DDA5002_HW1

by Xiaocao_225040374

Problem 1 - Solution

- (a)

According to the given information, we could conclude the following table.

info type	the 1st product	the 2nd product	limitation
assembly labor	1/4	1/3	90
testing hours	1/8	1/3	80
worth of raw material	$1.2	$0.9	-
market value	$9	$8	-
profit	$7.8	$7.1	-

Denote the yield of the first product and the second product as x and y separately.

This problem could be formulated as the following linear program.

$$\begin{aligned} \max_{x,\,y} \quad & 7.8x + 7.1y \\ \text{s.t.} \quad & \frac{1}{4}x + \frac{1}{3}y \le 90 \\ & \frac{1}{8}x + \frac{1}{3}y \le 80 \\ & x, y \ge 0 \end{aligned}$$

- (b)

(i) can be easily incorporated into the linear program formulation.

Based on the formulation in (a), denote the hours of overtime assembly labor as z. The adding hours releases the limitation of assembly labor from 90 to $90 + z$. However, the overtime assembly labor costs $7 per hour, which should be subtracted from the objective function.

Then the new linear program is as follows.

$$
\begin{aligned}
\max_{x,\,y,\,z} \quad & 7.8x + 7.1y - 7z \\
\text{s.t.} \quad & \frac{1}{4}x + \frac{1}{3}y \le 90 + z \\
& \frac{1}{8}x + \frac{1}{3}y \le 80 \\
& x,\, y \ge 0 \\
& 0 \le z \le 50
\end{aligned}
$$

Problem 2 - Solution

- (a)

Denote distance as y and power consumption as x. The residual could be denoted as $e_i = y_i - (mx_i + b)$, where $i \in \mathbb{I}$. The elements in set $\mathbb{I}$ are the indexes of the data points.

- **Indices and set**

  $i \in \mathbb{I} = \{1, 2, \ldots, n\}$,n is the number of data points.

- **Parameters**

  x_i: power consumption of data point i

  y_i: distance of data point i

- **Decision Variables**

  e_i: the residual of prediction

  m: the slope of the model

  b: the intercept of the model

The absolute value of the residual could be transformed into a maximum function:

$$|e_i| = \max\{y_i - (mx_i + b), -(y_i - (mx_i + b))\}$$

Then the problem could be formulated as the following linear program.

$$
\begin{aligned}
\min_{m,b,e} \quad & \sum_{i \in \mathbb{I}} e_i \\
\text{s.t.} \quad & e_i \ge y_i - (mx_i + b), \quad \forall i \in \mathbb{I} \\
& e_i \ge -(y_i - (mx_i + b)), \quad \forall i \in \mathbb{I} \\
& m, b \ge 0
\end{aligned}
$$

- (b) The code is as follows.

In [1]:
```python
import coptpy as cp
from coptpy import COPT

env = cp.Envr()
linear_model_p2 = env.createModel("HW1_problem2")
# Disable logging
linear_model_p2.setParam(COPT.Param.Logging, 0)

# Load data
x = [10, 8, 13, 15, 9]
y = [60, 55, 75, 80, 64]
n = len(x)

# Define variables
e = [linear_model_p2.addVar(name=f"e_{i}", lb=0) for i in range(n)]
m = linear_model_p2.addVar(name="m", lb=0)
b = linear_model_p2.addVar(name="b", lb=0)

# Add constraints
for i in range(n):
    linear_model_p2.addConstr(e[i] >= y[i] - (m * x[i] + b), name=f
    linear_model_p2.addConstr(e[i] >= -(y[i] - (m * x[i] + b)), nam

# Set objective
linear_model_p2.setObjective(sum(e), sense=cp.COPT.MINIMIZE)

# Solve the model
linear_model_p2.solve()

# Output results
if linear_model_p2.status == COPT.OPTIMAL:
    print(f"Optimal value: {linear_model_p2.objval}")
    print(f"Optimal m: {m.x}")
    print(f"Optimal b: {b.x}")
```

```
Cardinal Optimizer v7.2.11. Build date Aug  1 2025
Copyright Cardinal Optimizer 2025. All Rights Reserved

Optimal value: 9.714285714285715
Optimal m: 3.571428571428571
Optimal b: 26.428571428571434
```

Explanation:

Given the data points, we are optimizing the parameters m and b of the linear model $y = mx + b$ to minimize the sum of absolute residuals between the observed distances and the distances predicted by the model.

According to the output of the code, the optimal value of the objective function is 9.71, which means adjusting the parameters m and b could reduce the total absolute error to 9.71 as much as possible.

