Optimal Project Allocation

Ru Zhang* Thomas A. Weber†

August 31, 2024

Abstract

The project allocation problem is a critical task in academic and organizational settings, where the objective is to match a set of students to a set of projects based on preferences while satisfying various constraints. This paper explores and compares three algorithms: a Greedy Algorithm, the Stable Matching Algorithm (Gale-Shapley), and a Score-Based Allocation Method. We assess their performance in terms of computational efficiency, stability of the match, and overall satisfaction. Our findings suggest that while the Stable Matching Algorithm ensures stability, the Score-Based Method provides a flexible approach to optimizing different criteria.

1 Introduction

The optimal assignment of students to projects is a fundamental problem in both educational and professional contexts. The problem involves matching students to projects based on their preferences while satisfying certain constraints like project capacities or specific skill requirements. This paper investigates three approaches to solving the project allocation problem: the Greedy Algorithm, the Stable Matching Algorithm (also known as the Gale-Shapley Algorithm), and a Score-Based Allocation Method.

1.1 Background

In educational institutions, especially at the graduate level, students are often required to participate in projects as part of their curriculum. These projects typically have a limited number of slots available, and students have preferences over which projects they wish to undertake. The challenge is to assign students to projects in a manner that is both efficient and fair, taking into account their preferences and the constraints of the projects.

*École Polytechnique Fédérale de Lausanne, Station 5, CH-1015 Lausanne, Switzerland. E-mail: ru.zhang@epfl.ch.

†Chair of Operations, Economics and Strategy, École Polytechnique Fédérale de Lausanne, Station 5, CH-1015 Lausanne, Switzerland. Phone: +41 21 693 0141. E-mail: thomas.weber@epfl.ch.

1.2 Objective

The objective of this study is to develop and evaluate different algorithms for solving the student-project allocation problem. We aim to compare these algorithms in terms of their computational efficiency, stability, and the overall satisfaction of the participants.

2 Problem Modeling

2.1 Data Structure

The dataset used for testing and evaluating the algorithms was derived from a comprehensive survey conducted among students during the Fall 2023 semester. This survey captured detailed information about student preferences, educational backgrounds, and skills. For a complete description of the survey structure and the data samples used, please refer to Appendix A.1.

The problem can be represented using the following primary data structures:

- **Student Preference Lists:** Each student has a list of projects ranked in order of preference. Suppose there are n students and m projects. We represent the students' preferences using a matrix $P \in \mathbb{R}^{n \times m}$, where P_{ij} denotes the preference score of student i for project j (the higher the score, the stronger the preference).
- **Project Capacity Constraints:** Each project has a capacity constraint that indicates the maximum number of students that can be assigned. This is represented by a vector $C \in \mathbb{R}^m$, where C_j denotes the maximum capacity of project j.
- **Student Assignment Variables:** We introduce a binary decision variable x_{ij}, where $x_{ij} = 1$ if student i is assigned to project j, and $x_{ij} = 0$ otherwise.

2.2 Constraints

The main constraints in the allocation process include:

- **Project Capacity Constraint:** The number of students assigned to each project must not exceed its capacity, i.e., for each project j, we have

$$\sum_{i=1}^{n} x_{ij} \leq C_j, \quad \forall j \in \{1, 2, \ldots, m\}.$$

- **Student Uniqueness Constraint:** Each student can only be assigned to one project, i.e., for each student i, we have

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i \in \{1, 2, \ldots, n\}.$$

- **Fairness Constraint:** The allocation should aim to minimize envy, ensuring that no student would prefer another student's assigned project more than their own. This can be achieved by maximizing overall satisfaction or minimizing the maximum dissatisfaction.

2.3 Objective Function

The objective function to be maximized varies depending on the algorithm. We can define different objective functions corresponding to different allocation strategies:

- **Greedy Algorithm:** This algorithm aims to maximize immediate student satisfaction. The objective function can be expressed as:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} \cdot x_{ij},$$

  where P_{ij} represents the preference score of student i for project j, and x_{ij} is the assignment variable.

- **Stable Matching Algorithm:** The goal of this algorithm is to avoid any blocking pairs, thereby ensuring a stable match. A stable match is defined as one in which, for any unassigned student i and project j, if i prefers j and j has the capacity, then j should not prefer another assigned student more than i.

- **Score-Based Allocation:** This method seeks to maximize a weighted score, which could include factors such as student satisfaction, project preferences, and overall fairness. The objective function can be defined as:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{m} w \cdot P_{ij} \cdot x_{ij} + \sum_{j=1}^{m} \sum_{i=1}^{n} w' \cdot Q_{ji} \cdot x_{ij},$$

  where w and w' are weights, and Q_{ji} denotes the preference score of project j for student i.

These mathematical formulations provide a formal framework for analyzing and solving the project allocation problem, allowing for a systematic evaluation and optimization of the allocation outcomes.

