Math 177 Project Report

Brandon Palomino

Math 177: Linear and Non-Linear Programming
Sogol Jahanbekam

November 15, 2020

Problem Statement

Linear programming is a set of mathematical techniques used to solve systems of linear equations and inequalities while maximizing or minimizing some linear function for a desired result. Over the years, linear programming has become important in fields such as scientific computing, economics, and other management systems. Because of the increase in demand for calculations to be done in this manner, reliance on powerful tools such as Python programming is becoming a necessity when optimizing linear programming problems.

For the purpose of this project, the following problem was assigned:

> "An Auto company must meet the following demands for cars during the next 4 months: month 1 - 4,000; month 2 - 2,000; month 3 - 5,000; month 4 - 1,000. At the beginning of month 1, there are 300 autos in stock, and the company has the capacity to produce at most 3,000 cars per month. At the beginning of each month, the company can change production capacity by one car. It costs $100 to increase monthly production capacity. It costs $50 per month to maintain one car of production (even if it is unused during the current month). The variable cost of producing a car is $3,000. A holding cost of $150 per car is assessed against each month's ending inventory. It is required that at the end of month 4, plant capacity must be at least 4,000 cars. Formulate an LP to minimize the total cost incurred during the next four months."

In order to solve the problem, it is essential that the problem is understood carefully and modeled appropriately into a linear program. After modeling the objective function and constraints of the linear programming problem, it must be run on Python in order to obtain the optimal solution. For the purpose of this problem in particular, I will be using PuLP as the main linear programming API in Python due to it being less prone to errors when compared to SciPy. The following link is the steps to installing PuLP and setting up constraints and functions in python: https://realpython.com/linear-programming-python/

For the problem itself, my approach was to identify and define variables that contribute to the objective function of minimization. Because the problem revolved around an automobile industry's management of inventory, key variables that were created involved the cars produced, cars in inventory at the end of each month, production increases per month, and the extra cars produced under the production increase. With that in mind, I was able to construct constraints that matched the description of the problem. When creating the constraints, it is important to take each separate month as its own constraints since the results at the end of every month influences the following month. Upon finalizing the constraints, the objective function was implemented with the costs that were defined by the problem.

With everything into consideration, I believe the optimal solution that was formulated from this linear program should be the policy that is followed by the company to optimize their benefits.

Variable Definitions

The decision variables that will be implemented onto the problem's constraints and objective function are as followed and defined accordingly:

- C1 = number of cars produced during month 1
- C2 = number of cars produced during month 2
- C3 = number of cars produced during month 3
- C4 = number of cars produced during month 4
- LC1 = number of loaded cars in inventory at the end of month 1
- LC2 = number of loaded cars in inventory at the end of month 2
- LC3 = number of loaded cars in inventory at the end of month 3
- LC4 = number of loaded cars in inventory at the end of month 4
- PC1 = number of production capacity increases for month 1
- PC2 = number of production capacity increases for month 2
- PC3 = number of production capacity increases for month 3
- PC4 = number of production capacity increases for month 4
- EC1 = number of extra capacity cars produced during month 1
- EC2 = number of extra capacity cars produced during month 2
- EC3 = number of extra capacity cars produced during month 3
- EC4 = number of extra capacity cars produced during month 4

Formulation of Constraints and Objective Function

Objective Function:

min 150(LC1 + LC2 + LC3 + LC4) + 3000(C1 + C2 + C3 + C4 + EC1 + EC2 + EC3 + EC4) + 100(PC1 + PC2 + PC3 + PC4) + 50(EC1 + EC2 + EC3 + EC4)

Constraints:

subject to (s.t.) C1 <= 3000 "Constraints for cars produced: Month 1"
C2 <= 3000 "Constraints for cars produced: Month 2"
C3 <= 3000 "Constraints for cars produced: Month 3"
C4 <= 3000 "Constraints for cars produced: Month 4"
EC1 = PC1 "Constraints for extra cars and production capacity: Month 1"
EC2 = PC2 "Constraints for extra cars and production capacity: Month 2"
EC3 = PC3 "Constraints for extra cars and production capacity: Month 3"
EC4 = PC4 "Constraints for extra cars and production capacity: Month 4"
300 + C1 + EC1 = 4000 + LC1 "Demands and holdings constraints: Month 1"
LC1 + C2 + EC2 = 2000 + LC2 "Demands and holdings constraints: Month 2"
LC2 + C3 + EC3 = 5000 + LC3 "Demands and holdings constraints: Month 3"
LC3 + C4 + EC4 = 1000 + LC4 "Demands and holdings constraints: Month 4"
LC4 >= 4000 "Constraint for final checking at end of month 4"

