MAT3007 Assignment 1

117010008 曹正家

Question 1

> A company produces two kinds of products. A product of the first type requires 1/6 hours of assembly labor, 1/2 hours of testing, and $1.2 worth of raw materials. A product of the second type requires 1/4 hours of assembly, 1/6 hours of testing, and $0.9 worth of raw materials. Given the current personnel of the company, there can be at most 90 hours of assembly labor and 80 hours of testing each day. Products of the first and second type have a market value of $9 and $8 respectively.
>
> (a) Formulate a linear optimization that maximizes the daily profit of the company. (For simplicity, you do not need consider the completeness of products. That is, the variables may take float values.)
>
> (b) Write the standard form of the LP you formulated in part (a)
>
> (c) Consider the following modification to the original problem: Suppose that up to 40 hours of overtime assembly labor can be scheduled, at a cost of $8 per hour. Can it be easily incorporated into the linear optimization formulation and how?
>
> (d) Solve the LP using MATLAB (for the original problem).

1(a)

In this question, we have the following table:

	Assembly Hours	Testing hours	Raw material	Market Value
Product A	1/6	1/2	$1.2	9
Product B	1/4	1/6	$0.9	8

And we know that the total hours of assembly can be at most 90 hours and the total number of testing can at most be 80 hours.

The decision variables: the number of product A: X_a, the number of product B: X_b

Objective: Maximize the profit given by: $(9-1.2)*X_a+(8-0.9)*X_b=7.8X_a+7.1X_b$.

Constraints:

- Total assembly labour hours: $\frac{1}{6}*X_a+\frac{1}{4}*X_b\leq 90$
- Total testing labour hours: $\frac{1}{2}*X_a+\frac{1}{6}*X_b\leq 80$
- $X_a\geq 0, X_b\geq 0$

1(b)

To write this in standard LP form:

$$\begin{aligned} minimize_{x_a,x_b} \; & -(7.8X_a + 7.1X_b) \\ subject\ to \; & \frac{1}{6}X_a + \frac{1}{4}X_b + s = 90 \\ & \frac{1}{2}X_a + \frac{1}{6}X_b + t = 80 \\ & X_a, X_b, s, t \geq 0 \end{aligned}$$

where s and t are slack variables

1(c)

We can add a new decision variable X_c to denote the amount of extra hours used after the 90 hours of assembly labor. We have our cost
$X_c = ReLU(\frac{1}{6}X_a + \frac{1}{4}X_b - 90), where\ ReLU(x) = max(0, x)$

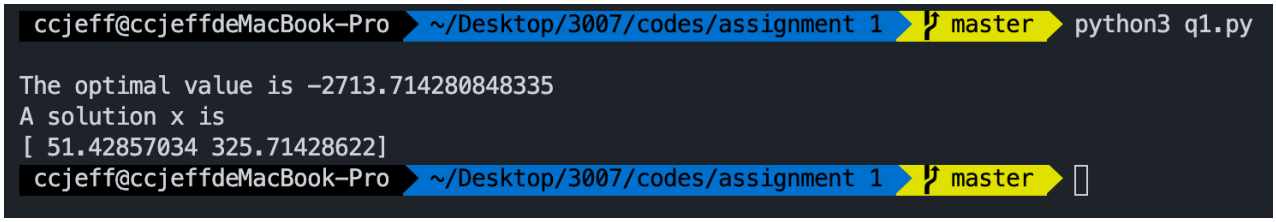
$$\begin{aligned} minimize_{x_a,x_b} \; & -(7.8X_a + 7.1X_b - 8X_c) \\ subject\ to \; & \frac{1}{6}X_a + \frac{1}{4}X_b + s = 130 \\ & \frac{1}{2}X_a + \frac{1}{6}X_b + t = 80 \\ & X_a, X_b, X_c, s, t \geq 0 \end{aligned}$$

1(d)

