Modified Hopcroft-Karp algorithm to solve University Course Assignment Problem

Rohit Raj

November 2023

ABSTRACT: This research addresses the optimization of the University Course Assignment System, focusing on categorizing faculty into three groups and assigning courses based on preferences and category-specific constraints. Using a bipartite graph and the Hopcroft-Karp algorithm(1), our approach maximizes course assignments while accommodating faculty preferences. The project project focuses on using iterations of the algorithms in a certain way that can be generalized to a large set of problems.

1 Introduction

The problem revolves around the assignment of n courses among m faculty members. Each faculty member has a preference order and a category. There are three categories: x1, x2, and x3 and they correspond to a max course load of 0.5, 1, and 1.5 courses, respectively. If a faculty takes 0.5 courses the rest of the course must be assigned to another faculty with at least 0.5 course space available.

This problem is different from a typical assignment problem because it involves sharing courses, which complicates our lives. We require a solution that adopts both keeping track of faculty requirements and course sharing. This should ensure that each course's second half doesn't end up unassigned. Moreover, there are other constraints in the problem, such as ensuring all CDCs are assigned.

2 Literature Review

There are n number of research papers revolving around one-one assignments with respect to bipartite graphs. What makes our problem different from a typical job-worker assignment problem is the aspect that a course can be shared and multiple assignment is possible for a course. This(3) research discusses this aspect with the help of a modified Hungarian method(2) for solving assignment problem with multiple job assignment per worker. Another paper(4) by the same author discusses the assignment of a job to multiple machines. This paper tells us how net opportunity cost is lower for multiple assigned system and provided inspiration for our work.

3 Problem Formulation

The optimization problem addressed in this research focuses on enhancing the University Course Assignment System. Within a department, faculty members are categorized into three groups— x1, x2, and x3—each with distinct course load constraints. The objective is to develop an assignment scheme that maximizes the number of courses assigned to faculty, considering their preferences and adhering to category-based constraints.

3.1 Objective Function:

$$\text{Maximize} \quad Z = \sum_{i \in U} \sum_{j \in V} x_{ij}$$

3.2 Decision Variables:

$$x_{ij} = \begin{cases} 1 & \text{if faculty member } i \text{ is assigned to course } j \\ 0 & \text{otherwise} \end{cases}$$

3.3 Constraints:

1. **Faculty Assignment Constraints:**

$$\sum_{j \in V} x_{ij} \leq 2, \quad \forall i \in U$$

Ensures each faculty member is assigned to at most two courses.

2. **Course Assignment Constraints:**

$$\sum_{i \in U} x_{ij} \leq 2, \quad \forall j \in V$$

Ensures each course is assigned to at most two faculty members.

3. **Category-Based Constraints:**

$$\sum_{j \in V} x_{ij} \leq y_i, \quad \forall i \in U_{x1}$$

$$\sum_{j \in V} x_{ij} \leq 2y_i, \quad \forall i \in U_{x2}$$

$$\sum_{j \in V} x_{ij} \leq 3y_i, \quad \forall i \in U_{x3}$$

4. **Faculty Preference Constraints:**

$$x_{ij} = 0 \quad \text{if course } j \text{ is not in the preference list of faculty member } i$$

5. **Non-negativity Constraints:**

$$x_{ij} \geq 0, \quad \forall i \in U, j \in V$$

Parameters:

- U: Set of faculty members.
- V: Set of courses.
- U_{x1}, U_{x2}, U_{x3}: Sets of faculty members in categories $x1$, $x2$, and $x3$ respectively.
- y_i: Maximum number of courses faculty member i can be assigned based on category-specific constraints.

4 Methodology

We approached the problem using two distinct methods. These approaches differ significantly: the first employs the Hungarian Algorithm(2) with some modifications, while the second utilizes the Hopcroft-Karp algorithm(1) with similar adjustments. Although the first method provided valuable insights, we decided to focus on the second approach due to its notably higher success rate.

