Midterm 1

Marcus Rose
MATH 87 Math Modeling

March 7, 2021

1 Minimizing Shipping Costs

1.1 Network Flow

The first goal of this report is to determine at which cost shipping for a certain supply chain system is minimized for a given month while maximizing the overall profit for that month.

- SF, Santa Fe
- EP, El Paso
- TB, Tampa Bay
- C, Chicago
- LA, Los Angeles
- NY, New York
- H, Houston
- A, Atlanta

Note, these variable names are only applied to simplify the network flow diagram in section 1.1. Different variable names will be established later to simplify the inflows and outflows of each city. The following 17 variables will be defined in the linear program

- sc
- sl
- sh
- sa
- el
- eh
- ea
- tn

- th
- ta
- hc
- hl
- hn
- ha
- ac
- an
- ah

The following is a network flow diagram of shipping centers and destinations given the following variable names. This will assist in the creation of the linear program by highlight the costs of the 17 routes.

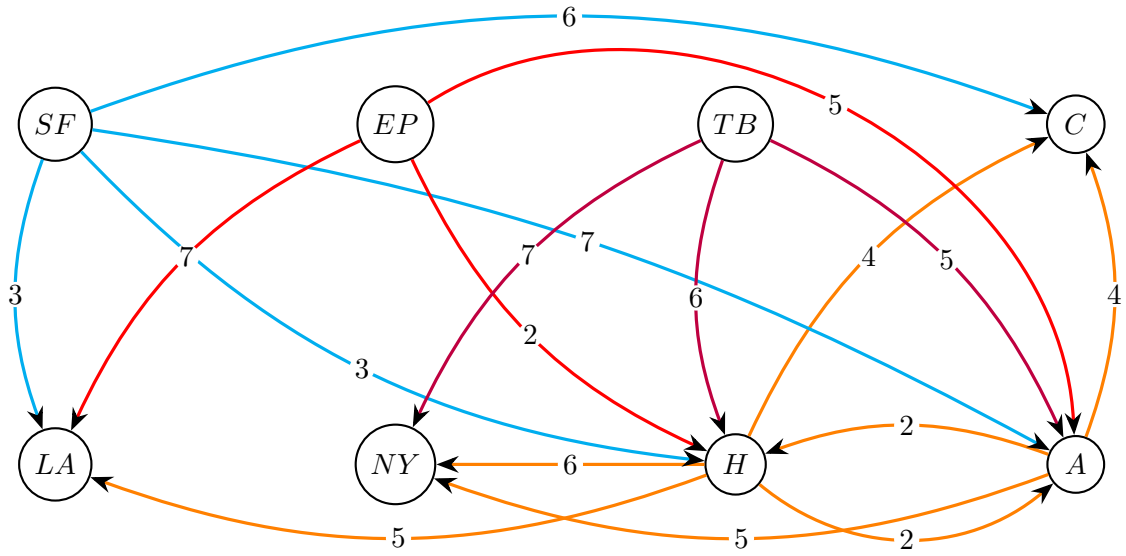

Figure 1: a representation of the exchange of dollars among cities

1.2 Formulating the Linear Program

The first objective of this report is to minimize shipping costs to each location given certain constraints. That is, our goal is to minimize the function

$$\min f(sc, sl, sh, sa, el, eh, ea, tn, th, ta, hc, hl, hn, ha, ac, an, ah) = 6sc + 3sl + 3sh + 7sa +$$
$$7el + 2eh + 5ea + 7tn + 6th + 4ta + 4hc + 5hl + 6hn + 2ha + 4ac + 5an + 2ah \quad \text{s.t.}$$
$$\begin{aligned}
6sc + 3sl + 3sh + 7sa &= 700,\\
7el + 2eh + 5ea &= 200,\\
7tn + 6th + 5ta &= 200,\\
6sc + 4hc + 4ac &= 200,\\
3sl + 7el + 5hl &= 200,\\
7tn + 6th + 5ta &= 250,\\
3sh + 2eh + 6th + 2ah - 2ha - 4hc - 6hn - 5hl &= 300,\\
7sa + 5ea + 5ta + 2ha - 2ah - 4ac - 5an &= 150,\\
sc, sl, sh, sa, el, eh, ea, tn, th, ta, hc, hl, hn, ha, ac, an, ah &\leq 200
\end{aligned}$$

