**Application of Graph Optimization-**

**University Course Allotment Project**



Group Members:

Bhuvan Satrasala-2022A7PS0018G

Ishita Godani-2022A7PS0007G

Trisha Reddy-2022A7PS0009G

**ABSTRACT**

This research addresses the optimization of a University Course Assignment System, considering faculty categorization, variable course loads, and individual preferences. Faculty members, categorized into x1, x2, and x3, face course loads of 0.5, 1, and 1.5 per semester. The challenge involves maximizing course assignments to faculty while adhering to category-based constraints and preferences. Hungarian Algorithm is explored for potential solutions. The implementation involves constructing bipartite graphs, converting to maximization form, and considering faculty categories, course load capacities, and preferences. The proposed system aims to offer a flexible and adaptive approach to maximize faculty satisfaction.

**Contents**

1. **Problem Formulation**
2. **Methods Adopted**
3. **Results under different test cases**
4. **Crash Test**
5. **Consistency Report**
6. **Bibliography**
7. **Problem formulation**
   1. **Overview**

University course assignment is one of the most crucial parts of a university system, influencing both faculty satisfaction and the efficient utilization of resources. The University Course Assignment System aims to optimize the allocation of courses to faculty members within a department. The primary objective is to maximize the number of courses assigned to faculty while considering individual preferences, professor categories, and course load constraints. In addition, the system should explore and provide multiple acceptable combinations of assignments.

**1.2 Problem Statement**

**Mathematical Representation:**

* Sets: Let I be the set of courses, where i∈I.

Let J be the set of professors, where j∈J

* Parameters: xij ​ is an arbitrary value representing whether course i is assigned to professor j. (1 – 10 if assigned, 1000 otherwise)
* Decision variable: cij​ is a arbitrary value indicating whether professor j has course i in their preference list. (1 if in preference, 1000 otherwise)
* Objective Function: Minimize Z = ∑ i∈I ∑ j∈J cij.xij

**Faculty Categorization:**

* Each category has different course load capacities: 0.5 courses for "x1," 1 course for "x2," and 1.5 courses for "x3."
* No professor can have 0 courses in their preference list: ∑ i∈I cij≥1∀j∈J.

**Faculty Preferences:**

* Faculty members maintain preference lists of courses ordered by personal preference.

**Assignment Flexibility:**

* Depending on the group the professor is in, the minimum and maximum number of courses he/she can take differs.
* A single course can be assigned to a maximum of two faculty members, with a shared load.

**Optimization Objective:**

* Maximize the overall satisfaction of faculty members by aligning course assignments with their preferences.
* First, all the CDCs need to be assigned, followed by electives. Maximum number of courses need to be assigned.
* Explore and provide single best solution with CDCs as their priority and another solution where CDCS and electives are given equal priority.

**1.3 Constraints Category-Based Constraints:**

* Professors from category "x1” can have only 0.5 courses. (only one possibility): ∑ j∈J xij=0.5cij ∀i where i is in category x1
* Professors from category “x2” can either take two 0.5 courses or one complete course. (two possibilities): ∑ j∈J xij=cij ∀i where i is in category x2.
* Professors from category “x3” can take less than or equal to 1.5 courses which means they can take two 0.5 courses or one complete course or three 0.5 courses or take a complete course and two 0.5 courses. (4 possibilities):

∑ j∈J xij≤ 1.5cij ∀i where i is in category x3

**Course Load Constraints:**

* Ensure that the total course load for each faculty member aligns with their category-specific capacity: ∑ j∈J xij≤ Capacity of category of j for course i.

**Preference Consideration:**

* Courses can only be assigned if they are present in the faculty member's preference list: xij≤ cij ∀i∈I, ∀j∈J,

1. **Methods Adopted**

**2.1 Graph Optimization Approach:**

* To address the complexity of the University Course Assignment System, we adopted a graph optimization approach. This involves representing the problem as a bipartite graph, where professors and courses are nodes, and edges represent relationships based on faculty preferences.