The optimal parameters are $m = 3.57$ and $b = 26.43$. That is, the two values minimize the error of approximating the distance based on power consumption using a linear model. So they are the best fit parameters for the given data points using linear programming.

Problem 3 - Solution

Definition of the variables, parameters and sets:

- **indices and sets**

  $i \in \mathbb{I} = \{1, 2\}$ denote the period 1 and 2.

  $j \in \mathbb{J} = \{1, 2, 3\}$ denote the sales outlets I, II and III.

  $k \in \mathbb{K} = \{1, 2\}$ denote the manufacturing plants A and B.

- **parameters (constants)**

  c_{jk}: cost of shipping one unit of the product from the plant k to the sales outlet j.

  h: holding cost per unit of product at the factory at the end of each period.

  H: inventory capacity at the factory.

  P_{ik}: production capacity at the factory in each period.

  d_{ij}: maximum demand at sales outlet j in period i.

  s_{ij}: price per unit of the product at sales outlet j in period i.

  p_{ik}: cost per unit of producing at plant k in period i.

- **decision variables**

  x_{ijk}: amount shipped from plant k to sales outlet j in period i.

  y_{ik}: amount produced and stored at plant k in period $i-1$, preparing for sales in period i. So y_{1k} represents the inventory at the beginning of period 1 (produced at 0 period and for sales at 1 period), meaning no initial inventory.y_{3k} represents the inventory produced at period 2 and stored at period 2, but for sales at period 3. The two values are both 0 according to the problem description.

  z_{ik}: amount produced at plant k in period i.

The problem could be formulated as the following linear program.

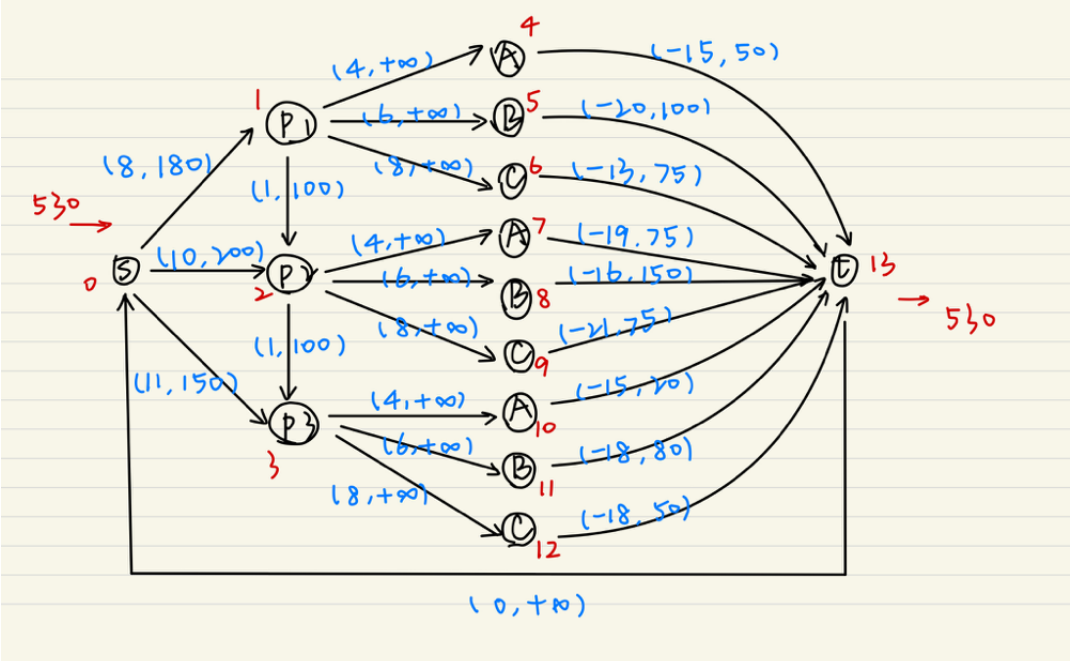
$$
\begin{aligned}
\max_{x,\,y} \quad & \sum_{i\in\mathbb{I},j\in\mathbb{J},k\in\mathbb{K}} s_{ij}x_{ijk} - \sum_{i\in\mathbb{I},j\in\mathbb{J},k\in\mathbb{K}} c_{jk}x_{ijk} - \sum_{i\in\mathbb{I},k\in\mathbb{K}} hy_{ik} - \sum_{i\in\mathbb{I},k\in\mathbb{K}} p_{ik}z_{ik} \\
\text{s.t.} \quad & \sum_{k\in\mathbb{K}} x_{ijk} \le d_{ij} \quad \forall i \in \mathbb{I}, j \in \mathbb{J} \\
& y_{ik} \le H \quad \forall i \in \mathbb{I}, k \in \mathbb{K} \\
& z_{ik} \le P_{ik} \quad \forall k \in \mathbb{K} \\
& x_{ijk}, y_{ik} \ge 0 \quad \forall i \in \mathbb{I}, k \in \mathbb{K} \\
& \sum_{j\in\mathbb{J}} x_{ijk} = z_{ik} + y_{ik} - y_{i+1,k} \quad \forall i \in \mathbb{I}, k \in \mathbb{K} \\
& y_{1k} = 0, \quad \forall k \in \mathbb{K} \\
& y_{3k} = 0, \quad \forall k \in \mathbb{K}
\end{aligned}
$$