Initial Simplex (Constraints Entered In Python Program)

Constraints Entered:

```
# constraints for cars produced
model += (C1 <= 3000, "month 1 car production constraint")
model += (C2 <= 3000, "month 2 car production constraint")
model += (C3 <= 3000, "month 3 car production constraint")
model += (C4 <= 3000, "month 4 car production constraint")
# constraints for extra cars produced from changes in production capacity
model += (EC1 == PC1, "month 1 car increase production constraint")
model += (EC2 == PC2, "month 2 car increase production constraint")
model += (EC3 == PC3, "month 3 car increase production constraint")
model += (EC4 == PC4, "month 4 car increase production constraint")
# constraints for demands and holding inventory at end of month
model += (300 + C1 + EC1 == 4000 + LC1, "month 1 demands and excess loading constraint")
model += (LC1 + C2 + EC2 == 2000 + LC2, "month 2 demands and excess loading constraint")
model += (LC2 + C3 + EC3 == 5000 + LC3, "month 3 demands and excess loading constraint")
model += (LC3 + C4 + EC4 == 1000 + LC4, "month 4 demands and excess loading constraint")
# requirement at the end of month 4
model += (LC4 >= 4000, "FINAL month 4 capacity constraint")
```

automobiles.py

Objective Function Entered:

```
# Add the objective function to the model
obj_func = 150 * (LC1 + LC2 + LC3 + LC4) + 3000 * (C1 + C2 + C3 + C4 + EC1 + EC2 + EC3 + EC4) + 100 * (PC1 + PC2 + PC3 + PC4) + 50 * (EC1 + EC2 + EC3 + EC4)
model += obj_func
```

automobiles.py

*Note: The constraints and objective function entered into the program represents the constraints and objective function that was formulated from the previous section. From the perspective of a linear programmer, these constraints can be reconstructed to a simplex matrix, which could then be solved using the simplex method by hand on paper. For simplicity and project purposes, the simplex of this problem will be represented by the constraints entered into the Python program. Each constraint is defined accordingly in quotations. After all constraints have been entered, the objective function is created and is defined as "obj_func." Afterwards, the constraints and objective function are loaded onto to model for PuLP to solve

Final Simplex (Outcome/Optimal Minimization)

Once the constraints and object function have been inserted into the Python program (*i.e. automobiles.py*), open up the computer's terminal and change directories to where the program is stored. As soon as the program is located, run the program by entering "python automobiles.py." The optimal minimization of the linear programming problem will then be outputted.

Outcome:

```
ECLSOTO:Project sotom$ python automobiles.py
status: 1, Optimal
objective: 48405000.0
C1: 3000.0
C2: 2000.0
C3: 3000.0
C4: 3000.0
EC1: 700.0
EC2: 0.0
EC3: 2000.0
EC4: 2000.0
LC1: 0.0
LC2: 0.0
LC3: 0.0
LC4: 4000.0
PC1: 700.0
PC2: 0.0
PC3: 2000.0
PC4: 2000.0
month_1_car_production_constraint: 0.0
month_2_car_production_constraint: -1000.0
month_3_car_production_constraint: 0.0
month_4_car_production_constraint: 0.0
month_1_car_increase_production_constraint: 0.0
month_2_car_increase_production_constraint: 0.0
month_3_car_increase_production_constraint: 0.0
month_4_car_increase_production_constraint: 0.0
month_1_demands_and_excess_loading_constraint: 0.0
month_2_demands_and_excess_loading_constraint: 0.0
month_3_demands_and_excess_loading_constraint: 0.0
month_4_demands_and_excess_loading_constraint: 0.0
FINAL_month_4_capacity_constraint: 0.0
```

Terminal output of automobiles.py

*Note: The results of our Python program gives an optimal solution of **$48,405,000.** The values and constraints for each variable are also listed as well. The solution and other representations will be discussed in the next section.

Analysis of Final Optimal Solution, Interpretations, & Representations

After inputting the constraints and object function into the Python program, I believe the automobile company should enforce a policy that is represented by the values calculated from the linear program. Should they follow the policy through, the automobile company would minimize their total costs to **$48,405,000**. To fully understand why the linear program obtained this result, this section of the report will analyze the values assigned to each variable and how it correlates to the constraints given by the problem.

