Automatic Allocation of Group Projects

Student Name: Haris Javed
Supervisor Name: Thomas Erlebach
Submitted as part of the degree of BSc Computer Science to the
Board of Examiners in the Department of Computer Sciences, Durham University

Abstract—**Context/Background:** The Group Student-Project Allocation (Group-SPA) problem poses a notable challenge that involves the task of assigning students to project topics, taking into account their ranked preferences, while respecting stringent constraints. Traditional methods employed involve a trial-and-error approach which can be time-consuming and inefficient.
Aims: The primary objective of this research is to develop a network flow-based solution for the Group-SPA problem. The algorithm aims to maximize the preference satisfaction of students while respecting the uniform constraints on group sizes, and the stipulated number of groups per project topic. Further, our goal extends to the integration of this solution within a user-friendly web interface.
Method: We leverage real-world data from a university's computer science department and synthetic data to analyze the performance of our model. We further investigate the impact of varying parameters such as group size and project popularity on the quality of the allocation. The quality of the solution is quantified by a defined score function.
Results: Applying our model to real-world data, we observed that the network flow-based solution provides a good allocation within seconds. While the manual method demonstrated a better allocation quality given the defined constraints for this case, our model nonetheless produced a good solution. It strictly adhered to all constraints ensuring that no student remained unassigned. Comprehensive evaluations with synthetic data reveal insightful trade-offs and optimal conditions that govern the allocation process.
Conclusions: The findings demonstrate that our network flow-based solution to the Group-SPA problem handles constraints proficiently while maximizing preference satisfaction. This approach, when extended, can potentially address a broader spectrum of group allocation problems. The developed web interface enables the automated generation of good group allocations.

Index Terms—Graph algorithms, Network problems, Optimization, Graphical user interfaces

1 Introduction

IN ORDER to fulfill their upper-level degree requirements, university students must typically be assigned to individual or group projects under a supervisor. Students typically have ordinal preferences over the projects they find acceptable while supervisors can cover certain projects. Each student can be assigned to at most one project and there are usually constraints on the maximum number of students that can be assigned to each project and supervisor. Additionally, supervisors may have preferences over students, or over the projects they offer, or they may not have explicit preferences at all. The problem then is to assign students to projects in a manner that satisfies these constraints while taking preferences into account. The problem is referred to as the Student-Project Allocation problem in the literature [1], [2], [3]. The focus of our project lies within the SPA problem with group projects (Group-SPA) [4], [5].

Finding an optimal allocation of students to projects manually is time-consuming and error-prone. Given the large number of students that are typically involved in such an allocation process, many university departments seek to automate the allocation of students to projects. Examples include the University of Glasgow [6], the University of York [7], the University of Southern Denmark [4], and the Geneva School of Business Administration [8]. It is generally believed that matching problems are best solved through centralized schemes in which agents submit their preferences and a central authority calculates an optimal matching that meets all specified criteria [9]. These matching problems carry significance beyond academia, impacting the quality of life of the participants based on the outcome of the allocation process. For instance, if students are assigned to projects that do not align with their interests, their motivation could wane, potentially leading to low-quality outcomes and strained student-supervisor relationships [8]. Reference [6] suggest that students are more engaged and perform better when they are allocated to projects that align with their interests. Therefore, it is essential for the algorithms employed in these schemes to maximize participant satisfaction relative to their stated preferences as far as possible.

Matching problems require us to allocate a set A of agents to another set B of agents in order to find a matching that is optimal subject to certain criteria [3]. These types of problems can be represented graphically, with vertices representing the elements to be matched and edges representing the constraints or relationships between those elements. The Group-SPA problem is a bipartite matching task [10], encompassing two disjoint sets: students and project topics. Each edge in this bipartite graph represents a potential assignment of a student to a project group. The goal is to establish a many-to-one mapping between students and project topics, respecting specific conditions such as a project's capacity limit. Multiple students can be assigned to a single project topic. In this framework, students have preferences amongst project topics, but the reverse does not hold; project topics do not express preferences toward students. In many real-world SPA scenarios, like at The University of Glasgow [6], preferences are one-sided; students express their project preferences, while supervisors remain indifferent. Such settings result in a bipartite matching problem with one-sided preferences which are the focus of our paper.

Practical applications of the SPA problem present a variety of complex challenges, as discussed in [10], [12]. Multiple students collaborate on a single project, intensifying competition for top-choice projects. When numerous students express a preference for the same project, it is inevitable that some students may be assigned a project lower on their preference list. This can trigger a ripple effect, as these less preferred projects could be the first choice of others. The supervisor's workload is another crucial consideration in this context. Supervisors often propose more projects than they can feasibly manage, aiming to provide a wider selection for students. The popularity of a specific supervisor's projects can lead to a situation where some students inevitably miss out on their first-choice project. It further underscores the importance of load balancing to prevent supervisors from being overburdened due to a project's popularity. Real-world implementations also necessitate that all students are assigned a project, which can pose a problem when some students fail to submit their complete preferences.

To accommodate these challenges, we integrate strict upper and lower bounds on group sizes and impose restrictions on topic availability into our model. This ensures a balance across project topics with groups of comparable sizes. Topic capacity constraints are employed to ascertain that no project topic accommodates more groups than permitted. Further, we assume that each supervisor overlooks a distinct project topic. In situations where students fail to submit some or all preferences, we allocate a random project topic to them while adhering to the defined constraints.

Incorporating lower bounds into the equation adds an element of complexity, possibly leading to scenarios where a feasible solution is not necessarily guaranteed, as posited in [10]. Moreover, [6] suggest that enforcing this constraint might inadvertently diminish student satisfaction levels. Nonetheless, in real-world scenarios, the priority lies in ensuring a fair distribution of supervisory responsibilities among the lecturing staff, given that it significantly impacts their performance in other academic and administrative duties. Consequently, our focus transitions towards obtaining a good solution as opposed to an optimal one. The goal is to construct allocations that can maximise student satisfaction to the greatest extent possible. To achieve this, careful consideration of strict constraints is required while also giving due respect to student preferences.

Bipartite matching problems can be aptly conceptualized as networks, where the objective is to identify a flow that translates into a matching fulfilling predetermined criterion. SPA has been investigated in the network flow context [6], [10] where a minimum cost maximum flow algorithm is used to find a minimum cost maximum matching, demonstrating polynomial time complexity in the context of bipartite matching with one-sided preferences. Moreover, the models depicted in [6], [11] display their adaptability by incorporating lower quotas, demonstrating their robustness in accommodating a wide array of constraints. While these methods have demonstrated success in standard SPA situations, their utilization in the context of group allocations remains unexplored. This gap provides a powerful rationale for the exploration of network flow models within the Group-SPA problem. Owing to their intrinsic adaptability in handling varied constraints, network flows propose a promising strategy for tackling the constraints of the Group-SPA problem.

