midterm-report1

March 13, 2024

[27]:
```python
import numpy as np
from scipy.optimize import linprog
from math import inf
from itertools import product
```

[28]:
```python
warehouse_cities = [ 'Santa Fe',
                     'El Paso',
                     'Tampa Bay'
                    ]

store_cities = [ 'Chicago',
                 'LA',
                 'NY',
                 'Houston',
                 'Atlanta'
                ]

hubs = [ 'Houston', 'Atlanta' ]

vertices=[ 'Source',
           *warehouse_cities,
           *store_cities,
           'Demand'
          ]

supplies = { 'Santa Fe': 700,
             'El Paso': 200,
             'Tampa Bay': 200
            }

demand = { 'Chicago': 200,
           'LA': 200,
           'NY': 250,
           'Houston': 300,
           'Atlanta': 150
          }
```

```
[29]: def ship_costs(f,t):
    match (f,t):
        case 'Source',_:              # no shipping cost for "shipments" from␣
↪source to warehouse
            return 0

        case _,'Demand':              # no shipping costs for "shipments" from␣
↪store to customers
            return 0


        case 'Santa Fe','Chicago':
            return 6
        case 'Santa Fe','LA':
            return 3
        case 'Santa Fe','Houston':
            return 3
        case 'Santa Fe','Atlanta':
            return 7

        case 'El Paso','LA':
            return 7
        case 'El Paso','Houston':
            return 2
        case 'El Paso','Atlanta':
            return 5

        case 'Tampa Bay','NY':
            return 7
        case 'Tampa Bay','Houston':
            return 6
        case 'Tampa Bay','Atlanta':
            return 4

        case _:
            return inf

def relay_costs(f,t):
    match (f,t):
        case 'Houston','Chicago':
            return 4
        case 'Houston','LA':
            return 5
        case 'Houston','NY':
            return 6
        case 'Houston','Atlanta':
            return 2
```

```python
        case 'Atlanta','Chicago':
            return 4
        case 'Atlanta','NY':
            return 5
        case 'Atlanta','Houston':
            return 2

        case _:
            return inf
```

[30]:

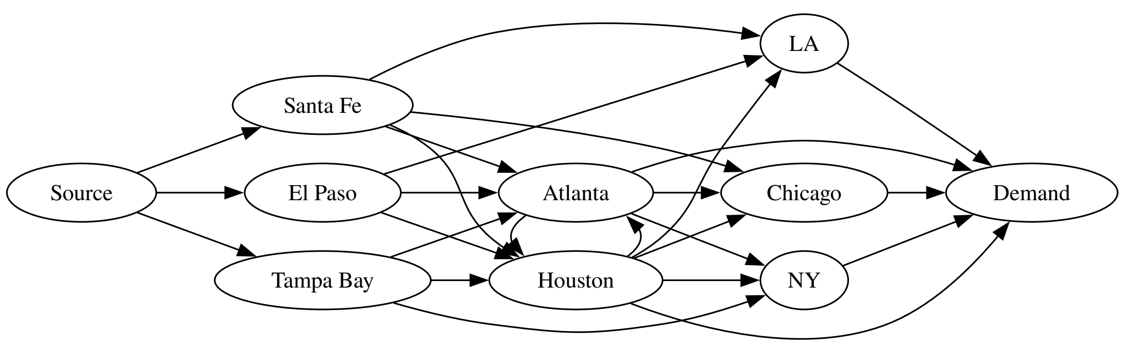
```python
edges_source = [ { 'from': 'Source',
                   'to': c,
                   }
                for c in warehouse_cities ]


edges_demand = [ { 'from': c,
                   'to': 'Demand',
                   }
                 for c in store_cities
                ]


edges_ship = [ { 'from': source,
                 'to': dest,
                }
               for source,dest in product(warehouse_cities,store_cities)
               if ship_costs(source,dest) != inf
              ]


edges_relay = [ { 'from': source,
                  'to': dest,
                 }
                for source,dest in product(store_cities,store_cities)
                if relay_costs(source,dest) != inf
               ]



edges =  edges_source + edges_ship + edges_relay + edges_demand
```