1.3 Creating the Linear Program

In order to create the linear program, matrices will need to be established corresponding to the related minimizing objective function, equality constraints, and inequality constraints. Four matrices will be established as follows

$$A_{eq} = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0\\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 1\\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & -1 & -1
\end{bmatrix}$$

$$A_{ub} = \begin{bmatrix}
1 & 0 & \dots & \dots & \dots & 0 & 0\\
0 & 1 & \dots & \dots & \dots & 0 & 0\\
\vdots & \vdots & I_{1,1} & & & \vdots & \vdots\\
\vdots & \vdots & & \ddots & & \vdots & \vdots\\
\vdots & \vdots & & & I_{17,17} & \vdots & \vdots\\
0 & 0 & \dots & \dots & \dots & 1 & 0\\
0 & 0 & \dots & \dots & \dots & 0 & 1
\end{bmatrix} \in \mathbb{R}^{17\times 17}$$

$$\mathbf{x} = \begin{bmatrix} sc \\ sl \\ sh \\ sa \\ el \\ eh \\ ea \\ tn \\ th \\ ta \\ hc \\ hl \\ hn \\ ha \\ ac \\ an \\ ah \end{bmatrix} \in \mathbb{R}^{17\times 1}$$

$$b_{ub} = \begin{bmatrix} 200 \\ 200 \\ 200 \\ \vdots \\ 200 \end{bmatrix} \in \mathbb{R}^{17\times 1}$$

$$b_{eq} = \begin{bmatrix} 700 \\ 200 \\ 200 \\ 200 \\ 200 \\ 250 \\ 300 \\ 150 \end{bmatrix} \in \mathbb{R}^{8\times 1}$$

$$c^T = \begin{bmatrix} 6 & 3 & 3 & 7 & 7 & 2 & 5 & 7 & 6 & 4 & 4 & 5 & 6 & 2 & 4 & 5 & 2 \end{bmatrix} \in \mathbb{R}^{1\times 17}$$

A_{eq} represents the equality matrix that represents the "on" values that correspond to a city's inflows and outflows, and $\mathbf{x}$ represents the vector of variable paths (totaling 17). Applied together, these matrices must equal b_{eq}, which represents the actual values of the equality matrix, as each city needs to receive its respective demand of ducks, and each supplier needs to begin with the amount of ducks stated in the problem. Computing the linear program using `linprog` in the `scipy` library

```
import numpy as np
from scipy.optimize import linprog

Ae = np.array([[1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
               [0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0],
               [1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0],
```

```
                [0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0],
                [0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,0],
                [0,0,1,0,0,1,0,0,1,0,-1,-1,-1,-1,0,0,1],
                [0,0,0,1,0,0,1,0,0,1,0,0,0,1,-1,-1,-1]])

Ai = np.array([[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
                [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],
                [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0],
                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],
                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0],
                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]])

be = np.array([700, 200, 200, 200, 200, 250, 300, 150])
bi = np.array
    ([200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200])

c = np.array([6,3,3,7,7,2,5,7,6,4,4,5,6,2,4,5,2])

res = linprog(c, A_ub=Ai, b_ub=bi, A_eq=Ae, b_eq=be)

print(res)
```

Listing 1: The original LP

The array corresponding to the output of the original linear program above is as follows

```
x:  array([2.00000000e+02, 2.00000000e+02, 2.00000000e+02, 1.00000000e+02,
7.49721145e-09, 2.00000000e+02, 1.55524723e-08, 2.00000000e+02, 8.00491389e-09,
7.77042722e-08, 1.14017508e-08, 5.15120341e-09, 5.00000000e+01, 4.99999999e+01,
8.23746022e-09, 8.38412695e-08, 8.87638704e-09])
```