The research question we aim to address is: Can we develop an effective framework for the Group-SPA problem using a network flow model, such that it not only produces a good allocation based on individual preferences but also adheres to group size and topic capacity constraints?

The overarching aim of our research is to investigate how the application of network flow models, can optimize the intricate process of assigning student groups to their respective project topics in real-world academic settings. The major objectives we aim to accomplish within the scope of our research are as follows:

1) **Modeling the Group-SPA problem:** Our first objective is to model the problem as a bipartite matching graph, which is subsequently transformed into a flow network. This approach aids in understanding the constraints of the problem and incorporating them into a network flow model.
2) **Develop an optimized algorithm:** Next, we focus on devising a robust algorithm, leveraging the min-cost max-flow model. The goal of this algorithm is to generate allocations that adhere to defined constraints while maximizing respect for student preferences.
3) **Implementation in a user-friendly tool:** Once the algorithm is developed, we seek to implement it within a user-friendly web application. This tool is designed to allow easy input of students' preferences and capacities and, after processing, provides a downloadable file detailing the allocation, along with a log file explaining the allocation process.
4) **Comprehensive evaluation of the algorithm:** Our final objective involves assessing the algorithm's performance in academic allocation scenarios. By utilizing real-world data and complementing it with adjustable synthetic datasets, we seek to assess the algorithm's ability to meet student preferences while adhering to academic constraints. Key benchmarks include the quality of matches (cost) and student satisfaction levels (degree of matching to preferences), with experiments tailored to explore varying group capacities and project popularity levels.

This paper is organized as follows. We review the literature related to our problem in Section 2. Our main contribution is in Section 3 where we model the student-project assignment problem within the framework of network flows. Section 4 presents numerical experiments to evaluate the performance of the proposed algorithm; Section 5 provides an evaluation of these results. Concluding remarks and limitations are provided in Section 6.

2 Related Work

Our research into the Group Student Project Allocation (Group-SPA) problem draws from established combinatorial optimization techniques applied in diverse academic contexts such as the universities of Southampton, Mauritius, Glasgow, Southern Denmark, and Usmanu Danfodiyo. These techniques, encompassing integer programming [2], [4], constraint programming [5], [7], and network flows [6], [10], [11], provide the foundational understanding of our work. Furthermore, we investigate comparable problems in other domains, such as conference management [14], medical resident allocations [17], and the college admissions problem [16]. Each of these contexts presents unique allocation challenges and methodologies that can be adapted and applied to the Group-SPA problem.

Reference [2] present two integer programming models for project assignment. The first model is a one-to-one assignment, involving two phases: an initial phase that minimizes the maximum number of projects supervised by a single lecturer to balance workload (using a minimax criterion), and a subsequent phase that optimizes a weighted sum of preferences to maximize first-choice assignments while maintaining balance. The second model sets lower and upper bounds on group sizes, optimizing a weighted sum criterion considering both individual preferences and group constraints. Both models are computationally efficient, successfully applied to Individual and Group Projects at the University of Southampton. While these approaches align with maximizing first-choice assignments, considering one-sided preferences, and including group capacity constraints, they do not address multiple groups per project topic or workload balancing in group allocation, which are the focus of our paper.

Reference [18] present a project allocation framework for the University of Mauritius, focusing on two core objectives: maximizing students' satisfaction by granting their top preferences, and evenly distributing the supervisory and examination workload. The study demonstrated that the system's efficiency allowed 82% of students to secure their first-choice project among the 30 proposed by 15 supervisors across 11 teams. Our project derives inspiration from these concepts of student preference maximization and equitable load balancing, aiming to incorporate them into our own unique context and requirements.

In [6], the traditional SPA problem is restructured within a network flow framework, utilizing a minimum cost maximum flow algorithm to identify a minimum cost maximum matching. The flexibility of this new model is emphasized through its capability to accommodate varying project capacities, lecturers' workloads, and the integration of diverse constraints. Our research echoes this approach as we similarly restructure a one-sided bipartite matching problem into a network, utilizing a minimum cost maximum flow algorithm. The approach presented in [6], presents a blueprint adaptable to the Group-SPA problem.

Similarly, [10] focus on the development of network flow algorithms to find greedy maximum matchings and generous maximum matchings within the SPA context. The Greedy maximum matching strives for a profile that is lexicographically maximum, aiming to assign as many students as possible to their top-choice projects. Conversely, the Generous matching seeks a profile that is lexicographically minimum in reverse, favouring a distribution that considers the lower preference choices of students to ensure a fair allocation. Our focus lies in exploring how the network flow model can be applied to accommodate unique constraints within the Group-SPA problem, making matching concepts relevant. The approach also accounts for constraints like lecturer lower quotas, thus demonstrating adaptability in handling additional conditions. Even though lower quotas are applied in the context of individual projects, they can be restructured as lower bounds on group size.

Reference [4] focus on the student-project allocation problem at the University of Southern Denmark. Their methodology underscores group assignments, taking into account both the upper and lower bounds on group size and facilitating multiple groups for each project topic. This approach aligns with our problem definition; however, our constraints are uniform across all project groups and do not incorporate group registrations. Moreover, both our model and the one presented in [4] share a common objective: to establish fair project assignments that maximise overall satisfaction.

In [15], the scenario is evaluated where lecturers possess preferences over the pairing of students and projects. This model is commended for its flexibility and potential to decrease the quantity of unassigned students. However, in our specific use case, we find such preferences undesirable. It is argued that lecturers might favour certain students, which would result in an unjust assignment. Similarly, the need for lecturers to have preferences over project topics is seen as unnecessary since we assume that each lecturer proposes a unique project.

Reference [5] propose a distinct approach to the Group Student-Project Allocation problem by leveraging constraint optimization methodologies. In their initial testing round with 95 students, an impressive 90% were successfully allocated to their projects. Their model emphasizes two critical factors: maximizing first-choice allocations and abiding by group size constraints, both of which are key considerations in our project.

Our problem shares similarities with multiple domains that deal with allocation problems involving capacity constraints. In the domain of conference management, for example, the assignment of papers to reviewers presents a related challenge. Just as each student group is assigned a project in Group-SPA, each reviewer in a conference setting is assigned multiple papers to review [14]. This paper-reviewer assignment must balance the load of papers per reviewer while ensuring that each paper receives sufficient reviews, hence the presence of lower and upper capacity constraints. This is reminiscent of the capacity constraints in the Group-SPA problem where each project has a minimum and maximum number of students it can accommodate.

In the medical field, the allocation of students to hospitals has been modeled as a bipartite matching problem with lower and upper quotas. This allocation problem, known as the Hospitals / Residents problem with Lower Quotas [17], seeks a stable matching of residents to hospitals with a balance in the number of residents each hospital can take. While the context is different, the strategies developed for this allocation problem can provide insights into the Group-SPA problem.