[31]:

```python
#---------------------------------------------------------------------------------
from graphviz import Digraph as GVDigraph
```

```
dot = GVDigraph("example",format='png')
dot.attr(rankdir='LR')

dot.node('Source')

with dot.subgraph(name='warehouse') as c:
    c.attr(rank='same')
    for vertex in warehouse_cities:
        c.node(vertex)

with dot.subgraph(name='hubs') as c:
    c.attr(rank='same')
    for vertex in hubs:
        c.node(vertex)

with dot.subgraph(name='stores') as c:
    c.attr(rank='same')
    for vertex in store_cities:
        if not (vertex in hubs):
            c.node(vertex)


c.node('Demand')

for e in edges:
#  dot.edge(e["from"],e["to"],label=f"costs {e['ship_costs']}")
  dot.edge(e["from"],e["to"])

#dot.render('graph')
dot
```

[31]:



[32]:
```
# return a standard basis vector
# these are "0-indexed" e.g. sbv(0,3) == [1,0,0]
def sbv(index,size):
```

```
    return np.array([1.0 if i == index else 0.0 for i in range(size)])


# create the objective vector for the "costs" linear program
ship_costs_obj = sum([ ship_costs(e['from'],e['to'])*sbv(edges.
 ↪index(e),len(edges))
                       for e in edges_ship])

relay_costs_obj = sum([ relay_costs(e['from'],e['to'])*sbv(edges.
 ↪index(e),len(edges))
                        for e in edges_relay])

costs_obj = ship_costs_obj + relay_costs_obj

#------------------------------------------------------------------------------
def getIncoming(vertex,edges):
    return [ e for e in edges if e["to"] == vertex ]

def getOutgoing(vertex,edges):
    return [ e for e in edges if e["from"] == vertex ]

def isSource(vertex,edges):
    return getIncoming(vertex,edges) == []

def isSink(vertex,edges):
    return getOutgoing(vertex,edges) == []

def interiorVertices(vertices,edges):
    return [ v for v in vertices if not( isSource(v,edges) or isSink(v,edges) )␣
 ↪]

#------------------------------------------------------------------------------

def conservationLaw(vertex,edges):
    ii = sum([ sbv(edges.index(e),len(edges)) for e in␣
 ↪getIncoming(vertex,edges) ])
    oo = sum([ sbv(edges.index(e),len(edges)) for e in␣
 ↪getOutgoing(vertex,edges) ])
    return ii - oo

conservationMatrix =np.array([conservationLaw(v,edges) for v in␣
 ↪interiorVertices(vertices,edges) ])
```

[33]:
```
list(zip(edges,costs_obj))
```

[33]:
```
[({'from': 'Source', 'to': 'Santa Fe'}, 0.0),
 ({'from': 'Source', 'to': 'El Paso'}, 0.0),
 ({'from': 'Source', 'to': 'Tampa Bay'}, 0.0),
 ({'from': 'Santa Fe', 'to': 'Chicago'}, 6.0),
 ({'from': 'Santa Fe', 'to': 'LA'}, 3.0),
 ({'from': 'Santa Fe', 'to': 'Houston'}, 3.0),
 ({'from': 'Santa Fe', 'to': 'Atlanta'}, 7.0),
 ({'from': 'El Paso', 'to': 'LA'}, 7.0),
 ({'from': 'El Paso', 'to': 'Houston'}, 2.0),
 ({'from': 'El Paso', 'to': 'Atlanta'}, 5.0),
 ({'from': 'Tampa Bay', 'to': 'NY'}, 7.0),
 ({'from': 'Tampa Bay', 'to': 'Houston'}, 6.0),
 ({'from': 'Tampa Bay', 'to': 'Atlanta'}, 4.0),
 ({'from': 'Houston', 'to': 'Chicago'}, 4.0),
 ({'from': 'Houston', 'to': 'LA'}, 5.0),
 ({'from': 'Houston', 'to': 'NY'}, 6.0),
 ({'from': 'Houston', 'to': 'Atlanta'}, 2.0),
 ({'from': 'Atlanta', 'to': 'Chicago'}, 4.0),
 ({'from': 'Atlanta', 'to': 'NY'}, 5.0),
 ({'from': 'Atlanta', 'to': 'Houston'}, 2.0),
 ({'from': 'Chicago', 'to': 'Demand'}, 0.0),
 ({'from': 'LA', 'to': 'Demand'}, 0.0),
 ({'from': 'NY', 'to': 'Demand'}, 0.0),
 ({'from': 'Houston', 'to': 'Demand'}, 0.0),
 ({'from': 'Atlanta', 'to': 'Demand'}, 0.0)]
```

