Tufts University - Department of Mathematics
Math 87 Midterm Solutions

Note that this is only a set of partial solutions and is not intended to represent a fully complete write up for the midterm. To receive full points on format, you should include details to explain each of the problems. Someone who is unfamiliar with the assignment should be able to read your report and have a clear understanding of the goal and results. Similarly, you should clearly outline each of the alternate scenarios.

In this problem, we must first minimize the shipping costs of a supply chain system between several cities, and then maximize the overall profit, taking into account various situations.

As the logistics manager for a supply-chain company that makes and sells rubber ducks, we have 3 main warehouses in Santa Fe, El Paso, and Tampa Bay. At each warehouse, there are a certain number of rubber ducks that must be shipped to your stores in various cities across the US. The number of supplies (in units of ducks) for each warehouse is listed here:

Santa Fe	El Paso	Tampa Bay
700	200	200

There are have 5 stores located across the US that will sell these ducks to your customers. The demands at each store are as follows (again in units of ducks):

Chicago	LA	NY	Houston	Atlanta
200	200	250	300	150

In order to ship the rubber ducks, to each of these cities, we use an air-shipping service that charges different prices between different cities depending on how much you ship. Some routes are not available. The following grid indicates the cost (in dollars) to ship 1 duck between a warehouse and a store. Note, these routes are one-way:

	Chicago	LA	NY	Houston	Atlanta
Santa Fe	\$6	\$3	-	\$3	\$7
El Paso	-	\$7	-	\$2	\$5
Tampa Bay	-	-	\$7	\$6	\$4

Additionally, Houston and Atlanta are hubs that, in addition to their own demands, can transship material to other customers. Those routes are indicated here:

	Chicago	LA	NY	Houston	Atlanta
Houston	\$4	\$5	\$6	-	\$2
Atlanta	\$4	-	\$5	\$2	-

Finally, we should note that shipping on each route is restricted to a maximum of 200 units.

The problem is to determine an optimum shipping plan that minimizes the total cost of shipping while meeting all customer demands with available supplies. We must formulate and solve a linear program to solve this problem, subject to the constraints explained above.

1. Draw a clearly labeled network-flow model for the linear program. You must explain all the constraints that you have included and why you have included them. You are strongly encouraged to include a node for the source of ducks (an initial node) and a node for the customers (a terminal node) even though these nodes are not really involved in the air-freight these nodes.

Answer:
For the network flow map, I will use each city as a node, plus a node for the source of rubber ducks and a node for the sink, which comes from selling off to customers. The shipping routes between each city represents each variable in the system. The weights on each arc indicate the cost to ship the duck, and the edges from the source to warehouses, and stores to customer also include the demand at the warehouse/store.

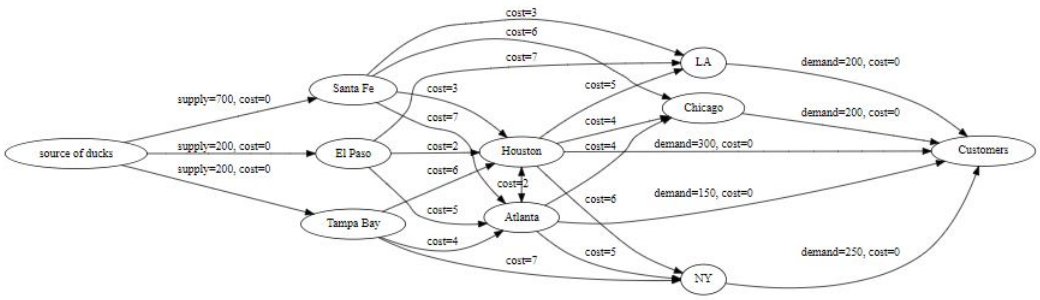

Figure 1: Network flow for shipping rubber ducks

Notice that we have 17 routes and on each shipping route, we have a upper bound constraint of 200 from the shipping constraints, and of course a lower bound of 0. Additionally, the arcs from the source to warehouses (3 total) have no costs, but equality constraints of manufacturing numbers. The arcs to the Demand node (5 total) have no costs and equality constraints of the customer demand for each city.