Moreover, [16] present a remarkable parallel to group student project allocation. The authors explore two extensions to the College Admissions problem, both sharing similarities with project group allocation dynamics. Firstly, they investigated a scenario where colleges hold both lower and upper quotas, akin to the minimum and maximum student limits within a project group. Secondly, they considered a situation where subsets of colleges share common quotas, analogous to cases where there are a set number of groups per project topic. Hence, these approaches are clearly relevant to our project objectives.

In conclusion, our problem definition shares substantial similarities with the approach outlined in [4], with the notable exception of group registrations. The network graph model we have constructed is informed by and can extend the methodology presented in [6]. While this approach considers the SPA problem without groups, it concentrates on the challenges of bipartite matching with one-sided preferences and lower quotas for load balancing - areas that are pertinent to our research. Additionally, the empirical evaluation techniques found in [10], applicable to both real-world and randomly-generated datasets, align with our project objectives.

3 Methodology

Our methodology in addressing the Group-SPA problem incorporates mathematical modeling, computational algorithms, and user-friendly application design. The process initiates with the definition of the Group-SPA problem, capturing its intricate constraints and diverse preferences. The problem is then translated into a network graph, which serves as the primary computational model. The Minimum-Cost Maximum-Flow (MCMF) problem, a classical problem in network flow optimization, is adapted to find an initial assignment that maximizes preference satisfaction and adherence to the constraints. To reconcile the outcome with constraints not directly factored in the initial MCMF model, a tailored redistribution strategy is devised and applied. Subsequently, the algorithm and model are embodied in an accessible and interactive Graphical User Interface (GUI). This interface enables the input of project preference data, specification of constraints, and output of generated allocations. Our methodology converges on a comprehensive solution, balancing the objectives of maximizing preference satisfaction, constraint adherence, and user-friendly applicability, apt for deployment in real-world academic settings.

3.1 Design

3.1.1 Problem Formulation

We provide a formal definition of the Group Student-Project Allocation (Group-SPA) problem as it pertains to real-world academic settings. In this framework, we introduce a strategy that, while requiring only minimal modifications, demonstrates significant applicability for resolving a comprehensive array of similar group allocation problems.

An instance I of the Group-SPA problem is defined by a set $S = \{s_1, s_2, \ldots, s_n\}$ of n students, and a set $P = \{p_1, p_2, \ldots, p_m\}$ of m project topics. Let R be a preference relation such that for each student $s_i \in S$, there is an associated strict preference list ranking exactly four project topics.

Every project topic p_j is uniquely associated with a supervisor, and it can be partitioned into a fixed number of groups, denoted by t_{limit}, where each project topic forms a distinct set of groups.

There are uniform upper and lower quotas, namely g_{max} and g_{min}, across all project topics. These parameters determine the minimum and maximum number of students that can be assigned to each group, effectively defining the size constraints of the groups for all project topics.

We introduce a binary variable X_{ij}:

- $X_{ij} = 1$ if student i is assigned to project topic j.
- $X_{ij} = 0$ if student i is not assigned to project topic j.

Our objective is to assign students to project topics in such a way as to maximize student satisfaction, subject to the following constraints:

1) **Single Assignment Constraint**: For each student $s_i \in S$, the sum of binary variables X_{ij} over all topics $p_j \in P$ must equal one, i.e., $\sum_{j=1}^{m} X_{ij} = 1$. This constraint ensures that each student is assigned to exactly one project topic.
2) **Group Size Constraint**: For each project topic $p_j \in P$, the sum of X_{ij} over all students $s_i \in S$ must fall within the range $[g_{min}, g_{max}]$, i.e., $g_{min} \leq \sum_{i=1}^{n} X_{ij} \leq g_{max}$. This guarantees that the size of each project group aligns with the pre-specified lower and upper quotas.
3) **Topic Limitation Constraint**: For each project topic $p_j \in P$, the number of distinct groups, each defined as a subset of students assigned to the same topic, must not exceed the defined t_{limit}.

The objective is to optimize the assignment between students' preferences and their assigned project topics, aiming to maximize overall satisfaction. The satisfaction metric is defined by the proximity of the assigned topic to the students' preferred choices. A student assigned to their first choice will experience higher satisfaction compared to a student assigned to their fourth choice. This prioritizes fulfilling higher-ranked preferences while still considering lower-ranked ones.

To formulate this as an optimization problem, we define the objective function as a weighted sum of students' preferences:

$$Maximize \quad Z = w_1 \cdot X_1 + w_2 \cdot X_2 + w_3 \cdot X_3 + w_4 \cdot X_4 - w_r \cdot X_r \tag{1}$$

Here, X_k (for $k = 1, 2, 3, 4$) represents the number of students assigned to their k-th preference, and X_r represents the number of students assigned randomly. The weights w_k quantify the importance of each preference level.

This formulation enables us to capture the complexities of the Group-SPA problem while allowing for a systematic approach toward a solution that respects student preferences and administrative constraints. A depiction of a typical problem instance is shown in Fig1.

3.1.2 Network Graph Modeling

The Group-SPA problem focuses on a bipartite matching task, where students are matched to project topics. Each edge represents a potential student assignment to a project group, with the goal of achieving a many-to-one mapping, subject to capacity constraints and student preferences.

students' preferences:

s_1: $p_2\ p_3\ p_1\ p_4$

s_2: $p_2\ p_4\ p_3\ p_1$ project offered: $\{p_1, p_2, p_3, p_4, p_5\}$

s_3: $p_5\ p_3\ p_4\ p_2$ group capacities: $g_{min} = 2$, $g_{max} = 3$

s_4: $p_5\ p_1\ p_3\ p_2$ topic capacities: $t_{limit} = 1$

s_5: $p_5\ p_4\ p_3\ p_1$

Fig. 1. A Group-SPA instance I.

Our goal is to find a maximum matching in this setting. A maximum matching refers to a matching of maximum size, which entails the maximum number of edges in a graph such that no two edges share an endpoint [11]. In the context of the Group-SPA problem, this would mean finding an optimal assignment of students to project topics so that as many students as possible get assigned to a project of their preference.

To tackle this Maximum Bipartite Matching (MBP) problem, we apply a transformation that involves converting it into a flow network problem. By modelling our problem as a network, we can leverage network flow techniques to achieve our goal. This transformation involves converting our bipartite graph into a directed graph.

Following the approach outlined in [6], [19], the transformation process can be broken down into several stages, leading to a directed graph (or network):

- **Problem Representation**: The original bipartite graph $G = (V, E)$, where $V = S \cup P$ (students and project topics), is transformed into a directed graph, $G^N = (N, A)$, including additional source $s0$ and sink t.
- **Nodes Definition**: $N = S \cup P \cup \{s_0, t\}$ is the set of nodes in the network, comprising student nodes (S), project topic nodes (P), a super-source (s_0), and a super-sink (t).
- **Edges Formation**: Directed edges are formed extending from the source to all students, from students to each acceptable project topic, and from all project topics to the sink. The set of directed arcs is:

  $$A = (s_0, s_i) \cup (s_i, p_j) \cup (p_j, t)$$

  for $i = \{1, \dots, n_1\}, j = \{1, \dots, n_2\}$. Here:

  - (s_0, s_i) represents the set of arcs from the source to every student in S,
  - $(s_i, p_j) \in S(s)$ represents the union of the sets $S(s_i)$ of all student-project pairs (arcs) such that p_j are the projects which student s_i finds acceptable,
  - (p_j, t) represents the set of arcs connecting projects to the sink.