3 Algorithms Overview

3.1 Greedy Algorithm

The Greedy Algorithm is a straightforward approach that iteratively assigns each student to their most preferred project with available capacity. Although simple, it often fails to find globally optimal solutions, especially in complex scenarios.

3.2 Stable Matching Algorithm

The Stable Matching Algorithm, specifically the Gale-Shapley Algorithm, is designed to avoid instability in the allocation. It ensures that no student-project pair would rather be matched with each other than with their current assignments, thus avoiding "blocking pairs".

3.3 Score-Based Allocation

The Score-Based Allocation method assigns a score to each possible allocation, based on a predefined scoring function. The method then selects the allocation with the highest total score, aiming to balance various objectives like satisfaction and fairness.

4 Greedy Algorithm

4.1 Algorithm Description

The Greedy Algorithm begins by sorting the students based on their preference rankings. It then allocates each student to their highest-ranked available project, moving down the list until all students are assigned or all projects are filled.

4.2 Pseudo-Code

Algorithm 1 Greedy Algorithm for Project Allocation

Data: Students' preferences, Project capacities

Result: Assignment of students to projects

```
Sort students by preference ranking  for each student s in sorted list do
    for each project p in s's preference list do
        if p has available slots then
            Assign s to p  Update capacity of p  break
return Final assignment
```

4.3 Analysis

The Greedy Algorithm is computationally efficient, with a time complexity of $O(S \times P)$, where S is the number of students and P is the number of projects. However, it may result in suboptimal allocations because it does not consider the overall fairness or stability of the match. It prioritizes immediate assignment based on preferences, which can lead to scenarios where some students are left with less desirable projects if their top choices are already filled.

5 Stable Matching Algorithm

5.1 Algorithm Description

The Stable Matching Algorithm, commonly referred to as the Gale-Shapley Algorithm, is designed to find a stable matching between students and projects. In the context of student-project allocation, a matching is considered stable if there is no student-project pair who would both prefer to be matched with each other over their current assignments. This algorithm ensures that no "blocking pairs" exist, making it a widely used method in scenarios where stability is a key concern.

The algorithm operates by iterating over the students, allowing each to "propose" to their most preferred project that has not yet rejected them. Projects, in turn, tentatively accept the most preferred students up to their capacity and reject the rest. This process continues until all students are matched to projects, resulting in a stable outcome.

5.2 Pseudo-Code

Algorithm 2 Stable Matching Algorithm (Gale-Shapley)

Data: Students' preferences, Projects' capacities

Result: Stable assignment of students to projects

```
Initialize each project as unfilled  while there exists an unassigned student do
    Select the next unassigned student s  for each project p in s's preference list do
        if p has available slots then
            Assign s to p  Update capacity of p  break
        else if p prefers s over its least preferred current assignment then
            Replace the least preferred student in p with s  Reassign the replaced student  break
return Final stable assignment
```

5.3 Analysis

The Stable Matching Algorithm has a time complexity of $O(S \times P)$ in the worst case, where S is the number of students and P is the number of projects. It guarantees that the final allocation is stable, meaning there are no student-project pairs that would both prefer to be matched with each other over their current assignments. However, stability may come at the cost of overall satisfaction; students might not get their top choice, especially in cases where preferences are highly competitive.

Moreover, the outcome is sensitive to the order in which students make their proposals. Depending on the specific implementation (student-proposing vs. project-proposing), the final allocation may be

biased in favor of either the students or the projects.

6 Score-Based Allocation

6.1 Algorithm Description

The Score-Based Allocation method introduces a scoring function that evaluates the quality of each possible assignment based on multiple factors, such as student preferences, project demands, and overall fairness. The algorithm then searches for the assignment that maximizes the total score.

This method is flexible and allows for the incorporation of various metrics into the scoring function, providing a way to balance different objectives (e.g., satisfaction, fairness, diversity). Unlike the Greedy Algorithm or Stable Matching, which focus on immediate satisfaction or stability, the Score-Based Allocation method can optimize for a more comprehensive goal.

6.2 Pseudo-Code

Algorithm 3 Score-Based Allocation

Data: Students' preferences, Projects' capacities, Scoring function

Result: Optimal assignment of students to projects

Initialize an empty assignment **for** *each possible assignment* **do**

    Calculate the total score using the scoring function **if** *current score is higher than the previous best* **then**

        Update the best assignment

return *Assignment with the highest score*

6.3 Analysis

The time complexity of the Score-Based Allocation method depends on the complexity of the scoring function and the number of possible assignments. In general, it may be computationally expensive ($O(S! \times P!)$ in the worst case), especially for large-scale problems, because it requires evaluating every possible assignment. However, heuristics and optimization techniques, such as dynamic programming or genetic algorithms, can be employed to reduce the computational burden.