[34]:
```
# return the edge from the list `edges` with 'from': f and 'to': t
#
def lookupEdge(f,t):
    r = list(filter(lambda x: x['from'] == f and x['to'] == t, edges))
    if r != []:
        return r[0]
    else:
        return "error"


def lookupEdgeIndex(f,t):
    r = lookupEdge(f,t)
    return edges.index(r)


# create equality constraint matrix for the supply (at warehouses) and demand␣
↪(at stores)

Aeq_costs = np.concatenate([ conservationMatrix,
                             [ sbv(lookupEdgeIndex('Source',w),len(edges))
                               for w in warehouse_cities ],
```

```
                              [ sbv(lookupEdgeIndex(s,'Demand'),len(edges))
                                for s in store_cities ]
                             ],axis=0)
beq_costs = np.concatenate([ np.zeros(len(conservationMatrix)),
                              [ supplies[w] for w in warehouse_cities ],
                              [ demand[s] for s in store_cities ]
                             ])



# create inequality constraint matrix
# initially the only thing to account for is "can't ship more than 200 ducks"

Aub_costs = np.array([ sbv(edges.index(e),len(edges)) for e in edges_ship ]
                     + [ sbv(edges.index(e),len(edges)) for e in edges_relay ])

bub_costs = np.array([ 200 for e in edges_ship]
                     + [ 200 for e in edges_relay ] )
```

[35]:
```
## run the "costs" linear program

costs_result = linprog(costs_obj,
                       A_eq = Aeq_costs,
                       b_eq = beq_costs,
                       A_ub = Aub_costs,
                       b_ub = bub_costs
                       )



def display_result(res):
    print(f"Objective value: {abs(res.fun)}")
    print("============================")
    for e in edges:
        i = edges.index(e)
        print(f"{e['from']} -> {e['to']}: {res.x[i]}")

display_result(costs_result)
```

```
Objective value: 5300.0
============================
Source -> Santa Fe: 700.0
Source -> El Paso: 200.0
Source -> Tampa Bay: 200.0
Santa Fe -> Chicago: 200.0
Santa Fe -> LA: 200.0
Santa Fe -> Houston: 200.0
Santa Fe -> Atlanta: 100.0
El Paso -> LA: 0.0
El Paso -> Houston: 200.0
```

```
El Paso -> Atlanta: -0.0
Tampa Bay -> NY: 200.0
Tampa Bay -> Houston: 0.0
Tampa Bay -> Atlanta: 0.0
Houston -> Chicago: 0.0
Houston -> LA: 0.0
Houston -> NY: 50.0
Houston -> Atlanta: 50.0
Atlanta -> Chicago: 0.0
Atlanta -> NY: 0.0
Atlanta -> Houston: 0.0
Chicago -> Demand: 200.0
LA -> Demand: 200.0
NY -> Demand: 250.0
Houston -> Demand: 300.0
Atlanta -> Demand: 150.0
```

