Assignment 9.3 – The Problem

This assignment presents a management problem from Professor Rob Freund's textbook ***Data, Models, and Decisions: The Fundamentals of Management Science*** that you need to solve using a linear optimization model.

To access the problem, please download the following document.

Module 9 - Formulating a Management Problem as a Linear Optimization Model.PDF

Read the problem, and please answer the following questions:

1. How much coal should NBS contract from each supplier to minimize the cost of supplying coking coal to NBS?

2. What is NBS's total supply cost?

3. What is NBS's average supply cost?

Assignment 9.3 – The Solution

1. How much coal should NBS contract from each supplier to minimize the cost of supplying coking coal to NBS?

Problem Summary

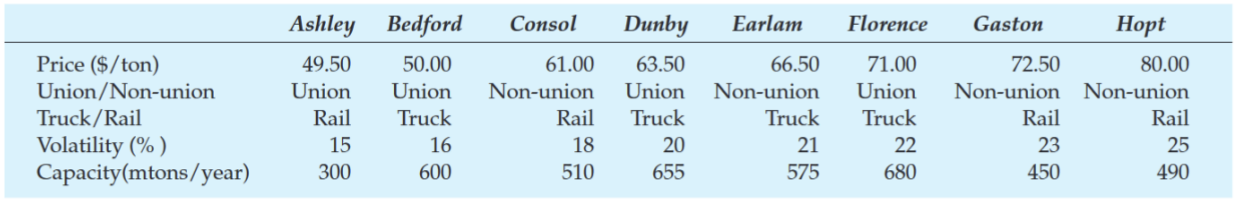
NBS needs to purchase **1,225 mtons** (1 mton equates to 1,000 tons) of coking coal from 8 suppliers. The goal is to:

1. **Minimize total cost of supply.**

2. **Ensure**

   - At least **50% of coal** comes from **union mines.**
   - **Average volatility ≥ 19% (**the volatility of the coal is the percent of volatile (burnable) matter in the coal)
   - Delivery by **rail ≤ 650 mtons**
   - Delivery by **truck ≤ 720 mtons**

3. **Relevant information:**



	Ashley	*Bedford*	*Consol*	*Dunby*	*Earlam*	*Florence*	*Gaston*	*Hopt*
Price ($/ton)	49.50	50.00	61.00	63.50	66.50	71.00	72.50	80.00
Union/Non-union	Union	Union	Non-union	Union	Non-union	Union	Non-union	Non-union
Truck/Rail	Rail	Truck	Rail	Truck	Truck	Truck	Rail	Rail
Volatility (%)	15	16	18	20	21	22	23	25
Capacity(mtons/year)	300	600	510	655	575	680	450	490

4. **Decision Variables:**

   Assuming x_i = amount of coal (in mtons) purchased from supplier $_i$ where $_i$ = 1, 2, …, 8

5. **Objective Function**

   - Minimize:

     Total Cost $= \sum_{i=1}^{8} c_i \,.\, x_i$

     Where c_i is the cost per ton from supplier *i*

6. **Constraints**

- **Total Quantity** constraint: 1225 mtons, acquired from different vendors.
- **Volatility constraint** (weighted average volatility ≥ 19%):

  (Sum (v_i * x_i)) / 1225 >= 0.19

  Which can be simplified to:

  (Sum (v_i * x_i)) >= 232.75
- **Union labor constraint** (at least 50% from union mines):

  Sum (x_i) >= 612.5
- **Transport Limits**

  Train Limit <= 650

  Truck Limit <= 720
- **Capacity constraint** for each supplier**:**

  0 <= x_i <= capacity_i

7. **The computation:**

Using linear programming via scipy.optimize.linprog in Python, which computes the values of xi that minimize total cost while satisfying all constraints.

