DDA5002_HW3

by Xiaocao_225040374

Problem 1 - Solution

(a) The dual of the two models are as follows:

Model 1:

$$\begin{aligned}\max_{\pi} \quad & 19\pi_1 + 55\pi_2 + 7\pi_3 \\ \text{s.t.} \quad & \pi_1 + \pi_3 = 5 \\ & \pi_1 + 4\pi_2 + 6\pi_3 \leq 1 \\ & \pi_1 - \pi_3 \leq -4 \\ & \pi_1 + 8\pi_2 \geq 0 \\ & \pi_1 \geq 0, \pi_2 \leq 0, \pi_3 \text{ free}\end{aligned}$$

Model 2: The primal model can be rewritten as:

$$\begin{aligned}\max_{x} \quad & 2x_1 - 7x_2 + 6x_3 + 5x_4 \\ \text{s.t.} \quad & 2x_1 - 3x_2 - 5x_3 - 4x_4 \leq 20 \\ & 7x_1 + 2x_2 + 6x_3 - 2x_4 = 35 \\ & 4x_1 + 5x_2 - 3x_3 - 2x_4 \geq 15 \\ & x_1 \leq 10 \\ & x_2 \leq 5 \\ & x_3 \geq 2 \\ & x_1, x_2, x_3, x_4 \geq 0\end{aligned}$$

So the dual model is:

$$\begin{aligned}\min_{\pi} \quad & 20\pi_1 + 35\pi_2 + 15\pi_3 + 10\pi_4 + 5\pi_5 + 2\pi_6 \\ \text{s.t.} \quad & 2\pi_1 + 7\pi_2 + 4\pi_3 + \pi_4 \geq 2 \\ & -3\pi_1 + 2\pi_2 + 5\pi_3 + \pi_5 \geq -7 \\ & -5\pi_1 + 6\pi_2 - 3\pi_3 + \pi_6 \geq 6 \\ & -4\pi_1 - 2\pi_2 - 2\pi_3 \geq 5 \\ & \pi_1 \geq 0, \pi_2 \text{ free}, \pi_3 \leq 0, \pi_4 \geq 0, \pi_5 \geq 0, \pi_6 \leq 0\end{aligned}$$

(b) The primal feasibility and complementary slackness conditions are as follows:

Model 1:

1. Primal feasibility:

$$
\begin{aligned}
&x_1 + x_2 + x_3 + x_4 = 19 \\
&4x_2 + 8x_4 \le 55 \\
&x_1 + 6x_2 - x_3 = 7 \\
&x_2, x_3 \ge 0 \\
&x_4 \le 0
\end{aligned}
$$

2. Complementary slackness:

$$
\begin{aligned}
&x_1(5 - \pi_1 - \pi_3) = 0 \\
&x_2(1 - \pi_1 - 4\pi_2 - 6\pi_3) = 0 \\
&x_3(-4 - \pi_1 + \pi_3) = 0 \\
&x_4(-\pi_1 - 8\pi_2) = 0 \\
&\pi_1(x_1 + x_2 + x_3 + x_4 - 19) = 0 \\
&\pi_2(4x_2 + 8x_4 - 55) = 0 \\
&\pi_3(x_1 + 6x_2 - x_3 - 7) = 0
\end{aligned}
$$

Model 2:

1. Primal feasibility:

$$
\begin{aligned}
&2x_1 - 3x_2 - 5x_3 - 4x_4 \le 20 \\
&7x_1 + 2x_2 + 6x_3 - 2x_4 = 35 \\
&4x_1 + 5x_2 - 3x_3 - 2x_4 \ge 15 \\
&x_1 \le 10 \\
&x_2 \le 5 \\
&x_3 \ge 2 \\
&x_1, x_2, x_3, x_4 \ge 0
\end{aligned}
$$

