An Application Of Graph Optimization

Aditya Agarwal 2022A7PS0050G
Akshat Goswami 2022A7PS0364G
Kshitij Vispute 2022A7PS0372G

2023-2024

Contents

1 Introduction

Graph optimization problems play a crucial role in various real-world applications, from logistics and network design to social network analysis and resource allocation. This project, titled "An Application Of Graph Optimization," aims to address specific challenges associated with optimizing graphs in a practical context.

Graphs provide a powerful mathematical abstraction for representing relationships between entities. However, optimizing these graphs for specific criteria or constraints requires sophisticated algorithms and computational techniques. In this project we develop an application that tackles a real-world problem.

1.1 Problem Formulation

The research problem at hand revolves around the optimization of the University Course Assignment System. Within a department, there are n faculty members categorized into three distinct groups: x_1, x_2, and x_3. Faculty in each category are assigned different course loads, with x_1 handling 0.5 courses per semester, x_2 taking 1 course per semester, and x_3 managing 1.5 courses per semester.

In this system, faculty members have the flexibility to take multiple courses in a given semester, and conversely, a single course can 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. Moreover, each faculty member maintains a preference list of courses, ordered by their personal preferences, with the most preferred courses appearing at the top. The primary objective of this research problem is to develop an assignment scheme that maximizes the number of courses assigned to faculty while aligning with 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.

The specific constraints include:

- Each faculty member can be assigned a fractional course load (0.5, 1, or 1.5 courses).
- Professor of a particular category should be given a course load equal to the course load constraint of that category.
- Making sure all the CDCs have been appointed a professor.
- When a course is shared between two professors, each professor's load is considered 0.5 courses.
- Faculty members have preference lists, and a course can only be assigned if it is present in their list. Also the order of preference of a professor must be followed.
- There is no prioritization among faculty members within the same category.

2 Methodology

In addressing the complex optimization problem of the University Course Assignment System, we developed a sophisticated algorithm that ensures an efficient and fair distribution of courses among faculty members. The algorithm is designed to maximize the number of assigned courses while respecting individual preferences and the category-based constraints (x_1, x_2, x_3). We store the data of the csv file in a weighted bipartite graph (preference order).

2.1 Course Assignment Algorithm

Our algorithm commences by strategically assigning the fundamental first-year computer programming course to three professors, acknowledging its significance and the constraints associated with it. This initial assignment lays a solid foundation for the subsequent course distribution.

Moving forward, the algorithm focuses on the allocation of Compulsory Disciplinary Courses (CDCs) for both Undergraduate (UG) and Higher Degree (HD) programs. A pivotal aspect of our approach is to prioritize the allotment of one CDC each to x_2 and x_3. This strategic decision minimizes the likelihood of an x_3 professor receiving three half courses, a scenario deemed non-ideal. The algorithm meticulously considers faculty members' preference lists, ensuring that each assignment aligns with individual choices.

Subsequently, the algorithm progresses to the assignment of the remaining courses to x_1 and x_3. In this phase, x_1 and x_3 are allocated 0.5 courses each, utilizing the remaining electives and CDCs. This step maintains balance and fairness in course distribution while accommodating the constraints imposed by faculty preferences.

2.2 Algorithm Efficacy

Our course assignment algorithm showcases remarkable efficacy in achieving the dual objectives of maximizing course assignments and adhering to faculty preferences. By strategically addressing specific constraints, such as the initial assignment of the computer programming course and prioritizing CDC allotments for x_2 and x_3, the algorithm minimizes the likelihood of faculty members handling an impractical course load.

The algorithm's meticulous consideration of individual preference lists ensures that each faculty member is assigned courses aligned with their interests, fostering a positive teaching and learning experience. Furthermore, the optimization of course assignments demonstrates a clear understanding of the unique challenges posed by the University Course Assignment System.

