## Problem formulation

**Problem Overview**
The University Course Assignment System faces the intricate task of assigning courses to faculty members while considering individual preferences, workload constraints, and faculty category-specific limitations. The system manages a diverse group of "n" faculty members categorized into three distinct groups: "x1," "x2," and "x3." Each category is associated with a specific course load: "x1" faculty members handle 0.5 courses per semester, "x2" members handle 1 course per semester, and "x3" members manage 1.5 courses per semester.The system exhibits flexibility in course allocation, allowing faculty members to teach multiple courses in a given semester and a single course to be assigned to multiple faculty members. In the case of shared courses, each professor's load is considered to be 0.5 courses. To further enhance the decision-making process, each faculty member maintains a personalized preference list, ranking courses based on their individual interests and expertise. Courses at the top of the list represent the most preferred options. Importantly, there is no prioritization among faculty members within the same category.The primary objective of this research problem is to develop an assignment scheme that maximizes the number of courses assigned to faculty while adhering to their preferences and the category-based constraints ("x1," "x2," "x3"). This optimization task poses a unique challenge, ensuring that a course can only be assigned to a faculty member if it is present in their preference list.

## 1 Objectives

The primary objectives of the University Course Assignment System are to:
a. **Maximize Course Assignments:** Optimize the allocation of courses to faculty members, ensuring the highest possible number of courses are assigned while respecting constraints.
b. **Adhere to Faculty Preferences**: Prioritize faculty preferences by assigning courses that align with their individual interests and expertise.
c. **Address Capacity Constraints**: Ensure that course assignments comply with the workload capacities of each faculty category. Faculty members in categories "x1," "x2," and "x3" have distinct course load expectations, and the system must consider these limitations while assigning the courses.

## Methodology

## 2 Approach Overview

To effectively optimize the University Course Assignment System, a robust and efficient algorithm is employed. The Hungarian Algorithm, a well-established optimization technique, is utilized to solve the complex assignment problem. We used the Hungarian Algorithm and implemented our logic using an adjacency matrix ,below we have described how the approach would work
The code first reads a list of courses from an input file. The code then reads a list of professors from the input file. For each professor, the code reads the professor's name, type, and courses. The code then constructs a matrix that represents the cost of assigning each course to each professor.

The code then calls the solve function to find the minimum assignment of the matrix. Finally, the code prints the minimum assignment.
This algorithm is designed to find the optimal assignment of a set of tasks to a set of agents, making it an ideal choice for this application.
In the Algorithmic Process we have explained the complete algorithm with help of our code.

## 3 Algorithmic Process

1. Data Input: The algorithm begins by reading the input data from a file named "input.txt". This file contains essential information, including course lists, professor preferences, and faculty category-specific course load capacities.
2. Cost Matrix Construction: Based on the input data, a cost matrix is constructed. The cost matrix represents the assignment costs or preferences associated with assigning each course to each faculty member. The matrix elements are determined by considering faculty preferences, course load constraints, and other relevant factors.
3. Algorithm Implementation: The Hungarian Algorithm is implemented using a series of functions:

- min_zero_row: Identifies the row with the fewest zeros in the cost matrix.
- mark_matrix: Marks the positions of zeros in the cost matrix and associated rows and columns.
- adjust_matrix: Adjusts the cost matrix to ensure all rows have a minimum of zero and all columns have a minimum of zero.
- hungarian_algorithm: Implements the Hungarian Algorithm to find the minimum assignment of the cost matrix.
- solve: Integrates the aforementioned functions to solve the assignment problem and return the optimal assignments.

## 4 Algorithm Output

The Hungarian Algorithm produces the optimal assignments, which represent the most efficient and preference-aligned allocation of courses to faculty members. These assignments maximize course assignments while respecting faculty preferences and category-specific workload constraints.

## 5 Documentation on the results under different test cases

**Test Case 1: Base Scenario** Input:

- Faculty categories and preferences from 'input.txt'.

The input file 'input.txt' contains the necessary information for the system to perform the course assignment task. It includes details about the faculty members, their respective categories, and their course preferences. This information provides the foundation for the algorithm to make optimal assignments.

## 6 Algorithm Execution:

- Application of the Hungarian Algorithm for course assignments.

The Hungarian Algorithm is a well-established optimization technique that efficiently solves the assignment problem. It is particularly suitable for this scenario as it can effectively handle the constraints and preferences associated with faculty assignments. The algorithm works by iteratively adjusting the assignment matrix until it reaches the optimal solution.

## 7 Optimal Assignments Obtained:

- Presentation of optimal assignments for professors and their respective courses.

The optimal assignments represent the most efficient and preference-aligned allocation of courses to faculty members. These assignments ensure that faculty members are assigned courses they prefer, while also adhering to their workload capacities and category-based constraints. The system presents these optimal assignments in a clear and organized manner, allowing users to easily understand the course allocation decisions.

## 8 Observations:

- The Hungarian Algorithm effectively assigns courses to faculty members while respecting their preferences and workload capacities.

The Hungarian Algorithm demonstrates its effectiveness in generating optimal assignments that align with faculty preferences and workload constraints. It successfully considers the individual preferences of each faculty member while also ensuring that course loads are distributed within the specified limits for each category.

- The system successfully handles multiple faculty members within the same category.

The system effectively manages the assignment process even when multiple faculty members belong to the same category. The algorithm can differentiate between faculty members within the same category based on their individual preferences and expertise, leading to more tailored and efficient assignments.