2. Use your network-flow model to formulate the linear program. Be sure to use descriptive variable names. Give the objective function, the equality constraints, and the inequality constraints.
Answer:
Define each arc to be a variable and order them in some fashion. Here, we use the following:

Source → Santa Fe	x_1
Source → El Paso	x_2
Source → Tampa Bay	x_3
Santa Fe → Chicago	x_4
Santa Fe → LA	x_5
Santa Fe → Houston	x_6
Santa Fe → Atlanta	x_7
El Paso → LA	x_8
El Paso → Houston	x_9
El Paso → Atlanta	x_{10}
Tampa Bay → NY	x_{11}
Tampa Bay → Houston	x_{12}
Tampa Bay → Atlanta	x_{13}

Houston → Chicago	x_{14}
Houston → LA	x_{15}
Houston → NY	x_{16}
Houston → Atlanta	x_{17}
Atlanta → Chicago	x_{18}
Atlanta → NY	x_{19}
Atlanta → Houston	x_{20}
Houston → Customers	x_{21}
Atlanta → Customers	x_{22}
Chicago → Customers	x_{23}
LA → Customers	x_{24}
NY → Customers	x_{25}

Now, we can easily write the linear program using the table and shipping costs:
Function to Minimize: shipping cost of the system, where the shipping cost on each route

was previously given, and the route variables are defined in the above table.

$$\begin{aligned} f(\boldsymbol{x}) = {} & 0x_1 + 0x_2 + 0x_3 + 6x_4 + 3x_5 + 3x_6 + 7x_7 + 7x_8 + 2x_9 + 5x_{10} + 7x_{11} + 6x_{12} \\ & + 4x_{13} + 4x_{14} + 5x_{15} + 6x_{16} + 2x_{17} + 4x_{18} + 5x_{19} + 2x_{20} + 0x_{21} + 0x_{22} + 0x_{23} \\ & + 0x_{24} + 0x_{25}. \end{aligned}$$

Note that routes from source to warehouse and stores to customer have no associated cost.

Inequality Constraints: All routes (except to/from the source/terminal nodes) have a lower bound of zero and upper bound of 200 ducks.

$$x_i \geq 0 \quad \text{for all } 4 \leq i \leq 20$$
$$x_i \leq 200 \quad \text{for all } 4 \leq i \leq 20$$

Equality Constraints:
Arcs from Source and Terminal nodes are known:

$$x_1 = 700, \quad x_2 = 200, \quad x_3 = 200,$$
$$x_{21} = 300, \quad x_{22} = 150, \quad x_{23} = 200, \quad x_{24} = 200, \quad x_{25} = 250.$$

Conservation at each node: the number of ducks received must be equal to the number of ducks leaving.
SFE: CHI + LA + HOU + ATL - SOURCE:

$$x_4 + x_5 + x_6 + x_7 - x_1 = 0.$$

CHI: TERMINAL - SFE - HOU - ATL:

$$x_{23} - x_4 - x_{14} - x_{18} = 0.$$

LA: TERMINAL - SFE - HOU - ELP:

$$x_{24} - x_5 - x_{15} - x_8 = 0$$

HOU: TERMINAL + CHI + LA + NY + ATL - SFE - ELP - TBY - ATL :

$$x_{21} + x_{14} + x_{15} + x_{16} + x_{17} - x_6 - x_9 - x_{12} - x_{20} = 0.$$

ATL: TERMINAL + CHI + NY + HOU - SFE - ELP - TB - HOU :

$$x_{22} + x_{18} + x_{19} + x_{20} - x_7 - x_{10} - x_{13} - x_{17} = 0.$$

NY: TERMINAL - HOU - ATL - TB:

$$x_{25} - x_{16} - x_{19} - x_{11} = 0.$$

ELP: LA + HOU + ATL - SOURCE:

$$x_8 + x_9 + x_{10} - x_2 = 0.$$

TB:NY+ HOU + ATL - SOURCE:

$$x_{11} + x_{12} + x_{13} - x_3 = 0.$$

3. Enter your model into `python` and use the `linprog` command (from `scipy.optimize`) to find an optimal solution. You are free to use colab or a python interpreter. Describe and include any code used in producing the matrices representing the equality and inequality constraints. Include the formulation of the call to `linprog` and the text of the output.

   Now that we have equations, we can plug this into linprog to solve. The inequality constraints will be represented by a 25×25 identity matrix for A_{ub}, with the RHS vector b_{ub} with 200 for $4 \leq i \leq 20$, and the values listed for equality constraints above as the other values. The equality constraints will be given by A_{eq}, an 8×25 matrix where each row represents and equality equation and RHS vector b_{eq} of all zeros. The vector c will be a vector of coefficients of the objective function $f(\boldsymbol{x})$.