  Capacities and costs are denoted u_{ij} and c_{ij} for all arcs $(i, j) \in A$.
- **Weighting and Capacity Constraints**: Edges between students and project topics are weighted according to preference rank (c_{ij}), while capacities (u_{ij}) encapsulate the constraints, such as unique topic allocation for students and topic capacity limits.

The weighted maximum bipartite matching problem has been efficiently transformed into a minimum-cost maximum flow problem. This allows us to address Group-SPA using a network flow approach. In this graph model, student preferences and project constraints are intricately embedded as weights and capacities on the edges.

The transformation from a bipartite undirected graph $G = (V, E)$ to a weighted directed graph $G^N = (N, A)$ is depicted in Figure 2.

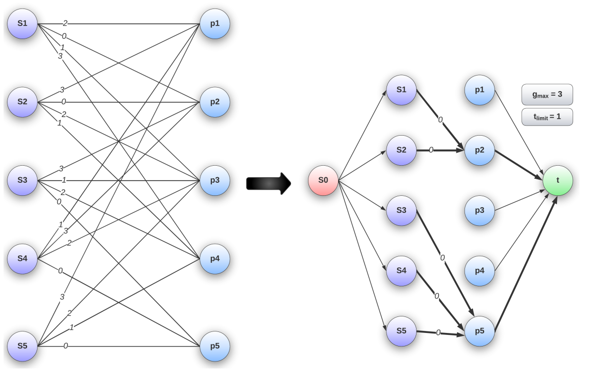

Fig. 2. Group-SPA network flow model for a bipartite matching (lowest weights are shown for visualization).

3.1.3 Minimum Cost Maximum Flow

By restructuring the Group-SPA problem in the context of a flow network, the Minimum Cost Maximum Flow (MCMF) algorithm emerges as an effective tool for obtaining a solution. The central challenge here is to ascertain the most cost-efficient method for conveying a specified flow quantity through the network while observing defined constraints on capacities and costs. Our aim is to identify the minimum-cost flow from the range of potential maximum flows.

The objective of the MCMF problem can be conceptualized as identifying a flow, denoted as $f : E \to R_{\geq 0}$, satisfying the following conditions [19]:

1) **Capacity Constraints:** $f(e) \leq c(e)$ for all $e \in E$. This ensures that the number of students assigned to any given project topic does not exceed its designated capacity.
2) **Conservation of Flow:** For all vertices except the source and sink, the total incoming flow equals the total outgoing flow. This simulates the complete assignment of students without any loss.
3) **Flow Maximization:** The total flow from the source is maximized. This corresponds to assigning as many students as possible to their preferred topics while adhering to constraints.
4) **Cost Minimization:** The chosen flow minimizes the total cost defined as $\sum_{e \in E} a(e) \cdot f(e)$. In the context of the Group-SPA problem, this aims to assign students to their higher-ranked preferences, thereby maximizing overall satisfaction.

In leveraging the MCMF algorithm to address the Group-SPA problem, two central objectives are accomplished:

1) **Minimizing Cost:** The costs in this formulation represent an inverse function of student preferences, with lower costs for higher preferences. Minimizing

the total cost corresponds to maximizing overall student satisfaction with their project topic assignments.

2) **Maximizing Flow:** The flow represents the assignment of students to project topics. Maximizing this flow ensures that all students are assigned to a group in line with their preferences, observing the constraints on group sizes and topic limitations.

3.1.4 Redistribution strategy

The initial solution, based on the Minimum Cost Maximum Flow (MCMF) algorithm, creates an assignment that satisfies the majority of constraints, such as upper bounds on group size and topic capacity. However, the initial construction of the directed flow network does not directly consider the minimum group size constraint. This discrepancy arises because the network flow algorithm optimizes for cost, which is influenced by student preferences and the maximum group size. The algorithm aims to find a flow that maximizes the capacity (satisfying the maximum group size) while also minimizing the cost (considering student preferences). This omission may result in some groups being smaller than the minimum group size prescribed, necessitating a more refined redistribution approach to ensure that the final assignments conform to all given constraints. The approach to redistribution follows a multi-step process:

1) **Combination of Topic-Based Groups:** The students assigned to each topic are combined into a single pool. This pool is subsequently partitioned into subgroups that adhere to a predetermined size range defined by the $[g_{min}, g_{max}]$ parameters. Any surplus students that cannot be accommodated within these constraints are marked for redistribution. This initial grouping serves as a foundational step to align the solution with the overarching group size constraints.
2) **Preference-Based Redistribution:** The redistribution phase is sensitive to student preferences. Unassigned students or those in groups that fall short of the minimum size are redistributed according to their original topic preferences. The redistribution process strictly observes the maximum group size and topic use limit constraints, ensuring that the reassignment does not violate the fundamental project requirements. This phase ensures that student preferences are respected to the fullest extent possible, given the constraints, maintaining a balance between preferences and strict constraints.
3) **Merging of Undersized Groups:** Despite the preference-based redistribution, some groups might still fail to meet the minimum size requirement. To rectify this, undersized groups within the same topic are either merged or replenished with students from larger groups. This step is performed iteratively until all groups satisfy the minimum size constraint, thus aligning the solution with one of the critical problem constraints.
4) **Random Assignment and Final Adjustments:** In instances where students could not be assigned to preferred topics due to full capacity, a controlled random assignment mechanism ensures that every student was allocated. This adjustment safeguards the other constraints, maintaining a delicate balance between individual preferences, deterministic allocation, and the need for random assignment when preferences cannot be satisfied.