Problem 4 - Solution

- (a)

The diagram is as follows.

On each arc, the annotations are in the form of (cost, upper bound).



- (b)

Definition of the variables, parameters and sets:

- **indices and sets**

  $i \in \mathbb{I} = \{0, 1, \ldots, 13\}$ denote all the nodes.

- **parameters (constants)**

  c_{ij}: cost of the arc which connects node i to node j.

  l_{ij}: lower bound of the amount for the arc which connects node i to node j.

  u_{ij}: upper bound of the amount for the arc which connects node i to node j.

- **decision variables**

  x_{ij}: amount shipped from node i to node j.

The problem could be formulated as the following linear program.

$$\begin{aligned} \min_{x,\,y} \quad & \sum_{i,j\in\mathbb{I}} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i\in\mathbb{I}} x_{ik} = \sum_{j\in\mathbb{I}} x_{kj} \quad \forall k \in \mathbb{I} \\ & l_{ij} \le x_{ij} \le u_{ij} \quad \forall i \in \mathbb{I}, j \in \mathbb{J} \end{aligned}$$

- (c) The code is as follows.

In [2]:
```
linear_model_p4 = env.createModel("HW1_problem4")
# Disable logging
linear_model_p4.setParam(COPT.Param.Logging, 0)

# Load data
c = [[8, 10, 11],   # cost of production
     [4, 6, 8],     # cost of shipping
     [4, 6, 8],
     [4, 6, 8],
     [-15, -20, -13], # price matrix at outlets
     [-19, -16, -21],
     [-15, -18, -18],
     [1, 1, 0],]    # inventory cost

H = 100 # inventory limit
P = [180, 200, 150] # capacity
d = [[50, 100, 75], # max demand matrix
     [75, 150, 75],
     [20, 80, 50],]

# Define variables
x = linear_model_p4.addVars(8, 3, nameprefix="x", lb=0)

# Add constraints on each node -- flow in = flow out

# S
linear_model_p4.addConstr(
```

```
    sum(x[0,j] for j in range(3)) == x[7, 2], name="0")

# P1, P2, P3
linear_model_p4.addConstr(
    x[0,0] == sum(x[1,j] for j in range(3)) + x[7, 0], name="1")
linear_model_p4.addConstr(
    x[0,1] + x[7,0] == sum(x[2,j] for j in range(3)) + x[7, 1], nam
linear_model_p4.addConstr(
    x[0,2] + x[7,1] == sum(x[3,j] for j in range(3)), name="3")

# A, B, C in 3 periods
linear_model_p4.addConstr(
    x[1, 0] == x[4, 0], name="A1")
linear_model_p4.addConstr(
    x[2, 0] == x[5, 0], name="A2")
linear_model_p4.addConstr(
    x[3, 0] == x[6, 0], name="A3")

linear_model_p4.addConstr(
    x[1, 1] == x[4, 1], name="B1")
linear_model_p4.addConstr(
    x[2, 1] == x[5, 1], name="B2")
linear_model_p4.addConstr(
    x[3, 1] == x[6, 1], name="B3")

linear_model_p4.addConstr(
    x[1, 2] == x[4, 2], name="C1")
linear_model_p4.addConstr(
    x[2, 2] == x[5, 2], name="C2")
linear_model_p4.addConstr(
    x[3, 2] == x[6, 2], name="C3")

# inventory flow
linear_model_p4.addConstr(
    sum(x[i,j] for i in range(4, 7) for j in range(3)) == x[7, 2],

# Constraints on value bounds
    # production capacity constraint
for j in range(3):
    linear_model_p4.addConstr(
         x[0, j] <= P[j],name=f"capacity_{j}")

    # A,B,C to t
for i in range(3):
     for j in range(3):
        linear_model_p4.addConstr(x[i+4, j] <= d[i][j], name=f"dema

    # inventory
for j in range(2):
     linear_model_p4.addConstr(x[7, j] <= H, name=f"inventory_{j}")

# Set objective
linear_model_p4.setObjective(
    sum(c[i][j] * x[i, j] for i in range(8) for j in range(3)),
    sense=cp.COPT.MINIMIZE
)
```

```
# Solve the model
linear_model_p4.solve()

# Output results
if linear_model_p4.status == COPT.OPTIMAL:
    print(f"Optimal value: {linear_model_p4.objval}")
    print("----------------------")
    print("** Decision Variables **")
    for i in range(8):
        print(x[i, 0].x, x[i, 1].x, x[i, 2].x)
    print("----------------------")
    print("** Period 1 **")
    print(f"A: {x[1,0].x}, \nB: {x[1,1].x}, \nC: {x[1,2].x}, \nstor
    print("** Period 2 **")
    print(f"A: {x[2,0].x}, \nB: {x[2,1].x}, \nC: {x[2,2].x}, \nstor
    print("** Period 3 **")
    print(f"A: {x[3,0].x}, \nB: {x[3,1].x}, \nC: {x[3,2].x}, \nstor
```

```
Optimal value: -1460.0
----------------------
** Decision Variables **
180.0 120.0 80.0
50.0 100.0 0.0
75.0 0.0 75.0
0.0 80.0 0.0
50.0 100.0 0.0
75.0 0.0 75.0
0.0 80.0 0.0
30.0 0.0 380.0
----------------------
** Period 1 **
A: 50.0,
B: 100.0,
C: 0.0,
store: 30.0
** Period 2 **
A: 75.0,
B: 0.0,
C: 75.0,
store: 0.0
** Period 3 **
A: 0.0,
B: 80.0,
C: 0.0,
store: 0
```