[36]:

```
## LA situation - demand scenario

def LA_demand_ship_costs(f,t):
    match (f,t):
        case (_,'LA'):
            return 2*ship_costs(f,t)      ## double shipping costs to LA
        case _:
            return ship_costs(f,t)


def LA_demand_relay_costs(f,t):
    match (f,t):
        case (_,'LA'):
            return 2*relay_costs(f,t)      ## double shipping costs to LA
        case _:
            return relay_costs(f,t)



# this results in a new objective function

LA_demand_ship_costs_obj = sum([␣
 ↪LA_demand_ship_costs(e['from'],e['to'])*sbv(edges.index(e),len(edges))
                       for e in edges_ship])

LA_demand_relay_costs_obj = sum([ relay_costs(e['from'],e['to'])*sbv(edges.
 ↪index(e),len(edges))
                        for e in edges_relay])

LA_demand_costs_obj = LA_demand_ship_costs_obj + LA_demand_relay_costs_obj
```

```
# results

LA_demand_costs_result = linprog(LA_demand_costs_obj,
                                 A_eq = Aeq_costs,
                                 b_eq = beq_costs,
                                 A_ub = Aub_costs,
                                 b_ub = bub_costs
                                 )

display_result(LA_demand_costs_result)
```

```
Objective value: 5900.0
============================
Source -> Santa Fe: 700.0
Source -> El Paso: 200.0
Source -> Tampa Bay: 200.0
Santa Fe -> Chicago: 200.0
Santa Fe -> LA: 200.0
Santa Fe -> Houston: 200.0
Santa Fe -> Atlanta: 100.0
El Paso -> LA: 0.0
El Paso -> Houston: 200.0
El Paso -> Atlanta: -0.0
Tampa Bay -> NY: 200.0
Tampa Bay -> Houston: 0.0
Tampa Bay -> Atlanta: 0.0
Houston -> Chicago: 0.0
Houston -> LA: 0.0
Houston -> NY: 50.0
Houston -> Atlanta: 50.0
Atlanta -> Chicago: 0.0
Atlanta -> NY: 0.0
Atlanta -> Houston: 0.0
Chicago -> Demand: 200.0
LA -> Demand: 200.0
NY -> Demand: 250.0
Houston -> Demand: 300.0
Atlanta -> Demand: 150.0
```

[37]:
```
## strike scenario



LA_strike_Aub_costs = np.array([ sbv(edges.index(e),len(edges)) for e in␣
  ↪edges_ship ]
```

```python
                              + [ sbv(edges.index(e),len(edges)) for e in␣
 ↪edges_relay ])

def LA_strike_capacity(e):
    match e['to']:
        case 'LA':
            return 100
        case _:
            return 200

LA_strike_bub_costs = np.array([ LA_strike_capacity(e) for e in edges_ship]
                              + [ LA_strike_capacity(e) for e in edges_relay ]␣
 ↪)


LA_strike_costs_result = linprog(costs_obj,
                                 A_eq = Aeq_costs,
                                 b_eq = beq_costs,
                                 A_ub = LA_strike_Aub_costs,
                                 b_ub = LA_strike_bub_costs
                                 )

display_result(LA_strike_costs_result)
```

```
Objective value: 6050.0
============================
Source -> Santa Fe: 700.0
Source -> El Paso: 200.0
Source -> Tampa Bay: 200.0
Santa Fe -> Chicago: 200.0
Santa Fe -> LA: 100.0
Santa Fe -> Houston: 200.0
Santa Fe -> Atlanta: 200.0
El Paso -> LA: 0.0
El Paso -> Houston: 200.0
El Paso -> Atlanta: 0.0
Tampa Bay -> NY: 200.0
Tampa Bay -> Houston: 0.0
Tampa Bay -> Atlanta: 0.0
Houston -> Chicago: 0.0
Houston -> LA: 100.0
Houston -> NY: 0.0
Houston -> Atlanta: 0.0
Atlanta -> Chicago: -0.0
Atlanta -> NY: 50.0
Atlanta -> Houston: 0.0
Chicago -> Demand: 200.0
```

```
LA -> Demand: 200.0
NY -> Demand: 250.0
Houston -> Demand: 300.0
Atlanta -> Demand: 150.0
```