Linear Program for Optimal Route

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

# Limit printing to 4 digits after decimal
float_formatter = "{:.2f}".format
np.set_printoptions(formatter={'float_kind':float_formatter})

# First let's create a dictionary to parametrize city numbers
cities=['Source','Santa_Fe','El_Paso','Tampa_Bay','Houston','Atlanta','Chicago','LA','NY','Customers']
ncities=len(cities)
city={cities[i]:int(i) for i in range(ncities)}

# We assign a variable to each of the 17 routes + 8 supply and demand edges
# (Houston to Atlanta and Atlanta to Houston count as different).
# To keep track of them we create a 2x25 array, where each column would correspond to a route
# first row (index=0) will contain the origin name, and second row (index=1) - destination
nvar=25 #number of routes, aka number of variables

routes=np.empty((2,nvar), dtype=int)
routes[0,0:3]=city['Source']
routes[1,0:3]=[city[name] for name in ['Santa_Fe', 'El_Paso', 'Tampa_Bay']] #destinations

routes[0,3:7]=city['Santa_Fe'] # origin
routes[1,3:7]=[city[name] for name in ['Chicago', 'LA', 'Houston', 'Atlanta']] #destinations

routes[0,7:10]=city['El_Paso']# origin
routes[1,7:10]=[city[name] for name in ['LA', 'Houston', 'Atlanta']] #destinations

routes[0,10:13]=city['Tampa_Bay']# origin
routes[1,10:13]=[city[name] for name in ['NY', 'Houston', 'Atlanta']] #destinations

routes[0,13:17]=city['Houston']# origin
routes[1,13:17]=[city[name] for name in ['Chicago', 'LA', 'NY', 'Atlanta']] #destinations

routes[0,17:20]=city['Atlanta']# origin
routes[1,17:20]=[city[name] for name in ['Chicago', 'NY', 'Houston']] #destinations

routes[0,20:25]=[city[name] for name in ['Houston', 'Atlanta', 'Chicago', 'LA', 'NY']]
routes[1,20:25]=city['Customers']

# Now we create report function that will parse the results
#  FYI, variables defined outside of the function, but not inside are called nonlocal
#  they can be accessed, but not modified
def report(results):
    print('Optimal_supplies_on_each_route:')
    for i in range(nvar):
        print('_',cities[routes[0,i]],'->',cities[routes[1,i]]+':',f"{results.x[i]:.4f}")
    print('Optimal_cost:', f"${results.fun:.4f}")

# Upper bound constraint corresponds to each route being bellow 200,
#so no need to track which is which here
# we also treat supply and demand as upper bound for the moment.
Aub=np.identity(nvar)
# Initiate the bounds by all 200, then correct what whould be different
bub=200*np.ones(nvar)
bub[(routes[0]==city['Source'])*(routes[1]==city['Santa_Fe'])]=700
bub[(routes[0]==city['Houston'])*(routes[1]==city['Customers'])]=300
bub[(routes[0]==city['Atlanta'])*(routes[1]==city['Customers'])]=150
bub[(routes[0]==city['NY'])*(routes[1]==city['Customers'])]=250

# Additionally to get equality for supply and demand we add them as the lower bound
Alb=np.copy(Aub[(routes[0]==city['Source'])+(routes[1]==city['Customers'])])
blb=np.copy(bub[(routes[0]==city['Source'])+(routes[1]==city['Customers'])])

# Cumulative inequality constraint is concatenation
A=np.concatenate([Aub,-Alb])
b=np.concatenate([bub,-blb])

# There is a zero balance equality constraint for each city
#(only source and customers nodes are unbalanced).
# initiate the matrix Aeq with all zeros, then assign nonzero elements
cities_with_equality=['Santa_Fe', 'El_Paso', 'Tampa_Bay', 'Houston', 'Atlanta', 'Chicago', 'LA', 'NY']
Aeq=np.zeros((len(cities_with_equality),nvar))
beq=np.zeros(len(cities_with_equality))

