Envy Free Hostel Room Allocation App

Tanvi Ajay Nerkar (180819)
Mentor: Siddharth Agrawal (150716)

Problem Statement

The problem statement is based on a fairly common issue faced in the real world. Several housemates move in together into a house with multiple rooms and need to decide who gets which room and at what price. The situation becomes interesting when the rooms differ in quality. The aim is to find a fair and envy free room and price allocation.

The final results contain the details of room with player allocation and the final price at which the room is rented out. The primary condition to be satisfied is that the total rent that the landlord receives remains the same. There should be no loss or no gain to the landlord.

The rent division problem involves a set of players [n] = {1, . . . , n}, and a set of rooms [n]. Each player i has a non-negative value $v_{ij} \in R+$ for each room j. We assume (without loss of generality) that the total rent is 1, and also assume (with loss of generality) that for all $i \in [n]$,

$$\sum_{j=1}^{n} v_{ij} = 1$$

An assignment of the rooms is a permutation $\sigma : [n] \to [n]$, where $\sigma(i)$ is the room assigned to player i.

Given a solution (σ, p) for a rent division problem V , the *utility* of player i is denoted as

$$u_i(\sigma, p) = v_{i\sigma(i)} - p_{\sigma(i)}.$$

A solution is *envy free (EF)* if the utility of each player for his/her room is at least as high as any other room. Formally, (σ, p) is EF if and only if

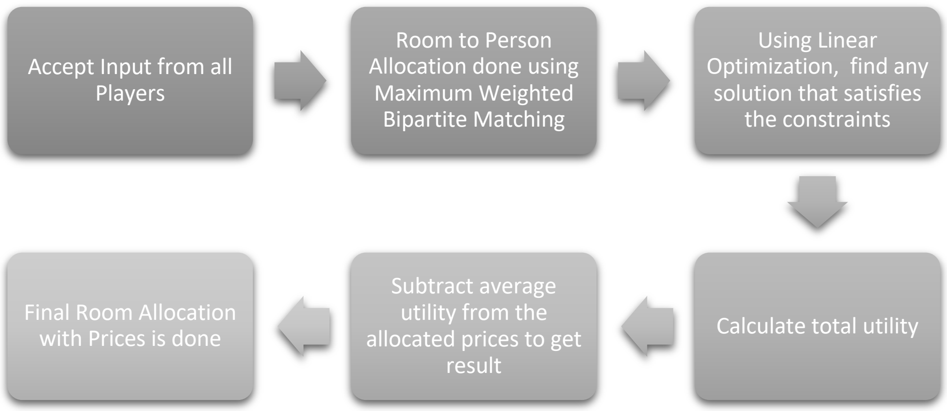
$$\forall\ i, j \in [n], \qquad v_{i\sigma(i)} - p_{\sigma(i)} \geq v_{ij} - p_j$$

The problem statement and its implementation are based on a research paper titled "Which Is the Fairest (Rent Division) of Them All?" authored by Ya'akov (Kobi) Gal, Israel Moshe Mash, Ariel D. Procaccia and Yair Zick.

Project Goals and Implementation

The algorithm implemented in this project is:

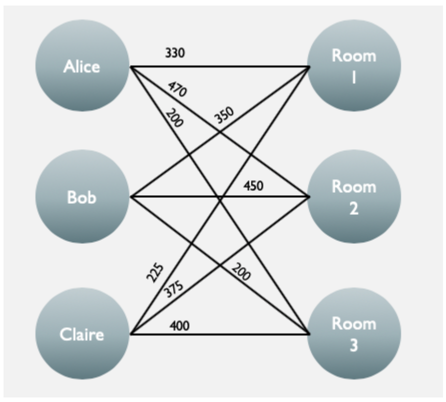
- Accept input from all players.
- Allot the rooms to the players by using a maximum weighted bipartite matching algorithm.
- Using Linear Optimisation, find a solution that satisfies the constraints.
- Calculate total utility of all players.
- Subtract average utility from the allotted prices to get results.


Accept Input from all Players
Room to Person Allocation done using Maximum Weighted Bipartite Matching
Using Linear Optimization, find any solution that satisfies the constraints
Calculate total utility
Subtract average utility from the allocated prices to get result
Final Room Allocation with Prices is done

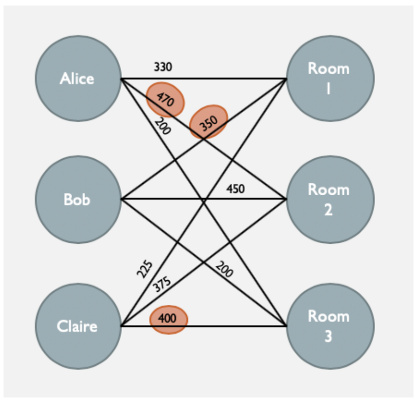
The input is accepted in the form of a n x n matrix, where n is the number of players and rooms. The above algorithm can be extended to k rooms, where k ≥ n. The algorithm implemented here is effective for an n x n input matrix.
Let us consider the following input. Assume total rent to be 1000.

	Room 1	**Room 2**	**Room 3**
Alice	330	470	200
Bob	350	450	200
Claire	225	375	400

A *bipartite graph* is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. The bipartite graph is formed here with the players as members of one set and rooms as members of the other disjoint set. The graph initially begins with n^2 branches. n branches originate from each player and n branches terminate at each room. The matching has to be done in such a way that n branches are chosen from the n^2 branches that the graph initially has.
It is assumed that each player volunteers to pay higher prices for rooms that are higher up on their preference lists. Therefore, by finding a *maximum weighted matching* in the bipartite graph, most rooms are allotted to the players who gave those rooms higher preference while finalising their inputs.
The algorithm used to find this matching is the *Hungarian Algorithm*, also known as the Munkres algorithm. This algorithm has time complexity $O(n^3)$. The Hungarian method finds a perfect matching and a potential such that the matching cost equals the potential value. This proves that both of them are optimal.
Using maximum weighted matching on the bipartite graph, fair results are obtained. The prices that each player was willing to pay for the room that they are assigned are noted. The total rent in this situation is found. This total sum is always greater than or equal to the total rent that is supposed to be paid to the landlord.