First of all, I would like to explain the values that were assigned to the variables reflecting the number of cars produced per month. As the problem already states, the automobile company can only produce 3,000 cars per month. In regards to minimizing the total cost of the company, the

cost of producing a car is $3,000. When looking at the results of our optimal solution, **month 1 (C1) produces 3,000 cars**, **month 2 (C2) produces 2,000 cars**, **month 3 (C3) produces 3,000 cars**, and **month 4 (C4) produces 3,000 cars**. When looking at the values assigned for each month, I was at first skeptical at the results due to all variables but month 2 (C2) has reached the max constraint at 3,000 cars being produced (i.e. constraint C# <= 3,000). However, after looking back at the description of the problem, it makes sense that the linear program would favor producing cars through these constraints instead of producing them through the expensive route of increased production capacity. As a reminder, cars can be stored in inventory when going into the next month with a holding cost of $150 per car at the end of a month. It is important to consider the transfer of cars into each month because the company asks us to have at least 4,000 cars at the end of month 4. With that into consideration, I believe the constraints were well designed to prioritize producing cars normally as a way of minimizing total costs.

Before moving forward, I want to note that creating two different variables (PC# and EC#) was only for distinguishing purposes between increases in production capacity and the extra cars produced from the change. Even though I could have designed the constraint and objective function to only need one of these variable types to represent both, I choose to separate them for easy clarification since the problem defined them both separately in terms of how it affects the total cost. Because of this, the constraint "EC# = PC#" was created to signify that an extra car was produced due to an increase in production capacity.

Next, I would like to go over the values that were assigned to the variables reflecting the production capacity increases and the extra cars that were produced as a result of it. When looking over the variables regarding the cars under the increased production capacity, **month 1 (PC1 & EC1) produces 700**, **month 2 (PC2 & EC2) produces 0 cars**, **month 3 (PC3 & EC3) produces 2,000 cars**, and **month 4 (PC4 & EC4) produces 2,000 cars**. When comparing these results with the number of cars that were produced under normal constraints, these results logically make sense. With the demands of the company being asked for large amounts of cars at the end of every month, the usual process of producing 3,000 per month is not enough to meet these demands. Because of this, the linear program opts to create extra cars produced under the increase in production capacity in order to meet these demands on time. Due to producing these extra cars, the cost of making them are significantly more expensive in comparison to the usual process. On top of paying the $3,000 cost to produce each car, it also costs $100 to increase the monthly production capacity per car and $50 to maintain these extra built cars. These extra costs are reflected in the objective function accordingly based on the variables in place.

Finally, I would like to go over the constraints and values in regards to meeting the company demands and the holding costs that result from it at the end of the month. When designing the constraints that reflect the demands and holdings for the company at the end of every month, it was important for the linear program to consider every other variable such as the produced cars

(C#) and extra produced cars (EC#). Because of this, both C# and EC# must be optimized correctly in order to meet the demands every month. As a result, there is the possibility of excess cars as leftover at the end of every month. These cars would be classified as loaded cars (LC#) that are carried over into the next month with the expense of paying a holding cost of $150 per car at the end of the month (except the first constraint where the company has an initial inventory of 300 cars in stock already). When looking over the values of the variables regarding the loaded cars at the end of every month, **month 1 (LC1) loads 0 cars**, **month 2 (LC2) loads 0 cars**, **month 3 (LC3) loads 0 cars**, and **month (LC4) loads 4,000 cars**. Fortunately for the linear program, these are the kind of results that the company should aim for. By not having to load cars in inventory at the end of a month, we are greatly reducing the total cost for the company. This is due to the linear program producing the necessary cars needed to meet the demands at the end of a month. The only time cars are loaded into inventory is in the final month because there is another constraint that wants the company to have at least 4,000 cars in inventory at the end.

In conclusion, I believe the company should adapt to a policy that reflects the variables that were assigned by the linear program. When applying the variable values into our objective function, the company should obtain an optimal minimal total cost of **$48,405,000**. By prioritizing the production of normal cars over the cars produced from the increased production capacity, the company should be able to minimize their cost in terms of producing cars. As for the loading cost, the linear program was designed for the company to only build cars to meet the demands at the end of a month which allowed the company to not pay any loading cost in most months. Ultimately, the company should follow the basis of this report and the python file as their optimal solution to their problem.