for i, name in enumerate(cities_with_equality):
```

```
    Aeq[i,routes[0]==city[name]]=-1
    Aeq[i,routes[1]==city[name]]=1

# Objective function would correspond to the cost for each route
c=np.zeros(nvar)

c[(routes[0]==city['Santa_Fe'])*(routes[1]==city['Chicago'])]=6
c[(routes[0]==city['Santa_Fe'])*(routes[1]==city['LA'])]=3
c[(routes[0]==city['Santa_Fe'])*(routes[1]==city['Houston'])]=3
c[(routes[0]==city['Santa_Fe'])*(routes[1]==city['Atlanta'])]=7
c[(routes[0]==city['El_Paso'])*(routes[1]==city['LA'])]=7
c[(routes[0]==city['El_Paso'])*(routes[1]==city['Houston'])]=2
c[(routes[0]==city['El_Paso'])*(routes[1]==city['Atlanta'])]=5
c[(routes[0]==city['Tampa_Bay'])*(routes[1]==city['NY'])]=7
c[(routes[0]==city['Tampa_Bay'])*(routes[1]==city['Houston'])]=6
c[(routes[0]==city['Tampa_Bay'])*(routes[1]==city['Atlanta'])]=4
c[(routes[0]==city['Houston'])*(routes[1]==city['Chicago'])]=4
c[(routes[0]==city['Houston'])*(routes[1]==city['LA'])]=5
c[(routes[0]==city['Houston'])*(routes[1]==city['NY'])]=6
c[(routes[0]==city['Houston'])*(routes[1]==city['Atlanta'])]=2
c[(routes[0]==city['Atlanta'])*(routes[1]==city['Chicago'])]=4
c[(routes[0]==city['Atlanta'])*(routes[1]==city['NY'])]=5
c[(routes[0]==city['Atlanta'])*(routes[1]==city['Houston'])]=2

result=linprog(c,A_eq=Aeq,b_eq=beq,A_ub=A,b_ub=b)
report(result)
```

```
Optimal supplies on each route:
  Source -> Santa Fe: 700.0000
  Source -> El Paso: 200.0000
  Source -> Tampa Bay: 200.0000
  Santa Fe -> Chicago: 200.0000
  Santa Fe -> LA: 200.0000
  Santa Fe -> Houston: 200.0000
  Santa Fe -> Atlanta: 100.0000
  El Paso -> LA: 0.0000
  El Paso -> Houston: 200.0000
  El Paso -> Atlanta: 0.0000
  Tampa Bay -> NY: 200.0000
  Tampa Bay -> Houston: 0.0000
  Tampa Bay -> Atlanta: 0.0000
  Houston -> Chicago: 0.0000
  Houston -> LA: 0.0000
  Houston -> NY: 50.0000
  Houston -> Atlanta: 50.0000
  Atlanta -> Chicago: 0.0000
  Atlanta -> NY: 0.0000
  Atlanta -> Houston: 0.0000
  Houston -> Customers: 300.0000
  Atlanta -> Customers: 150.0000
  Chicago -> Customers: 200.0000
  LA -> Customers: 200.0000
  NY -> Customers: 250.0000
Optimal cost: $5300.0000
```

Notice that not all routes are utilized. In fact, we don't even need to use Atlanta as a hub for the best results. The total cost of this route is $5,300.

4. Next, consider the following variant problems. Assume that shipping workers in LA are unhappy and contemplating a strike. They demand that all shipping costs to LA be doubled; if their demand is not met they will strike and the maximum number of supplies that can be shipped on all routes to LA is cut in half (i.e., from 200 to 100). Model both scenarios and see which one increases the cost more.

5. Test the same scenarios contemplated in 4. on the hub city of Houston. Is the result more or less drastic? Which city (LA or Houston) would cause the most problems for us if a work stoppage occurred? Remark: Technically you are only asked to change values corresponding to routes to Houston and not from. However specifically in this problem that difference does not affect the result.
Answer:

Meeting the demands of the workers in LA, the constraints from the original program stay the same, but the objective function changes:

$$f(\boldsymbol{x}) = 0x_1 + 0x_2 + 0x_3 + 6x_4 + 6x_5 + 3x_6 + 7x_7 + 14x_8 + 2x_9 + 5x_{10} + 7x_{11} + 6x_{12} + 4x_{13} + 4x_{14} + 10x_{15} + 6x_{16} + 2x_{17} + 4x_{18} + 5x_{19} + 2x_{20} + 0x_{21} + 0x_{22} + 0x_{23} + 0x_{24} + 0x_{25}.$$

The alternative is to halve the number of supplies on each route from LA. This corresponds to using the original program with the original objective function, with the change that $(b_{eq})_i = 100$ for $i = 5, 8, 15$.

LA Demands and Strike

