SYSEN 5200: Systems Analysis Behavior and Optimization

Optimization

Nick Kunz [NetID: nhk37] nhk37@cornell.edu

March 28, 2023

1. A figure represents an oil pipeline network. The nodes represent pumping and/or receiving stations. The lengths in miles of the different segments of the network are shown on the respective arcs. The bi-directional segments allow flows in both directions. The supplies at stations 1 and 3 are respectively 60 and 50 barrels per day. The demands at the stations 2 and 4 are respectively 85 and 25 barrels per day. Assume that the transportation cost is proportional to the distance and we are interested in minimizing the total transportation cost.

   (a) Let x_{ij} be the amount of flow over arc (i, j), so that a formulation of the problem can be expressed as:

$$\begin{aligned}
\text{min.}\ & 15x_{12} + 6x_{17} + 40x_{72} + 10x_{62} + 9x_{37} + 16x_{34} + \\
& 8x_{54} + 12x_{57} + 12x_{75} + 50x_{56} + 50x_{65} \\
\text{s.t.}\ & x_{12} + x_{17} = 60 \\
& -x_{12} - x_{72} - x_{62} = -85 \\
& x_{37} + x_{34} = 50 \\
& -x_{34} - x_{45} = -25 \\
& x_{62} - x_{56} = 0 \\
& x_{54} - x_{56} - x_{75} = 0 \\
& x_{72} + x_{75} - x_{17} - x_{37} - x_{57} = 0 \\
& x_{ij} \geq 0, \forall\, i, j
\end{aligned} \tag{1}$$

   (b) This is a linear optimization problem, as the objective function and constraints are linear functions. Therefore, the problem can be solved using linear programming techniques.

2. A tech startup was founded in 2021. The following table shows an estimate of active numbers of contractors the company needs for any given year (which includes new hires as well as those staying on due to a longer-term contract) over the next 5 years:

Year	2022	2023	2024	2025	2026
n SDE	110	130	70	165	50

   The company can choose to sign different length contracts with each contractor it hires and wants to keep more contractors than needed during some years of the planning horizon. The startup estimates that the cost of recruiting and maintaining the contractors is a function of their length of stay with the company. The following table summarizes these estimates:

Length of Employment Contract (Years)	1	2	3	4	5
Total Cost for All Years per Contractor	110	140	170	230	250

The startup is interested in the recruiting schedule to minimize the total cost.

(a) Let $x_{i,j}$ be n SDE's hired in year i with a contract of length j and $y_{i,j}$ be n SDE's already under contract, where $i \in 2022, 2023, 2024, 2025, 2026$ and $j \in 1, 2, 3, 4, 5$. Let c_j be the cost per contractor for a contract of length j, so that a formulation of the problem can be expressed as:

$$\begin{aligned} \text{min.} \quad & \sum_{i=2022}^{2026} \sum_{j=1}^{5} (x_{i,j} + y_{i,j}) \cdot c_j \\ \text{s.t.} \quad & \sum_{j=1}^{5} x_{i,j} \geq n_i - \sum_{k=1}^{5} (x_{i-k+1,k} + y_{i-k+1,k}), \forall\, i \\ & x_{i,j} \geq 0, y_{i,j} \geq 0, \forall\, i, j \end{aligned}$$

where n_i is the number of contractors needed in year i from the first table.

(b) This is a linear optimization problem, as the objective function and constraints are linear functions. Therefore, the problem can be solved using linear programming techniques.

(c) The optimal solution when solved, can determined with the following:

```
## library
from pulp import *

## years contracts and cost
years = [2022, 2023, 2024, 2025, 2026]
n = [110, 130, 70, 165, 50]
c = [110, 140, 170, 230, 250]

## optim model
model = LpProblem(
    name = "Tech Startup Hiring",
    sense = LpMinimize
)

## decision vars
x = LpVariable.dicts(
    "x", [(i, j) for i in years for j in range(1, 6)],
    lowBound = 0,
    cat = "Integer"
    )

y = LpVariable.dicts(
    "y", [(i, j) for i in years for j in range(1, 6)],
    lowBound = 0,
    cat = "Integer"
)

## obj func
model += lpSum(
    (x[(i, j)] + y[(i, j)]) * c[j - 1] for i in years for j in range(1, 6)
)

## constraints
for i in years:
    model += lpSum(x[(i, j)] for j in range(1, 6)) >= n[i - 2022] - \
```

```python
    lpSum(x[(i - k + 1, j)] + y[(i - k + 1, j)] for k in range(1, 6) \
    if i - k + 1 >= 2022)

## optim solver
model.solve()

## results
print(f"Minimum Labor Cost: {int(model.objective.value())}")
```