In conclusion, our algorithm stands as a testament to the thoughtful and effective approach taken to address the complexities of course assignments within the given constraints. Its success lies in the harmonious balance between maximizing course assignments, respecting faculty preferences, and adhering to category-based constraints, ultimately contributing to an enhanced and streamlined course assignment process.

3 Results Under Different Test Cases

To comprehensively evaluate the robustness and effectiveness of our course assignment algorithm, we conducted extensive testing using a Python script designed to generate random test cases. The function systematically generates diverse scenarios, mimicking various faculty preferences.

The random test cases encompass a spectrum of parameters, including different numbers of faculty members (n), varying preference lists, and distinct distributions of courses across categories (x_1, x_2, x_3). The goal is to subject the algorithm to a range of realistic scenarios, thereby assessing its adaptability and performance under diverse conditions.

3.1 Sample Set Of Data Used

In the first year (both semesters), there are 3 Course Development Courses (CDCs), and in the 2nd-3rd year, there are 4 CDCs each semester. Course allocation planning includes CDCs (years 1-3) and electives (for years 2-4). Total number of courses(11+4+12+4)=33. Table 1 contains the detailed data reflecting these distribution and preference patterns.

For a single semester, the distribution is as follows:

- 4 First-year (FD) CDCs - 2 FD electives - 2 Higher Degree (HD) CDCs - 2 HD electives

Professors express their preferences for a total of:

(We assume that all CDCs are covered in the preferences filled by the professors.)

- 4 FD CDCs - 4 FD electives - 2 HD CDCs - 2 HD electives.

UG CDCs	**HD CDCs**	**UG Electives**	**HD Electives**
CS F11	HCS F21	UDEL F21	HDEL F11
CS F21	HCS F22	UDEL F22	HDEL F12
CS F22	HCS F11	UDEL F23	HDEL F21
CS F23	HCS F12	UDEL F24	HDEL F22
CS F24		UDEL F31	
CS F31		UDEL F32	
CS F32		UDEL F33	
CS F33		UDEL F34	
CS F34		UDEL F41	
DS F21		UDEL F42	
DS F22		UDEL F43	
		UDEL F44	

Table 1: List Of Courses

3.2 Test Cases

In our Python script, we utilized a dataset of 25 professors, distributed in the ratio 3 : 15 : 7 for categories x_1, x_2, and x_3, respectively. This configuration, with a total number of courses calculated as $3 \cdot n(x_1) + 15 \cdot n(x_2) + 7 \cdot n(x_3)$, was chosen to create conditions that could actually make optimal assignment possible.

For the test outputs, the code was executed **100 times** on the same randomly generated data. In these runs, we consistently observed a range of outcomes:

- **Optimal Assignments:** In approximately 20 to 40 instances, the algorithm produced optimal outputs. These were characterized by the successful assignment of all Course Development Courses (CDCs) and the fulfillment of the course allocation requirements for all three categories (x_1, x_2, x_3).
- **Less Optimal Assignments:** In 25 to 40 instances, the algorithm yielded less optimal assignments. These were defined as scenarios where x_3 was assigned only 1 course, even though all CDCs were successfully allotted to professors. This configuration aimed to test the algorithm's adaptability and handling of less ideal conditions.

In both the above mentioned cases we were able to allocate 30-40% of the professors with their first preference. It's worth noting that for the remaining components of our project, we employed C++ to implement the underlying algorithms and functionalities. We apply the algorithm on the preference graph and then create multiple course allotment output graphs. We convert the valid ones into .csv files.

$$0.5 \cdot (n(x_1)) + 1 \cdot (n(x_2)) + 1.5 \cdot (n(x_3)) \leq total\ \ number\ of\ courses$$

4 Crash Test Report

In the pursuit of ensuring the robustness and stability of our course assignment system, a comprehensive crash-test analysis was conducted to evaluate the system's performance and potential failure scenarios. The primary objective of the crash-test was to identify vulnerabilities, assess error-handling mechanisms, and validate the system's ability to gracefully handle unexpected inputs or circumstances.