4.1 Modified Hungarian Algorithm

The Hungarian matching algorithm(2), also known as the Kuhn-Munkres algorithm, is an $O(|V|^3)$ algorithm designed for finding maximum-weight matchings in bipartite graphs, commonly referred to as the assignment problem. While bipartite graphs are conveniently represented by an adjacency matrix, our problem extends beyond the generic cases addressed by this algorithm. Our specific challenge involves constraints and course-sharing, necessitating a unique approach capable of handling these constraints effectively.

4.1.1 Category Profit Matrix

Figure 1 presents a matrix illustrating the opportunity cost for each faculty's assignment. This matrix ensures: (i) Maximizing the number of course assignments, (ii) Relatively higher costs for assignments below the maximum workload, and (iii) Promoting cost-effectiveness for larger workload assignments. After establishing weights, we incorporate each faculty's priority list to further refine the process.

$$\begin{pmatrix} c & \text{x1} & \text{x2} & \text{x3} \\ 0 & 7 & 8 & 9 \\ 0.5 & 3 & 5 & 6 \\ 1 & 0 & 2 & 4 \\ 1.5 & 0 & 0 & 1 \end{pmatrix}$$

Figure 1: Category Profit Matrix

Here, the column c refers to the four possibilities of course assignments. The rest of the columns have cost values depending on the quality of the assignment with respect to our system.

4.1.2 Complete Profit Matrix

Building upon the previous section, each faculty category now has a weight assigned to a specific course assignment. This weight, combined with the faculty's preference ranking, creates an m*n profit matrix.(m being the number of faculty members and n being the number of courses) For example, assigning a 1 course-load course to an x3 category professor with a weight of four, ranked second in preference, results in a net cost of 6. The Hungarian Algorithm(2) is then applied to this profit matrix to yield effective assignments.

4.1.3 Bottlenecks

While this approach appears correct, it overlooks the essence of the problem, particularly course-sharing, leading to standalone assignments for some x1 category professors, breaking constraints.

Acknowledging these limitations, we transition to the next approach—the modified Hopcroft-Karp Algorithm.

4.2 Modified Hopcroft-Karp Algorithm

The Hopcroft-Karp algorithm(1) efficiently solves the maximum cardinality bipartite matching problem, focusing specifically on bipartite graphs. Unlike the Hungarian algorithm's(2) $O(n^3)$ time complexity, the Hopcroft-Karp algorithm boasts $O(\sqrt{V} \cdot E)$ efficiency, making it well-suited for bipartite matching problem

4.2.1 Dealing with Course-Sharing

Addressing the course-sharing issue, we divide each course into two parts. Each part is treated as a separate course with half the course load, allowing us to assign based on faculty preferences. This leads to three possibilities: first, a course is shared between two different faculties; second, a course is assigned entirely to one faculty; and third, a course is assigned to a faculty, but the second half remains unassigned. We found an efficient way to avoid the third case.

4.2.2 Iterative Assignment

Assigning professors half-load courses based on preferences using the Hopcroft-Karp algorithm(1) results in $2n - m$ unassigned half-load courses. All x1 faculties reach their full capacity, x2 category faculties reach 0.5/1, and x3 category faculties reach 0.5/1.5 of their capacity. Now, the second half of some of the courses that were assigned in this iteration are still unassigned. In order to avoid the third case where half of a course remains unassigned, we run another iteration but only on courses whose first half are already assigned to ensure the entire course is assigned. Subsequent iterations involve using the Hopcroft-Karp algorithm on the remaining unassigned courses, providing a full workload for the x1 and x2 categories. The final assignment addresses the remaining 0.5 slots for x3 category faculty, resulting in a nearly complete set of assignments. Adding the extra steps of prioritizing the courses with their first halves pre-assigned ensures that we are not left with a half-assigned course.

4.2.3 Accounting for CDCs and Second Half of the Courses

To assign all CDCs, we initially run the algorithm solely on CDCs, guaranteeing their allocation. We then assign electives while ensuring that if one half of a course is assigned, the other half follows in the next iteration. This is done similar to how CDCs were assigned in the previous steps. If all iterations are done correctly in the order mentioned, is almost guaranteed to get a nearly full allocation while keeping the faculty preferences in mind.

5 Results

5.1 Input Data

Table 1 illustrates our input data for one of the test cases, including each faculty's category and preferences. The category aids in determining the maximum course load for a faculty member.

