GRASP-Based Approach to the Course Faculty Assignment Problem

Jainam Shah · Michael Lewis · Mahir Jimit Ghadiali

November 2023

1 Introduction

The focal point of our research revolves around optimizing the University Course Assignment System. Within a department, faculty members are classified into three distinct groups: x_1, x_2, and x_3. Each group has a designated maximum course load, with x_1 handling 0.5 courses per semester, x_2 managing 1 course per semester, and x_3 overseeing 1.5 courses per semester.

This system allows faculty members the flexibility to enroll in multiple courses during a semester, and conversely, a single course may be assigned to multiple faculty members. When a course is shared between two professors, each professor's load is considered to be 0.5 courses. Additionally, each faculty member maintains a preference list of courses, ordered by personal preference, with the most favored courses listed at the top. It is important to note that there is no prioritization among faculty members within the same category.

The primary objective of this research problem is to devise an assignment scheme that maximizes the number of courses assigned to faculty while adhering to their preferences and the category-based constraints (x_1, x_2, x_3). The challenge lies in ensuring that a course can only be assigned to a faculty member if it is present in their preference list. Also among the courses, the CDCs(Compulsory Disciplinary Courses) have higher priority to be assigned than the Electives. So the algorithm must assign all CDCs (unless not possible) while also maximising the total number of courses assigned.

This problem stands out due to the flexibility it offers regarding the number of courses faculty members can undertake, setting it apart from typical Assignment problems. Also keeping the preference of the faculties in mind while giving multiple sub optimal solutions is required so that the Department HOD can make the last call by choosing 1 among the possible solutions. Possible modifications may involve adjusting the maximum number of courses y for each category of professors, rather than requiring strict adherence, or expanding the number of professor categories beyond the existing three to create a more generalized solution.

2 Problem Constraints and Assumptions

2.1 Hard Constraints

The formulation of the University Course Assignment Problem is bound by the following hard constraints:

1. **Maximum Course Load:** The maximum course load for each faculty member must be maintained, distinguishing between three categories: x_1, x_2, and x_3, representing 0.5, 1, and 1.5 courses per semester, respectively.
2. **Faculty Assignment to Courses:** Each course should have either 1 or 2 faculty members assigned to it. This ensures that a course is either solely assigned to one faculty or shared between two.
3. **Complete Assignment:** A course should not remain partially assigned. For shared courses, it must be assigned to exactly 2 faculties, maintaining completeness in faculty assignment.
4. **CDC Assignment:** All Compulsory Disciplinary Courses (CDCs) must be assigned if possible. Priority is given to ensure the assignment of all CDCs.

2.2 Soft Constraints

The soft constraints, while not mandatory, add additional considerations for optimizing the assignment process:

1. **Teacher's Course Preferences:** Faculty members' course preferences should be taken into account during the assignment process. This involves considering the ordered lists of courses provided by each faculty member.
2. **Balanced Course Load:** The course load of faculty members should be balanced across all teachers as much as possible. This soft constraint aims to distribute the workload evenly among faculty members.

2.3 Assumptions

Certain assumptions have been made to facilitate the formulation of the problem:

1. **Equal Preference Priority:** In cases where multiple faculty members have the same position in the preference list for a particular course, there is no preference given to one professor over another. This assumption ensures fairness in course assignments.

3 Objective Function

The objective function for the problem is as follows:

$$\text{Maximize} \sum_i assigned\_courses_i \tag{1}$$

Subject to:

$$assigned\_courses_i \leq max\_course\_load_i \quad \forall i \tag{2}$$

$$assigned\_CDCs = total\_CDCs \tag{3}$$

Here,

where
$assigned\_courses_i$ is the number of courses assigned to professor i,
$max\_course\_load_i$ is the maximum course load for professor i,
$assigned\_CDCs$ is the number of CDCs assigned,
$total\_CDCs$ is the total number of CDCs,
and the summation is taken over all professors i.

