered 1- 11 are CDCs and Courses numbered 12- 20 are Electives. No other distinction between the two sets of courses has been done.

### Domain of Solution

The solution provided primarily generates output for the given set of preferences. Any other preference order might result into a crash template.

### Graphs

We have represented the courses and the professors using a Bipartite graph where each Course is mapped to some professors, and each professor is mapped to some courses. The way to represent it in the code is using a 2-Dimensional Matrix where the rows signify the professor and the columns signify the course. The value at a specific entry shows the preference of the professor to the course(1 being the highest, and 0 showing no preference).



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	4	2	1	0	0	0	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0	1	0	0	3	0	0	0	0	4	0
0	2	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4
0	3	0	0	0	4	0	0	0	0	0	0	0	1	0	0	0	0	2	0
0	0	0	0	0	4	3	0	0	0	0	0	0	0	1	0	0	0	2	0
0	0	2	0	0	0	1	0	0	0	0	4	0	0	0	3	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	3	0	4	0
0	3	0	2	0	0	0	4	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	3	1	0	0	0	0	0	2	0	0	0	0	4
0	0	0	0	0	0	4	0	0	3	0	0	0	0	2	1	0	0	0	0
0	3	0	0	0	0	0	0	0	4	1	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	4	3
0	0	0	0	0	4	1	0	0	0	0	0	0	0	2	0	0	3	0	0
0	0	0	4	0	3	0	0	0	0	0	0	0	0	0	0	0	1	2	0
0	0	2	0	0	0	0	0	0	0	1	0	3	0	0	0	0	0	0	4
0	0	0	0	3	2	0	0	0	4	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	4	3
0	1	0	0	0	4	0	0	0	0	0	0	2	3	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	4	0	3	0	0	0	0	2	0
0	0	0	0	2	0	4	0	3	0	0	0	0	0	0	0	0	0	0	1

Figure 1: An image of the grid

The thing to note is we have used 1-based indexing for both professors and courses.

## Methodology

Step 1: Adhering to the rules, we first made an algorithm that makes a reads input from a .csv file(which includes the preferences of the professors).

Step 2: We then used the following Algorithm:

```
for (P in profs)
  mini : 5
  if (P.category == 2) {
    for (c in courses ranging 1 to 20)
      if (CourseToAllot.c == 0 && grid[P][c] != 0 && mini > grid[P][c])
        c.CourseToAllot = 1
        ProfToCourse[P][0] = c
        mini = grid[P][c]
  }
```

Figure 2: Pseudocode for Category:x2

In the above pseudo code, we have checked whether the professor belongs to the x2 category. If yes, then we have allotted the highest available course in the preference list to the professor.

```
for (P in profs)
  mini : 5
  if (P.category == 3) {
    for (c in courses ranging 1 to 20)
      if (CourseToAllot.c == 0 && grid[P][c] != 0 && mini > grid[P][c])
        c.CourseToAllot = 2
        ProfToCourse[P][0] = c
        mini = grid[P][c]
  }
```

Figure 3: Pseudocode for Category:x3

Then in the above pseudo code, we have checked whether the professor belongs to the x3 category. If yes, then we have allotted the highest available course in the preference list to the professor just like the previous step.

```

for (P in profs)
    mini = 5
    if (P.category == 1) {
        for (c in courses ranging 1 to 20)
            if (CourseToAllot.c == 0 && grid[P][c] != 0 && mini > grid[P][c])
                c.CourseToAllot = 3
                ProfToCourse[P][0] = c
                mini = grid[P][c]
                allot c to random P:category = 3.
    }

```

Figure 4: Pseudocode for Category:x1

Finally, we have allotted the un-allotted courses to the professors of x1 category keeping their preference list in check. Once the course is allotted to the professor of x1 category, it is bound to be allotted to the professor of x3 category. No order is followed while allotting the course to the professor of x3 category.

### Crash Case

Based on the restrictions and assumptions that were taken by us at liberty, these are the following scenarios where our code is susceptible to crash/fail/not give the most optimal output.

1. If there are not enough available courses to distribute to the professors of the categories x1 and x3, then the professors of x3 category will have to take only 1 course instead of their initial condition (1.5 course as mentioned in the problem statement).
2. If the number of professors in x1 and x3 are not equal then even in the most optimal case, at least 1 professor of x3 category will have only 1 course allotted to them.
3. In some possible cases, wherein the preference order of the professors is in a certain order, may lead to some courses being left unallocated.

Statistically these cases are not very significant as the probability of them occurring comes down as we increase the number of courses and professors.