Comparative Analysis of Greedy, Gale-Shapley, and Score-Based Methods for Optimal Project Allocation

Abstract

The project allocation problem is essential in various fields, including academic and organizational settings,
where efficient, stable, and satisfying assignments are crucial for maximizing engagement and productivity.
This paper presents a comparative analysis of three distinct algorithms for project allocation: the Greedy
Algorithm, the Stable Matching Algorithm (Gale-Shapley), and a Score-Based Allocation Method. The
Greedy Algorithm prioritizes computational efficiency, while the Gale-Shapley algorithm focuses on stability,
and the Score-Based Allocation Method allows flexibility in balancing multiple criteria. Our findings reveal
that while the Stable Matching Algorithm ensures stable matches, the Score-Based Method optimizes for
overall satisfaction based on different objectives, offering valuable insights into the trade-offs involved in
choosing each approach.
Keywords: project allocation; student preferences; Greedy Algorithm; Stable Matching Algorithm; Score-Based Allocation;
computational efficiency; stability
2010 Mathematics Subject Classification: 90C27; 91B68; 68W40

1 Introduction

The optimal assignment of students to projects is a foundational problem in both educational and professional settings, where
matching participants to preferred options is necessary for enhancing engagement, performance, and satisfaction. The studentproject
allocation problem is a specific instance of this matching challenge, where students express preferences over a set of
available projects, and the allocation process must respect various constraints, such as project capacities and skill requirements.
Effective solutions to this problem have the potential to improve educational outcomes, support students ’ career aspirations,
and ensure fair and efficient utilization of project resources.
In this paper, we explore three distinct approaches to solving the student-project allocation problem: the Greedy Algorithm,
the Stable Matching Algorithm (often referred to as the Gale-Shapley Algorithm), and a Score-Based Allocation Method.
Each approach offers unique advantages: the Greedy Algorithm prioritizes computational efficiency, making it suitable for
quick allocations in large datasets; the Stable Matching Algorithm, based on Gale-Shapley ’ s work, focuses on stability,

avoiding any ”blocking pairs” where students and projects might prefer each other over their current matches; and the Score-
Based Allocation Method provides flexibility, allowing for custom scoring functions that can balance criteria such as student
satisfaction, fairness, and project preferences. This comparative analysis highlights the strengths and limitations of each
algorithm, aiming to guide practitioners in selecting the most appropriate approach for specific allocation contexts.

1.1 Background

In educational institutions, particularly at the undergraduate and graduate levels, students are frequently assigned to projectbased courses or research initiatives as part of their curriculum. These projects typically have limited availability, and students hold preferences for certain projects based on interests, career goals, or skill alignment. The challenge lies in designing an allocation process that maximally aligns with students ’ preferences while respecting project constraints and ensuring fair access to opportunities.

In practice, student-project allocation is a complex multi-agent problem with competing interests and often incompatible preferences, making it computationally challenging to find optimal solutions. To address this, institutions commonly adopt different algorithms to balance between optimality, fairness, and feasibility. For example, a simple greedy allocation can quickly assign students based on preferences, but it may not result in a stable or fair solution. More sophisticated approaches, like the Stable Matching Algorithm, aim to produce stable outcomes where no student and project would prefer each other over their current allocation. Meanwhile, score-based methods allow for nuanced prioritization based on criteria like academic performance or specific skill sets.

1.2 Objective

The objective of this study is to systematically evaluate the performance of three different algorithms for student-project allocation. Specifically, we compare the algorithms in terms of computational efficiency, allocation stability, and overall participant satisfaction. By assessing these aspects, we aim to determine which method best balances the competing priorities in student-project allocation scenarios and can most effectively accommodate student preferences and project constraints.

1.3 Literature Review

The student-project allocation problem has been extensively studied within the field of operations research, with applications ranging from education to workforce assignments [12, 2]. Recent studies have also explored hybrid and heuristic approaches, such as genetic algorithms, simulated annealing, and machine learning techniques, to address complex allocation challenges and improve performance in large datasets [9, 17]. These approaches show promise in optimizing allocation solutions when balancing criteria such as stability, fairness, and computational efficiency is critical. The rise of Industry 4.0 has further emphasized the importance of effective allocation methods, particularly with the integration of Internet of Things (IoT) and advanced data analytics in production processes [7, 6, 5].