Fig. Python 2(c)

Minimum Labor Cost ($): 33,550

3. A company would like to choose the site of a new distribution center that will service four sales centers located in Arlington (A), Bainbridge (B), Charlotte (C) and Dryden (D). The table below gives the location of these sales centers in the plane, where the units are in miles. The table also gives the number of daily deliveries to each sales center from the distribution center. Suppose that truck delivery costs are $1 per mile.

Sales Center	Code	Daily Deliveries	x coord.	y coord.
Arlington	A	9	-3	-3
Bainbridge	B	6	4	9
Charlotte	C	5	7	14
Dryden	D	5	2	7

(a) Let T_{ij} be the number of daily deliveries from the distribution center to sales center i and j, and d_{ij} the distance between sales center i and j, so that a formulation to minimize the daily transportation cost, assuming that the trucks travel in a straight line, can be expressed as:

$$\begin{aligned}
\text{min.} \quad & \sum_{i \in A,B,C,D} \sum_{j \in A,B,C,D} T_{ij} \cdot d_{ij} \\
\text{s.t.} \quad & \sum_{i \in A,B,C,D} T_{ij} = d_j, \text{ for all } j \in A, B, C, D \\
& \sum_{j \in A,B,C,D} T_{ij} = d_i, \text{ for all } i \in A, B, C, D \\
& T_{ij} \geq 0, \text{ for all } i, j \in A, B, C, D \\
& d_{ij} = \sqrt{(x_i - x_d)^2 + (y_i - y_d)^2} + \sqrt{(x_j - x_d)^2 + (y_j - y_d)^2}, \text{ for all } i, j \in A, B, C, D \\
& x_d, y_d, d_{ij}, q_i \geq 0
\end{aligned}$$

where q_i is the daily delivery quantity of sales center i.

(b) This is a non-linear optimization problem because of the presence of a non-linear constraint in the distance calculation. Therefore, linear programming techniques cannot be used to solve the problem.

(c) The optimal solution when solved, can determined with the following:

```python
## libraries
import numpy as np
from scipy.optimize import minimize

## coord and demand of the sales centers
```

```python
coords = np.array([[-3, -3], [4, 9], [7, 14], [2, 7]])
demand = np.array([9, 6, 5, 5])

## compute transport cost
def tran_cost(dist_cent):
    dist = np.zeros((4, 4))
    for i in range(4):
        for j in range(i + 1, 4):
            dist[i, j] = np.sqrt(
                (coords[i, 0] - dist_cent[0]) ** 2 \
                + (coords[i, 1] - dist_cent[1]) ** 2) \
                + np.sqrt((coords[j, 0] - dist_cent[0]) ** 2 \
                + (coords[j, 1] - dist_cent[1]) ** 2
            )
            dist[j, i] = dist[i, j]
    T = np.zeros((4, 4))
    for i in range(4):
        for j in range(i + 1, 4):
            T[i, j] = T[j, i] = (demand[i] + demand[j]) / 2
    return np.sum(T * dist)

## init approx
x0 = np.array([0, 0])

## minimize objective
obj = minimize(
    fun = tran_cost,
    x0 = x0
)

## optimal solution
print(’Optimal Distribution Center Location:’, obj.x)
print(’Minimum Transportation Cost ($):’, round(
    number = obj.fun,
    ndigits = 2)
)
```

Fig. Python 3(c)

Optimal Distribution Center Location: [2, 7]   Minimum Transportation Cost ($): 160.61