Toy Example: Simple Transportation Problem - Model Report

Sample Solution

2025-02-05

1 Purpose of This Document

This simplified example demonstrates the **report format and style** expected for your Midterm Report 1. Your actual problem will be more complex, with more nodes, routes, constraints, and multiple parts to address. Use this as a guide for:

- How to draw and explain network diagrams
- How to formulate and describe linear programs
- How to present results with appropriate code references
- How to organize a code appendix

2 Problem Statement

A company manufactures widgets at two factories (Factory A and Factory B) and needs to ship them to three retail stores (Store 1, Store 2, and Store 3). Factory A can supply up to 100 widgets, and Factory B can supply up to 80 widgets. The stores require 60, 70, and 50 widgets respectively.

Shipping costs (in dollars per widget) are shown in the following table:

Table 1: Shipping costs ($ per widget)

	Store 1	Store 2	Store 3
Factory A	2	3	5
Factory B	4	2	3

Our goal is to determine the optimal shipping plan that minimizes total shipping cost while meeting all store demands.

2.1 Your tasks

(a) Formulate and draw a network flow model describing this problem. Explain all nodes, edges, and constraints. Include a network diagram.

(b) Use your network flow model to formulate the linear program. Clearly describe the objective function, equality constraints, and inequality constraints.

(c) Implement your linear program in Python using `scipy.optimize.linprog` and find the optimal solution. Report the optimal cost and shipping plan.

3 Part (a): Network Flow Model

We model this problem as a network flow with the following structure:

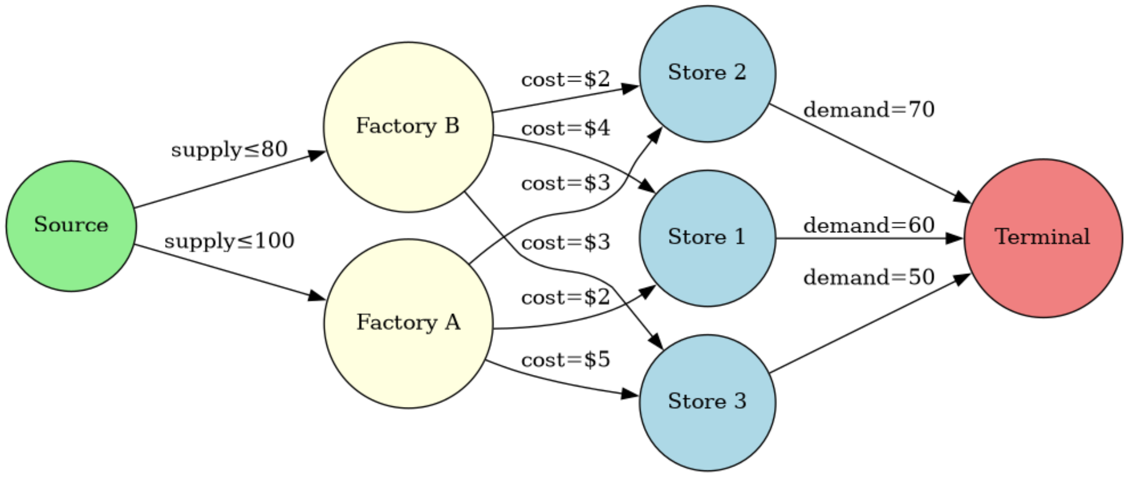
- **Source node**: Represents the initial supply of widgets (total: 180 units)
- **Factory nodes**: Factory A and Factory B (supply capacities: 100 and 80)
- **Store nodes**: Store 1, Store 2, Store 3 (demands: 60, 70, 50)
- **Terminal node**: Represents customer demand being met (total: 180 units)

  Note: While the source and terminal nodes don't represent physical locations, including them helps organize the problem structure and ensures supply equals demand.

The edges in our network represent possible shipping routes:

- From source to each factory (capacity = factory supply limit)
- From each factory to each store (capacity = unlimited, but cost varies)
- From each store to terminal (capacity = store demand)

All edges have associated costs. The source-to-factory and store-to-terminal edges have zero cost, while factory-to-store edges have costs given in the table above.



In your report: Include a clear network diagram showing all nodes and edges, with labels indicating capacities and costs where relevant.

4 Part (b): Linear Program Formulation

Let x_{ij} denote the number of widgets shipped from factory i to store j, where $i \in \{A, B\}$ and $j \in \{1, 2, 3\}$.

4.1 Objective Function

We seek to minimize the total shipping cost:

$$\text{minimize} \quad 2x_{A1} + 3x_{A2} + 5x_{A3} + 4x_{B1} + 2x_{B2} + 3x_{B3}$$

4.2 Constraints

Supply constraints: Each factory cannot ship more than its available supply:

$$x_{A1} + x_{A2} + x_{A3} \leq 100 \quad \text{(Factory A capacity)}$$
$$x_{B1} + x_{B2} + x_{B3} \leq 80 \quad \text{(Factory B capacity)}$$

Demand constraints: Each store must receive exactly its required demand:

$$x_{A1} + x_{B1} = 60 \quad \text{(Store 1 demand)}$$
$$x_{A2} + x_{B2} = 70 \quad \text{(Store 2 demand)}$$
$$x_{A3} + x_{B3} = 50 \quad \text{(Store 3 demand)}$$

Non-negativity: All shipping quantities must be non-negative:

$$x_{ij} \geq 0 \quad \text{for all } i, j$$

5 Part (c): Implementation and Results

We implemented this linear program using Python's `scipy.optimize.linprog` function. The complete code is provided in the Code Appendix below.