**2.2 Forming equation:**

* Each course is split into two equal divisions, each equivalent to 0.5. This splitting allows flexibility in course assignments, accommodating the preferences and capacities of different professor categories. Similarly, each category of professors is divided into fractional units of 0.5. Specifically, x1 is allocated one 0.5 division, x2 is allocated two 0.5 divisions, and x3 is allocated three 0.5 divisions. Combining the divisions of courses and professor categories, the equation is formulated as 2n= x1+2x2+3x3 where 2n represents the total number of course divisions available and x1,x2,x3 represent number of professors in each category.

**2.3 Hungarian Algorithm:**

* + The Hungarian Algorithm is chosen to address the weighted matching aspect of the problem. Each edge in the bipartite graph is assigned a weight corresponding to the priority of the course in the faculty member's preference list. The Hungarian Algorithm efficiently finds the maximum-weight matching, optimizing the overall satisfaction of faculty members based on their preferences

**2.4 Implementation Steps Graph Construction:**

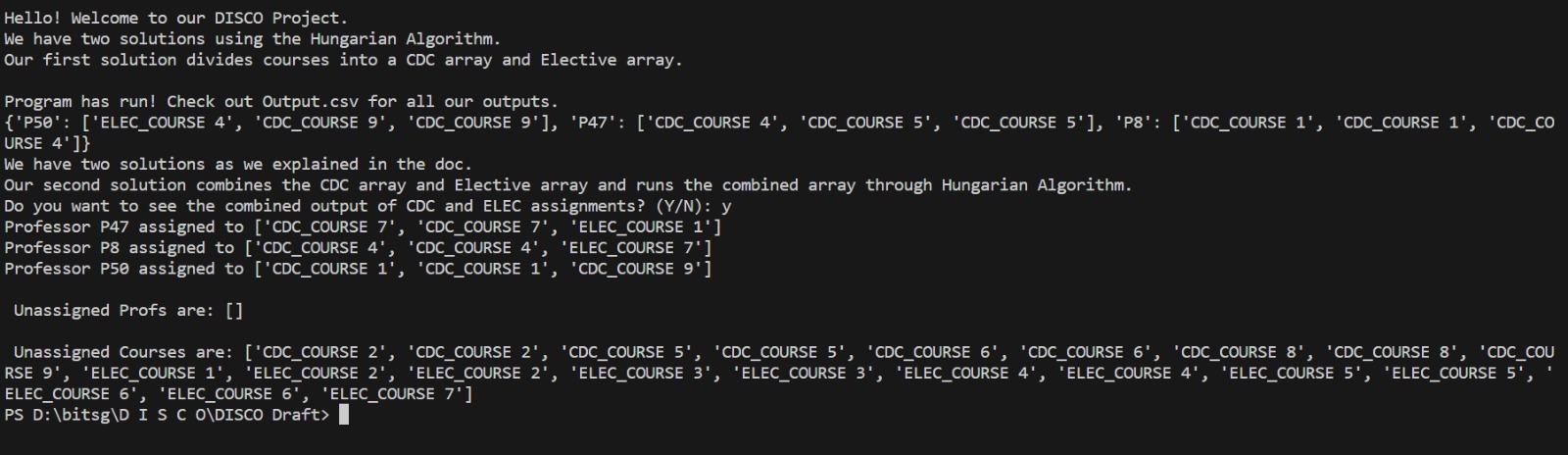
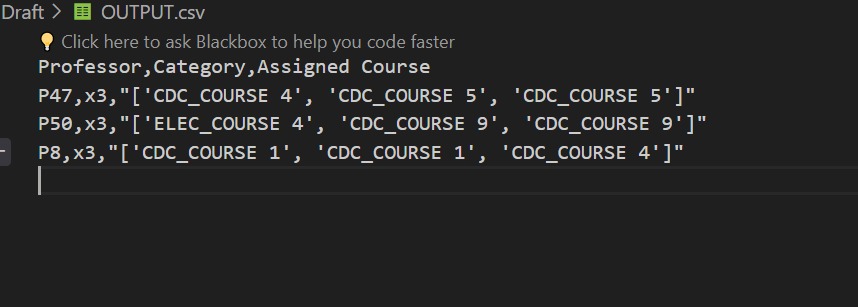
* Create a bipartite graph with faculty members and courses.
* Assign weights to edges based on faculty preferences.
* We have a cost matrix where we take row and course preferences and column as professor. Mathematically represented by {cij}N x N
* It is then converted to a binary matrix represented by {xij}N x N where xij=1 if and only if ith course preference is allotted to jth professor.
* Then, we find the minimum number of rows and columns required to cover all zeros.
* Since Hungarian algorithm works efficiently for square matrices, we add rows and columns necessary to ensure minimum coverage.
* The Hungarian Algorithm is applied to find the maximum-weight matching based on faculty preferences. Each faculty member is matched with the course that maximizes their satisfaction, considering the weights assigned to edges.
* Flexibility Adjustment: Explore modifications to enhance flexibility, such as adjusting the maximum number of courses for each category.
  1. **Functions in executable code**
* Main.py
  + - process\_csv() - a function to process data from the csv file to make 3 lists of course – CDC,ELEC,ALL COURSES..
    - cdc\_hung\_assignment() – a function used to process the CDC list to a CDC Array so it can be passed to the Hungarian algorithm and return the optimal assignment of cdc courses
    - list\_to\_dict() – a function that converts the lists to a common dictionary
    - assignments\_check() – a function to check if all profs and courses are assigned appropriately
    - elec\_hung\_assignments() - a function used to process the ELEC list to a ELEC Array so it can be passed to Hungarian algorithm and return the optimal assignment of ELEC courses
    - all\_assignments() - a function used to process the COURSES list to a COURSES array so it can be passed to the Hungarian algorithm and return the optimal assignment of all courses
* hungarian\_algo.py.
  + - An algorithm that helps us generate the most optimal solution for a given input matrix.