Name	Category	Preferences
prof1	x1	C3,C5,C8,C14,C2
prof2	x1	C9,C5,C3,C8,C14
prof3	x1	C5,C14,C9,C10,C3
prof4	x1	C13,C1,C5,C8,C15
prof5	x2	C12,C3,C8,C6,C15
prof6	x2	C15,C4,C11,C7,C2
prof7	x2	C3,C10,C12,C6,C7
prof8	x2	C6,C3,C12,C11,C4
prof9	x3	C10,C8,C6,C14,C7
prof10	x3	C8,C13,C9,C15,C4
prof11	x3	C2,C8,C6,C10,C11
prof12	x3	C12,C4,C8,C11,C5

Table 1: This test case is initially run for CDCs followed by HELs to ensure all CDCs are mandatorily assigned.

5.2 Output Data

Table 2 displays the generated output. Upon manual inspection of each assignment, it becomes evident that some faculty members received their top preferences while others did not. However, on a broader scale, it is noteworthy that most faculty managed to secure one of their top priorities, maintaining the entire system's efficiency. Alternative approaches, such as the Greedy algorithm, pose challenges due to their emphasis on immediate gratification. Our code prioritizes the long-term effectiveness of the Computer Science Department by employing an iterative preference-first algorithm. This approach ensures that faculty preferences are respected and course-sharing is effectively managed.

Upon evaluation under different test cases, we obtained the following results at an average:

- 100% of all faculties received one of their preferences.
- 58.3% of all faculties received one of their top two preferences.
- 77.5% of all faculties received one of their top three preferences.

5.3 Crash Test Results

In the crash test of our model, we encountered a scenario where a faculty member's preferences solely consisted of courses already assigned to other fully occupied faculty members. In such cases, the algorithm struggled to complete the full course assignment, resulting in unassigned courses, mainly due to the algorithm's dependency on preferences. Additionally, we observed a situation where a large number of faculty members shared the exact same preferences. In such instances, the algorithm faced challenges in distributing the courses evenly, potentially leaving one or two faculty members with incomplete assignments. Despite these specific scenarios, it's important to note that the algorithm maintained a high degree of accuracy across a diverse range of test cases, successfully optimizing course assignments for most faculty members.

5.4 Discussion

The efficacy of this approach lies in its ability to transform a many-is-to-many assignment problem into a series of iterative one-one assignments. This strategic shift, added to our emphasis on prioritizing courses with their first halves already assigned, minimizes the risk of leaving half-courses unas-

Name	Category	Preferences	Courses
prof1	x1	C3,C5,C8,C14,C2	C3
prof2	x1	C9,C5,C3,C8,C14	C8
prof3	x1	C5,C14,C9,C10,C3	C10
prof4	x1	C13,C1,C5,C8,C15	C1
prof5	x2	C12,C3,C8,C6,C15	C15,C15
prof6	x2	C15,C4,C11,C7,C2	C7,C4
prof7	x2	C3,C10,C12,C6,C7	C3,C10
prof8	x2	C6,C3,C12,C11,C4	C14,C11
prof9	x3	C10,C8,C6,C14,C7	C8,C14,C6
prof10	x3	C8,C13,C9,C15,C4	C11,C9,C9
prof11	x3	C2,C8,C6,C10,C11	C4,C2,C2
prof12	x3	C12,C4,C8,C11,C5	C7,C5,C5

Table 2: Here is the output for the given test case. We can see how the x2 category professors have one whole course either by having two different halves of courses or two halves of the same course. Similarly, x1 and x3 have 0.5 and 1.5 courses, respectively.

signed. Furthermore, the use of the Hopcroft-Karp algorithm(1), specifically designed for bipartite graphs, enhances computational efficiency compared to the Hungarian algorithm(2). The Hopcroft-Karp algorithm(1) streamlines the calculations, contributing to faster and more optimized solutions for our course assignment system. This approach yielded various observations, such as the revelation that, in large job-worker assignment problems, distributing the job among workers results in a lower net total cost. Addressing this issue involves dividing each course into two parts, each with half the course load. This strategy ensures that if a faculty member prefers to take a course individually, they will be assigned both instances of the course. Varying categories and rules could further decrease the net cost, posing an exciting avenue for future research.