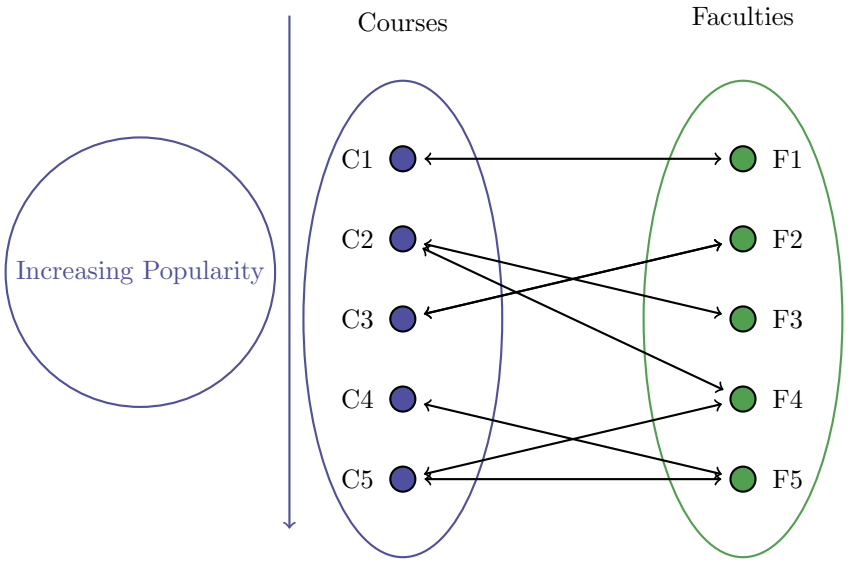

Figure 1: Faculty-Course Assignment.

4 Algorithmic Approach

Our algorithm initiates with a bipartite unidirectional graph, modeling the relationship between faculty members and courses, with courses initially assigned to faculties. The objective is to construct an initial solution through a partially greedy procedure. A key element of this approach is the utilization of a popularity function.

The popularity function is defined for both faculty members and courses. A higher popularity for a course indicates it is more favored among faculty members. The calculation involves the preference list, where, for example, if a course appears second in the preference list of a faculty, the added popularity is determined by the total number of courses in that preference list and the position of the course in the preference list of the faculty (shown as an equation below).

$$\text{Popularity of a course} = \sum_i (n_i - y_i + 1) \tag{4}$$

where

n_i : Number of unique courses in the preference list of all the faculties.

y_i : Position of the course in the preference list of the i-th faculty. If the course isn't present, it is given the $(n_i + 1)$-st position.

Similarly, for faculty members, the popularity is the sum of the popularity values of all courses in their preference list. The faculties and courses are then sorted in ascending order based on their popularity.

Subsequently, an iteration is performed through courses and, within that, through faculties. Courses are assigned to faculties if the course exists in their preference list and there are sufficient credits remaining. This process results in an initial solution.

To introduce randomness and explore alternative solutions, the initial solution undergoes a random change. This is achieved through a local search mechanism, identifying positions where two courses of equal popularity can be exchanged.

The algorithm performs efficiently even for extensive test cases due to its O(nlogn) time complexity. Consequently, it effectively transforms this NP-hard problem into a P-hard solution.

5 Pseudocode

Algorithm 1: Generate Initial Solution

```
   Data: graph
 1 while reading input do
 2 |   calculatePopularity(graph);
 3 |   sortFacultiesByPopularity();
 4 |   sortCoursesByPopularity();
 5 |   for each course in courses do
 6 |   |   for each faculty in faculties do
 7 |   |   |   if course in faculty.preferenceList and
   |   |   |    faculty.hasSufficientCredits(course) then
 8 |   |   |   |   assignCourseToFaculty(course, faculty);
 9 |   |   |   end
10 |   |   end
11 |   end
12 |   introduceRandomness();
13 |   localSearch();
14 end
```

6 Input-Output

The input is in the form of a CSV File of the folllowing format :
Column 1 : Professor name
Column 2 : Array of UGCDC choices
Column 3 : Array of UGELECT choices
Column 4 : Array of HDCDC choices
Column 5 : Array of HDELECT choices
Column 6 : Professor category (x1, x2, x3)

Do note that the arrays contain the indexes of the courses, the course name corresponding to each index can be found in the code.