Several key approaches have been proposed:

• Stable Matching and the Gale-Shapley Algorithm: The StableMatching Algorithm, developed by Gale and Shapley [2], has been foundational in allocation problems, particularly in contexts where stability is prioritized. Roth ’ s work [12] further established the importance of stability in matching problems, showing how instability can lead to dissatisfaction and inefficiency. In educational settings, stable matching is used to prevent scenarios where a student-project pair would

prefer each other over their assigned match, promoting a fair and consistent allocation.

• Greedy Algorithms: Greedy algorithms provide a straightforward approach to allocation problems by assigning students to projects sequentially based on preference rankings. Although simple, greedy methods often lack the stability of other approaches and may lead to suboptimal allocations where the overall satisfaction of participants is not maximized. Nonetheless, their computational efficiency makes them appealing for large datasets and real-time applications [10]. Extensions of greedy approaches have also been used in IoT and production quality control systems to balance operational efficiency with resource constraints [13, 4, 3].

• Score-Based Allocation Methods: In cases where students or projects can be ranked based on quantifiable criteria (such as test scores or prerequisite skills), score-based methods have been proposed as an alternative to preference-based allocation. These methods assign students based on scores, often prioritizing high-scoring students for competitive projects [1]. While score-based allocations can improve fairness by prioritizing merit, they may not fully consider individual preferences, potentially impacting student satisfaction [11]. Recent studies have applied similar score-based methods to prioritize tasks in crowdsourcing and co-creation contexts, enhancing the effectiveness of collaborative and distributed decision-making [8, 14].

• Hybrid and Heuristic Approaches: More recent studies have explored hybrid methods that combine aspects of greedy, stable matching, and score-based approaches to balance trade-offs. For example, heuristic algorithms such as simulated annealing or genetic algorithms have been used to optimize student-project allocations by iteratively improving a solution based on defined objectives [9]. Hybrid approaches integrating data-driven methods, such as kernel support vector machines and recurrent neural networks, have also been applied to improve performance in resource-intensive allocation tasks [16, 15].

In this paper, we build upon this body of research by testing and evaluating three representative algorithms: a Greedy Algorithm, the Gale-Shapley Stable Matching Algorithm, and a Score-Based Allocation Method. By comparing these approaches within a controlled framework, we aim to identify which algorithm offers the best balance of efficiency, stability, and participant satisfaction in the context of student-project allocation.

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 $P \in R_{n\times m}$, 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 $C \in R_m$, 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:

3

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

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

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

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

• 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, introduced by [2]. 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
1 Sort students by preference ranking;
2 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

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 largescale
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

7.2 Project Capacity Utilization

To evaluate the capacity utilization of each project under different algorithms, we plotted the ratio of assigned students to project
capacity. Figure 1 shows the utilization ratio for each project under the Greedy, Gale-Shapley, and Score-Based algorithms.

Figure 1: Project Capacity Utilization by Algorithm

As shown in Figure 1, all three algorithms manage to utilize project capacities effectively, with minor differences in
utilization ratios. The Greedy algorithm tends to slightly overfill certain projects, leading to maximum utilization in some
cases. The Score-Based algorithm, however, provides a more balanced approach, ensuring that all projects are adequately
filled without exceeding capacities. The Gale-Shapley algorithm shows stable utilization across projects but may underutilize
certain ones due to its emphasis on stability over strict capacity optimization.

7.3 Preference Satisfaction and Unassigned Students

In addition to capacity utilization, we evaluated each algorithm based on the average and median preference ranks of assigned
projects, as well as the number of unassigned students. Figure 2 summarizes these metrics for each algorithm.

Figure 2: Preference Satisfaction and Unassigned Students by Algorithm

8

From Figure 2, we observe that:
• Average Preference Rank: The Score-Based algorithm achieves the lowest average rank, suggesting higher satisfaction
in terms of students receiving projects closer to their top preferences. The Gale-Shapley algorithm, while stable, results
in a higher average rank, indicating that students may not always receive their most preferred projects.
• Median Preference Rank: Both the Greedy and Gale-Shapley algorithms show similar median ranks, while the Score-

Based algorithm achieves a slightly better median rank, which highlights its flexibility in optimizing satisfaction.

• Unassigned Students: All three algorithms perform similarly in terms of the number of unassigned students, indicating
that the algorithms are equally effective in maximizing assignments. However, the Score-Based algorithm marginally
reduces the number of unassigned students by better utilizing project capacities and balancing preferences.

