Project-Team Allocator

MAHIKA JAGUSTE* and NIPUN MAHAJAN*, Indian Institute of Technology, Gandhinagar

There are many scenarios where it is required to select a group of candidates from a pool for optimal usage of resources. Unfortunately, the task is intensive and expensive due to the exponentially large number of possible clusters. In our project, we focus on developing and deploying a tool that can solve the issue of optimal project allocation and team formation in a formal educational setting.

Key Words and Phrases: Project allocation, Team formation, Integer Linear Progamming, Gurobi, Django, Web-implementation

1 MOTIVATION

A *hedonic game* (also known as a hedonic coalition formation game) is a game that models the formation of coalitions (groups) of players when players have preferences over which group they belong to. Hedonic games have been used to model many interesting settings, such as research team formation, scheduling group activities, formation of coalition governments, and so on. The input to a hedonic game instance involves every agent a specifying their preferences over the space of all possible coalitions (i.e, subsets of agents) involving a. Typically, these preferences are given by a reflexive, complete, and transitive binary relation.

We consider the application of hedonic games to team formation in the context of project assignments. A team in a company or a class in an university is often in the following situation. From the perspective of a manager/instructor, there are a few projects that need to be done, each project has an ideal team size (possibly given by a range), and possibly needs multiple teams working on it. On the other hand, from the perspective of the employees/students, agents have preferences over the projects and teammates. While hedonic games consider these preferences only from the perspective of the comparing different coalitions, a well-studied generalization called the Group Activity Selection Problem (GASP) does account for preferences of agents for projects; but it is typically assumed that preferences over activities depend on the number of participants in the activity.

We introduce a new model where we decouple these preferences; the agents have separate preferences over other agents and over the projects. We focus on finding allocations that optimize a natural notion of welfare, and propose an ILP formulation of this objective. This is implemented through a web application that relies on the Gurobi solver and is built using the Django framework. A natural direction of future work is to consider solution concepts from a game-theoretic perspective (for instance, stablity notions).

2 INTRODUCTION

The *Project-Team Allocation* problem focuses on allocating projects and forming corresponding student teams for the projects floated by the instructor in a given course. The problem is scale-able to real-world classroom settings. Consider that an instructor I teaching a course C has floated a set of projects *Pi*. The students enrolled in the course C will have to select the projects they are interested in working (the ranking of preference is not accounted for by the algorithm). Each student shall also provide a set of students they want to work with for any project among their preferences, and another set of students they don't prefer working with. Our study aims to create a model that can provide an optimal allocation of projects while simultaneously forming optimal student teams to work on the projects. We focus on developing a scale-able platform to deploy our designed algorithm which can be used in organisations for group selection and activity allotment.

3 PROBLEM STATEMENT

3.1 Input

Let P = $\{p_1, p_2, p_3, ...p_m\}$ denote a set of m projects and S = $\{s_1, s_2, s_3, ...s_n\}$ denote a set of n students. Each project p_i is represented by a tuple (num_team_i, max_size_i) where num_team indicates the maximum number of teams that can be allocated to project p_i and max_size indicates the maximum number of members in one team working on project p_i. Each student s_a is represented by a tuple (H_a, G_a). Here, H_a is an array (indexed from 1 to m) representing s_a's preferred projects (1) and neutral projects (0). G_a is an array (indexed from 1 to n) representing s_a's preferred teammates (1), non-preferred teammates (-1), and neutral teammates (0). Neutral indicates that s_a does not have any good/bad opinion about the project/student under consideration.

3.2 Validity of Input

The project set should be such that there are enough positions for all students to be able to work on atleast one project.

$$\forall i \in [1, m](num\_team_i * max\_size_i) \geq n$$

*Both authors contributed equally to this research.

Authors' address: Mahika Jaguste, mahika.oj@iitgn.ac.in; Nipun Mahajan, mahajan.n@iitgn.ac.in, Indian Institute of Technology, Gandhinagar.