```
print('LA_demands_scenario:')
# copy the cost function then multiply LA costs by 2
c_LA_demand=np.copy(c)
c_LA_demand[routes[1]==city['LA']]*=2
result_LA_demand = linprog(c_LA_demand,A_eq=Aeq,b_eq=beq,A_ub=A,b_ub=b)
report(result_LA_demand)

print('--------------------')

print('LA_strike_scenario:')
# copy the route contraints then divide LA constraints by 2
bub_LA_strike=np.copy(bub)
bub_LA_strike[routes[1]==city['LA']]/=2
# Lower bound constraints are not modified so we just concatinate the total
b_LA_strike=np.concatenate([bub_LA_strike, -blb])

result_LA_strike = linprog(c,A_eq=Aeq,b_eq=beq,A_ub=A,b_ub=b_LA_strike)
report(result_LA_strike)
```

```
LA demands scenario:
Optimal supplies on each route:
  Source -> Santa Fe: 700.0000
  Source -> El Paso: 200.0000
  Source -> Tampa Bay: 200.0000
  Santa Fe -> Chicago: 200.0000
  Santa Fe -> LA: 200.0000
  Santa Fe -> Houston: 200.0000
  Santa Fe -> Atlanta: 100.0000
  El Paso -> LA: 0.0000
  El Paso -> Houston: 200.0000
  El Paso -> Atlanta: 0.0000
  Tampa Bay -> NY: 200.0000
  Tampa Bay -> Houston: 0.0000
  Tampa Bay -> Atlanta: 0.0000
  Houston -> Chicago: 0.0000
  Houston -> LA: 0.0000
  Houston -> NY: 50.0000
  Houston -> Atlanta: 50.0000
  Atlanta -> Chicago: 0.0000
```

```
  Atlanta -> NY: 0.0000
  Atlanta -> Houston: 0.0000
  Houston -> Customers: 300.0000
  Atlanta -> Customers: 150.0000
  Chicago -> Customers: 200.0000
  LA -> Customers: 200.0000
  NY -> Customers: 250.0000
Optimal cost: $5900.0000
-----------------
LA strike scenario:
Optimal supplies on each route:
  Source -> Santa Fe: 700.0000
  Source -> El Paso: 200.0000
  Source -> Tampa Bay: 200.0000
  Santa Fe -> Chicago: 200.0000
  Santa Fe -> LA: 100.0000
  Santa Fe -> Houston: 200.0000
  Santa Fe -> Atlanta: 200.0000
  El Paso -> LA: 52.5919
  El Paso -> Houston: 147.4081
  El Paso -> Atlanta: 0.0000
  Tampa Bay -> NY: 200.0000
  Tampa Bay -> Houston: 0.0000
  Tampa Bay -> Atlanta: 0.0000
  Houston -> Chicago: 0.0000
  Houston -> LA: 47.4081
  Houston -> NY: 0.0000
  Houston -> Atlanta: 0.0000
  Atlanta -> Chicago: 0.0000
  Atlanta -> NY: 50.0000
  Atlanta -> Houston: 0.0000
  Houston -> Customers: 300.0000
  Atlanta -> Customers: 150.0000
  Chicago -> Customers: 200.0000
  LA -> Customers: 200.0000
  NY -> Customers: 250.0000
Optimal cost: $6050.0000
```

Nothing much changed, since we were already only shipping to LA from one route, so only the prices went up. The optimal cost is \$5,900, a \$600 increase from the original program.

With the strike, our option of just sending ducks via Santa Fe doesn't work any more, so we must take advantage of the Houston connection, despite the higher costs. The optimal cost is \$6,050, a \$750 increase from the original program. Thus, we should probably just the demands and pay the LA workers more!

Similar modifications can be made for the same scenario in Houston. Meeting the demands of the workers, only the objective function differs from the original linear program:

$$\begin{aligned} f(\boldsymbol{x}) = {} & 0x_1 + 0x_2 + 0x_3 + 6x_4 + 3x_5 + 6x_6 + 7x_7 + 7x_8 + 4x_9 + 5x_{10} + 7x_{11} + 12x_{12} \\ & + 4x_{13} + 4x_{14} + 5x_{15} + 6x_{16} + 2x_{17} + 4x_{18} + 5x_{19} + 2x_{20} + 0x_{21} + 0x_{22} + 0x_{23} \\ & + 0x_{24} + 0x_{25}. \end{aligned}$$

The alternative is to let the workers strike and halve the number of supplies on each route from LA. This corresponds to using the original program with the original objective function, with the change that $(b_{eq})_i = 100$ for $i = 6, 9, 12, 20$.

Houston Demands and Strike