**2.6 Expected Outcome**

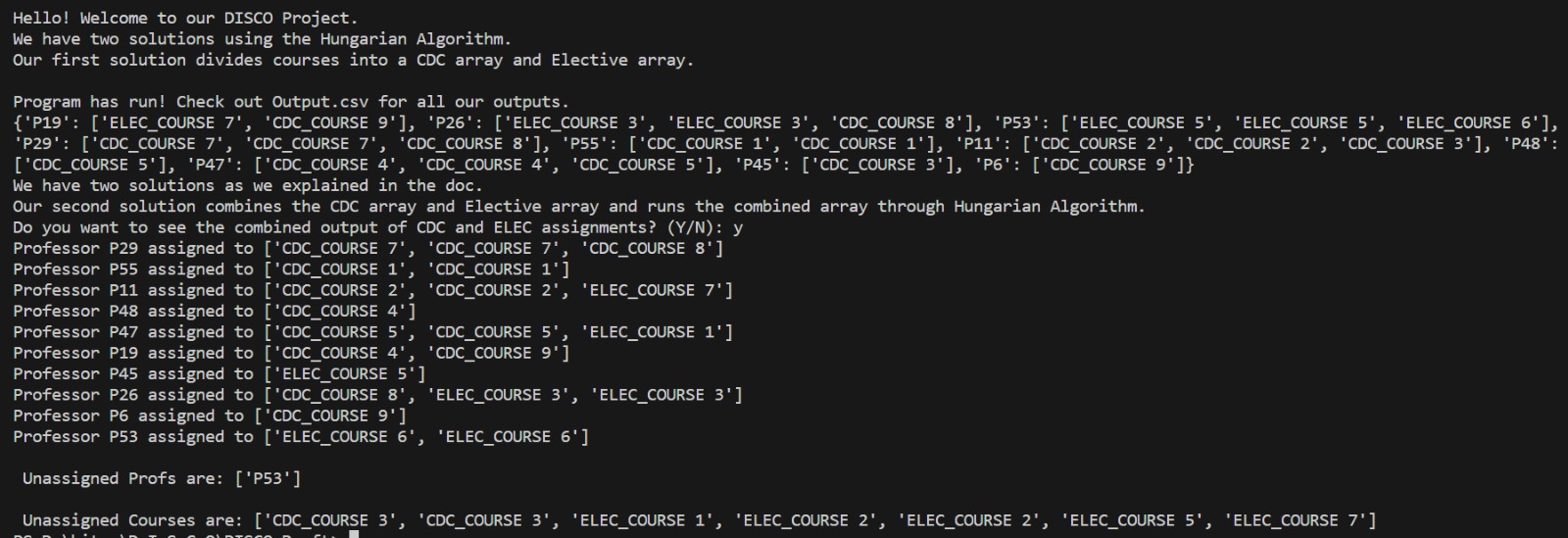
* + The expected outcome is an array of multiple acceptable combinations of assignments, each maximizing course assignments while respecting faculty preferences and category-based constraints. The graph optimization approach provides a robust framework for exploring diverse solutions. This report outlines the problem formulation and methods adopted for the University Course Assignment System, emphasizing the exploration of multiple acceptable combinations.

