Linear Programming using Python and its Application in Fair Room Allocation

Santosh Kavhar (22m0787@iitb.ac.in)
Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, India

***Abstract*—Linear programming(LP) is being widely used in many engineering domains because of its wide use in resource allocation techniques. In this work we focus on the implementation of linear programming using python and its application in fair room allocation. By fair room allocation we mean certain parameters like best preference and envy-freeness. The problem of fair room allocation usually occurs when a group of tenants have different valuations for different rooms, that is the exact problem we will be trying to solve. For this we will solve the maximum weight bipartite matching problem using LP in python using PuLP library. Towards the end we will also sow the seeds for envy-freeness implementation using LP, which is again the goal of fair room allocation. Here we will focus on maximizing the objective function using the constraints that are provided.**

***Index Terms*—linear programming, python, PuLP, maximum weight bipartite matching, envy-freeness, constraints, feasible region, objective function, optimization.**

I. Introduction

World war saw the boost of linear programming as it solved the problems of resource allocation with specific constraints in an efficient manner. Then on LP just picked up. LP's Simplex algorithm has been declared to be one of the top 10 most influential algorithms in the practice of science and engineering in twentieth century (Chong and Zak, 2013). Had LP not been invented and we would have still be stuck on brute-force solution and thus finding the best solution would have taken exponential time.

We will use python to implement the LP. We have chosen python as it is one of the most loved programming languages, and because of its immense support of external libraries, especially PuLP. PuLP is used to solve the LP problems because it is a very friendly library with all the basic functionalities that we need.

In most fair allocation tenants should be preferred to be given a room which they think is most valuable. That is exactly what we are trying to achieve first. The problem that we have solved is the maximum weight bipartite matching using LP. We have used PuLP library in python languages to solve the given problem. In the model section we will show how we have formulated the constraints for the problem. This problem is quite important in real world scenarios as well. As for our context consider different rooms where the renter has a chance to bid his price for that room and that sum of all the rooms price must meet the total rent. Then we can always find the most optimal matching using the LP. Had we used greedy then the matching wouldn't be most optimal. By optimal matching we mean the matching which gives maximum total rent.

Why are we using LP to solve this problem? Well, take the following example into consideration to get a sense of how much computing is required in a brute-force strategy. Consider a modest firm that employs 20 various machines to create 20 distinct parts. Assume that every machine is capable of producing every part. We also assume that each machine's production time for each part is known. The challenge is to allocate a part to each machine such that the total manufacturing time is kept as low as possible. There are 20! (20 factorial) potential assignments, as can be shown. Writing down every potential assignment and then comparing them all would be the brute-force method of addressing this assignment problem. Let's say we have a computer at our disposal that determines each assignment in 1 μs (10^{-6} seconds) Chong and Zak, 2013. Then, if this computer worked 24 hours a day, 365 days a year, it would take 77,147 years to identify the optimum (optimal) assignment. Even if you were to use the heuristics then too it would only give an approximate solutions and not the optimal ones and the difference between an approximate and an accurate solution in real world means shutting down the business.

In the next subsection we will see contributions where we will discuss the contributions of this report i.e maximum weight bipartite matching and envy-freeness, in the next subsection of related works we will discuss about three resources that were helpful for our study and understanding. Then we will discuss about the second section, background, which will help us understand the definition of all the keywords necessary to grasp our work. The third section will be model where we will dealing with the equations of LP then also the equations that we have used in our work to solve the maximum weight bipartite matching and also sow the seeds for envy-freeness constraints. In the fourth section, main results, we will show the results that we have got from using the implementation. Then in the final section, conclusion and future scope, we will conclude the paper with a taste of what future will hold for this work.

A. Contributions

This report has one main contributions in solving the problems of LP using python and one contribution is in process.

- **Maximum Weight Bipartite Matching**

This report shows us with the algorithm that is used to solve the problem of MWBM using LP. Though this is not the first material which does that but it is one of the official ones which does it for python using the PuLP library. This is the maximization problem and that means the optimal solution is the one which gives the maximal value. In (Gal et al., 2016) it is not mentioned how we are achieving the maximum welfare and it was our sole choice to use linear programming for it as it seemed an easy and affordable choice.