```
print('Houston demands scenario:')
# copy the cost function then multiply LA costs by 2
c_Houston_demand=np.copy(c)
c_Houston_demand[routes[1]==city['Houston']]*=2
# note that we are not doubling the shipping costs from Houston, only to
result_Houston_demand = linprog(c_Houston_demand,A_eq=Aeq,b_eq=beq,A_ub=A,b_ub=b)
report(result_Houston_demand)

print('------------------')

print('Houston strike scenario:')
# copy the route contraints then divide LA constraints by 2
bub_Houston_strike=np.copy(bub)
bub_Houston_strike[routes[1]==city['Houston']]/=2
# Lower bound constraints are not modified so we just concatinate the total
b_Houston_strike=np.concatenate([bub_Houston_strike, -blb])

result_Houston_strike = linprog(c,A_eq=Aeq,b_eq=beq,A_ub=A,b_ub=b_Houston_strike)
report(result_Houston_strike)
```

```
Houston demands scenario:
Optimal supplies on each route:
  Source -> Santa Fe: 700.0000
  Source -> El Paso: 200.0000
  Source -> Tampa Bay: 200.0000
  Santa Fe -> Chicago: 200.0000
  Santa Fe -> LA: 200.0000
  Santa Fe -> Houston: 198.0959
  Santa Fe -> Atlanta: 101.9041
  El Paso -> LA: 0.0000
  El Paso -> Houston: 125.2483
  El Paso -> Atlanta: 74.7517
  Tampa Bay -> NY: 200.0000
  Tampa Bay -> Houston: 0.0000
  Tampa Bay -> Atlanta: 0.0000
  Houston -> Chicago: 0.0000
  Houston -> LA: 0.0000
  Houston -> NY: 23.3442
  Houston -> Atlanta: 0.0000
  Atlanta -> Chicago: 0.0000
  Atlanta -> NY: 26.6558
  Atlanta -> Houston: 0.0000
  Houston -> Customers: 300.0000
  Atlanta -> Customers: 150.0000
  Chicago -> Customers: 200.0000
  LA -> Customers: 200.0000
  NY -> Customers: 250.0000
Optimal cost: $6250.0000
-----------------
Houston strike scenario:
Optimal supplies on each route:
  Source -> Santa Fe: 700.0000
  Source -> El Paso: 200.0000
  Source -> Tampa Bay: 200.0000
```