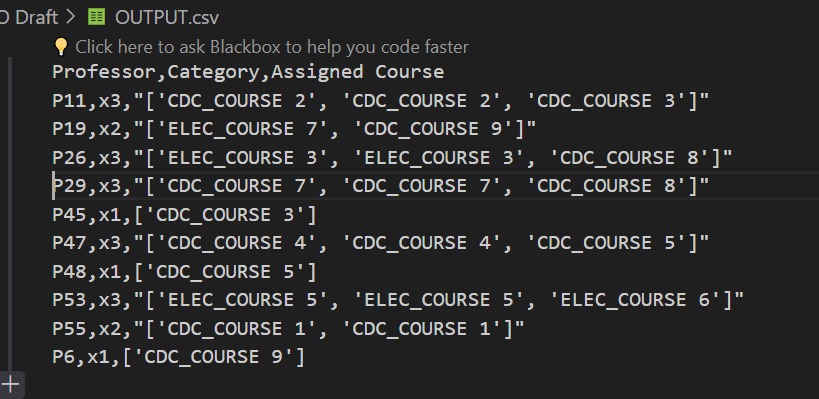
1. **Results under Different Test cases**

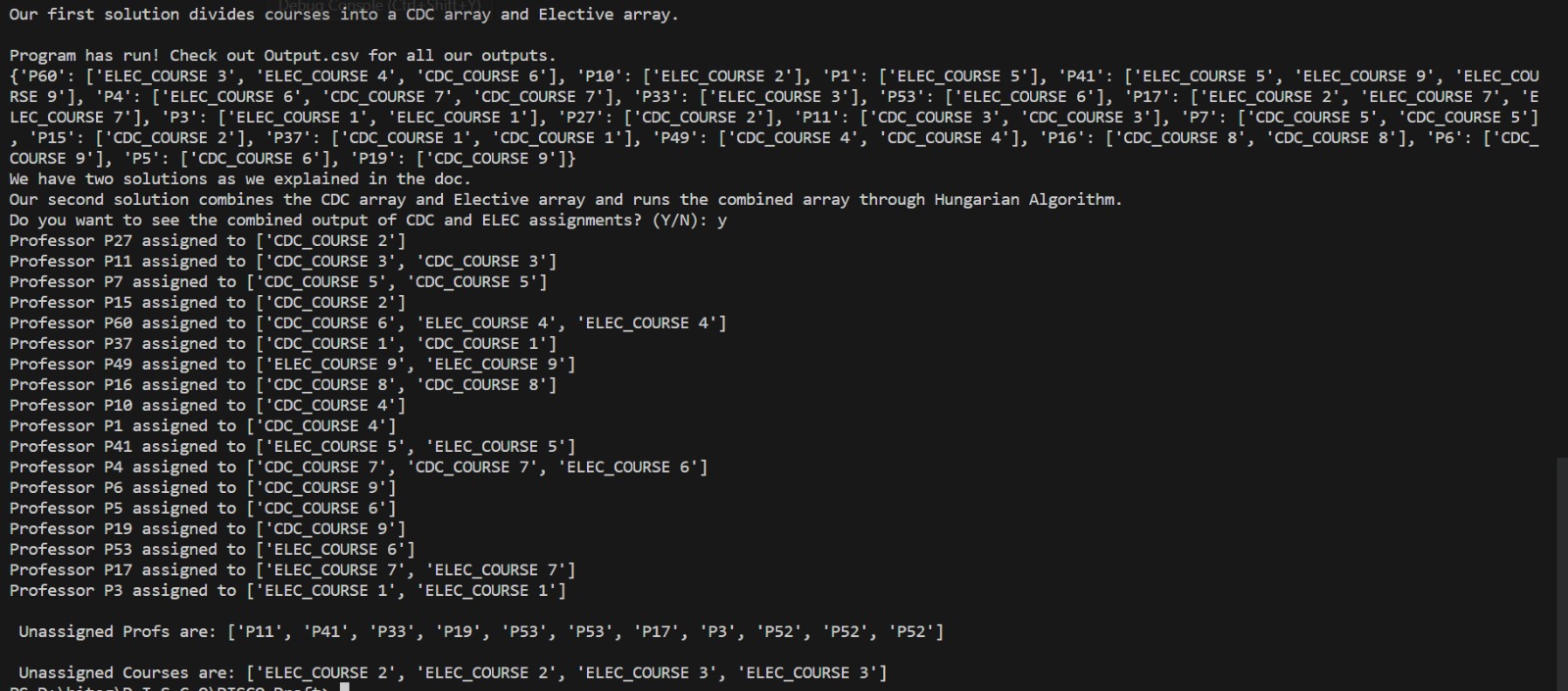
* **Testcase 1: 3 professors**

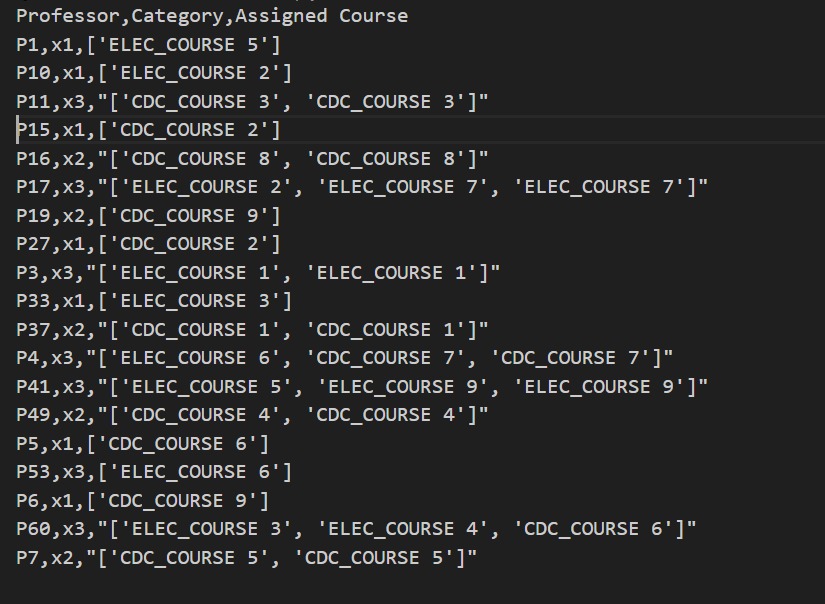
****

2. **Testcase 2: 10 professors**