Bipartite Graph



Results of Maximum Weighted Matching

In order to find a set of values that will satisfy the constraints of the problem, linear optimisation is used to find an optimal solution to fit these conditions. The simplex technique in optimise() method of scipy library in Python has been used for finding the optimal solution.
After the results of the linear optimization are analysed, it is checked if they satisfy the conditions of envy-freeness. The condition for envy-freeness is that every player should have the same utility in the final allocation. In a situation where they do not satisfy this condition, an algorithm is implemented to ensure that the results are envy-free.
In the algorithm used to ensure envy-freeness, the total utility is found. The total utility is the difference between the sum of values allocated by the maximum weighted matching and the total rent that final values are supposed to add up to.

Envy-freeness is guaranteed if the final utilities of all players are equal. This is achieved by finding the average utility. After the average utility is found, it is subtracted from each of the player's allocated room rent.

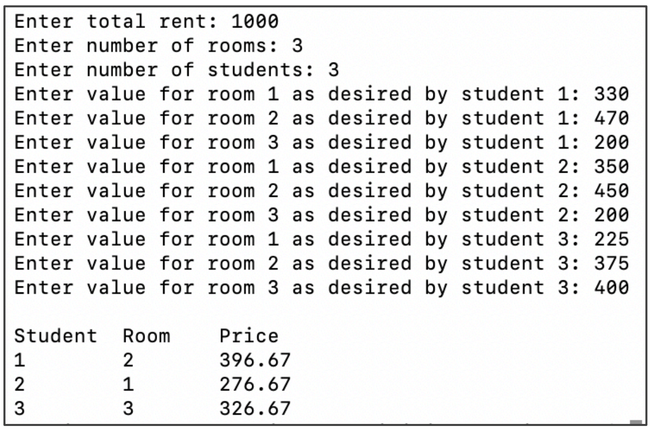
Results after average utility is subtracted

Person	Room	Allotted Price - Utility	Final Price
Alice	2	470.0 - 73.33	396.67
Bob	1	350.0 - 73.33	276.67
Claire	3	400.0 - 73.33	326.67

Results

The authors of the research paper have implemented their algorithm on the website www.spliddit.org. The results of the program used to implement the above algorithm are the same.

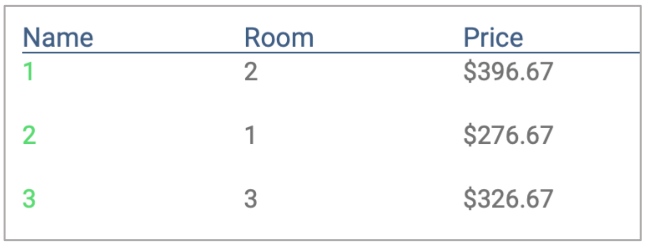
Project Results



```
Enter total rent: 1000
Enter number of rooms: 3
Enter number of students: 3
Enter value for room 1 as desired by student 1: 330
Enter value for room 2 as desired by student 1: 470
Enter value for room 3 as desired by student 1: 200
Enter value for room 1 as desired by student 2: 350
Enter value for room 2 as desired by student 2: 450
Enter value for room 3 as desired by student 2: 200
Enter value for room 1 as desired by student 3: 225
Enter value for room 2 as desired by student 3: 375
Enter value for room 3 as desired by student 3: 400

Student  Room    Price
1        2       396.67
2        1       276.67
3        3       326.67
```

Results from Spliddit for the same input (Prices assumed to be in $)



Name	Room	Price
1	2	$396.67
2	1	$276.67
3	3	$326.67

The sum of rents in both cases give the total rent (assumed to be 1000).
Validation of the final results is done by checking if the allocation and rents satisfy the following properties:

- If a player had been allotted any other room for the price at which that room is finally rented out, the utility would be less than or equal to their current utility. It shall never exceed their current utility.
- No player ever pays more rent for a room than what they are willing to pay as per the original input.

Utilities per room per player according to final room allocation results

	Room 1	Room 2	Room 3
Alice	53.33	*73.33*	-126.67
Bob	*73.33*	53.33	-126.67
Claire	-51.67	-21.67	*73.33*

It can be deduced from the results that each player is assigned the room where get the maximum utility. They are also not envious of the others since all utilities are equal.

Summary

The project results in an allocation of rooms to players at prices that satisfy all constraints. The results obtained are fair and envy-free. The conditions that were to be satisfied were:

- Total rent is maintained.
- Utilities of players are maximised.
- There does not exist another solution which could satisfy all of the above conditions.

Python was the language chosen to write the code in. This was due to the availability of the *pulp* and *numpy* libraries which were very useful in implementing the algorithm. File Handling was also used. The results of the maximum weighted bipartite graph matching are written to a *.lp* file, and the results are read from this file for the subsequent operations.

This project can be taken further by extending it to n players and k rooms, where k > n. The implementation finds wide use in hostels.

This implementation is meant for singly occupied rooms. It could be extended to multiple sharing rooms which would increase the complexity of the problem. There is wide scope for improvement and suggestion on the current algorithm as well as the different directions in which this project can be taken.

I would like to thank my mentor Siddharth Agrawal and Prof. Swaprava Nath for giving me their valuable suggestions and inputs on this project. It would not have been possible to complete this without their guidance.