```
  Santa Fe -> Chicago: 200.0000
  Santa Fe -> LA: 200.0000
  Santa Fe -> Houston: 100.0000
  Santa Fe -> Atlanta: 200.0000
  El Paso -> LA: 0.0000
  El Paso -> Houston: 100.0000
  El Paso -> Atlanta: 100.0000
  Tampa Bay -> NY: 200.0000
  Tampa Bay -> Houston: 0.0000
  Tampa Bay -> Atlanta: 0.0000
  Houston -> Chicago: 0.0000
  Houston -> LA: 0.0000
  Houston -> NY: 0.0000
  Houston -> Atlanta: 0.0000
  Atlanta -> Chicago: 0.0000
  Atlanta -> NY: 50.0000
  Atlanta -> Houston: 100.0000
  Houston -> Customers: 300.0000
  Atlanta -> Customers: 150.0000
  Chicago -> Customers: 200.0000
  LA -> Customers: 200.0000
  NY -> Customers: 250.0000
Optimal cost: $6050.0000
```

The higher prices at Houston cause the highest value of the shipping cost. We can shift some of the Houston routes to Atlanta, but because Houston was a highly used hub in the original program, it's more difficult the avoid using it despite the high prices.

It turns out letting Houston strike is better than giving into their demands. Letting them strike, we can just re-route things, and actually take advantage of Atlanta at this time. Overall, disruptions in Houston will be more costly than disruptions in LA since Houston is a hub and has more routes (and more ducks) entering and leaving.

6. Finally, return to the non-strike scenario, and consider the value of the goods being made and sold. The following table shows the profit made at each city from selling 1 rubber duck. Note that in the warehouse cities, you are making the goods, which amounts to a cost rather than revenue. Now, use a related linear program to maximize the total profit, taking into account the shipping costs. Here assume that supply and demand are only upper bounds for the actual production and sales correspondingly. Then compare it with the profit based on the flow from part 3. Report which cities underproduced and which were undersupplied.

Santa Fe	El Paso	Tampa Bay	Chicago	NY	Houston	Atlanta	LA
-$8	-$5	-$10	$15	$25	$10	$10	$20

Answer:
Now, we assume there is a production cost (negative profits at the warehouses) and profit for ducks sold at the store. We want to now maximize profit subject to the same constraints in the original linear program. First, we need to determine the objective function. Taking the price of transportation as a cost (negative profit), we need to change the signs of all

coefficients in the original program. Then, we change the cost of the routes from source to warehouse and from store to customer from zero to the profit listed in the table.

$$\begin{aligned} f(\boldsymbol{x}) = &-8x_1 - 5x_2 - 10x_3 - 6x_4 - 3x_5 - 3x_6 - 7x_7 - 7x_8 - 2x_9 - 5x_{10} - 7x_{11} - 6x_{12} \\ &- 4x_{13} - 4x_{14} - 5x_{15} - 6x_{16} - 2x_{17} - 4x_{18} - 5x_{19} - 2x_{20} + 10x_{21} + 10x_{22} + 15x_{23} \\ &+ 20x_{24} + 25x_{25}. \end{aligned}$$

All of the constraints remain the same. Since this a maximization problem, we input the negative of the coefficient vector of the above function into the call to `linprog`.

For the case where undersupply and underproduction is allowed, we change all equality constraints in the original program to inequality constraints.

Maximizing profit

```
# to evaluate profit we take transportation costs as negative
c_profit=np.copy(-c)

# assign production costs as negative (routes originating from source used to have zero cost)
c_profit[routes[1]==city['Santa Fe']]=-8
c_profit[routes[1]==city['El Paso']]=-5
c_profit[routes[1]==city['Tampa Bay']]=-10


#assign profits to all routes to customers (which used to have zero values)
c_profit[(routes[1]==city['Customers'])*(routes[0]==city['Chicago'])]=15
c_profit[(routes[1]==city['Customers'])*(routes[0]==city['LA'])]=20
c_profit[(routes[1]==city['Customers'])*(routes[0]==city['NY'])]=25
c_profit[(routes[1]==city['Customers'])*(routes[0]==city['Houston'])]=10
c_profit[(routes[1]==city['Customers'])*(routes[0]==city['Atlanta'])]=10

# reporting profit function would change a little
def report_profit(results):
    print('Optimal supplies on each route:')
    for i in range(nvar):
        print(' ',cities[routes[0,i]],'->',cities[routes[1,i]]+':',f"{results.x[i]:.4f}")
    print('Optimal profit:', f"${-results.fun:.4f}")


# Now we consider 2 scenarios
print('(A) maximizing the profit with fixed production and sold amounts')

result_profit_A=linprog((-1)*c_profit,A_eq=Aeq,b_eq=beq,A_ub=A,b_ub=b)
report_profit(result_profit_A)

# clearly in this scenario the optimal solution is the same as for the very first case,
# and only the objective value is affected by the new costs/profits
print('Profit evaluated using earlier solution:', f"{np.dot(c_profit,result.x):.4f}")

print('----------------------------')
print('(B) maximizing the profit, while allowing to underproduce and not fully satify the demand')
# in this case both supply and demand become upper bounds
# Thus we just omit them as the lower bounds
result_profit_B=linprog((-1)*c_profit,A_eq=Aeq,b_eq=beq,A_ub=Aub,b_ub=bub)
report_profit(result_profit_B)

# Now let us see which how actual production and sales differ from possible supply and demand
# this is given by the slack in the corresponding inequality
print('Discrepancy by city:')
slack=result_profit_B.slack

production_cities=['Santa Fe', 'El Paso', 'Tampa Bay']
for name in production_cities:
    ind=np.arange(nvar)[routes[1]==city[name]][0]
    print(' ', name+':',f"{slack[ind]:.4f}",'underproduced')

sale_cities=['Houston', 'Atlanta', 'Chicago', 'LA', 'NY']
for name in sale_cities:
    ind=np.arange(nvar)[(routes[0]==city[name])*(routes[1]==city['Customers'])][0]
    print(' ', name+':',f"{slack[ind]:.4f}",'undersupplied')
```