Running the optimization code (see Code Appendix), we obtain an optimal total shipping cost of **$450**.

The optimal shipping plan is as follows:

Table 2: Optimal shipping quantities (widgets)

	Store 1	Store 2	Store 3
Factory A	60	40	0
Factory B	0	30	50

This solution makes intuitive sense: Factory A ships to Stores 1 and 2 where it has lower costs ($2 and $3 respectively), while Factory B ships to Stores 2 and 3 where it is more cost-effective ($2 and $3 respectively). Store 2's demand of 70 units is split between both factories to take advantage of the low shipping cost from each.

6 Code Appendix

6.1 Network diagram generation

```
"""
```

```python
Network Flow Diagram for Toy Example
Visualize the transportation problem structure
"""

from graphviz import Digraph

# Create a new directed graph
dot = Digraph(comment='Toy Example Network Flow')
dot.attr(rankdir='LR')  # Left to right layout

# Define node styles
dot.attr('node', shape='circle', style='filled', fillcolor='lightblue')

# Add nodes
# Source and terminal with different color
dot.node('Source', 'Source', fillcolor='lightgreen')
dot.node('Terminal', 'Terminal', fillcolor='lightcoral')

# Factory nodes
dot.node('FA', 'Factory A', fillcolor='lightyellow')
dot.node('FB', 'Factory B', fillcolor='lightyellow')

# Store nodes
dot.node('S1', 'Store 1', fillcolor='lightblue')
dot.node('S2', 'Store 2', fillcolor='lightblue')
dot.node('S3', 'Store 3', fillcolor='lightblue')

# Add edges from source to factories (with supply capacities)
dot.edge('Source', 'FA', label='supply≤100')
dot.edge('Source', 'FB', label='supply≤80')

# Add edges from factories to stores (with shipping costs)
dot.edge('FA', 'S1', label='cost=$2')
dot.edge('FA', 'S2', label='cost=$3')
dot.edge('FA', 'S3', label='cost=$5')
dot.edge('FB', 'S1', label='cost=$4')
dot.edge('FB', 'S2', label='cost=$2')
dot.edge('FB', 'S3', label='cost=$3')

# Add edges from stores to terminal (with demands)
dot.edge('S1', 'Terminal', label='demand=60')
dot.edge('S2', 'Terminal', label='demand=70')
dot.edge('S3', 'Terminal', label='demand=50')

# Save the diagram
filename = "MidRep1--2025-02-16--toy-network-diagram"
dot.render(filename, format='png', cleanup=True)
print(f"Network diagram saved on disk as\n '{filename}.png'")
```

```
Network diagram saved on disk as
 'MidRep1--2025-02-16--toy-network-diagram.png'
```

6.2 Linear Program implementation

```python
"""
Toy Example: Simple Transportation Problem
Optimal widget shipping from 2 factories to 3 stores
"""

import numpy as np
from scipy.optimize import linprog

# Define the problem data
# Shipping costs (dollars per widget): rows = factories, cols = stores
costs = np.array([
    [2, 3, 5],  # Factory A to Stores 1, 2, 3
    [4, 2, 3]   # Factory B to Stores 1, 2, 3
])

# Flatten cost matrix for objective function
#   (order: A1, A2, A3, B1, B2, B3)
c = costs.flatten()

# Supply capacities
supply = np.array([100, 80])  # Factory A, Factory B

# Store demands
demand = np.array([60, 70, 50])  # Store 1, Store 2, Store 3

# Build constraint matrices
# Supply constraints (inequality): sum over stores <= supply
A_supply = np.array([
    [1, 1, 1, 0, 0, 0],  # Factory A
    [0, 0, 0, 1, 1, 1]   # Factory B
])
b_supply = supply

# Demand constraints (equality): sum over factories = demand
A_demand = np.array([
    [1, 0, 0, 1, 0, 0],  # Store 1
    [0, 1, 0, 0, 1, 0],  # Store 2
    [0, 0, 1, 0, 0, 1]   # Store 3
])
b_demand = demand
```

```python
# Solve the linear program
result = linprog(
    c=c,
    A_ub=A_supply,
    b_ub=b_supply,
    A_eq=A_demand,
    b_eq=b_demand,
    method='highs',
    bounds=(0, None)   # Non-negativity constraints
)
# Display results
if result.success:
    print("Optimal shipping cost: $%.2f" % result.fun)
    print("\nOptimal shipping plan:")
    print("Factory A -> Store 1: %.1f widgets" % result.x[0])
    print("Factory A -> Store 2: %.1f widgets" % result.x[1])
    print("Factory A -> Store 3: %.1f widgets" % result.x[2])
    print("Factory B -> Store 1: %.1f widgets" % result.x[3])
    print("Factory B -> Store 2: %.1f widgets" % result.x[4])
    print("Factory B -> Store 3: %.1f widgets" % result.x[5])

    # Reshape for display as matrix
    shipping_matrix = result.x.reshape(2, 3)
    print("\nShipping matrix:")
    print(shipping_matrix)
else:
    print("Optimization failed:", result.message)
```

```
Optimal shipping cost: $450.00

Optimal shipping plan:
Factory A -> Store 1: 60.0 widgets
Factory A -> Store 2: 40.0 widgets
Factory A -> Store 3: 0.0 widgets
Factory B -> Store 1: 0.0 widgets
Factory B -> Store 2: 30.0 widgets
Factory B -> Store 3: 50.0 widgets

Shipping matrix:
[[60. 40.  0.]
 [ 0. 30. 50.]]
```

Note: In your actual submission, your code should be organized with clear sections corresponding to each part of the problem (parts a-f). Use comments to explain your approach and any key decisions.