4 NOTATIONS

P	Set of m projects $\{p_1, p_2, p_3, ...p_m\}$
S	Set of n students $\{s_1, s_2, s_3, ...s_n\}$
H	Project preference matrix (n x m) where each row represents a student and each column represents a project
G	Teammate preference matrix (n x n) where the rows and columns both represent the students
$y_{(i,t,a)}$	Binary Decision variable for whether student s_a has been allocated to the t^{th} team of project p_i
$L_{(i,t,a_1,a_2)}$	Binary Decision variable for whether students s_{a_1} and s_{a_2} have been allocated to the t^{th} team of project p_i
α	Weight allocated to project score
β	Weight allocated to group score

5 CONSTRAINTS

(1) Every student gets allocated to exactly one team of exactly one project.

$$\forall a \in [1, n] \sum_{i=1}^{m} \sum_{t=1}^{num\_team_i} y_{(i,t,a)} == 1$$

(2) Variable 'L' should represent the bitwise AND of the corresponding 'y' variables.

$$\forall i \in [1, m], t \in [1, num\_team_i], a_1 \in [1, n], a_2 \in [1, n]$$

$$L_{(i,t,a_1,a_2)} \leq y_{(i,t,a_1)}$$

$$L_{(i,t,a_1,a_2)} \leq y_{(i,t,a_2)}$$

$$L_{(i,t,a_1,a_2)} \geq y_{(i,t,a_1)} + y_{(i,t,a_2)} - 1$$

(3) Each team formed should follow the max_size constraint for that particular project.

$$\forall i \in [1, m], t \in [1, num\_team_i] \sum_{a=1}^{n} y_{(i,t,a)} \leq max\_size_i$$

6 OBJECTIVE

6.1 Definition of Project Score

We want to maximise the number of students who get allocated to one of their preferred projects.

$$project\_score = \sum_{i=1}^{m} \sum_{t=1}^{num\_team_i} \sum_{a=1}^{n} (y_{(i,t,a)} * H[a][i])$$

6.2 Definition of Team Score

We want to consider how many of a particular student s_a's preferred and non-preferred students become his teammates. We provide an impact of +1 for every preferred student and an impact of -1 for every non-preferred student. We consider such impact for all s_a's.

$$team\_score = \sum_{i=1}^{m} \sum_{t=1}^{num\_team_i} \sum_{a_1=1}^{n} \sum_{\substack{a_2=1 \\ a_1 \neq a_2}}^{n} (L_{(i,t,a_1,a_2)} * G[a_1][a_2])$$

6.3 Objective function

We want to construct a model that tries to optimise both the project score and the team score.

$$maximise \quad \alpha * project\_score + \beta * team\_score$$

7 TESTING

To check the robustness of our implementation we designed three testing structures for which the objective function can be calculated manually. We set up a testing environment to manually check if the constraints were satisfied and the expected objective function value was computed. These tests were run for n ranging from 15 to 200 students (here $\alpha = 0.5$ and $\beta = 0.5$ is assumed).

7.1 Structure I

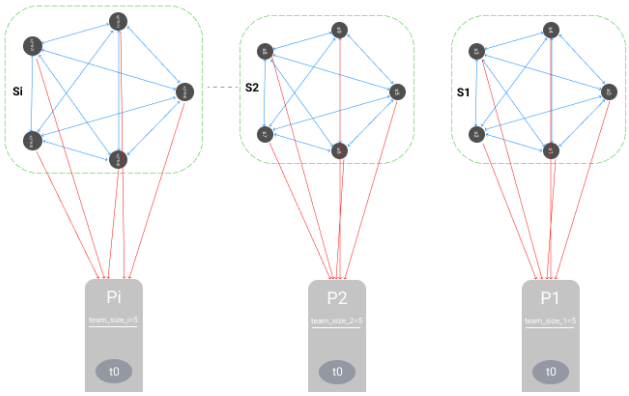

Fig. 1. Structure-1 for Testing Phase

In the first structure, the students are divided into groups of 5 and each such set of 5 students form a clique, that is, they are mutual friends. The absence of any other edge indicates that everyone has a neutral relationship with every person outside their friend group. For simplicity, all students within the friend group have the same project preference and each friend group has a different project preference. The maximum number of teams allowed for each project is 1 and the maximum team size is 5. In this case, each friend group should form a project team and should be allocated their preferred project. The objective function should equal $(n + 20 * m)/2$.