[38]:

```
## Houston situation

## demand scenario

def Houston_demand_ship_costs(f,t):
    match (f,t):
        case (_,'Houston'):
            return 2*ship_costs(f,t)      ## double shipping costs to Houston
        case _:
            return ship_costs(f,t)


def Houston_demand_relay_costs(f,t):
    match (f,t):
        case (_,'Houston'):
            return 2*relay_costs(f,t)      ## double shipping costs to Houston
        case _:
            return relay_costs(f,t)



# this results in a new objective function

Houston_demand_ship_costs_obj = sum([␣
 ↪Houston_demand_ship_costs(e['from'],e['to'])*sbv(edges.index(e),len(edges))
                       for e in edges_ship])

Houston_demand_relay_costs_obj = sum([ relay_costs(e['from'],e['to'])*sbv(edges.
 ↪index(e),len(edges))
                        for e in edges_relay])

Houston_demand_costs_obj = Houston_demand_ship_costs_obj +␣
 ↪Houston_demand_relay_costs_obj

# results


Houston_demand_costs_result = linprog(Houston_demand_costs_obj,
                                      A_eq = Aeq_costs,
                                      b_eq = beq_costs,
                                      A_ub = Aub_costs,
                                      b_ub = bub_costs
```

```
                                               )
display_result(Houston_demand_costs_result)
```

```
Objective value: 6250.0
============================
Source -> Santa Fe: 700.0
Source -> El Paso: 200.0
Source -> Tampa Bay: 200.0
Santa Fe -> Chicago: 200.0
Santa Fe -> LA: 200.0
Santa Fe -> Houston: 100.0
Santa Fe -> Atlanta: 200.0
El Paso -> LA: 0.0
El Paso -> Houston: 200.0
El Paso -> Atlanta: -0.0
Tampa Bay -> NY: 200.0
Tampa Bay -> Houston: 0.0
Tampa Bay -> Atlanta: 0.0
Houston -> Chicago: 0.0
Houston -> LA: 0.0
Houston -> NY: 0.0
Houston -> Atlanta: 0.0
Atlanta -> Chicago: 0.0
Atlanta -> NY: 50.0
Atlanta -> Houston: 0.0
Chicago -> Demand: 200.0
LA -> Demand: 200.0
NY -> Demand: 250.0
Houston -> Demand: 300.0
Atlanta -> Demand: 150.0
```

[39]:
```
# houston strike

## strike scenario



strike_Aub_costs = np.array([ sbv(edges.index(e),len(edges)) for e in␣
↪edges_ship ]
                            + [ sbv(edges.index(e),len(edges)) for e in␣
↪edges_relay ])

def strike_capacity(e):
    match e['to']:
        case 'Houston':
            return 100
        case _:
```

```python
            return 200

strike_bub_costs = np.array([ strike_capacity(e) for e in edges_ship]
                            + [ strike_capacity(e) for e in edges_relay ] )


Houston_strike_costs_result = linprog(costs_obj,
                                 A_eq = Aeq_costs,
                                 b_eq = beq_costs,
                                 A_ub = strike_Aub_costs,
                                 b_ub = strike_bub_costs
                                 )
display_result(Houston_strike_costs_result)
```

```
Objective value: 6050.0
============================
Source -> Santa Fe: 700.0
Source -> El Paso: 200.0
Source -> Tampa Bay: 200.0
Santa Fe -> Chicago: 200.0
Santa Fe -> LA: 200.0
Santa Fe -> Houston: 100.0
Santa Fe -> Atlanta: 200.0
El Paso -> LA: -0.0
El Paso -> Houston: 100.0
El Paso -> Atlanta: 100.0
Tampa Bay -> NY: 200.0
Tampa Bay -> Houston: 0.0
Tampa Bay -> Atlanta: 0.0
Houston -> Chicago: 0.0
Houston -> LA: 0.0
Houston -> NY: 0.0
Houston -> Atlanta: 0.0
Atlanta -> Chicago: -0.0
Atlanta -> NY: 50.0
Atlanta -> Houston: 100.0
Chicago -> Demand: 200.0
LA -> Demand: 200.0
NY -> Demand: 250.0
Houston -> Demand: 300.0
Atlanta -> Demand: 150.0
```

[40]:
```python
def profit(e):
    match e['from'],e['to']:
        case 'Source','Santa Fe':
            return -8
        case 'Source','El Paso':
```

```
            return -5
        case 'Source','Tampa Bay':
            return -10
        case 'Chicago','Demand':
            return 15
        case 'NY','Demand':
            return 25
        case 'Houston','Demand':
            return 10
        case 'Atlanta','Demand':
            return 10
        case 'LA','Demand':
            return 20
        case _:
            return 0

#sales = np.array([ profit(e) for e in edges])
sales = sum([profit(e)*sbv(edges.index(e),len(edges)) for e in edges ])

profit_obj = sales - costs_obj
```