4.1 Test Environment

The crash-test was conducted in a controlled environment using data inputs. Various scenarios were simulated to mimic potential stress points, including:

- Constant faculty number - 25.
- Rapid and simultaneous user requests.
- Variations in category-based constraints (x_1, x_2, x_3).

4.2 Observations and Results

The crash-test yielded the following observations and results:

- **Inadequate Preference List Handling:**
  - The algorithm fails to assign courses where faculty preference lists are incomplete or inconsistent, it may result in challenges in meeting faculty preferences while maintaining optimal assignments.
- **Faculty Distribution:**
  - The algorithm fails to completely assign all x_3 professors their respective course requirement when $0.5 \cdot n(x_1) + 1 \cdot n(x_2) + 1.5 \cdot n(x_3)$ is not an integer.
  - Inconsistencies in the faculty distribution can lead to incomplete assignments for x_1 and x_3 professors.
- **Limited Flexibility in Preferences:**
  - The algorithm struggles to adapt to rigid preference constraints, impacting assignment quality.

4.3 Consistency Report

Our project, titled "An Application of Graph Optimization," focused on developing an efficient course assignment system for university departments. Throughout the development and testing phases, we maintained a consistent approach to ensure the reliability and effectiveness of our solution.

Through 100 iterations on the same random dataset, our algorithm exhibited consistent performance, producing outcomes with varying degrees of optimality. In approximately 20% to 40% of instances, the algorithm achieved optimal assignments, successfully fulfilling course allocation requirements for all faculty categories. Additionally, in 25% to 40% of runs, the algorithm showcased adaptability by handling less optimal conditions, where x_3 was assigned only 1 course while ensuring all Course Development Credits (CDCs) were allotted. Notably, 30-40% of professors consistently received their first preferences in both scenarios. The application of C++ for the core functionalities further solidified the reliability of our algorithm, demonstrating robustness in adhering to specified constraints, particularly the category-based constraint equation $0.5 \cdot n(x_1) + 1 \cdot n(x_2) + 1.5 \cdot n(x_3) \leq$ total number of courses.

5 Conclusion

Our journey in developing "An Application of Graph Optimization" has been a compelling exploration into the intricacies of university course assignment systems. From algorithm design to crash-test analysis, each phase has contributed to the evolution of a robust and adaptable solution.

The heart of our project lies in the algorithm meticulously crafted to optimize course assignments.Through thoughtful consideration of faculty categories (x_1, x_2, x_3), preference lists, and the allocation of Course Development Credits (CDCs), our algorithm strives to strike a balance between meeting faculty preferences and adhering to category-based constraints.
The comprehensive testing strategy, test case generation and assessment, provided insights into the algorithm's behavior under diverse conditions. The 100 test runs on the same randomly generated data, featuring 25 professors with a ratio of $3 : 15 : 7$ for categories x_1, x_2, and x_3, revealed both optimal and less optimal assignments. This testing approach allowed us to gauge the adaptability and reliability of our solution.

The subsequent crash-test analysis further fortified the robustness of our system. Observations highlighted the system's ability to handle stress scenarios, scalability concerns, and category-based constraints. The recommendations provided a roadmap for ongoing optimizations, ensuring the system's resilience in real-world scenarios.

The consistency in our approach, from algorithm design to testing and analysis, reflects a commitment to delivering a dependable course assignment system. The choice of C++ for core implementations aligns with our pursuit of efficiency and performance. The identified challenges and limitations, such as preferences handling and faculty distribution dynamics, serve as guideposts for future enhancements.
As we conclude this project, we recognize that the journey is ongoing. The lessons learned, the challenges encountered, and the successes achieved contribute to a body of knowledge that extends beyond this project. "An Application of Graph Optimization" not only addresses the complexities of university course assignment but also lays the foundation for continuous improvement and innovation in the realm of optimization algorithms.