Optimizing University Course Assignment: A Penalty-Based Hungarian Algorithm Approach

Siddharth Chaitra Vivek (2022A7PS0569G), Amogh Balepur (2022A7PS1228G), Chaitanya Joshi (2022A7PS0730G)

November 30, 2023

Abstract

This paper attempts to find a solution for optimization of university course assignments within a department, where faculty members can handle multiple course loads and have their own preference order. The primary objective revolves around fulfilling course allocations while adhering to individual preferences and category-based constraints. The problem is solved employing a penalty-based assignment strategy and utilizing the Hungarian algorithm for weighted graphs. The findings underscore the suitability and effectiveness of this tailored approach in solving the complex course assignment optimization problem. The paper presents a few test cases and a set of close to optimal solutions generated by the code implementation.

Problem Statement

The aim of the problem is to implement a University Course Assignment System. Within a department, faculty members are grouped into distinct categories ("x1-teaching half a course," "x2-teaching one full course," and "x3-teaching one and a half courses") with varying course loads. When a course is taught by 2 professors, it is considered that each professor taught 0.5 of the course. Each professor also submits a preference list of the courses they are willing to teach arranged according to their priority. The goal is to create an assignment strategy that optimizes course allocation to faculty while adhering to their preferences and category-based constraints. This involves ensuring that a course is only assigned to a faculty member if it appears on their preference list and that all professors are assigned the exact number of courses based on which category they belong to.

Approaching the Problem

A potential solution to the given problem lies in employing a bipartite perfect matching, which involves constructing a graph representing professors and available university courses and linking them based on the various constraints. One way to implement this solution is through the Hungarian algorithm for weighted graphs. This method involves assigning increasing penalties to courses based on professors' preferences, aiming to minimize the total penalty generated upon assignment. Another potential approach, the minimum cost maximum flow algorithm, might not be well-suited for this specific course assignment optimization problem. This problem not only involves maximizing course assignments but also accommodating faculty preferences, making the Hungarian algorithm a better choice due to its capacity to address these nuanced constraints effectively. and easily provide an optimal solution.

Problem Formulation

The Hungarian Algorithm

The Hungarian algorithm, developed by Hungarian mathematicians Dénes Kőnig and Jenő Egerváry in the 1930s, stands as a pioneering method for efficiently solving assignment problems.Its efficiency and ability to handle larger problem sets make it an invaluable tool across various domains where optimization and efficient assignment problems are prevalent. This algorithm adeptly addresses the challenge of determining the optimal assignment in bipartite graphs. It can also be applied to square matrices representing the assignment, operating on the principle that if a number is added to or subtracted from all entries in any single row or column of a matrix, the resulting optimal assignment for this adjusted matrix remains optimal for the original matrix. This optimized matrix can be then represented as a bipartite graph. After each stage of the Hungarian algorithm requires O(n^2) arithmetic operations (if implemented with the appropriate data structures [4, 6]). Consequently, the computational complexity of the entire algorithm involving n stages is O(n^3)..

Hungarian:

Input: A bipartite graph, V, U, E (where |V | = |U| = n) and an n × n matrix of edge costs C.

Output: A complete matching, M

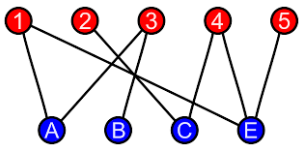

Figure 1: *A Bipartite Graph*

Objective Functions and Graph Constraints: Hungarian Algorithm

Given a graph G where each edge e connects two vertices and every vertex v is covered exactly once, we have the following inequality:

$$w(M') = \sum_{e \in E} w(e) \leq \sum_{e \in E} (l(e_x) + l(e_y)) = \sum_{v \in V} l(v)$$

where M' is any perfect matching in G created by a random assignment of vertices, $l(x)$ is a numeric label to node x. This implies that $\sum_{v \in V} l(v)$ is an upper bound on the cost of any perfect matching.

Now let M be a perfect match in G, then:

$$w(M) = \sum_{e \in E} w(e) = \sum_{v \in V} l(v)$$

So $w(M') \leq w(M)$ and M is optimal.
A feasible labelling includes the following condition:

$$l(x) + l(y) \geq w(x, y) \quad \forall x \in X, y \in Y$$

where X is the set of nodes on one side of the bipartite graph, Y is the other set of nodes, $l(x)$ is the label of x, and $w(x, y)$ is the weight of the edge between x and y.
Our solution to the problem involves the usage of an Adjacency Matrix which implements the Hungarian algorithm.

Methodology

The primary objective is to adhere to the constraints and professors' preference lists. To do this we are creating a penalty based system.