Algorithm 1 *Minimum Cost Maximum Flow (MCMF) with Redistribution*

```
Require:
  - Preference dataset PD in CSV format where each row represents a student's ordered topic preferences
  - Integer parameters:
    - g_min: represents the minimum allowable size of a group.
    - g_max: represents the maximum allowable size of a group.
    - t_limit: represents the usage limit for each topic.
1: /* Initialization */
2: Read student preferences from PD and store in S
3: Initialize an empty set of topics, T
4: for each student $s_i$ in S do
5:    Extract preferences and populate set of topics T
6: /* Constructing the network flow graph */
7: Create a directed graph G with nodes comprising of students, topics, and additional source and sink
   nodes
8: Add edges from source to all students $s_i$ with capacity 1 indicating that each student can be allocated to
   one topic only
9: for each student $s_i$ in S do
10:    for each topic $t_j$ in their preferences do
11:       Add edge from $s_i$ to $t_j$ with capacity 1, weight inversely proportional to preference ranking, i.e.,
          higher the rank, lower the weight
12: for each topic $t_j$ in T do
13:    Add edge from tj to sink with capacity g_max * t_limit, weight 0
14: /* Solve for minimum cost maximum flow */
15: Compute minimum cost maximum flow on G using the Successive Shortest Path algorithm to obtain
    flow dictionary F
16: /* Group formation based on flow results */
17: Initialize empty topic group dictionary GD
18: for each topic $t_j$ in T do
19:    for each student si in S do
20:       if flow from $s_i$ to $t_j$ in F is positive then
21:          Assign $s_i$ to group in GD for $t_j$
22:          if group size reaches g_max then
23:             Create a new group for $t_j$ in GD
24: /* Redistribute students to adhere to group size constraints */
25: Function redistribute_students(GD, S, g_min, g_max)
26:    - Combine undersized groups in GD and redistribute students to create groups of size between g_min
         and g_max, based on student preferences and availability. If a student can't be assigned based on
         preferences, assign them to a random group
27: /* Compute assignment statistics */
28: Function assignment_count(S, GD)
29:    - Calculate count of students assigned by preference ranking
30: /* Store results in a file */
31: Function result_file(GD)
32:    - Generate a CSV file 'allocation.csv' containing final group assignments
33: /* Generate a log file with summary */
34: Function log_file(S, GD, g_min, g_max, t_limit)
35:    - Create a text file 'log_file.txt' summarizing the assignment process and statistics
36: /* Execute steps */
37: Call functions redistribute_students, result_file, assignment_count, and log_file
```

3.2 Implementation

Our solution to the Group-SPA problem incorporates the Minimum Cost Maximum Flow (MCMF) algorithm with a redistribution strategy. To achieve this, we have designed a Python program that leverages the capabilities of the NetworkX[1] and pandas[2] libraries for modelling flow networks and processing data respectively. Moreover, we provide a user-friendly web interface developed using Flask[3].

3.2.1 Implementation of MCMF with redistribution

The algorithmic solution to the Group-SPA problem is demonstrated in Algorithm 1. Given preference data, group size constraints, and topic usage limits, it outputs the finalized allocation alongside a detailed log capturing the score function's behavior.

Data Parsing: Using the pandas library, student-topic preference data was parsed, transforming it into a Python dictionary structure. The dictionary maps students to their

1. https://networkx.github.io/
2. https://pandas.pydata.org/
3. https://flask.palletsprojects.com/en/1.0.x/

respective ranked topic preferences. Simultaneously, a set of distinct topics was extracted and sorted.

Cost Function Definition: Central to the MCMF approach is the cost function c_{ij}. For our use-case, c_{ij} depicts the cost of allocating student i to topic j. This is inversely proportional to the student's preference rank, formalized as $c_{ij} = 1/rank_{ij}$.

Algorithm Variants: The MCMF algorithm can operate under two variants: a weighted and an unweighted version. In the weighted version, edge weights between the students and topics are assigned based on the preference ranking (weight = i), implying that assigning a student to their n^{th} preference topic incurs a cost of n. This ensures that the algorithm prioritizes assigning students to their higher-ranked topics.

Conversely, the unweighted variant assigns a uniform weight of 1 to all student-topic edges, focusing on maximizing the total number of students assigned irrespective of their preference ranking. While this variant may result in a larger number of assignments, it could compromise satisfaction rates as it may assign more students to their lower-ranked preferences.

Network Graph Construction: The subsequent phase involves constructing a directed network flow graph $G(V, E)$, where V signifies the set of vertices (or nodes), and E denotes the set of edges. Vertices V were segmented into four distinct types: the source node s, the sink node t, individual student nodes S, and unique topic nodes T. The edges E were created following several criteria:

- An edge (s, i) for every $i \in S$, with a capacity $u_{si} = 1$, representing that each student could be assigned one topic.
- Edges (i, j) for every $i \in S$ and $j \in T$, with a capacity $u_{ij} = 1$. The weight w_{ij} for these edges is defined as $w_{ij} = c_{ij}$, the cost associated with each topic according to the cost function, encapsulating the cost of assigning student i to topic j.
- An edge (j, t) for every $j \in T$, with a capacity u_{jt} equivalent to the maximum number of students permissible per topic.

Flow Computation: The NetworkX's nx.max_flow_min_cost function was applied to $G(V, E)$, which calculates an optimal flow, ensuring maximum flow with minimum associated cost.

Group Formation and Redistribution: Using the flow dictionary, preliminary student-topic groupings were constructed. Given certain initial constraints, not all groups complied with the minimum size condition, prompting redistribution. A specialized redistribute_students function was employed to iteratively reassign students from smaller groups, respecting student preferences during reassignment. Students are reassigned to less-preferred or random topics only when essential.

3.2.2 *Graphical User Interface*

In accordance with the project objective to actualize the Min-Cost Max-Flow (MCMF) algorithm in a user-friendly tool, an interactive and comprehensive graphical user interface (GUI) was constructed. The interface was created utilizing Flask, a lightweight yet versatile web framework well-regarded for its Python-based applicability.

Data Upload Functionality: Central to the UI's design is its ability to facilitate the import of preference data in a CSV file format. Each CSV file encapsulates a student's project preferences, delineated in a ranked sequential order. This functionality is crucial, given the high volume of preference data typically encountered in real-world academic or institutional environments. This feature ensures that the UI remains scalable and efficient, capable of processing large datasets effectively.

Constraints Input: After uploading the preference data, the UI permits users to define the constraints pivotal to the Group-SPA problem. Users can specify parameters for group sizes, setting both minimum and maximum limits, as well as stipulate the availability of topics - essentially, how many groups a specific topic can accommodate. This aspect of the GUI enhances its versatility and allows the MCMF algorithm to be tailored to the unique requirements of different scenarios, making it adaptable to diverse real-world contexts.

Allocation Generation: Upon entering the data and defining the constraints, the system applies the MCMF algorithm with redistribution to compute the optimal student-project allocation. The resulting allocation can then be directly exported from the GUI in a structured CSV format, simplifying further review and usage. The CSV file contains crucial information, such as the allocation of each student to a specific project, the composition of each group, and the corresponding preference rankings.

Log File Generation: Concurrently with the allocation file, a log file is generated. This log, which is exported in a text file format, provides a detailed account of the allocation process executed by the algorithm.

The developed Flask-based user interface, as illustrated in Fig3 , bridges the gap between the computational model and practical application. The interface's features allow users to seamlessly input data and constraints, and conveniently download the allocation and log files produced by the system. This successful implementation of the Group-SPA framework in a user-friendly tool aligns with the set project objectives, paving the way for real-world applications.