This is partially reused from my previous Bootcamp course at Rutgers University.

```
# Linear Programming for Coal Purchasing Optimization

import numpy  as np  # type: ignore
import pandas as pd # type: ignore
from scipy.optimize import linprog # type: ignore
# Supplier data
# Suppliers: Ashley, Bedford, Consol, Dunby, Earlam, Florence, Gaston, Hopt
# Prices: Cost per ton
# Union mines: 1 = Yes, 0 = No
# Transport modes: Rail or Truck
# Volatility: Percent volatility of each supplier
# Capacity: Max tons each supplier can deliver
# Data
suppliers      = ["Ashley",
               "Bedford",
               "Consol",
               "Dunby",
               "Earlam",
               "Florence",
               "Gaston",
               "Hopt"]
price          = [49.5, 50.0,
               61.0, 63.5,
               66.5, 71.0,
               72.5, 80.0]  # Cost per ton
is_union       = [1, 1, 0, 1, 0, 1, 0, 0]                # Union mine: 1 = Yes, 0 = No
transport_mode = ["Rail", "Truck",
               "Rail", "Truck",
               "Truck", "Truck",
               "Rail", "Rail"]
volatility     = [15, 16, 18, 20, 21, 22, 23, 25]              # Volatility percent
```

```
capacity      = [300, 600, 510, 655, 575, 680, 450, 490]        # Max tons each supplier can deliver
#
# Objective function coefficients (minimize cost)
# Coefficients for the objective function (cost per ton)
# We want to minimize the total cost of purchasing coal
#
c = price
# Constraints
# Equality constraint: total coal must equal 1225 mtons
# We need to purchase exactly 1225 mtons of coal
# Equality constraints (A_eq x = b_eq)
# Coefficients for the equality constraint
A_eq = [[1] * 8]
b_eq = [1225]
# Inequality constraints
# Inequality constraints (A_ub x <= b_ub)
A_ub = []
b_ub = []
#
# Volatility constraint: weighted volatility >= 19% => -volatility*x <= -232.75
# We want the weighted volatility to be less than or equal to 19%
# This is calculated as the sum of (volatility * purchased amount) / total purchased amount
# The total purchased amount is 1225 mtons, so we need to ensure that
# sum(volatility * purchased amount) <= 19% * 1225
# Rearranging gives us the inequality:
# sum(volatility * purchased amount) <= 232.75
# We can express this as:
#
A_ub.append([-v for v in volatility])
b_ub.append(-232.75)
#
# Union constraint: at least 50% from union mines => -union*x <= -612.5
# We want at least 50% of the total purchased amount to come from union mines
#
A_ub.append([-u for u in is_union])
b_ub.append(-612.5)
#
```

```
# Rail capacity constraint: x_rail <= 650
# We need to ensure that the amount purchased by rail does not exceed 650 mtons
#
A_ub.append([1 if m == "Rail" else 0 for m in transport_mode])
b_ub.append(650)
#
# Truck capacity constraint: x_truck <= 720
# We need to ensure that the amount purchased by truck does not exceed 720 mtons
#
A_ub.append([1 if m == "Truck" else 0 for m in transport_mode])
b_ub.append(720)
#
# Bounds for each supplier (0 <= x_i <= capacity_i)
# We need to ensure that the amount purchased from each supplier
# is between 0 and their capacity
#
bounds = [(0, cap) for cap in capacity]
#
# Solve using linear programming
#
print("Running optimization...")
result = linprog(c=c, A_ub=A_ub,
                 b_ub=b_ub,
                 A_eq=A_eq,
                 b_eq=b_eq,
                 bounds=bounds,
                 method='highs')
#
# Process results
#
if result.success:
    purchased = result.x
    total_cost = np.dot(purchased, price)
    average_cost = total_cost / 1225
    df = pd.DataFrame({
        "Supplier": suppliers,
        "Price": price,
        "Purchased (mtons)": purchased,
        "Cost": [p * x for p, x in zip(price, purchased)]
    })
    print("\nOptimal Purchase Plan:")
    print(df.round(2).to_string(index=False))
    print(f"\nTotal Cost: ${total_cost:,.2f}")
    print(f"Average Cost per Ton: ${average_cost:.2f}")
else:
    print("Optimization failed:", result.message)
```

Running the Python Script, we have the answers for questions 1, 2, and 3
The table below shows how much coal is needed by each supplier.

```
Running optimization...

Optimal Purchase Plan:
Supplier  Price  Purchased (mtons)    Cost
  Ashley   49.5              300.0 14850.0
 Bedford   50.0              600.0 30000.0
  Consol   61.0              325.0 19825.0
   Dunby   63.5                0.0     0.0
  Earlam   66.5                0.0     0.0
Florence   71.0                0.0     0.0
  Gaston   72.5                0.0     0.0
    Hopt   80.0                0.0     0.0

Total Cost: $64,675.00
Average Cost per Ton: $52.80
```

2. **What is NBS's total supply cost?**

```
Total Cost: $64,675.00
```

3. **What is NBS's average supply cost?**

```
Average Cost per Ton: $52.
```