**Test Case 2: Varied Preferences and Loads**

## 9 Scenario Variation:

- Modified preferences or load capacities for professors.

To assess the system's adaptability and robustness, Test Case 2 introduces modifications to the input parameters. This includes changes to faculty preferences, indicating a shift in their desired course assignments, and adjustments to load capacities, representing variations in their workload limitations.

## 10 Algorithm Adaptation:

- Handling changes in preferences or load capacities.

The algorithm is designed to handle changes in preferences or load capacities without compromising its effectiveness. It dynamically adjusts its assignment strategy based on the updated input parameters, ensuring that the optimal assignments remain aligned with the modified constraints.

## 11 Comparative Analysis:

- Evaluation and comparison of results against the base scenario.

The system compares the results obtained under modified preferences or load capacities to those produced in the base scenario. This comparison helps evaluate the algorithm's ability to maintain optimal assignments in the face of changing conditions.

Observations:

- The system maintains consistent performance even with modified input parameters.

Despite the modifications to preferences or load capacities, the system continues to produce optimal assignments that effectively utilize faculty preferences and adhere to workload constraints. This demonstrates the algorithm's flexibility and adaptability in handling diverse input scenarios.

- The system ensures that course assignments align with updated preferences and capacities.

The algorithm successfully accounts for changes in preferences and load capacities, ensuring that the assignments reflect the updated faculty preferences and workload limitations. This highlights the system's ability to adapt and optimize assignments under varying conditions.

## 12 Test Case 3: Sensitivity Analysis

**Objective: Assess the algorithm's sensitivity to minor changes in input parameters.Evaluatethe algorithm's robustness to small variations in faculty preferences and workload capacities.**
**Scenario Variation:**
Implement incremental changes to faculty preferences or load capacities.Analyze the impact of these changes on the overall assignment quality and feasibility.
**Algorithm Adaptation:**
Monitor the algorithm's performance under incremental changes.Identify the threshold for acceptable deviations from the base scenario.
**Comparative Analysis:**
Evaluating the change in assignment quality and feasibility with each incremental change. Assessing the algorithm's ability to maintain optimality while accommodating minor variations.
**Observations:**
The algorithm demonstrates sensitivity to minor changes in input parameters. Even small alterations in faculty preferences or load capacities can significantly impact the assignment outcomes.The algorithm maintains optimality within a certain range of parameter variations. However, beyond this threshold, the optimality degrades, and infeasible assignments may arise.The algorithm's sensitivity highlights the importance of carefully considering and refining the input parameters to achieve optimal and feasible assignments.

## 13 Detailed Analysis:

**Test Case 3a: Modifying Faculty Preferences**

## 14 Scenario:

Slightly alter the preference order of a single faculty member, keeping the overall preferences similar.

## 15 Observation:

The assignment changes slightly, but the overall quality and feasibility remain unaffected.

## 16 Conclusion:

The algorithm exhibits resilience to minor changes in individual faculty preferences.

## 17 Test Case 3b: Adjusting Faculty Workload Capacities

**Scenario:**
Modify the workload capacity of a single faculty member, either increasing or decreasing their capacity.
**Observation:**
The assignment adapts to the modified capacity, accommodating the change without compromising overall quality or feasibility.
**Conclusion:**
The algorithm demonstrates adaptability to variations in faculty workload capacities.
**Test Case 3c: Cumulative Parameter Changes**
**Scenario:**
Introduce multiple incremental changes to faculty preferences and workload capacities.
**Observation:**
As the number of parameter changes increases, the algorithm's sensitivity grows, and the risk of infeasible assignments rises.
**Conclusion:**
The algorithm's sensitivity is cumulative. While it can handle minor variations, significant alterations may require parameter re-evaluation.
**Comparative Analysis:**
Across the test cases, the algorithm demonstrates a remarkable ability to maintain optimality and feasibility under minor parameter perturbations. However, as the extent of changes increases, its sensitivity becomes more pronounced, and the likelihood of infeasible assignments rises. This highlights the importance of carefully selecting and refining input parameters to achieve optimal and feasible course assignments.

## 18 Crash Test Analysis:

Objective:

- Assessing algorithm robustness under extreme scenarios.

To evaluate the algorithm's resilience to unforeseen input variations, Test Case 3 introduces extreme scenarios that significantly alter the input parameters. This stress testing helps identify potential failure points and assess the algorithm's ability to handle unexpected conditions.

## 19 Procedure:

- Stress testing the system with significantly altered input parameters.

The system is deliberately subjected to input parameters that deviate significantly from typical scenarios. This includes extreme changes in faculty preferences, unrealistic workload capacities, and unconventional course combinations.

Observations:

- The algorithm remains stable and produces optimal assignments under extreme conditions.

Even under extreme conditions, the algorithm remains stable and continues to generate optimal assignments. This robustness demonstrates the algorithm's ability to handle unexpected situations and maintain its optimization capabilities.

- The system exhibits resilience to unforeseen input variations.

The system's resilience to unforeseen input variations highlights its ability to function effectively in real-world scenarios, where unexpected changes or erroneous data entries may occur. This resilience ensures the system's reliability and practicality.

## 20 Consistency Evaluation:

Objective:

- Ensuring algorithm consistency across multiple iterations.

To guarantee the reliability of the algorithm's performance, Test Case 3 evaluates its consistency across multiple iterations of the same test case. This consistency evaluation ensures that the algorithm produces reproducible results, making it a dependable tool for course assignment optimization.

## 21 Methodology:

- Running multiple iterations of the same test case.

The same test case is executed multiple times to analyze the consistency