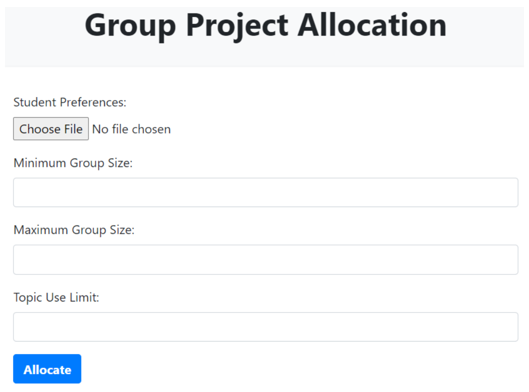

Fig. 3. Snapshot of the Web Interface

3.3 Verification and Validation

3.3.1 *Testing Strategy*

To validate the solution, we employed a robust testing strategy encompassing edge cases, large-scale scenarios, and random data sets. Edge cases tackled outliers like students lacking preferences or skewed group size constraints. Large-scale tests assessed the algorithm's scalability by varying the number of students and topics, while random data tests evaluated its adaptability to real-world variations in student choices and group conditions. Outputs were compared to manual calculations or anticipated outcomes, and a consistent sanity check ensured adherence to key constraints.

3.3.2 *Validity of Solution*

The testing affirmed the solution's robustness, adaptability, and reliability. Its ability to navigate edge case extremes, handle large-scale scenarios, and cater to randomized data showcases its versatility. Notably, even in random scenarios, students were predominantly aligned with their top preferences within the defined constraints. Moreover, the algorithm's adherence to project requirements, from topic assignments to ensuring group size compliance and preference satisfaction, underscores its efficacy and relevance for the Group-SPA problem.

3.4 Discussion of Potential Issues

Several potential issues could arise in the practical application of the solution. These include handling instances where students have no topic preference or when the constraints lead to potentially challenging scenarios such as overpopulated or unpopular topics.

Strict constraints on group sizes and topic availability could lead to situations where it becomes impossible to accommodate all students while adhering to the constraints. For instance, if the constraints result in a lower total available group capacity than the number of students, some students will inevitably remain unassigned. Extremely strict constraints might even lead to scenarios where groups of size 1 become unavoidable. We acknowledge these scenarios as limitations and suggest that constraints be set carefully to avoid such situations, considering the balance between preference satisfaction and practical limitations.

The current solution assumes identical group size and topic availability for all topics, which might not always hold true. Different topics could have distinct group size limitations and topic availability based on factors like resource availability or complexity of the topic. The current approach of dealing with lower group size constraints through a redistribution process after the application of the MCMF algorithm could be revised. In future iterations, these constraints could be directly incorporated into the net-work model.

Additionally, the current solution does not account for supervisors, who could serve as an additional constraint as each supervisor can typically oversee a limited number of projects. A potential enhancement to our model could include adding supervisors as nodes in the network, thereby enabling it to tackle scenarios where supervisor availability is a limiting factor.

4 Results

In this section, we describe the application of our MCMF with redistribution algorithm to both real-world and synthetically generated data. The use of both types of data allows for rigorous testing under conditions that mirror actual use cases as well as controlled settings.

4.1 Experimental Setup

Real-World Data. Our experiment utilizes data derived from a university's computer science department. This real-world data, comprising of actual student preferences for project topics and concrete constraints related to group sizes and topic availability, offers a realistic glimpse into an academic context. Through this data, our objective is to assess the performance and efficacy of our algorithm under practical operating conditions. This empirical evaluation is crucial, as it provides insights into the algorithm's applicability that may not be exposed through synthetic data alone.

Synthetic Data. In addition to real-world data, we have also generated synthetic datasets that mimic practical scenarios. These datasets permit controlled modifications in several parameters including the number of students, the diversity of project topics, group size constraints, and topic usage limits. These synthetic datasets, whilst simulating a broad spectrum of real-world conditions, also provide us the flexibility of altering parameters. Consequently, this aids us in exhaustively testing our algorithm under various conditions, thereby offering a thorough understanding of its performance and resilience.

An instance generator was utilized to construct random group-spa instances, which served as input for the algorithm implementations. This generator allows for the adjustment of specific properties of the instances produced. The configurable properties are:

1) The number of students n (default value $n = 100$).
2) The number of project topics m (default value $m = 10$).
3) The length of any student's preference list R (default value $R = 4$).
4) The popularity of the projects, defined by the ratio between the number of students applying for the most popular and least popular projects (default value of 5).
5) The minimum g_{min} and maximum g_{max} group size (default values $g_{min} = 9$ and $g_{max} = 10$).
6) The topic capacity limit t_{limit} (default value $t_{limit} = 1$).

In the subsequent subsection, we detail the experiments conducted using these datasets and discuss the results obtained. These experiments were designed to evaluate not only the performance of the algorithm under diverse conditions but also its effectiveness in satisfying student preferences and meeting constraints, the primary objectives of our study.

4.2 Experimental Results

4.2.1 *Real-World Data*

Experiment 1: Comparing Manual Solution to Algorithmic Solution In our preliminary experiment, we contrasted manual student allocation methods with two algorithmic approaches—weighted and unweighted. Our objective was to assess the distribution of students across their preferred choices under these differing allocation strategies.

The experiment was conducted with three parallel methodologies: a manual process, a weighted algorithmic process, and an unweighted algorithmic process. The weighted algorithm was designed to prioritize students' first preferences, while the unweighted algorithm equally considered all preferences, regardless of rank.

Our observations spanned the allocation of students to their first through fourth choices, as well as students who were randomly allocated when none of their preferred choices were available.

The results, shown in Fig4 indicated that the manual process assigned the highest number of students with their top preference. This was followed by the weighted algorithm and finally the unweighted algorithm. However, it's noteworthy that the unweighted algorithm achieved a more balanced distribution across the first four choices. In contrast, the weighted algorithm exhibited a steep decline in allocations after the first preference.

Notably, the number of students randomly allocated was significantly higher under the weighted algorithm compared to the other two methods. When examining the total preference-based allocations, we observed: 136 allocations via the manual method, 100 allocations via the weighted algorithm, and 115 allocations via the unweighted algorithm.

From a strategic perspective, the weighted algorithmic approach can be viewed as "greedy"—maximizing the number of first-choice allocations, even if that meant that some students were assigned their fourth preference. Conversely, the unweighted algorithm aimed for more holistic satisfaction by balancing allocations across all preferences. This approach avoided assigning a first-choice to a student if it meant forcing another student to accept their fourth preference. In this particular scenario, the unweighted or "generous" approach resulted in a more satisfactory allocation.

Significantly, all three methods successfully assigned all 141 students to a project group. Three students were excluded due to the absence of preference submission in this experiment.

However, strict constraints, such as group sizes of six or seven, rendered it unfeasible to allocate all students to their top-choice projects, thereby leading to a considerable number of random assignments. Despite the manual method yielding fewer random assignments than the algorithmic methods, the latter still provided satisfactory allocations.

Given the stringent constraints, it's a challenge to find a solution that enables maximum student allocation within their preference range. However, our aim was not to identify an optimal solution but rather to determine a good allocation that respects both the given constraints and student preferences.