2. Complementary slackness:

$$
\begin{aligned}
&x_1(2 - 2\pi_1 - 7\pi_2 - 4\pi_3 - \pi_4) = 0 \\
&x_2(-7 + 3\pi_1 - 2\pi_2 - 5\pi_3 - \pi_5) = 0 \\
&x_3(6 + 5\pi_1 - 6\pi_2 + 3\pi_3 - \pi_6) = 0 \\
&x_4(5 + 4\pi_1 + 2\pi_2 + 2\pi_3) = 0 \\
&\pi_1(2x_1 - 3x_2 - 5x_3 - 4x_4 - 20) = 0 \\
&\pi_2(7x_1 + 2x_2 + 6x_3 - 2x_4 - 35) = 0 \\
&\pi_3(4x_1 + 5x_2 - 3x_3 - 2x_4 - 15) = 0 \\
&\pi_4(x_1 - 10) = 0 \\
&\pi_5(x_2 - 5) = 0 \\
&\pi_6(x_3 - 2) = 0
\end{aligned}
$$

Problem 2 - Solution

(a) The definition of sets and variables are as follows:

- **Indices and Sets**:

  $i \in I = \{1, 2, 3\}$: set of different seat classes, representing First Class, Business Class, and Coach-fare Class respectively.

  $j \in J = \{1, 2, 3\}$: set of scenarios, representing (i), (ii), and (iii) respectively.

- **Parameters**:

  p_i: profit per tickets for class i, where $p_1 = 3$, $p_2 = 2$, and $p_3 = 1$.

  c_i: consumption of coach-fare seat spaces for class i, where $c_1 = 2$, $c_2 = 1.5$, and $c_3 = 1$.

  d_{ij}: demand for class i tickets in scenario j.

  s: total available coach-fare seat spaces, where $s = 200$.

  n: total number of scenarios, where $n = 3$.

- **Decision Variables**:

  - x_i: number of seats for class i.
  - $y_{i,j}$: number of class i seats sold in scenario j.

Then the problem can be formulated as the following linear programming model:

$$
\begin{aligned}
\max_{x,y} \quad & \frac{1}{n} \sum_{i \in I, j \in J} p_i y_{i,j} \\
\text{s.t.} \quad & \sum_{i \in I} c_i x_i \leq s \\
& y_{i,j} \leq x_i, \quad \forall i \in I, j \in J \\
& y_{i,j} \leq d_{ij}, \quad \forall i \in I, j \in J \\
& x_i \geq 0, \quad \forall i \in I \\
& y_{i,j} \geq 0, \quad \forall i \in I, j \in J
\end{aligned}
$$

(b) Solve the problem using Python with COPT as follows:

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

env = cp.Envr()
linear_model_p2 = env.createModel("HW3_problem2")
# Disable logging
linear_model_p2.setParam(COPT.Param.Logging, 0)
```

```python

# load data
p = [3, 2, 1]
c = [2, 1.5, 1]
d = [
    [20, 10,  5],   # First
    [50, 25, 10],   # Business
    [200,175,150],  # Coach
]
s = 200
m = 3 # number of seats types
n = 3 # number of scenarios

# Define variables
x = linear_model_p2.addVars(m, nameprefix="x", lb=0)
y = linear_model_p2.addVars(m, n, nameprefix="y", lb=0)

# Add constraints
linear_model_p2.addConstr(sum(x[i] * c[i] for i in range(m)) <= s,
name="space_constraint")

for j in range(n):
    linear_model_p2.addConstrs((y[i,j] <= x[i] for i in range(m)),
nameprefix=f"seats_constraints_scenario_{j}")

linear_model_p2.addConstrs((y[i, j] <= d[i][j] for j in range(n)
for i in range(m)), nameprefix="demand_constraints")

# Define objective function
linear_model_p2.setObjective(1/n * sum(p[i] * sum(y[i, j] for j in
range(n)) for i in range(m)), sense=cp.COPT.MAXIMIZE)