The main advantage of this approach is its flexibility. By adjusting the scoring function, the algorithm can be tailored to prioritize different criteria, making it suitable for scenarios where multiple objectives must be balanced. However, finding the right balance in the scoring function can be challenging and may require domain expertise.

7 Results and Conclusion

7.1 Algorithm Comparison

We compared the three algorithms based on several criteria, as summarized in Table 1.

Table 1: Comparison of Algorithms: Greedy, Stable Matching, and Score-Based Allocation

Criteria	Greedy Algorithm	Stable Matching Algorithm	Score-Based Allocation
Computational Efficiency	High	Medium	Low
Stability	Low	High	Depends on scoring function
Satisfaction	High (immediate)	Medium	High (tunable)

- **Computational Efficiency:** The Greedy Algorithm is the most computationally efficient, followed by the Stable Matching Algorithm. The Score-Based Allocation, while flexible, can be computationally intensive.
- **Stability:** The Stable Matching Algorithm guarantees a stable match, making it ideal in situations where stability is crucial. The Greedy Algorithm does not guarantee stability, and the Score-Based Allocation's stability depends on the scoring function.
- **Satisfaction:** The Greedy Algorithm often maximizes immediate satisfaction but may not lead to an optimal overall outcome. The Score-Based Allocation can be tuned to maximize satisfaction across different criteria, potentially offering a better overall solution.

7.2 Final Choice

The choice of algorithm depends on the specific needs of the allocation scenario:

- For scenarios where stability is paramount (e.g., avoiding blocking pairs), the Stable Matching Algorithm is the best choice.
- In situations where computational efficiency is the priority, and a quick, satisfactory solution is needed, the Greedy Algorithm is suitable.
- For complex scenarios requiring a balance of multiple objectives, the Score-Based Allocation offers the most flexibility, albeit at the cost of increased computational complexity.

A Appendix

A.1 Data Samples

We utilized the 2023 Fall semester student survey data, which was specifically designed to gather detailed information for the project allocation process. The survey was meticulously crafted to capture a broad range of student preferences, educational backgrounds, skills, and availability. Below is an overview of the survey structure:

- **Personal Information:**
  - *Last Name, First Name:* Basic identification details of the students.
  - *Email:* Contact information, used for correspondence.
- **Registration Status:**
  - Students were asked whether they were already successfully registered for the course, with options including ”Yes,” ”No,” or ”Other.”
- **Educational Background:**
  - The students were required to specify their educational background, choosing from a list that included various engineering disciplines, management, data science, and more.
  - *Major (Program):* Students selected their current major from options like ”Management of Technology and Entrepreneurship,” ”Mechanical Engineering,” ”Electrical and Electronical Engineering,” and others.
  - *MTE Minor:* Students indicated whether they were pursuing a minor in ”Management of Technology and Entrepreneurship” or another field.
- **Skills Assessment:**
  - Students self-rated their proficiency in various skills on a scale from Basic (1), Good (2), to Excellent (3). The skills included ”Business Economics,” ”Prototyping (Hardware),” ”Prototyping (Software),” ”System Engineering,” and ”Design Thinking.”
- **Availability (Schedule):**
  - Students were asked to indicate their general availability for team meetings throughout the week, specifying availability across different time slots from Monday to Friday.
- **Project Choices:**

- Students were asked to rank their top five project preferences from a list of ten projects, with each choice corresponding to a different industry partner or project focus. Examples of these projects included:
    * *Boschung Mecatronic AG - FAST IoT:* A project focused on IoT solutions for fast data transmission.
    * *Datwyler - Electrically Conductive Rubber for Strain Sensors:* Development of innovative materials for sensor applications.
    * *Logitech - Eco-Design Plastic:* Exploring sustainable materials for product design.
    * *ZF-Group - Next Generation Battery Electric Fuel Cell Commercial Vehicle:* Advancing energy solutions for commercial vehicles.

These survey responses provided crucial inputs for the student-project allocation algorithms. The preference rankings, along with the students' educational backgrounds and skill levels, were used to develop a matching strategy that maximizes overall satisfaction and project success rates. The data was instrumental in testing and refining the algorithms discussed in this report, ensuring they could handle real-world scenarios effectively.

Sample datasets used in this analysis are provided as supplementary material to this report, including anonymized student preference lists and project capacities.

A.2 Code Implementation

Here we provide sample implementations of the discussed algorithms in Python. The code includes functions for reading input data, processing the allocation, and outputting the results.

References

[1] GALE, D., AND SHAPLEY, L. S. (1962) "College Admissions and the Stability of Marriage," *The American Mathematical Monthly*, 69(1): 9-15.

[2] ROTH, A. E., AND SOTOMAYOR, M. (1992) "Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis," *Econometric Society Monographs*, Cambridge University Press.