Thus, multiplying each of the shipping prices corresponding to each of the optimal shipping route costs in the form of a linear combination, the total cost of the optimal solution is

$$200 \times 6 + 200 \times 3 + 200 \times 3 + 100 \times 7 + 0 \times 7 + 200 \times 2 + 0 \times 5 + 200 \times 7 + 0 \times 6 + 0 \times 4 + 0 \times 4 + 0 \times 5 + 50 \times 6 + 50 \times 2 + 0 \times 4 + 0 \times 5 + 0 \times 2 = 5300 \text{ dollars}$$

1.4 Accounting for Strikes in Los Angeles

In the case where there is no strike in Los Angeles, the city will meet the demand of the laborers by doubling the shipping costs on each path to Los Angeles. This increases each of the values corresponding to that city by a factor of two. However, if the city decides not to meet the demand of the workers, they will strike and the supply of all the routes to Los Angeles will be reduced by a factor of 0.5. Hence, the new c^T (affected by the case of workers not striking) matrix, b_{ub} matrix (affected by the case of workers striking) code changes are represented in the linear program with the two new vectors $c2_l$ and $bi2_l$, respectively. The code and edited matrices are as follows

$$c^{T\prime} = \begin{bmatrix} 6 & 6 & 3 & 7 & 14 & 2 & 5 & 7 & 6 & 4 & 4 & 10 & 6 & 2 & 4 & 5 & 2 \end{bmatrix} \in \mathbb{R}^{1\times 17}$$

$$b_{ub} = \begin{bmatrix} 200 \\ 100 \\ 200 \\ 200 \\ 100 \\ 200 \\ 200 \\ 200 \\ 200 \\ 200 \\ 200 \\ 100 \\ 200 \\ 200 \\ 200 \\ 200 \\ 200 \end{bmatrix} \in \mathbb{R}^{17\times 1}$$

```
c2_l = np.array([6,6,3,7,14,2,5,7,6,4,4,10,6,2,4,5,2])
bi2_l = np.array([200,100,200,200,100,200,200,200,200,200,200,100,
                  200,200,200,200,200])

res_no_strike_l = linprog(c2_l, A_ub=Ai, b_ub=bi, A_eq=Ae, b_eq=be)
res_strike_l = linprog(c, A_ub=Ai, b_ub=bi2_l, A_eq=Ae, b_eq=be)

print(res_no_strike_l)
print()
print(res_strike_l)
```

Listing 2: Changes to costs due to striking in Los Angeles

The following arrays correspond to the program outputs for the optimal shipping routes given the workers do not strike and given the workers strike, respectively

```
x:  array([2.00000000e+02, 2.00000000e+02, 2.00000000e+02, 1.00000000e+02,
3.11793147e-09, 2.00000000e+02, 7.85906249e-09, 2.00000000e+02, 5.03438659e-09,
4.66228447e-08, 8.72464514e-09, 3.02957317e-09, 5.00000000e+01, 5.00000000e+01,
4.27623774e-09, 4.60880839e-08, 5.28146554e-09])
```

```
x:  array([2.00000000e+02, 9.99999998e+01, 2.00000000e+02, 2.00000000e+02,
5.25918709e+01, 1.47408129e+02, 7.35909321e-08, 1.99999999e+02, 1.62829759e-07,
1.42236251e-08, 4.98313402e-09, 4.74081289e+01, 1.83238828e-07, 1.12564684e-07,
8.77825447e-10, 4.99999999e+01, 2.05917981e-07])
```

Thus, multiplying each of the shipping prices corresponding to each of the optimal shipping route costs, the total cost for each scenario is

Given the workers in Los Angeles do not strike:

$$200 \times 6 + 200 \times 6 + 200 \times 3 + 100 \times 7 + 0 \times 14 + 200 \times 2 + 0 \times 5 + 200 \times 7 + 0 \times 6 + 0 \times 4 + 0 \times 4 + 0 \times 10 + 50 \times 6 + 50 \times 2 + 0 \times 4 + 0 \times 5 + 0 \times 2 = 5900 \text{ dollars}$$