According to the output of the code, the optimal value is -1460.0, which means the minimum cost that could be achieved is -$1460.0 among all the alternatives.

The detailed production and shipping plan is shown on the following table.

period	A	B	C	production	inventory
1	50	100	0	180	30

2	75	0	75	150		0
3	0	80	0	80		0

Explicitly, in period 1, the production is 180 units, shipping 50 units to outlet A and 100 units to outlet B, and storing 30 units for the next period.

In period 2, the production is 150 units, shipping 75 units to outlet A and 75 units to outlet C, and no inventory for the next period.

In period 3, the production is 80 units, shipping 80 units to outlet B, and no inventory left.

Problem 5 - Solution

Given the decision variables, define the index sets as follows.

- $\mathbb{T} = \{0, 1, \ldots, 23\}$

The objective function (i.e. the total cost of daily shift schedule) is the product of shift cost and number of clerks shifting at that time.

$$\min_{x} \sum_{t \in \mathbb{T}} c_t x_t$$

At time i, the number of clerks at work is determined by numbers of the newly-come clerks, the clerks going for meal at i and the remaining clerks.

- **The newly-come clerks**

  The newly-come clerks could be easily defined by x_i.

- **The clerks going for meal**

  The clerks for meal could be defined by $\sum_{t=i-5}^{i-3} y_{t,i}$, for each clerk working only for 9 hours.

  To make sure it still at work during i^{th} hour and can catch up the last meal time, it could start work at $i-5$ as the earliest. Also, if it wants to have a meal at i, it has be at work no later than $i-3$.

- **The remaining clerks**

  The remaining clerks started work before i, so they could started work at $i-8$. If it goes to meal, it would be counted as the second situation. This number would be

$$\sum_{t=i-8}^{i-1} x_t$$

So we can derive the number of clerks working at k as follows.

$$\begin{aligned} & x_i - \sum_{t=i-5}^{i-3} y_{t,i} + \sum_{t=i-8}^{i-1} x_t \\ = & \sum_{t=i-8}^{i} x_t - \sum_{t=i-5}^{i-3} y_{t,i} \end{aligned}$$

The LP model could be formulated as followed.

$$\begin{aligned} \min_{x,\,y} \quad & \sum_{t\in\mathbb{T}} c_t x_t \\ \text{s.t.} \quad & \sum_{t=i-8}^{i} x_t - \sum_{t=i-5}^{i-3} y_{t,i} \geq r_i \quad \forall i \in \mathbb{T} \\ & y_{t,i} \leq x_t \quad \forall t \in \mathbb{T}, i = t+3, t+4, t+5 \\ & x_t, y_{t,i} \geq 0 \end{aligned}$$

Question about the model:

Though we can simply define the decision variables as zero when the index is negative, but actually the time is circular. For example, the time before 0 is 23, 22 and so on. But I don't know how to further improve the model by considering the circular time, and still not satisfied with the current definition of the indices.