# Solve the model
linear_model_p2.solve()

if linear_model_p2.status == COPT.OPTIMAL:
    print(f"Optimal value: {linear_model_p2.objval}")
    print("Optimal seats partition:")
    for i in range(m):
        print(f"x[{i}] = {x[i].x}")
    print("Optimal ticket sales in each scenario:")
    for j in range(n):
        print(f"Scenario {j+1}:")
        for i in range(m):
            print(f"y[{i},{j}] = {y[i,j].x}")
```

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

Optimal value: 208.33333333333331
Optimal seats partition:
x[0] = 10.0
x[1] = 20.0
x[2] = 150.0
Optimal ticket sales in each scenario:
Scenario 1:
y[0,0] = 10.0
y[1,0] = 20.0
y[2,0] = 150.0
Scenario 2:
y[0,1] = 10.0
y[1,1] = 20.0
y[2,1] = 150.0
Scenario 3:
y[0,2] = 5.0
y[1,2] = 10.0
y[2,2] = 150.0
```

```
Optimal value: 208.33333333333331
Optimal seats partition:
x[0] = 10.0
x[1] = 20.0
x[2] = 150.0
Optimal ticket sales in each scenario:
Scenario 1:
y[0,0] = 10.0
y[1,0] = 20.0
y[2,0] = 150.0
Scenario 2:
y[0,1] = 10.0
y[1,1] = 20.0
y[2,1] = 150.0
Scenario 3:
y[0,2] = 5.0
y[1,2] = 10.0
y[2,2] = 150.0
```

According to the optimization results, the optimal number of seats for each class are:

- First Class: 10 seats, taking 20 coach-fare seat spaces.
- Business Class: 20 seats, taking 30 coach-fare seat spaces.
- Coach-fare Class: 150 seats, taking 150 coach-fare seat spaces.

Due to the assumption that each scenario occurs with equal probability, so we divide the total expected profit by 3 to get the final expected profit. And based on the ticket price of coach-fare class being 1, business class being 2, and first class being 3, we calculate the total expected profit.

Using the optimization results, the total expected profit is 208.33.

(c) The shadow price of the maximum available coach-fare seat spaces b=200 is 0.8889.

In [2]:
```
round(linear_model_p2.getDuals()[0], 4)
```

Out[2]:
```
0.8889
```

(d)

$$\Delta z^* = \pi_i^* \Delta b = 0.8889 \Delta b = 0.8889$$

So the predicted z^* would be $208.33 + 0.8889 = 209.2189$ if the available coach-fare seat spaces increase by 1 unit to 201.

(e) Resolve the problem as follows:

In [3]:
```
linear_model_p2e = env.createModel("HW3_problem2")
# Disable logging
linear_model_p2e.setParam(COPT.Param.Logging, 0)

s = 201

# Define variables
x = linear_model_p2e.addVars(m, nameprefix="x", lb=0)
y = linear_model_p2e.addVars(m, n, nameprefix="y", lb=0)

# Add constraints
linear_model_p2e.addConstr(sum(x[i] * c[i] for i in range(m)) <=
s, name="space_constraint")

for j in range(n):
    linear_model_p2e.addConstrs((y[i,j] <= x[i] for i in
range(m)), nameprefix=f"seats_constraints_scenario_{j}")

linear_model_p2e.addConstrs((y[i, j] <= d[i][j] for j in range(n)
for i in range(m)), nameprefix="demand_constraints")

# Define objective function
linear_model_p2e.setObjective(1/n * sum(p[i] * sum(y[i, j] for j
in range(n)) for i in range(m)), sense=cp.COPT.MAXIMIZE)

# Solve the model
linear_model_p2e.solve()

if linear_model_p2e.status == COPT.OPTIMAL:
    print(f"Optimal value: {linear_model_p2e.objval}")
```

```
Optimal value: 209.2222222222222
```

According to the output result, the optimal value is 209.22, which is consistent with the prediction in part (d).

(f)

```
In [4]: for i in linear_model_p2.getDuals()[10:]:
            print(round(i, 4))
```

```
-0.0
-0.0
-0.0
0.2222
-0.0
-0.0
1.0
0.6667
0.1111
```

From the top to bottom, the output dual varibles correspond to the shadow prices of:

1. The first scenario's demand constraints for First Class, Business Class, and Coach-fare Class.
2. The second scenario's demand constraints for First Class, Business Class, and Coach-fare Class.
3. The third scenario's demand constraints for First Class, Business Class, and Coach-fare Class.

Take the first scenario and the first class demand constraint as an example. In the optimal solution, the number of tickets sold for First Class in the first scenario reaches its demand limit of 10. According to the complementary slackness condition, the corresponding dual variable represents the corresponding slackness variable, so it has to be zero to reach optimality.