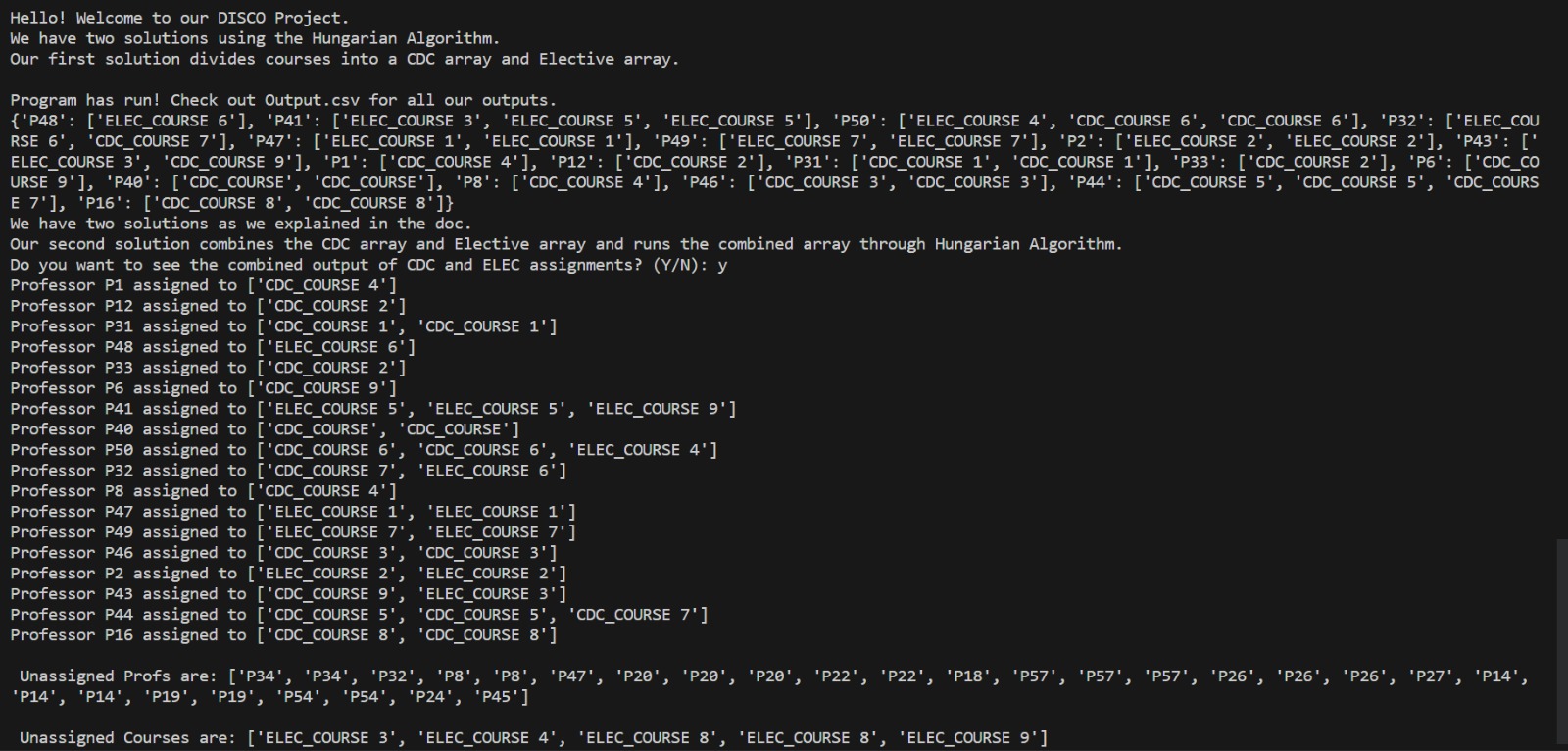
****

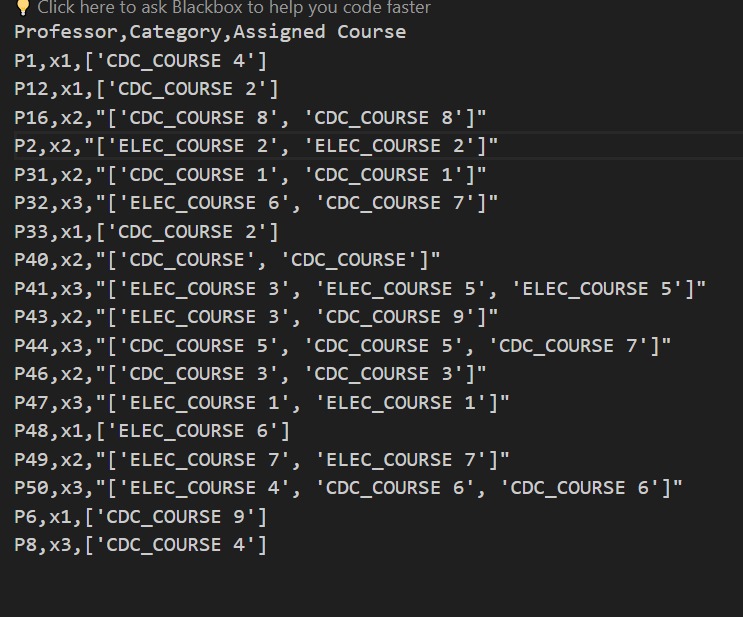


**Testcase 3:** **20 professors**



**Testcase 4: 30 professors**





**4.Crash Test**

Our algorithm remains consistent with the equation 2n>= x1+2x2+3x3 where n is the number of courses, x1 is the number of x1 professors, x2 is the number of x2 professors and x3 is the number of x3 professors. If 2n is lesser than x1+2x2+3x3, profs will go unassigned, which is recognized as a crash case.