- **Envy-free constraint**

We are yet to formulate as how to obtain an envy-free constraints solution using LP, which will also be later on implemented in the python using PuLP. Finding envy-free solution using LP will make the solution more fair and satisfying. We propose to implement this envy-free constraint in later part of work and that will be next our contribution.

B. Related Works

We have referred many resources out of which the three are the most for our work are cited here.

(Gal et al., 2016) talk about the fairest rent division problem and the first step in their algorithm is the welfare-maximizing assignment. It is from this article that we think of solving the matching using LP instead of traditional approach where we convert this problem into maximal flow problem which then can be solved by ford-fulkerson algorithm.

(Chong and Zak, 2013) gives detailed chapters about LP from where we learned the concepts of LP, it also has a chapter on simplex algorithm as well as on concept of duality.

(Mitchell et al., 2011) gives the documentation for PuLP which helped us use the functionality of PuLP. It is a self sufficient documentation for help.

II. Background

In this section we will highlight the terms and their definition which will be useful to understand our work. We have three subsections, in the first we will review the relevant topics in LP then in second subsection we will review on python and PuLP and then finally the third subsection, where we will explain about MWBM and envy-free constraint.

A. Linear Programming(LP)

The goal of **linear programming (LP)** is to get the best possible allocation of decision variables which maximizes or minimizes the linear objective function. Decision variables are subject to linear constraints. The technique of linear programming is used in wide range of applications, including transportation, military, industry, agriculture, social science, economic, and healthcare system.

In linear programming, a **constraints** refers to a limitation or condition that restricts the possible solutions of a problem. These constraints are usually expressed as linear inequalities or equations that must be satisfied by the decision variables of the problem.

Optimization refers to getting the best or most favorable outcome or solution that can be obtained in a particular context or situation. An optimal solution is one that provides the best possible value for a given objective function, subject to a set of constraints.

The **feasible region** refers to the set of all possible values of the decision variables that satisfy all the constraints in a given problem. It is the set of all feasible solutions or points that can be considered as potential optimal solutions for the problem. If the feasible region is unbounded then we say that there is not any bounded solution to our problem.

The **objective function** is a mathematical expression that represents the quantity to be maximized or minimized in the optimization problem. It is a linear combination of the decision variables that describes the goal or the objective of the problem.

B. Python and PuLP

Python is a high-level, interpreted programming language that is widely used in a variety of applications and domains. Python is a versatile language that is used in web development, scientific computing, data analysis, machine learning, artificial intelligence, and many other areas.

PuLP is a popular open-source linear programming (LP) modeling package in python. It provides an interface to create LP models in python, which can be solved by various LP solvers like GLPK, CBC, CPLEX, Gurobi, etc. PuLP allows users to create LP models using a more user-friendly syntax compared to other LP modeling tools.

We declare variables for our LP with help of *LpVariable*. To declare whether a problem is maximizing or minimizing we use *LpProblem*. In this the first parameter is the name of the problem and the second parameter is *LpMaximize* or *LpMinimize* depending on our problem. *LpProblem* returns us a variable in which we can append our LP constraints and maximizing or minimizing function. We can have only a single maximizing or minimizing function and rest have to be constraints. PuLP separates out the constraints from the function by looking at the equality condition that is input in case of constraints and is absent for the maximizing or minimizing functions. In the end we use *solve()* method to solve the LP problem. The *LpStatus* Option will print the type of solution like *Optimal, Not Solved, Infeasible, Unbounded, Undefined* etc.

We have also used functions like *lpSum* to get the sum of the elements from a list. This function helped us program at ease.

C. MWBM and Envy-Freeness

Maximum Weight Bipartite Matching (MWBM) is a classical optimization problem in graph theory. It is a variant of the bipartite matching problem, where the edges between two disjoint sets of vertices are weighted, and the goal is to find a maximum-weight matching, i.e., a matching of maximum total weight.

Envy-freeness is a concept in social choice theory that refers to a situation where no individual in a group prefers the allocation of resources or goods received by another individual in the same group. In other words, an allocation is envy-free if everyone perceives their own allocation to be at least as good as someone else's allocation.