```
(A) maximizing the profit with fixed production and sold amounts
Optimal supplies on each route:
  Source -> Santa Fe: 700.0000
  Source -> El Paso: 200.0000
  Source -> Tampa Bay: 200.0000
  Santa Fe -> Chicago: 200.0000
  Santa Fe -> LA: 200.0000
```

```
  Santa Fe -> Houston: 200.0000
  Santa Fe -> Atlanta: 100.0000
  El Paso -> LA: 0.0000
  El Paso -> Houston: 200.0000
  El Paso -> Atlanta: 0.0000
  Tampa Bay -> NY: 200.0000
  Tampa Bay -> Houston: 0.0000
  Tampa Bay -> Atlanta: 0.0000
  Houston -> Chicago: 0.0000
  Houston -> LA: 0.0000
  Houston -> NY: 50.0000
  Houston -> Atlanta: 50.0000
  Atlanta -> Chicago: 0.0000
  Atlanta -> NY: 0.0000
  Atlanta -> Houston: 0.0000
  Houston -> Customers: 300.0000
  Atlanta -> Customers: 150.0000
  Chicago -> Customers: 200.0000
  LA -> Customers: 200.0000
  NY -> Customers: 250.0000
Optimal profit: $3850.0000
Profit evaluated using earlier solution: 3850.0000
-------------------------
(B) maximizing the profit, while allowing to
    underproduce and not fully satify the demand
Optimal supplies on each route:
  Source -> Santa Fe: 400.0000
  Source -> El Paso: 200.0000
  Source -> Tampa Bay: 50.0000
  Santa Fe -> Chicago: 200.0000
  Santa Fe -> LA: 200.0000
  Santa Fe -> Houston: 0.0000
  Santa Fe -> Atlanta: 0.0000
  El Paso -> LA: 0.0000
  El Paso -> Houston: 200.0000
  El Paso -> Atlanta: 0.0000
  Tampa Bay -> NY: 50.0000
  Tampa Bay -> Houston: 0.0000
  Tampa Bay -> Atlanta: 0.0000
  Houston -> Chicago: 0.0000
  Houston -> LA: 0.0000
  Houston -> NY: 200.0000
  Houston -> Atlanta: 0.0000
  Atlanta -> Chicago: 0.0000
  Atlanta -> NY: 0.0000
  Atlanta -> Houston: 0.0000
  Houston -> Customers: 0.0000
  Atlanta -> Customers: 0.0000
  Chicago -> Customers: 200.0000
  LA -> Customers: 200.0000
```

```
  NY -> Customers: 250.0000
Optimal profit: $4799.9996
Discrepancy by city:
  Santa Fe: 300.0000 underproduced
  El Paso: 0.0000 underproduced
  Tampa Bay: 150.0000 underproduced
  Houston: 300.0000 undersupplied
  Atlanta: 150.0000 undersupplied
  Chicago: 0.0000 undersupplied
  LA: 0.0000 undersupplied
  NY: 0.0000 undersupplied
```

In the first case with fixed production, the optimal route is the same as the original linear program. The optimal value changes (since the objective function did) to give us a profit of $3,850. This makes sense, as the original program found the route to minimize the shipping cost, and all we've done is add costs to routes that have a fixed number of ducks being transported on them.

In the case where underproduction and undersupply is allowed, we get a profit of $4,800, which is higher than the profit with fixed supply and demand. Surprisingly, we are better off not producing and selling all of the ducks. However, looking at the optimal flow, it shouldn't be that surprising. At Houston and Atlanta, we only make $10 per duck, which compared to prices to produce and get ducks there leaves little to no room for a profit at all. We see that at the stores with higher selling prices for ducks, the full demand is met since they are the most profitable. The underproduction results also make sense looking at costs of production and routes from each warehouse. El Paso has the lowest production cost, so it makes sense to make the full amount of ducks there. Furthermore, shipping to Houston is cheap, and from Houston we can go to NY which has the highest profit per duck. While Tampa Bay has the highest production cost, we can go straight to NY to get the highest profit, so while we don't produce all ducks at Tampa Bay, we produce enough to turn the max profit with a direct route to NY. Finally, Santa Fe has a relatively high production cost, but ducks can be shipped directly to Chicago and LA, the other two most profitable stores.