Usually, each course is divided into divisions of 0.5 to accommodate preferences of x1, x2 and x3 categories. Ideally, a one to one mapping would occur with each professor being assigned a 0.5 division of a course. This is feasible when the number of available courses is greater than or equal to the sum of capacities of professor categories.

A crash case occurs when there are more professors than available courses, making it impossible to maintain a one-to-one mapping. This excess of professors may lead to unassigned individuals or unallocated course divisions. The crash case arises from a shortage of available course divisions to accommodate all professors and the algorithm faces a challenge in balancing the mapping. It must decide which professors receive course divisions and which may go unassigned. For example, if there are only 10 course divisions (each of 0.5) available, but there are 15 professors in total, a one-to-one mapping cannot be achieved.

The algorithm will provide feedback indicating the unassigned professors and potentially suggest alternative strategies for mapping in situations where a one-to-one mapping cannot be maintained. Users may need to adjust input parameters, redistribute capacities, or consider alternative preferences to achieve a more balanced assignment. By addressing this crash case, the algorithm becomes more robust and user-friendly, providing valuable insights and options for users to handle scenarios where the number of professors surpasses the available course divisions.

**5. Consistency Report**

The consistency report will emphasize the algorithm’s handling of various scenarios and performance under different conditions. Three important cases to consider:

* Unassigned electives: If 2n is greater than x1+2x2+3x3, electives will go unassigned, but all CDC’s will still be assigned.
* Maximum assignment: If 2n is equal to x1+2x2+3x3, every course is assigned.
* Minimum assignment: If 2n is lesser than x1+2x2+3x3, profs will go unassigned, which is recognized as a crash case.

The algorithm's consistency is tested by evaluating its sensitivity to changes in the number of courses (n) and the distribution of professors in different categories (x1,x2,x3) The algorithm is assessed on how it behaves in extreme cases, such as when one professor category dominates. When one professor category dominates, there can be several implications. The dominating category's preferences might disproportionately influence the overall assignment. If their preferences align with available courses, they may secure a higher number of preferred courses compared to other categories. Categories with fewer professors may face challenges in getting their preferred courses assigned if the dominating category exhausts the available courses. This could lead to unassigned professors in the underrepresented categories.

The algorithm's code is well-documented, including explanations for how the equation is implemented and how crash cases are handled. If professors go unassigned, it thoroughly analyzes and reports the reasons for unassignments. Verifies that the output format remains consistent across multiple runs. Describes the feedback or error messages provided by the algorithm when constraints are not met. Ensures clear communication of issues related to unassigned professors or courses.

The university course allotment algorithm maintains consistency by addressing crash test cases, providing feedback and offering optimization strategies.

Bibliography

* GUNAWAN, Aldy and NG, Kien Ming, “Solving the teacher assignment problem by two Metaheuristics”, (2011),<https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=5005&context=sis_research>
* Elvira E. Ongy,”Optimizing Student Learning: A Faculty-Course Assignment Problem using Linear Programming”,Journal of Science, Engineering and Technology, 2017, <https://www.researchgate.net/profile/Elvira-Ongy/publication/327721528_Optimizing_Student_Learning_A_Faculty-Course_Assignment_Problem_Using_Linear_Programming/links/5ba0cc6945851574f7d2a668/Optimizing-Student-Learning-A-Faculty-Course-Assignment-Problem-Using-Linear-Programming.pdf>
* Yash Varyani, “Hungarian Algorithm for Assignment Problem”,2023,<https://www.geeksforgeeks.org/hungarian-algorithm-assignment-problem-set-1-introduction/>
* Victor Keesey Fuentes, “Assigning Teacher Assistants to Courses: Mathematic Models,<https://www.math.ucdavis.edu/~webfiles/undergrad_thesis/201403_Victor_Fuentes_DeLoera.pdf>
* Guiming Luo,”The importance of Fuzzy Preference in Course Assignment Problem”, 25 Oct 2015, <https://www.hindawi.com/journals/mpe/2015/106727/>
* Eason, “Hungarian Algorithm Introduction and Python Implementation”,2021, <https://plainenglish.io/blog/hungarian-algorithm-introduction-python-implementation-93e7c0890e15>