III. Model

Finding the most optimum solution for a linear function (also known as the objective function) under a set of restrictions (a set of linear inequality constraints) is made possible by linear programming problems (LPP). The five basic requirements of linear programming are: constraints, objective function, linearity, non-negativity, finiteness. The most popular methods to solve LPPs are the simplex method and interior point methods. For the detailed explanations see (Chong and Zak, 2013).

A standard linear program is an optimization problem of form

$$\text{minimize} \quad \boldsymbol{c^T x}$$
$$\text{subject to} \quad \boldsymbol{Ax = b}$$
$$\boldsymbol{x \geq 0}$$
$$\text{where } \mathbf{c} \in \mathbb{R}^{\mathrm{N}}, \mathbf{b} \in \mathbb{R}^{\mathrm{M}}, \text{and } \mathbf{A} \in \mathbb{R}^{\mathrm{M*N}},$$

$$\text{here } \boldsymbol{c^T} = [c_1, c_2, ..., c_n],$$
$$\boldsymbol{x} = [x_1, x_2, ..., x_n]^T \ ,$$
$$\boldsymbol{b} = [b_1, b_2, ..., b_m]^T \ , \text{ and}$$
$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{m1} & a_{m2} & ... & a_{mn} \end{bmatrix}$$

x $\geq$ 0 means every component of x is non-negative. Instead of minimizing, we can maximize, or the constraints may be in the form of inequalities, such as Ax $\geq$ b or Ax $\leq$ b. We also refer to these variations as linear programs.

A. Example of LP

Example 1. *This example is adapted from (Chong and Zak, 2013). A manufacturer produces four different products: X_1, X_2, X_3, and X_4. There are three inputs to this production process: labor in person-weeks, kilograms of raw material A, and boxes of raw material B. Each product has different input requirements. In determining each week's production schedule, the manufacturer cannot use more than the available amounts of labor and the two raw materials. Every production decision must satisfy the restrictions on the availability of inputs. These constraints can be written down as*

$$x_1 + 2x_2 + x_3 + 2x_4 \leq 20$$
$$6x_1 + 5x_2 + 3x_3 + 2x_4 \leq 100$$
$$3x_1 + 4x_2 + 9x_3 + 12x_4 \leq 75$$

As the negative production levels are not meaningful, we must impose the following non-negativity constraints on the production levels: $x_i \geq 0, i = 1, 2, 3, 4$

Now, suppose that one unit of product X_1 sells for ₹6, and X_2, X_3, and X_4 sell for ₹4, ₹7, and ₹5, respectively. Then, the total revenue for any production decision (x_1, x_2, x_3, x_4) is

$$f(x) = 6x_1 + 4x_2 + 7x_3 + 5x_4$$

We can observe that m=3 and n=4. The problem can be written in compact form

$$\textit{maximize} \quad \boldsymbol{c^T x}$$
$$\textit{subject to} \quad \boldsymbol{Ax \leq b}$$
$$\boldsymbol{x \geq 0}$$
$$\textit{where } \boldsymbol{c^T} = [6, 4, 7, 5],$$

$$\boldsymbol{x} = [x_1, x_2, x_3, x_4]^T \ ,$$
$$\boldsymbol{b} = [20, 100, 75]^T \ , \textit{ and}$$
$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 & 2 \\ 6 & 5 & 3 & 2 \\ 3 & 4 & 9 & 12 \end{bmatrix}$$

*The Optimal solution on solving this is on PuLP is **x**= [15, 0, 3, 0] and thus the Optimal value of profit is = 111₹.*

B. Envy-Freeness

- **Maximum Weight Bipartite Matching(MWBM)**

We can frame the MWBM in the form with some constraints.

$$\max \sum_{i \in N} \sum_{j \in M} v_{ij} \cdot x_{ij} \tag{1}$$
$$\text{s.t.} \sum_{j \in M} x_{ij} \leq 1, \ \forall i \in N \tag{2}$$
$$\sum_{i \in N} x_{ij} \leq 1, \ \forall j \in M \tag{3}$$
$$x_{ij} \geq 0, \ \forall i \in N, j \in M. \tag{4}$$