4.2.2 Synthetic Data

Experiment 2: Varying Group Capacity constraints. In order to delve into the impacts of group size on both the quantity and quality of matches generated, we executed a controlled experiment. The number of students (n) was kept constant at 100, and other default values remained unchanged. The capacity of each group, in this case, was varied in steps of 2, ranging from 2 to 20.

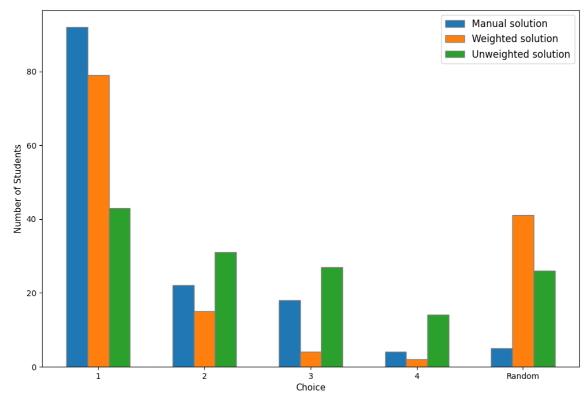

Fig. 4. Assignment Profiles Across Preference Choices Using Manual, Weighted, and Unweighted Methods

The experiment was repeated 1,000 times for each specified group size, leading to a total of 1,000 unique instances for each capacity value. Importantly, the minimum and maximum group sizes were held equal ($g_{min} = g_{max} = 2$) for this experiment, indicating that all groups had a fixed size.

The purpose of this experiment was to investigate the interplay between group capacity and the assignment algorithm's performance. By altering group size, we aimed to reveal how this parameter influences the matching quality in terms of both overall cost and student satisfaction, measured via the fulfillment of topic preferences.

During the initial phase of the experiment, as the group size progressed from 2 to 10, we observed a linear increase in both average cost and matching degree. Larger groups naturally accommodate more students, enabling higher chances for individuals to secure their preferred topics. Interestingly, a perfect matching degree of 1.0 was achieved at a group size of 10, implying that every student was allocated to a group that encompassed one of their preferred topics. However, the dynamics of the results began to shift as the group size exceeded 10.

For group sizes larger than 10, a paradoxical decrease in matching degree coupled with a sharp escalation in cost was noted. The dip in matching degree signifies that fewer students were matched to their preferred topics, while the surge in cost suggests an increased likelihood of students being assigned to less-preferred topics or even randomly.

This inversion in the trend beyond a group size of 10 hints towards a potential inefficiency when the group sizes become excessive. Several explanations may account for this, such as the algorithm's limitation in assigning students to their top-choice topics due to oversubscription, or a saturation point being reached for highly popular topics.

Collectively, the results depicted in Fig5 imply the existence of an optimal group size (in this case, around 10) that represents a balance between cost (reflective of the quality of the match) and the matching degree (indicative of the number of students matched to a preferred topic). Beyond this ideal group size, a notable decline in assignment quality tends to ensue.

Experiment 3: Varying Project Popularity. The relative popularity of the project topics can have substantial effects on the quality of matches produced. To understand these effects, we conducted an experiment, keeping the number of students (n) at 100, and other default values constant. We varied the project popularity from 0 to 9 in steps of 1, creating 1,000 random instances for each popularity level.

Fig6 demonstrates that with an increase in project popularity, we observe a corresponding increase in the average matching cost. This is an anticipated outcome, as higher popularity implies a larger number of students vying for the same top-preferred projects. This competition, in turn, results in more students being allocated to their lower-preference projects, leading to a higher matching cost.

Until the popularity level of 4, all students were assigned one of their preferred projects, resulting in a matching degree of 1.0. However, as popularity rose beyond 4, the matching degree started to decline. This decline implies that as more popular projects fill up quicker, some students are compelled to settle for their less-preferred projects or even projects not among their initial preferences.

Interestingly, beyond a popularity level of 7, we noticed a drop in the average matching cost. This decrease suggests that when a large proportion of students have similar preferences, they tend to get distributed evenly across highly popular topics. As more students get their high-preference projects, the total cost diminishes.

Nevertheless, despite the cost reduction, the average matching degree continued its downward trend, indicating that a higher number of students are failing to secure their preferred projects as popularity increases. This pattern infers that even though the average cost is decreasing, the overall satisfaction among students may not be improving since more students aren't getting their desired assignments.

In conclusion, there exists a trade-off between the matching cost and the matching degree. The balance between these two factors depends on the priorities of the organizers, for instance, whether they wish to minimize cost or maximize overall satisfaction. Consequently, different levels of project popularity may be more advantageous depending on the desired outcome.

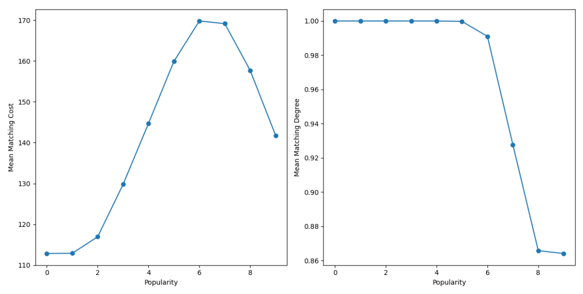

Fig. 6. Mean matching cost and degree vs popularity

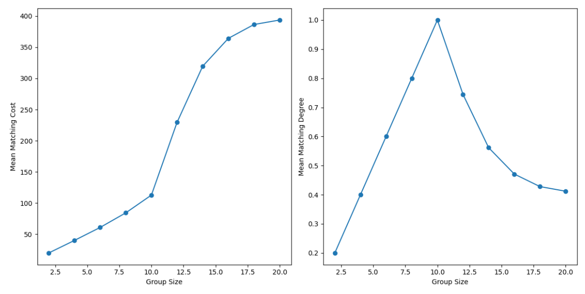

Fig. 5. Mean matching cost and degree vs group capacity

5 Evaluation

Upon reflection on our original research question, 'Can we develop an effective framework for the Group-SPA problem using a network flow model, such that it not only produces a good allocation based on individual preferences but also adheres to group size constraints and topic availability limitations?' it becomes clear that the developed Min-Cost Max-Flow (MCMF) algorithm with redistribution indeed provides a comprehensive and effective solution to this multifaceted problem.

One of the principal strengths of this framework is its ability to balance the optimization of individual student preferences and the adherence to constraints of group sizes and topic availability. This is particularly evident in real-world experiments, where the MCMF algorithm, particularly the unweighted variant, yielded a more systematic and equitable distribution of students across groups and topics compared to the manual method.

The adaptability of the MCMF algorithm represents another strength, as the synthetic data experiments showcased the robustness of this framework across varying group sizes and different levels of project popularity. Importantly, the algorithm was able to identify an optimal group size that effectively balances cost and matching degree, which is a critical insight for organizations with flexible group sizes.