We've depicted a visual representation in Figure 2, using the tool mentioned here(5), to clarify the workings of our method. In Figure 2, the top-left image showcases the bipartite graph we aim to achieve. Here, red dots denote faculties, and green dots represent courses, illustrating a many-to-many assignment setup. Our approach involves iterative steps. We start with a basic one-one assignment, as seen in the top-right image of Figure 2. In each subsequent iteration, we introduce two new courses, leading to another one-one assignment, illustrated in the bottom-left image of Figure 2. Finally, another one-one assignment is done in the bottom-right image, achieving the desired bipartite graph. We have achieved a many-to-many assignment simply by using several iterations of one-to-one assignments.

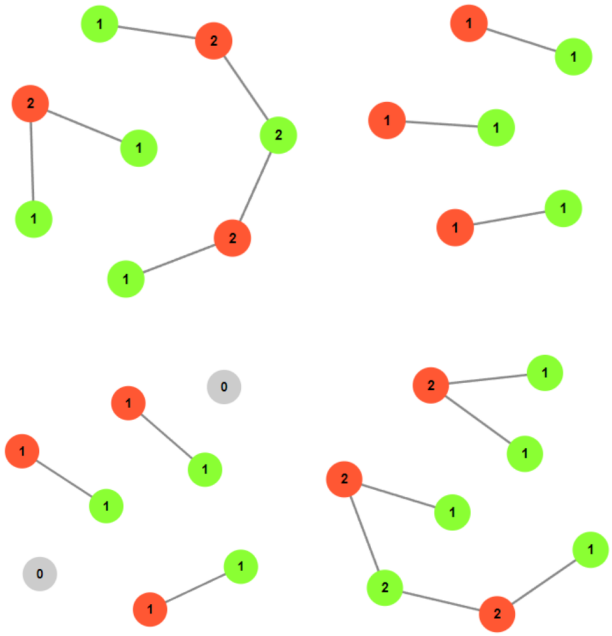

Figure 2: Visual Representation of our method

Here, green dots are faculties, and red dots are courses. To reach the top-left image, we progress through the initial iteration (top-right), followed by the second iteration (bottom-left), culminating in the final iteration (bottom-right). All iterations are simple one-one assignments.

6 Conclusions

In conclusion, our study delved into the intricacies of optimizing the University Course Assignment System, presenting a novel approach to maximize the number of courses assigned to faculty members.

Utilizing the Hungarian Algorithm(2) in the first model and the Hopcroft-Karp algorithm(1) in the second, we accommodate faculty preferences while adhering to category-specific constraints, offering a flexible and efficient course assignment scheme.

The uniqueness of this problem lies in its departure from traditional assignment problems, granting faculty members the flexibility to handle varying course loads, with each course assigned to one or two faculties. This is achieved by dividing each course into two parts, treating each as a separate course with half the course load. This transformation simplifies the problem into a one-one assignment problem, which is easier to solve. The solution is then applied iteratively: first on all three categories, then on x2 and x3, followed by just x3 to ensure maximum course load constraints while satisfying all problem constraints and requirements. Moreover, we account for mandatory assignments of CDCs by modifying the input course list to ensure complete courses are assigned in a semester.

In summary, this research contributes to enhancing the University Course Assignment System, providing a foundation for further advancements in optimizing faculty course assignments.

References

[1] John E. Hopcroft and Richard M. Karp. *A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*. SIAM J. Comput, 4(1973), 225-231

[2] H. W. Kuhn. *The Hungarian method for the assignment problem*. Nawal Research LogisticQuarterly, 2(1955), 83-97.

[3] Quazzafi Rabbani, Aamir Khan, Abdul Quddoos. *Modified Hungarian method for unbalanced assignment problem with multiple jobs*. Applied Mathematics and Computation, 361(2019), 493-498.

[4] Quazzafi Rabbani, Aamir Khan, Abdul Quddoos. *Assignment of multiple jobs scheduling to a single machine*. Advances in Mathematics Scientific Journal, 10(2), 1003-1011.

[5] mrpandey@github. *Bipartite Graph Generator*. D3 Graph Theory