From eq. (1) we can see that the MWBM is a maximization problem. Here v_{ij} refers to the weights of the edges of the graph between the source as node i and the destination as node j. For our example v_{ij} means the value of room i from the perspective of student j. x_{ij} refers to the choice whether the source node i is connected with node j. If there were no constraints then all left hand side nodes(denoted by i) would have been connected with right hand side nodes(denoted by j). But we have three constraints. The first constraint, denoted by eq. (2), states that the number of nodes connecting to any node j (in right hand side) is less than or equals to one. Similarly the second constraint, denoted by eq. (3), states that the number of nodes connecting to any node i (in left hand side) is less than or equals to one. Because of these constraints we see that only the nodes which produce highest overall weight are considered for connection. The third eq. (4) gives us non-negativity condition. This constraint seems redundant as we are trying to maximize the eq. (1) but we will keep it for better clarity.

Please note that we have not considered that the sum of user's preference must be the total rent, that will be part of later work. For the result of MWBM refer main results section.

	Room 1	Room 2	Room 3
Student 1	300	300	400
Student 2	400	200	200
Student 3	250	250	500

Table I
STUDENTS PREFERENCE FOR ROOMS

- **Envy-freeness constraint**

We need to focus on two main things in case of envy-freeness.

maximize R such that:

$$R \leq f_q(v_{1\sigma(1)} - p_{\sigma(1)}, ..., v_{n\sigma(n)} - p_{\sigma(n)}) \quad (5)$$

$$v_{i\sigma(i)} - p_{\sigma(i)} \geq v_{ij} - p_j \qquad \forall i, j \in [n] \quad (6)$$

Where f_q is a function which collects the minimum of all the values. σ refers to the allocated mapping, in case of table I $\sigma(1)$ means the room mapped to first student. v_{ij} means the value of room i from the perspective of student j. The variables $p_1, p_2, ...p_n$ denote the prices that will be set once these conditions are solved by linear programming.

Thus here we are trying to maximize the minimal profit earned by each student by eq. (5).

By eq. (6) we are trying to make the allocation envy-free. The envy-free constraint for first student will be satisfied when $v_{12} - p_2 \geq v_{13} - p_3$ and $v_{12} - p_2 \geq v_{11} - p_1$, which means his share of profit for the room he has been allocated is as good as anyone else's. Similarly for other cases.

IV. Main Results

Student's preference for their choice of rooms is represented in table I . In this example, three students prefer three rooms at different rates. This rent is achieved by means of maximum weight bipartite matching. The bold ones in table II are the assigned mapping such that maximum rent is achieved. This type of mapping is also called the maximum welfare assignment. For e.g in table II first student is mapped to second room, second student to first room, and third student to third room.

	Room 1	Room 2	Room 3
Student 1	300	**300**	400
Student 2	**400**	200	200
Student 3	250	250	**500**

Table II
STUDENT ALLOTTED TO ROOM

V. Conclusion and Future Scope

Linear programming has wide variety of applications and in here we use it to help us solve the fair room allocation with help of PuLP library in python. We solve the fair allocation problem in two steps. In first we find the maximum weight bipartite matching and in the second (which is still in process) we solve envy-freeness. As per the observations from MWBM we can declare that we have found an efficient solution. We are yet to solve envy-freeness constraint using LP.

As part of future scope we will implement the envy-freeness property using linear programming, for this we will be using PuLP and python. Then we will extend on this concept to build a web application like spliddit which is used for dividing the rent fairly. The rent division algorithm defined in (Gal et al., 2016) will be at the kernal of our application.

References

Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 75. John Wiley & Sons, 2013.

Ya'akov (Kobi) Gal, Moshe Mash, Ariel D. Procaccia, and Yair Zick. Which is the fairest (rent division) of them all? In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, page 67–84, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450339360. doi: 10.1145/2940716.2940724. URL https://doi.org/10.1145/2940716.2940724.

Stuart Mitchell, Michael OSullivan, and Iain Dunning. Pulp: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, 65, 2011.