7 Test Cases

Table 1: Professor Course Preferences

Name	UGCDC	UGELEC	HDCDC	HDELEC	Category
Snehanshu Saha	4 5 3 7	2 0 1 6	2 3 0 4	3 1 2 4	x3
Ramprasad S. Joshi	0 1 2 8	2 3 0 4	3 1 2	1 2 3	x1
Harikrishnan N. B.	2 3 1 6	3 1 2 4	3 2 1	2 0 1	x2
A. Baskar	1 3 2 4	1 3 0 5	3 1 2 4	1 2 3 4	x3
Aditya Challa	2 3 0 9	3 1 2 5	2 3 0	2 0 1 4	x2
Arnab Kumar Paul	0 1 2 5	2 3 1 5	3 1 2	1 2 3	x3
Ashwin Srinivasan	1 3 0 4	3 1 2 6	3 2 1	2 5 0 1	x3
Biju K. Raveendran Nair	0 1 3 6	2 3 0 4	2 3 0	1 2 6 3	x2
Devashish Gosain	1 2 3	1 2 3 4	3 2 0	3 0 1	x2
Sougata Sen	2 3 1	3 1 2 3	3 2 1	2 0 1	x2
Hemant Rathore	1 3 2	1 3 0 4	3 1 2 4	1 2 3	x3
Basabdatta Bhattacharya	2 3 0	3 1 2 5	2 3 0	2 0 1	x2
Bharat Madhusudan Deshpande	0 1 2	2 3 0 6	3 1 2	6 1 2 3	x3
Diptendu Chatterjee	1 3 0	3 1 2 5	3 2 1	2 0 1	x3
Kanchan Manna	0 1 2	2 3 0 6	3 1 2	1 2 3	x3
Kunal Kishore Korgaonkar	2 3 1	3 1 2	3 2 1 4	2 0 1	x2
Neena Goveas	1 3 2	1 3 0	3 1 2	1 2 3	x3
Rajesh Kumar	0 1 2	2 3 0	3 1 2	1 2 3	x3
Santonu Sarkar	2 3 1	3 1 2	3 2 1	2 0 1	x2
Vinayak Naik	1 2 3	1 2 3	3 2 0 4	3 0 1	x2
Sujith Thomas	0 1 2	2 3 0	3 1 2	5 1 2 3	x2
Swaroop Joshi	2 3 0	3 1 2	2 3 0	2 0 1 4	x2
Swati Agarwal	1 3 2	1 3 0	3 1 2	1 2 3	x3
Tanmay Tulsidas Verlekar	2 3 1	3 1 2	3 2 1	5 2 0 1	x2

7.1 Test Case 1

```
Output: 1

1. Ramprasad S. Joshi
    1. CS F342 CompArch
2. Aditya Challa
    1. CS F351 ToC
3. Snehanshu Saha
    1. CS F301 PoPL
    2. CS C623 Advanced Operating Systems
4. Swaroop Joshi
    1. CS G513 Network Security
    2. CS G525 Advanced Computer Networks
5. Basabdatta Bhattacharya
    1. CS G513 Network Security
```

    2. CS G525 Advanced Computer Networks
6. Biju K. Raveendran Nair
    1. CS F215 DD
    2. CS F11 CP1
7. Sujith Thomas
    1. CSG524 Advanced Computer Architecture
    2. CS F11 CP1
8. Diptendu Chatterjee
    1. CS G526 Advanced Algorithms and Complexity
    2. CS F214 LCS
    3. CS F12 CP2
9. Ashwin Srinivasan
    1. CS F222 DisCo
    2. CS F214 LCS
    3. CS F12 CP2
10. Bharat Madhusudan Deshpande
    1. SS G554 Distributed Data Systems
    2. BITS F463 CRYPTOGRAPHY
11. A. Baskar
    1. CS F222 DisCo
    2. SS G513 Network Security
12. Hemant Rathore
    1. BITS F386 QUANTUM INFO & COMPUTING
13. Arnab Kumar Paul
    1. CS F213 OOP
    2. BITS F452 BLOCKCHAIN TECHNOLOGY
14. Tanmay Tulsidas Verlekar
    1. CS G523 Software for Embedded Systems
15. Rajesh Kumar
    1. CSG524 Advanced Computer Architecture
    2. CS G526 Advanced Algorithms and Complexity
    3. BITS F311 Image Processing
16. Devashish Gosain
    1. BITS F386 QUANTUM INFO & COMPUTING
    2. SS G527 Cloud Computing
17. Vinayak Naik
    1. CS C623 Advanced Operating Systems
    2. SS G527 Cloud Computing

18. Kanchan Manna
    1. BITS F463 CRYPTOGRAPHY
    2. BITS F311 Image Processing
    3. CS G568 Network Security Projec
19. Sougata Sen
    1. CS F13 CP3
    2. BITS F343 FUZZY LOGIC & APPL
20. Santonu Sarkar
    1. CS F13 CP3
    2. BITS F343 FUZZY LOGIC & APPL
21. Harikrishnan N. B.
    1. CS F215 DD
    2. CS G553 Reconfigurable Computing
22. Neena Goveas
    1. CS G568 Network Security Projec
    2. BITS F312 NEURAL NET & FUZZY LOGIC
    3. BITS G553 Real-Time Systems
23. Kunal Kishore Korgaonkar

```
    1. CS G553 Reconfigurable Computing
    2. BITS F364 HUMAN COMP INTERACTION
24. Swati Agarwal
    1. BITS F312 NEURAL NET & FUZZY LOGIC
    2. BITS G553 Real-Time Systems
    3. BITS F364 HUMAN COMP INTERACTION

Total Number of Courses Assigned is: 27

... (further outputs) ...

Time efficiency : 304 outputs in 0.0202256 seconds
```

In this particular scenario, the algorithm successfully assigned 27 out of the 29 available courses, taking into account the individual course preferences and categories specified by each faculty member. The assignments were tailored to match the unique preferences and categories of each professor, ensuring a comprehensive distribution of courses.