Problem 3 - Solution

(a) The dual problem is as follows:

$$

$$
\begin{aligned}
\min_{\pi} \quad & 12\pi_1 + 10\pi_2 + 10\pi_3 \\
\text{s.t.} \quad & 2\pi_1 + \pi_2 + 3\pi_3 \geq 4 \\
& 3\pi_1 + 4\pi_2 + \pi_3 \geq 2 \\
& \pi_1 + 2\pi_2 + \pi_3 \geq 3 \\
& \pi_1, \pi_2, \pi_3 \geq 0
\end{aligned}
$$

(b) The complementary slackness conditions are as follows:

$$\begin{aligned}
&x_1(4-2\pi_1-\pi_2-3\pi_3)=0\\
&x_2(2-3\pi_1-4\pi_2-\pi_3)=0\\
&x_3(3-\pi_1-2\pi_2-\pi_3)=0\\
&\pi_1(2x_1+3x_2+x_3-12)=0\\
&\pi_2(x_1+4x_2+2x_3-10)=0\\
&\pi_3(3x_1+x_2+x_3-10)=0
\end{aligned}$$

Given that the optimal solution of the primal problem is $x_1=2$, $x_2=0$, and $x_3=4$, we can deduce that the corresponding dual variables must satisfy:

$$\begin{aligned}
&4-2\pi_1-\pi_2-3\pi_3=0\\
&3-\pi_1-2\pi_2-\pi_3=0\\
&\pi_1=0
\end{aligned}$$

Solve the above equations, we get $\pi_2=1$ and $\pi_3=1$.

So the optimal solution of the dual problem is $(\pi_1^*,\pi_2^*,\pi_3^*)=(0,1,1)$.

(c)

The optimal value of the dual problem is $12\pi_1^*+10\pi_2^*+10\pi_3^*=20$.

The optimal value of the primal problem is $4x_1+2x_2+3x_3=20$.

So the strong duality holds.

Problem 4 - Solution

(a) The definition of sets and variables are as follows:

- **Indices and Sets**:

  $i\in I=1,2,\dots,5$: set of starting hole locations.

  $j\in J$: set of ending hole locations.

- **Parameters**:

  d_{ij}: distance from starting hole location i to ending hole location j.

- **Decision Variables**:

  - x_{ij}: binary variable indicating whether to dig a hole from location i to location j (1 if yes, 0 if no).

Then the problem can be formulated as the following integer programming model:

$$\begin{aligned}
\min_{x} \quad & 13x_{12} + 8x_{13} + 15x_{14} + 7x_{15} + \\
& 13x_{21} + 15x_{23} + 7x_{24} + 14x_{25} + \\
& 8x_{31} + 5x_{32} + 15x_{34} + 7x_{35} + \\
& 15x_{41} + 7x_{42} + 15x_{43} + 8x_{45} + \\
& 7x_{51} + 14x_{52} + 7x_{53} + 8x_{54} \\
\text{s.t.} \quad & \sum_{j \in J \setminus \{i\}} x_{ij} = 1, \quad \forall i \in I \\
& \sum_{i \in I \setminus \{j\}} x_{ij} = 1, \quad \forall j \in J \\
& x_{ij} \in \{0, 1\}, \quad \forall i \in I, j \in J
\end{aligned}$$

(b) Solve the problem using Python with COPT as follows:

In [5]:
```
linear_model_p4 = env.createModel("HW3_problem4")
# Disable logging
linear_model_p4.setParam(COPT.Param.Logging, 0)

# Load data
D = [[0, 13, 8, 15, 7],
[13, 0, 5, 7, 14],
[8, 5, 0, 15, 7],
[15, 7, 15, 0, 8],
[7, 14, 7, 8, 0]]

num_holes = 5

# Define variables
x = linear_model_p4.addVars(num_holes, num_holes, nameprefix="x",
vtype=cp.COPT.BINARY)

# Add constraints
linear_model_p4.addConstrs((sum(x[i,j] for j in range(num_holes)
if j != i) == 1 for i in range(num_holes)),
nameprefix="forward_constraints")
linear_model_p4.addConstrs((sum(x[i,j] for i in range(num_holes)
if i != j) == 1 for j in range(num_holes)),
nameprefix="backward_constraints")

# Define objective function
linear_model_p4.setObjective(sum(D[i][j] * x[i,j] for i in
range(num_holes) for j in range(num_holes) if i != j),
sense=cp.COPT.MINIMIZE)

# Solve the model
linear_model_p4.solve()

if linear_model_p4.status == COPT.OPTIMAL:
    print(f"Optimal value: {linear_model_p4.objval}")
    print("Optimal assignment:")
    for i in range(num_holes):
        for j in range(num_holes):
            if i != j and x[i,j].x > 0.5:
                print(f"x_{{{i+1},{j+1}}} = 1")
```

```
Optimal value: 35.0
Optimal assignment:
x_{1,3} = 1
x_{2,4} = 1
x_{3,2} = 1
x_{4,5} = 1
x_{5,1} = 1
```

According to the optimization results, the optimal digging plan is:

Dig from location 1 --> 3 --> 2 --> 4 --> 5 --> 1.

The minimum total digging distance is 35 millimeters.

Problem 5 - Solution

(a) The definition of sets and variables are as follows:

- **Indices and Sets**:

  $i \in I = \{1, 2, \ldots, 7\}$: set of courses indexed by i.

  $j \in J = \{1, 2, 3\}$: set of requirements, representing Math, OR, and Computer respectively.

- **Parameters**:

  a_{ij}: binary parameter indicating whether course i satisfies requirement j (1 if yes, 0 if no).

- **Decision Variables**:

  x_i: binary variable indicating whether to offer course i (1 if yes, 0 if no).

Then the problem can be formulated as the following integer programming model:

$$
\begin{aligned}
\min_{x} \quad & \sum_{i \in I} x_i \\
\text{s.t.} \quad & \sum_{i \in I} a_{ij} x_i \geq 2, \quad \forall j \in J \\
& x_3 \leq x_6 \\
& x_4 \leq x_1 \\
& x_5 \leq x_6 \\
& x_7 \leq x_4 \\
& x_i \in \{0, 1\}, \quad \forall i \in I
\end{aligned}
$$

(b) Solve the problem using Python with COPT as follows:

```
In [6]: linear_model_p5 = env.createModel("HW3_problem5")
        # Disable logging
        linear_model_p5.setParam(COPT.Param.Logging, 0)

        # load data
        A = [[1, 0, 0],
             [1, 1, 0],
             [1, 0, 1],
             [1, 1, 0],
             [0, 1, 1],
             [0, 0, 1],
             [1, 1, 0]]
        r = [2, 2, 2]
        m = 7  # number of courses
        n = 3  # number of requirements

        # Define variables
        x = linear_model_p5.addVars(m, vtype=cp.COPT.BINARY,
        nameprefix="x")

        # Add constraints
        for j in range(n):
            linear_model_p5.addConstr(sum(A[i][j] * x[i] for i in
        range(m)) >= r[j], name=f"requirement_{j}_constraint")

        # The prerequisite courses constraints
        linear_model_p5.addConstr(x[2] <= x[5],
        name="prerequisite_course_3")
        linear_model_p5.addConstr(x[3] <= x[0],
        name="prerequisite_course_4")
        linear_model_p5.addConstr(x[4] <= x[5],
        name="prerequisite_course_5")
        linear_model_p5.addConstr(x[6] <= x[3],
        name="prerequisite_course_7")

        # Define objective function
        linear_model_p5.setObjective(sum(x[i] for i in range(m)),
        sense=cp.COPT.MINIMIZE)

        # Solve the model
        linear_model_p5.solve()

        if linear_model_p5.status == COPT.OPTIMAL:
             print(f"Optimal value: {linear_model_p5.objval}")
             print("Selected courses:")
             for i in range(m):
                  if x[i].x > 0.5:
                       print(f"Course {i+1}")
```

```
Optimal value: 4.0
Selected courses:
Course 2
Course 3
Course 5
Course 6
```

According to the optimization results, the minimum number of courses to be offered is 4.

That is, one can take courses 2, 3, 5, 6 to fulfill all the requirements and take the least number of courses. In the meantime, the prerequisite constraints are also satisfied.