Our method aims to create a perfect bipartite match, but this implies that it is a function that is both onto and one-one. However we cannot ensure that the number of course slots that are being offered in a semester are exactly equal to the number of professors' available teaching slots.

Hence we prioritize:

-Our first aim is to ensure that all the CDCs (Compulsory disciplinary courses) in a given semester have professors assigned to them.

-Further the unused professor teaching slots that remain are then assigned to Electives, all while trying to minimize penalty based on the preference order given by the professors.

2 terms to be familiar with are:

1. Professor slots: We will be dividing professors into slots of 0.5 called a professor slot, based on which category they belong to. x1 is a single slot, x2 is 2 slots and x3 is 3 slots since the can teach course loads of 0.5, 1 and 1.5 respectively.
2. Course slots: Courses are split into halves each worth 0.5 of a course. This means that each course slot will be assigned 1 professor slot

A given professor submits an ordered list of courses they would like to teach. The first preference in that list is assigned a penalty of 1, the second has penalty of 2, and so on. Once we have assigned these penalties for all the preferences of all the professors, we create a matrix: where each row corresponds to a given professor and each column to a given course. The assigned values are then input into this matrix.

Further courses which aren't in a professors preference list are assigned an arbitrary large penalty (in this case 1000). This is to highly de-incentivize this assignment from being made.

This now leaves us with a 2-dimensional matrix containing penalties for each possible assignment.

(i.e. A value of n in the i-th row and j-th column means that the i-th professor teaching slot being used for the j-th course slot has a penalty value of n)

The final requirement before we can apply the Hungarian algorithm is to ensure that this penalty matrix is a square matrix. Since we will adjust the number of courses depending on number of teaching slots available, this means there are always greater or equal teaching slots than course slots. So to compensate, we fill the remaining portion of the table with the penalty values for electives. This way the extra professor slots are immediately assigned to electives and CDCs are ensured to be all allotted professors. This elective process is discussed in detail in the next section. Finally this leaves us with a square penalty matrix to which we can apply the Hungarian Algorithm.

Assigning Electives

After assigning all Compulsory Disciplinary Courses (CDCs), the remaining teaching slots are allocated to professors based on their elective course preferences. In order to fill the electives, we need to assign the spillover professor slots in such a way that single halves of electives are not filled. To do this, we generate all permutations of elective course slots where both parts of the course are always together. (eg: If there are 4 extra professor slots, we would generate permutations of all 2 full elective courses, which

gives us 4 halves).These elective halves are also assigned penalties in a similar manner to the CDCs and then used to fill up the rest of the matrix until we have a square matrix. For each combination, we run the Hungarian algorithm and after evaluating all combinations, we select the one with the lowest total penalty. This method enables us to manage the assignment of both electives and CDCs effectively, which ensuring nobody gets assigned courses they do not want to take.

Generating Multiple Solutions

Applying the Hungarian Algorithm on a penalty matrix returns a bipartite perfect matching of the different professor and course slots. However this is just one optimal solution. In order to do generate multiple solutions, we manually tweaked the penalty matrix and forced a different output to be generated. Taking the first optimal solution generated, we set the penalty of the edge between the first professor slot and its corresponding assigned course slot to 1000. This ensures that when we rerun the Hungarian algorithm on this modified penalty matrix, this specific edge is not created. Doing this forces a new solution to be generated which may be equally or slightly less optimal.

We can generate a set of solutions by merely changing one penalty value for one professor slot at a time without compromising on optimisation.

Crash Cases

There are certain cases in which bipartite perfect matches cannot be generated, we call these "Crash cases". The program will crash in the following cases:

1. When the number of professor slots is fewer than the number of CDC slots in the semester. This means that all the CDC will not get assigned professors which is a violation of the constraints.
2. When the number of professor slots is more than the total course slots in the semester (CDCs + Electives). In this case, There is no way to assign courses to all professor such that they meet their category requirements of 0.5,1.and 1.5 for x1,x2,x3 respectively.
3. If the remaining number of professor slots after assigning all the CDCs is an odd number. This would result in atleast one elective being assigned partially (only one have would be assigned a professor) and since this is not within the constraints, the program crashes.

Examples for the Crash test and their outputs are given in the Results and Crash-Test Consistency report.

Conclusion

The paper has gone over a structured approach to the given problem statement. The implementation of the above is present in the Github repository and a "Results and Crash-Consistency report" highlights few test cases run on the program with varying outputs. A video explanation of the various parts of the code implementation and working demonstration is also linked in the Github README file.

References

1. **H. W. Kuhn, Bryn Yaw College: The Hungarian Method For the Assignment Problem**
2. **CMU Publication: The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs**
3. **Subash Suri: Bipartite Matching & the Hungarian Method**