Given the workers in Los Angeles strike:

$$200 \times 6 + 100 \times 3 + 200 \times 3 + 200 \times 7 + 52.6 \times 7 + 147.4 \times 2 + 0 \times 5 + 200 \times 7 + 0 \times 6 + 0 \times 4 + 0 \times 4 + 47.4 \times 5 + 0 \times 6 + 0 \times 2 + 0 \times 4 + 50 \times 5 + 0 \times 2 = 6050 \text{ dollars}$$

Therefore, the logical solution to minimize costs would be to submit to the workers' demands.

1.5 Accounting for Strikes in Houston

The same logic and procedure as in section 1.4 can be implemented for the city of Houston. Rearranging the inequality constraint and shipping costs slightly result in the following linear program

```
c2_h = np.array([6,3,6,7,7,4,5,7,12,4,8,10,12,4,4,5,4])
bi2_h = np.array([200,200,100,200,200,100,200,200,100,200,100,100,
                  100,100,200,200,100])

res_no_strike_h = linprog(c2_h, A_ub=Ai, b_ub=bi, A_eq=Ae, b_eq=be)
res_strike_h = linprog(c, A_ub=Ai, b_ub=bi2_h, A_eq=Ae, b_eq=be)

print(res_no_strike_h)
print()
print(res_strike_h)
```

Listing 3: Changes to costs due to striking in Houston

Thus, multiplying each of the shipping prices corresponding to each of the optimal shipping route costs, the total cost for each scenario is

Given the workers in Houston do not strike:

$$200 \times 6 + 200 \times 3 + 100 \times 3 + 100 \times 7 + 0 \times 7 + 100 \times 2 + 100 \times 5 + 200 \times 7 + 0 \times 6 + 0 \times 4 + 0 \times 4 + 0 \times 5 + 0 \times 6 + 0 \times 2 + 0 \times 4 + 50 \times 5 + 0 \times 2 = 6250 \text{ dollars}$$

Given the workers in Houston strike:

$$200 \times 6 + 200 \times 6 + 100 \times 3 + 200 \times 7 + 0 \times 14 + 100 \times 2 + 100 \times 5 + 200 \times 7 + 0 \times 6 + 0 \times 4 + 0 \times 4 + 0 \times 10 + 50 \times 6 + 0 \times 2 + 0 \times 4 + 50 \times 5 + 100 \times 2 = 6050 \text{ dollars}$$

Therefore, the logical solution in Houston differs from that of Los Angeles, as it would be more sensible to ignore the workers' demands to minimize costs.

2 Maximizing Profit

Given new profit information regarding each cities marginal loss and gain per duck, finding the maximum profit will involve creating 8 new paths corresponding to the profits of the eight cities (warehouse cities having negative profit in order to produce their ducks). Thus, there will be a total of 25 variables and a new linear program will be needed. These eight additional variables will be thought of as paths in the network flow; however, they will only represent each cities individual profit. Unlike section one, supply and demand will be treated as inequality constraints and as such, warehouse cities have the ability to under-produce and store cities may have an under-supply. New

inequality constraints and a new objective function are needed to solve this problem. Furthermore, because we are maximizing and not minimmizing this function, the `linprog` function in `scipy` will also need to be modified. The following is the linear program and associated code

$$A_{eq} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & -1 & -1 \end{bmatrix}$$

$$A_{ub} = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 & 0 \\ 0 & 1 & \dots & \dots & \dots & 0 & 0 \\ \vdots & \vdots & I_{1,1} & & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & & I_{25,25} & \vdots & \vdots \\ 0 & 0 & \dots & \dots & \dots & 1 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 & 1 \end{bmatrix} \in \mathbb{R}^{25\times 25}$$

$$b_{ub} = \begin{bmatrix} 200 \\ 200 \\ 200 \\ \vdots \\ 200 \end{bmatrix} \in \mathbb{R}^{25\times 1}$$

$$c^{T\prime\prime} = \begin{bmatrix} -8 & -5 & -10 & 15 & 20 & 5 & 10 & 10 & 6 & 3 & 3 & 7 & 7 & 2 & 5 & 7 & 6 & 4 & 4 & 5 & 6 & 2 & 4 & 5 & 2 \end{bmatrix} \in \mathbb{R}^{1\times 25}$$