7.2 Structure II

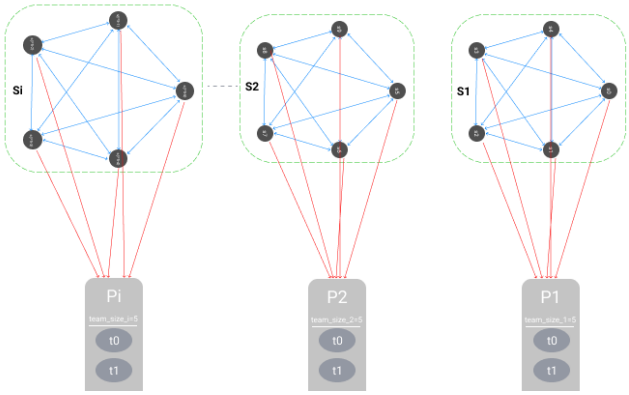

Fig. 2. Structure-2 for Testing Phase

For our next structure, we keep the same teammate preference settings as before. However, the maximum number of teams allowed for each project is 2 and the maximum team size is 3. Again, all students within the friend group have the same project preference and each friend group has a different project preference. In this case, each friend group should be split into 2 teams of sizes 2 and 3 respectively and should get allocated their preferred project. The objective function should equal $(n + 8 * m)/2$.

7.3 Structure III

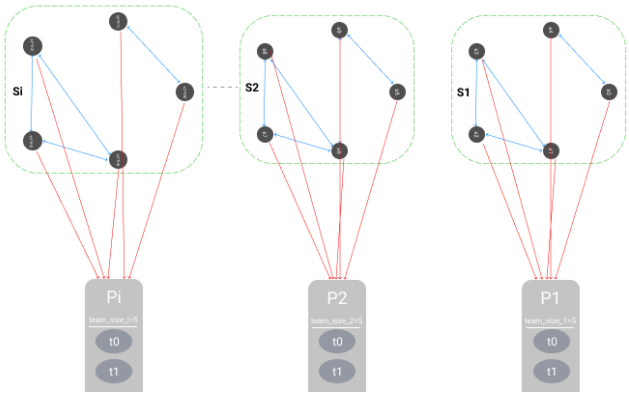

Fig. 3. Structure-3 for Testing Phase

For the third structure, we keep the project settings and the project preferences as in structure II. However, the earlier friend group of 5 is split such that 2 of them are friends with only each other and the other 3 are friends only among themselves. The goal was to add even more constraint to structure II as to how the 2-3 split among friends should take place. Since the setting is the same, the objective function here should also equal $(n + 8 * m)/2$.

8 TOOLS USED

8.1 ILP Implementation

To solve the described problem, we first experimented with PULP, a Python library for linear optimisation. By default, PULP uses the COIN-OR Branch-and-Cut MIP Solver (CBC) for integer and linear programming. However, for structured test cases with n 100 and m 20, the solver took more than 1 hour to compute and still didn't provide an output. Since there was less flexibility to alter any parameters or relax some constraints in Pulp, we decided to shift to the GEKKO package in Python which is designed for large-scale optimisation and can solve both linear and non-linear problems. This model was working correctly on the structured test cases and gave the output within reasonable times as shown in the graph, however, when n 150 and m 30, the model could not perform computation. The number of decision variables and constraints became too large for the model to handle.

The requirement of a model that could handle a huge number of variables and constraints, while simultaneously providing the solution in a reasonable time brought us to our final approach. We implemented our problem using the Gurobi Optimizer, a mathematical solver written in Python. Gurobi Optimizer is a commercial optimization solver which is used to solve linear optimization problems including Integer Linear Programming (ILP). Guorbi is a high-performance, flexible mathematical programming solver. Figure 4 is a graph that shows how long the different libraries take to compute the optimal objective value. As the number of decision variables increases, the Pulp and Gekko model are unable to function correctly.

