SCM 518 Module 3 Homework

Paul Bernert

September 4, 2020

Question 35

Question: A company manufactures two types of trucks. Each truck must go through the painting shop and the assembly shop. If the painting shop were completely devoted to painting Type 1 Trucks, 650 per day could be planted, whereas if the painting shop were completely devoted to painting Type 2 Trucks, 550 per day could be painted. If the assembly shop were completely devoted to assembling Truck 1 engines, 1400 per day could be assembled, whereas if the assembly shop were completely devoted to assembling Truck 2 Engines, 1000 per day could be assembled. It is possible, however, to paint *both* types of trucks in the painting shop. Similarly, it is possible to assemble both types in the assembly shop. Each Type 1 Truck contributes to $2,500 to profit; each Type 2 Truck contributes $3,000. Maximize the company's profit.

Solution: This problem focuses on maximizing the number of Trucks of two types to produce to maximize profit. In order to solve this, we are given a set of initial values:

$$P_{(x,y)} = (2500, 3000)$$

$$A_{(x,y)} = (1400, 1000)$$

$$M_{(x,y)} = (650, 550)$$

In order to solve this, we must calculate the time to produce one unit of each truck:

$$PTime = \frac{1}{rp} + \frac{1}{ra}$$

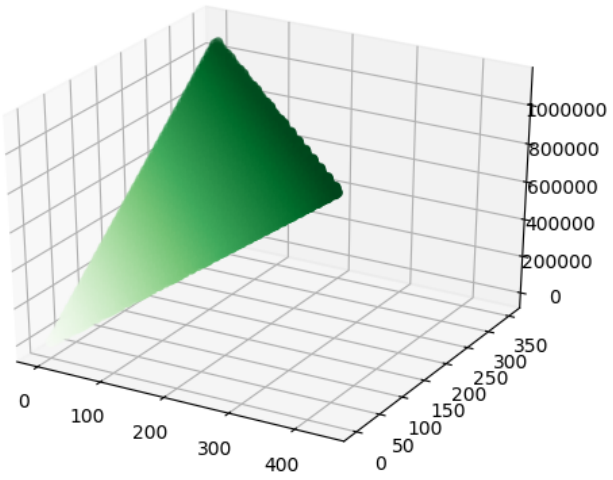
Where rp is the rate of Painting and ra is the rate of assembly. The total profit is calculated using the following function:

$$\mathcal{P} = P_1 * N_1 + P_2 * N_2$$

Where P is the profit per truck and N is the number of trucks produced. With this information, we have everything needed to write a parameter-scan script (doing this question with Python):

```python
import numpy as np
from math import floor

p1 = 2500
p2 = 3000

a1 = 1400
a2 = 1000

m1 = 650
m2 = 550

def production_time(rp, ra):
    return 1 / rp + 1 / ra

def profit(n1, n2):
    return p1*n1 + p2*n2

def max_number(T):
    return 1 / T

n_skipped = 0
skipped = 0

data = []

for N1 in range(floor(max_number(production_time(a1, m1)) + 1)):
    for N2 in range(floor(max_number(production_time(a2, m2)) + 1)):
        prod_1 = production_time(a1, m1) * N1
        prod_2 = production_time(a2, m2) * N2
        if (prod_1 + prod_2) > 1:
            skipped += 1
            continue
        data.append([N1, N2])

data = np.array(data)

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

ax = plt.axes(projection='3d')

zdata = profit(data[:,0], data[:,1])[::10]
xdata = data[:,0][::10]
ydata = data[:,1][::10]
ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
plt.show()
```

The optimal number of trucks to produce according to this model would be to maximize the number of Truck 1 up until near capacity, and finish off by producing simply a few of Truck two to make sure capacity is fully met.



Question 36

Question: A company manufactures mechanical heart valves from the heart valves of pigs. Different heart operations require valves of different sizes. The company purchases pig valves from three different suppliers. The cost and size mix of the valves purchased from each supplier are given by the file provided. The file also contains the maximum number of valves available from each supplier per month. Each month, the company places an order with each supplier. At least 1,300 large, 900 medium and 400 small valves must be purchased each month.

Solve to determine how the company can minimize the cost of acquiring the needed valves.

Solution:

The number of valves to purchase can be listed as follows:

$$N_x = N_{xs} + N_{xm} + N_{xl}$$

While the supply constraints for each supplier are the following:

$$N_1 = 2400$$

$$N_2 = 2000$$

$$N_3 = 1000$$

The following purchasing constraints must also be considered:

$$N_s = 400$$

$$N_m = 900$$

$$N_l = 1300$$

Thus, the total cost can be calculated using the following structure:

$$T_x = C_1(N_1) + C_2(N_2) + C_3(N_3)$$

With this information, we are able to calculate the minimum cost. We can do that with the following parameter-scanning function (written in C++):

```cpp
#include <iostream>

int c1 = 75;
int c2 = 65;
int c3 = 50;

int cost(int ns1, int ns2, int ns3, int nm1, int nm2, int nm3, int nl1, int nl2, int nl3) {
    return c1*(ns1 + nm1 + nl1) + c2*(ns2 + nm2 + nl2) + c3*(ns3 + nm3 + nl3);
}

int main() {
    int current = 6000000;

    int ns1, ns2, ns3;
    int nm1, nm2, nm3;
    int nl1, nl2, nl3;

    ns3 = 200;
    nm3 = 200;
    nl3 = 600;

    ns2 = 0;
    nm2 = 0;
    nl2 = 0;

    ns1 = 0;
    nm1 = 0;
    nl1 = 0;

    int iter_cost = 0;

    for (int ns1 = 0; ns1 <= 960; ns1++){
        std::cout << ns1 << " " << current << std::endl;

        for (int ns2 = 0; ns2 <= 600; ns2++){
            if (! ((ns1 + ns2 + ns3) == 400)) continue;

            for (int nm1 = 0; nm1 <= 960; nm1++){
                for (int nm2 = 0; nm2 <= 960; nm2++){
                    if (! ((nm1 + nm2 + nm3) == 900)) continue;

                    for (int nl1 = 0; nl1 <= 960; nl1++){
                        for (int nl2 = 0; nl2 <= 960; nl2++){
                            if (! ((nl1 + nl2 + nl3) == 1300)) continue;
                            iter_cost = cost(ns1, ns2, ns3, nm1, nm2, nm3, nl1, nl2, nl3);
                            if (iter_cost < current) current = iter_cost;
                        }
                    }
                }
            }
        }
    }

}
```

This parameter-scanning function returns that the optimal number of purchases to minimize cost is as follows:

$$N_{(1s,1m,1l)} = (0, 0, 0)$$

$$N_{(2s,2m,2l)} = (200, 700, 700)$$

$$N_{(3s,3m,3l)} = (200, 200, 600)$$

For a minimum cost of: \$154,000.00.