[41]:
```
list(zip(edges,costs_obj))
```

[41]:
```
[({'from': 'Source', 'to': 'Santa Fe'}, 0.0),
 ({'from': 'Source', 'to': 'El Paso'}, 0.0),
 ({'from': 'Source', 'to': 'Tampa Bay'}, 0.0),
 ({'from': 'Santa Fe', 'to': 'Chicago'}, 6.0),
 ({'from': 'Santa Fe', 'to': 'LA'}, 3.0),
 ({'from': 'Santa Fe', 'to': 'Houston'}, 3.0),
 ({'from': 'Santa Fe', 'to': 'Atlanta'}, 7.0),
 ({'from': 'El Paso', 'to': 'LA'}, 7.0),
 ({'from': 'El Paso', 'to': 'Houston'}, 2.0),
 ({'from': 'El Paso', 'to': 'Atlanta'}, 5.0),
 ({'from': 'Tampa Bay', 'to': 'NY'}, 7.0),
 ({'from': 'Tampa Bay', 'to': 'Houston'}, 6.0),
 ({'from': 'Tampa Bay', 'to': 'Atlanta'}, 4.0),
 ({'from': 'Houston', 'to': 'Chicago'}, 4.0),
 ({'from': 'Houston', 'to': 'LA'}, 5.0),
 ({'from': 'Houston', 'to': 'NY'}, 6.0),
 ({'from': 'Houston', 'to': 'Atlanta'}, 2.0),
 ({'from': 'Atlanta', 'to': 'Chicago'}, 4.0),
 ({'from': 'Atlanta', 'to': 'NY'}, 5.0),
 ({'from': 'Atlanta', 'to': 'Houston'}, 2.0),
 ({'from': 'Chicago', 'to': 'Demand'}, 0.0),
 ({'from': 'LA', 'to': 'Demand'}, 0.0),
 ({'from': 'NY', 'to': 'Demand'}, 0.0),
 ({'from': 'Houston', 'to': 'Demand'}, 0.0),
```

```
 ({'from': 'Atlanta', 'to': 'Demand'}, 0.0)]
```

[42]:
```
costs_obj
```

[42]:
```
array([0., 0., 0., 6., 3., 3., 7., 7., 2., 5., 7., 6., 4., 4., 5., 6., 2.,
       4., 5., 2., 0., 0., 0., 0., 0.])
```

[43]:
```
Aeq_profit = conservationMatrix
beq_profit = np.zeros(len(conservationMatrix))


Aub_profit = np.concatenate([ [ sbv(edges.index(e),len(edges)) for e in␣
 ↪edges_ship ],
                              [ sbv(edges.index(e),len(edges)) for e in␣
 ↪edges_relay ],
                              [ sbv(lookupEdgeIndex('Source',w),len(edges))
                                for w in warehouse_cities ],
                              [ sbv(lookupEdgeIndex(s,'Demand'),len(edges))
                                for s in store_cities]
                              ]
                            , axis=0)

bub_profit = np.concatenate([ [ 200 for e in edges_ship],
                              [ 200 for e in edges_relay ],
                              [ supplies[w] for w in warehouse_cities ],
                              [ demand[s] for s in store_cities ]
                             ])
```

[44]:
```
profit_result = linprog((-1)*profit_obj,
                        A_eq = Aeq_profit,
                        b_eq = beq_profit,
                        A_ub = Aub_profit,
                        b_ub = bub_profit)

display_result(profit_result)
```

```
Objective value: 4800.0
============================
Source -> Santa Fe: 400.0
Source -> El Paso: 200.0
Source -> Tampa Bay: 50.0
Santa Fe -> Chicago: 200.0
Santa Fe -> LA: 200.0
Santa Fe -> Houston: 0.0
Santa Fe -> Atlanta: 0.0
El Paso -> LA: 0.0
El Paso -> Houston: 200.0
El Paso -> Atlanta: 0.0
```

```
Tampa Bay -> NY: 50.0
Tampa Bay -> Houston: 0.0
Tampa Bay -> Atlanta: 0.0
Houston -> Chicago: 0.0
Houston -> LA: 0.0
Houston -> NY: 200.0
Houston -> Atlanta: 0.0
Atlanta -> Chicago: 0.0
Atlanta -> NY: -0.0
Atlanta -> Houston: 0.0
Chicago -> Demand: 200.0
LA -> Demand: 200.0
NY -> Demand: 250.0
Houston -> Demand: 0.0
Atlanta -> Demand: 0.0
```

[ ]:
```

```