7.2 Test Case 2

INPUT :

Table 2: Professor Course Preferences

Name	UGCDC	UGELEC	HDCDC	HDELEC	Category
A. Baskar	1 3 2 4	1 3 0 5	3 1 2 4	1 2 3 4	x3
Aditya Challa	2 3 0 9	3 1 2 5	2 3 0	2 0 1 4	x2
Arnab Kumar Paul	0 1 2 5	2 3 1 5	3 1 2	1 2 3	x3
Ashwin Srinivasan	1 3 0 4	3 1 2 6	3 2 1	2 5 0 1	x3
Biju K. Raveendran Nair	0 1 3 6	2 3 0 4	2 3 0	1 2 6 3	x2

OUTPUT :

```
(Terminating Program)
CRASH TEST: NO PROFESSOR HAS TAKEN COURSE CS F301 PoPL
```

8 Crash Test

The algorithm concludes its execution under two conditions: either when all assignments are successfully completed, or when it identifies an impractical scenario where no feasible assignment aligns with both the professors' preference lists and the hard constraints. To be more specific, termination occurs in the following scenarios:

1. If there are UGCDCs that remain unchosen by any professor.
2. If there are HDCDCs that remain unchosen by any professor.

Under these circumstances, the algorithm returns the course that is unchosen and terminates.

9 Consistency Report

To evaluate the consistency of the algorithm, we define the following metrics and criteria:

- **Accuracy:** The correctness of the assignments made by the algorithm is 100% as each assignment makes sure that the faculty category and preference order are taken care of and also that a course is completely assigned.
- **Precision:** The algorithm gives multiple solutions out of which most of the first few outputs will always be sub-optimal solutions of the problem.

The input format for the algorithm is the preference order of faculty for the courses they decide to teach, and the output is the assignment of faculty with consideration for their category constraints. Additionally, the algorithm ensures that a course is not assigned partially.

10 Future Possible Extensions

10.1 Genetic Algorithm Integration:

The algorithm's capabilities can be further extended by incorporating a genetic algorithm paradigm. Post the initial assignment, the introduction of fitness, mutate, and crossover functions within a genetic algorithm framework can enhance the optimization process, potentially leading to improved solutions.

10.2 Generalization for Diverse Professor Categories:

The algorithm's adaptability can be further demonstrated by extending its functionality to accommodate a broader spectrum of professor categories. This generalization would enhance its utility across various academic scenarios, allowing for more inclusive and flexible assignment solutions.

10.3 Application in Other Operations Research Problems:

Leveraging the algorithm's effectiveness in solving the course faculty assignment problem, it can be applied to address additional Operations Research problems like Employee shift scheduling, Exam Scheduling,etc. Its versatility can potentially streamline solutions for problems beyond academic scheduling, showcasing its applicability in diverse operational scenarios.

10.4 Development of User-Friendly Software:

A comprehensive software solution can be developed with a user-friendly frontend interface. This software would serve as a practical tool for faculties to submit their preferences and for Head of Departments (HODs) to efficiently assign courses. This integration of technology can enhance user experience, making the entire assignment process more accessible and intuitive.

By exploring these future extensions, the algorithm not only expands its scope and applicability but also contributes to the development of advanced solutions in the field of academic scheduling and beyond.

References

1. Andres Ramos and Tomas Santero, "The biquadratic assignment problem: A new mathematical programming model and computational results," *Reliability Engineering & System Safety*, vol. 189, pp. 106823, 2019. DOI: `https://www.sciencedirect.com/science/article/pii/S0305054819300486`

2. A. A. Popov et al, "A GRASP for the biquadratic assignment problem," *Journal of Physics: Conference Series*, 2020.

3. Author Name, "A GRASP for the biquadratic assignment problem," Year. `https://www.academia.edu/2933036/A_GRASP_for_the_biquadratic_assignment_problem`

Index