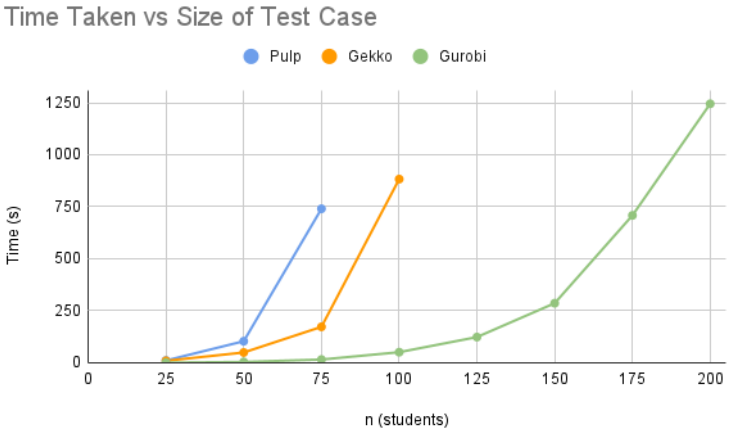

Fig. 4. Plot comparing the time-taken by 3 solvers for structure III instances

8.2 Website Implementation

Django, a Python web framework was used to implement the front-end, maintain a database, and connect to the back-end ILP algorithm. Figma software was used for wire-framing and designing the website. A thorough walk-through and the demo video of the web implementation of our project can be found here: *YouTube Link*
Our implementation of the tool and the code repository is linked here: *GitHub Link*

9 CONCLUSION

We developed a tool that facilitates the task of generating groups (forming teams) and allocating projects to students in an educational context. After working on the algorithm, we developed a web application based similar to a standard management portal for real-world use. The portal allows the interaction of the instructors and students within the system. The workflow of our project website tool can be divided into 3 stages.

In the first stage, the administrator provides the list of students in the educational organisation, information about the instructors, courses offered by all instructors in a given enrollment season and student enrollments for those courses. All this information is stored in a SQLite3 database in the backend. In the second stage, the instructor logs in on the portal using Google authentication. Among the list of courses, the instructor chooses any particular course, creates a project list and makes it available to the students enrolled in that course. In the third stage, the student logs in on the portal using Google authentication. For any course that they are enrolled in, once the project list is floated, they can choose the set of projects they are interested in, the set of students they prefer working with and students they don't prefer working with.After a certain deadline, the instructor can command the tool to invoke the solver and the allocation results are saved. The result for a course can be viewed by the professor. The student can view his own allocated project and team.

We have tested the tool for different scenarios and used randomly generated data for simulations. We also argue that such an optimal allocation is always possible. This tool can be used in different organisational settings. In future, we plan to add complexities in terms of skills honed by the students and those required for the project. We also plan to study which other problems can be captured by just slightly modifying our formulations.

10 REFERENCES

[1] Khandelwal, S.. "Building Teams of Experts using Integer Linear Programming." (2019).
[2] Noguera, Elena del Val, Juan M. Alberola, Víctor Sánchez-Anguix, Alberto Palomares and Maria Dolores Teruel. "A Team Formation Tool for Educational Environments." PAAMS (2014).
[3] Knez, Tina, Martina Holenko Dlab and Natasa Hoic-Bozic. "Implementation of Group Formation Algorithms in the ELARS Recommender System." iJET 12 (2017): 198-207.
[4] Cruz, Wilmax Marreiro and Seiji Isotani. "Group Formation Algorithms in Collaborative Learning Contexts: A Systematic Mapping of the Literature." CRIWG (2014).
[5] Darmann, Andreas, Edith Elkind, Sascha Kurz, Jérôme Lang, Joachim Schauer and Gerhard J. Woeginger. "Group Activity Selection Problem." WINE (2012).
[6] Eiben, E., Robert Ganian and Sebastian Ordyniak. "A Structural Approach to Activity Selection." IJCAI (2018).