7.4 Final Choice

The choice of algorithm ultimately depends on the specific needs and constraints of the allocation scenario. Our findings
offer practical guidance: the Stable Matching Algorithm is ideal for scenarios where stability and fairness are paramount,
such as in educational institutions seeking to prevent preference conflicts. The Greedy Algorithm, with its computational
efficiency, suits scenarios requiring quick, satisfactory solutions with minimal resources. The Score-Based Allocation Method,
while computationally intensive, offers flexibility, making it applicable for complex scenarios where multiple criteria must be
balanced, such as workforce allocations or specialized project assignments in industry. This comparative analysis provides
practitioners with a robust framework for selecting algorithms that align with their unique allocation requirements.

• 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] Atila Abdulkadiroglu and Tayfun S¨onmez. College admissions with affirmative action. International Journal of Game
Theory, 33(4):535–549, 2005.

[2] David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. The American Mathematical
Monthly, 69(1):9–15, 1962.
[3] H. Guo, R. Zhang, X. Chen, Z. Zou, T. Qu, G. Huang, J. Shi, M. Chen, and H. Gu. Quality control in production process
of product-service system: A method based on turtle diagram and evaluation model. In Procedia CIRP, volume 83, pages
389–393, 2019.
[4] H. Guo, R. Zhang, Z. Lin, T. Qu, G. Huang, J. Shi, M. Chen, H. Gu, and C. Deng. Research on task pricing of self-service
platform of product-service system. In Procedia CIRP, volume 83, pages 380–383, 2019.
[5] H. Guo, C. Xu, R. Zhang, J. Shi, T. Qu, C. Li, Y. Cai, X. Luo, and Z. He. Bibliometric analysis of internet of things

based on citespace. In Proceedings of the 25th International Conference on Industrial Engineering and Engineering
Management (IE&EM 2019), 2020.
[6] H. Guo, R. Zhang, T. Qu, C. Li, Z. Zou, Y. Zhou, Q. Chen, H. Jiang, and B. Chao. Research on the development situation
of industrial internet of things based on mapping knowledge domain. In Proceedings of the 25th International Conference
on Industrial Engineering and Engineering Management (IE&EM 2019), 2020.
[7] H. Guo, R. Zhang, Y. Zhu, T. Qu, M. Zou, X. Chen, Y. Ren, and Z. He. Sustainable quality control mechanism of
heavy truck production process for plant-wide production process. International Journal of Production Research, 58
(24):7548–7564, 2020.
[8] L. Lin, X. Chen, Y. Lou,W. Zhang, and R. Zhang. Task pricing optimization model of crowdsourcing platform. Business
and Management Studies, 4(3):44–51, 2018.
[9] Yannis Marinakis and Magdalini Marinaki. A hybrid algorithm for the project allocation problem. Computers &
Operations Research, 87:225–240, 2017.
[10] David L. Martin and Christopher Roberts. A heuristic for assignment problems with student preferences. In Proceedings
of the Annual Conference on Engineering Education, 2004.
[11] Parag A. Pathak and Tayfun S¨onmez. School admissions reform in chicago and england: Comparing mechanisms by
their vulnerability to manipulation. American Economic Review, 103(1):80–106, 2013.
[12] Alvin E. Roth. The two-sided matching problem: Origins, developments, and current research. Econometrica: Journal
of the Econometric Society, 60(4):1029–1050, 1992.
[13] Y. Wang, J. Wu, R. Zhang, S. Shafiee, and C. Li. A “user-knowledge-product” co-creation cyberspace model for product
innovation. Complexity, 2020:7190169, 2020.
[14] R. Zhang, C. Huang, and S. Chen. Futures trend strategy model based on recurrent neural network. Applied Economics
and Finance, 5(4):95–101, 2018.
[15] R. Zhang, C. Huang, W. Zhang, and S. Chen. Multi factor stock selection model based on lstm. International Journal of
Economics and Finance, 10(8):36, 2018.
[16] R. Zhang, Z. Lin, S. Chen, Z. Lin, and X. Liang. Multi-factor stock selection model based on kernel support vector
machine. Journal of Mathematical Research, 10(9), 2018.
[17] Yanqing Zhu, Bo Xu, and Hui Li. Machine learning for resource allocation. Journal of Operations Research, 34(3):
335–347, 2019.