Still using all of the packages and variables listed in the blocks of code from section 1,

```
Ae_maximize = np.array([[1,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0],
                        [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0],
                        [0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0],
                        [0,0,0,0,1,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0],
                        [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,0],
                        [0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,-1,-1,-1,-1,0,0,1],
                        [0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,0,1,-1,-1,-1]])

Ai_maximize = np.array([[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                        [0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
```

```
                          [0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0],
                          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
                         ]) # Identity, 25x25


# this is a maximizing objective function, so the signs of the original LP are
    reversed in    order to maximize profit
c_maximize = np.array
    ([-8,-5,-10,15,20,25,10,10,-6,-3,-3,-7,-7,-2,-5,-7,-6,-4,-4,-5,-6,
                -2,-4,-5,-2])

bi_maximize = np.array
    ([200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,
                200,200,200,200,200,200,200,200,200])

be_maximize = np.array([700,200,200,200,200,250,300,150])

# the maximizing linear program will look different as shown below
res_maximize = linprog(c_maximize*(-1), A_ub=Ai_maximize, b_ub=bi_maximize, A_eq=
    Ae_maximize, b_eq=be)
print(res_maximize)
```

Listing 4: Maximizing profit given costs

The array corresponding to the output of the profit-maximizing linear program above is as follows

```
x:  array([1.99999998e+02, 1.99999998e+02, 1.99999998e+02, 1.00000000e+002,
1.99999996e+02, 1.99999998e+02, 9.99999984e+01, 3.31206263e-07, 9.99999976e+01,
1.53507184e-06, 1.99999998e+02, 1.99999998e+02, 4.67192136e-08, 1.92989503e-07,
1.02038251e-07, 7.13617861e-08, 1.47962271e-07, 1.51630560e-07, 8.76436416e-08,
5.16146398e-08, 1.50724330e-07, 2.69255849e-07, 1.05541600e-07, 4.99999993e+01,
8.18647597e-07])
```

Hence the profit maximizing costs with these new constraints are

$$-(200 \times 8 + 200 \times 5 + 200 \times 10) + 100 \times 15 + 200 \times 20 + 200 \times 25 = 6900 \text{ dollars}$$
$$100 \times 6 + 200 \times 3 + 200 \times 5 + 50 = 2220 \text{ dollars}$$
$$\text{Total Profit} = 6900 - 2200 = 4700 \text{ dollars}$$

A profit of 4700 is obtained through reducing shipments from Santa Fe to Chicago and Los Angeles by 100 and 200, respectively. In addition, Tampa Bay's shipments will be reduced to New York by 200.

3 Summary

The purpose of this report was to minimize shipping costs while maximizing profit. In the first part, the total cost found by optimizing the total shipping costs was 5300 dollars. When striking occurred in different cities, different results were needed. In Los Angeles, the incurred costs of striking were greater than that of Houston, and it was more sensible to meet the workers demands. In Los Angles, the cost of striking was 6050 dollars versus a no strike cost of 5900. In Houston, the no strike cost was actually higher at 6250 dollars versus a strike cost of 6050. Hence, it would be more sensible in Houston to ignore the workers' demands and just cut supply. In terms of maximizing profit, there was a total of 25 variables to include the new profit information. Finding profit here involved subtracting total cost from the profit found (or the positive values of the objective function) in the liner program. This outputted a maximizing optimal solution of 4700 dollars, with routes from Santa Fe to Chicago, Santa Fe to Los Angeles, and Tampa Bay to New York all failing to supply their destination cities with enough ducks. Thus, shipping costs have been minimized, profit maximized, and two labor crises averted.