Also noteworthy is the algorithm's ability to shed light on the interplay between project popularity, matching cost, and overall satisfaction. It highlighted that, although an initial increase in project popularity results in higher matching costs, these costs eventually decrease as popularity continues to rise.

However, despite these strengths, we also identified certain limitations. The algorithm tends to fall short when first preference satisfaction is the primary objective, as the manual method provided a higher top preference assignment rate. Furthermore, the algorithm's performance showed signs of inefficiency when the group sizes were excessively large or when project popularity escalated significantly, resulting in lower matching degrees and overall satisfaction.

While the synthetic data experiments contributed valuable insights into the algorithm's performance under varying conditions, it's important to note that real-world scenarios involving variability of preference intensity among students are not completely captured in these experiments. This factor limits the comprehensiveness of our testing.

To summarize, our research positively addresses the original question, demonstrating that an MCMF with redistribution framework can indeed provide an effective solution for the Group-SPA problem. This is underpinned by the framework's strengths in handling complex constraints, its adaptability to varying scenarios, and its insightful understanding of the interplay between cost, satisfaction, and popularity. However, there are limitations, notably in handling scenarios with very large groups, high project popularity, and a focus on first-preference satisfaction. Future research could focus on refining the algorithm or exploring new adjustment strategies to optimize student allocation and satisfaction further, addressing these limitations.

6 Conclusion

In our study, we delved into the complex issue of student group allocation for project topics. Given the limited number of groups and the formation constraints due to both upper and lower bounds on group capacity, the primary challenge lies in maximizing student satisfaction within these strict constraints, a task reflective of real-world settings.

To tackle this problem, we employed a network flow framework and utilized a strategy involving minimum cost maximum flow (MCMF) along with redistribution techniques. This demonstrated its effectiveness in deriving good-quality allocations that align with the set constraints and also maximize student satisfaction.

While the redistribution approach served our purpose, future work could involve directly modeling the lower bounds into the network, enabling a more comprehensive understanding and implementation of the allocation problem. A further comparison of different MCMF algorithms would also contribute significantly to the development of more efficient and refined allocation strategies.

Although our focus was primarily on the algorithmic aspects, the practical application of our work can be enhanced by improving the user interface. Facilitating direct registration for project topics by students and enabling administrators to perform allocations via a web interface could significantly streamline the process, making it more user-friendly and efficient.

In this study, we used uniform capacity constraints across all project topics, which may not accurately reflect the varying supervisory capacities in actual scenarios. Future research could incorporate separate capacity constraints for each project topic, considering individual supervisory capacities. This approach can provide a more realistic and effective allocation of groups to different project topics.

Further exploration should delve into a detailed investigation of different metrics of optimality. This should include a comprehensive study of profile-based matchings and a nuanced understanding of the roles of generosity and greediness in the student allocation context.

Our findings have implications extending beyond the educational setting. Allocation problems are inherent in many societal operations, ranging from resource distribution in economics to organ transplants in healthcare [20]. By offering a more efficient and fair solution to the group-student project allocation problem, we can potentially improve resource allocation in various contexts, leading to broader societal benefits.

References

[1] D. J. Abraham, R. W. Irving, & D. F. Manlove, "Two algorithms for the Student-Project Allocation problem," *Journal of Discrete Algorithms*, vol. 5, pp. 73-90, 2007.

[2] A.A. Anwar & A.S. Bahaj, "Student project allocation using integer programming," *IEEE Transactions on Education*, vol. 46, pp. 359-367, 2003.

[3] D. F. Manlove, "Algorithmics of Matching Under Preferences," *Series on Theoretical Computer Science*, 2012.

[4] M. Chiarandini, R. Fagerberg, & S. Gualandi, "Handling preferences in student-project allocation," *Annals of Operations Research*, vol. 275, pp. 39-78, 2017.

[5] L. Yahaya, A. Y. Tambuwal, & Z. Sani, "Group Student-Project Allocation Using Constraint Optimization Techniques," *International Journal of Science for Global Sustainability*, vol. 2, pp. 8–8, 2016.

[6] M. Zelvyte & U. Kraehmer, "The Student-Project Allocation Problem: a Network Flow Model," *BSc Honours project dissertation, University of Glasgow, School of Mathematics and Statistics*, 2014.

[7] J. Dye, "A constraint logic programming approach to the stable marriage problem and its application to student-project allocation," *BSc Honours project dissertation, University of York, Department of Computer Science*, 2001.

[8] D. Srinivasan & L. Rachmawati, "Efficient Fuzzy Evolutionary Algorithm-Based Approach for Solving the Student Project Allocation Problem," *IEEE Transactions on Education*, vol. 51, pp. 439-447, 2008.

[9] M. Dyer, D. Gusfield, & R. W. Irving, "The Stable Marriage Problem: Structure and Algorithms," *The Journal of the Operational Research Society*, vol. 42, p. 263, 1991.

[10] A. Kwanashie, R. W. Irving, D. F. Manlove, & C. T. S. Sng, "Profile-Based Optimal Matchings in the Student/Project Allocation Problem," *Lecture Notes in Computer Science*, pp. 213-225, 2015.

[11] D. J. Abraham, "Algorithmics of two-sided matching problems," *Master's thesis, University of Glasgow, Department of Computing Science*, 2003.

[12] A. H. Chown, C. J. Cook, & N. B. Wilding, "A simulated annealing approach to the student-project allocation problem," *American Journal of Physics*, vol. 86, pp. 701-708, 2018.

[13] A. Arulselvan, Á. Cseh, M. Groß, D. F. Manlove, & J. Matuschke, "Matchings with lower quotas: Algorithms and complexity," 2014.

[14] N. Garg, T. Kavitha, A. Kumar, K. Mehlhorn, J. Mestre, "Assigning Papers to Referees," *Algorithmica*, vol. 58, pp. 119-136, Jan. 2010.

[15] A. H. Abu El-Atta & M. I. Moussa, "Student Project Allocation with Preference Lists over (Student, Project) Pairs," *IEEE Xplore*, vol. 1, pp. 375–379, 2009.

[16] P. Biró, T. Fleiner, R. W. Irving, & D. F. Manlove, "The College Admissions problem with lower and common quotas," *Theoretical Computer Science*, vol. 411, pp. 3136-3153, 2010.

[17] K. Hamada, K. Iwama, & S. Miyazaki, "The Hospitals/Residents Problem with Lower Quotas," *Algorithmica*, vol. 74, pp. 440-465, 2014.

[18] S. Pudaruth, M. Bhugowandeen, & V. Beepur, "A Multi-Objective Approach for the Project Allocation Problem," *International Journal of Computer Applications*, vol. 69, pp. 26-30, 2013.

[19] R. K. Ahuja, T. L. Magnanti, & J. B. Orlin, "Network Flows," *Pearson*, 1993.

[20] P. Biró, "Applications of Matching Models under Preferences. In Ulle Endriss (editor)," *Trends in Computational Social Choice*, ch. 18, pp. 345–373, 2017.