Here attached is the Python code:

```
import cvxpy as cp
import numpy as np

# minimize c^Tx
# having Ax <= b
c = np.array([-7.8, -7.1])
A = np.array([[1/6, 1/4], [1/2, 1/6]])
b = [90,80]

x = cp.Variable(2)

prob = cp.Problem(cp.Minimize(c.T@x),
                  [A @ x <= b,
                  x >= 0])
prob.solve()

# Print result.
print("\nThe optimal value is", prob.value)
print("A solution x is")
print(x.value)
```

Console screenshot:



Optimal value at -2713.7, which means the largest profit is 2713.7, with the solution 51.4 and 325.7, since both X_a and X_b have the integer constraints, we obtain $X_a = 51$, $X_b = 326$, and the optimal value is 2712.4

Question 2

> Consider a school district with I neighborhoods, J schools, and G grades at each school. Each school j has a capacity of C_{jg} for grade g. In each neighborhood i, the student population of grade g is S_{ig}. Finally, the distance of school j from neighborhood i is d_{ij} . Formulate a linear optimization problem whose objective is to assign all students to schools, while minimizing the total distance traveled by all students.

The **decision variables** for this question: the amount of student who lives in neighbourhood i, goes to school j and is in grade g, denoted by S_{ijg}

objective: to minimize the total distance traveled by all students

constraints:

sum of students of all grades equals to total amount of students, having $\sum_{j \in J} S_{ijg} = S_{ig}$

sum of students of all grades less than or equals to school capacity, having $\sum_{i \in I} S_{ijg} \leq C_{ig}$

we should also have poitive number of students, $S_{ijg} \geq 0$

this can be turned to standard form linear optimization problem:

$$
\begin{aligned}
minimize_{S_{ijg}} \sum_{i \in I} \sum_{j \in J} \sum_{g \in G} S_{ijg} \times d_{ij} \\
subject\ to \sum_{j \in J} S_{ijg} = S_{ig}, \forall i, g \\
\sum_{i \in I} S_{ijg} \leq C_{ig}, \forall j, g \\
S_{ijg} \geq 0, \forall i, j, g
\end{aligned}
$$

Question 3

> The China Railroad Ministry is in the process of planning relocations of freight cars among 5 regions of the country to get ready for the fall harvest. Table 1 shows the cost of moving a car between each pair of regions. Table 2 shows the current number of cars in each region and the number needed for harvest shipping.

From/To	1	2	3	4	5
1	-	10	12	17	34
2	10	-	18	8	46
3	12	18	-	9	27
4	17	8	9	-	20
5	34	46	27	20	-

Table 1: Costs of moving a car

	1	2	3	4	5
Present	130	385	400	480	610
Need	200	400	600	200	300

Table 2: Number of current and needed cars

> Write down a linear optimization to compute the least costly way to move the cars such us the need is met. Solve the problem using MATLAB.

We have X_{ij} denoting the number of cars moving from i to j, and Y_{ij} denotes the cost of moving one car from i to j

we can form our linear optimization problem as:

$$
\begin{aligned}
minimize_{X_{ij},Y_{ij}} \sum_{j=1}^{5}\sum_{i=1}^{5} X_{ij} * Y_{ij} \\
subject\ to \sum_{i=1}^{5} X_{i1} - \sum_{j=1}^{5} X_{1j} \geq 200 - 130 \\
\sum_{i=1}^{5} X_{i2} - \sum_{j=1}^{5} X_{2j} \geq 400 - 385 \\
\sum_{i=1}^{5} X_{i3} - \sum_{j=1}^{5} X_{3j} \geq 600 - 400 \\
\sum_{i=1}^{5} X_{i4} - \sum_{j=1}^{5} X_{4j} \geq 200 - 480 \\
\sum_{i=1}^{5} X_{i5} - \sum_{j=1}^{5} X_{5j} \geq 300 - 610 \\
X_{ij} \geq 0, \forall i,j
\end{aligned}
$$

Based on the given cost, we form our matrix $Y = \{Yij\}$, as:

$$
Y = \begin{bmatrix}
999 & 10 & 12 & 17 & 34 \\
10 & 999 & 18 & 8 & 46 \\
12 & 18 & 999 & 9 & 27 \\
17 & 8 & 9 & 999 & 20 \\
34 & 46 & 27 & 20 & 999
\end{bmatrix}
$$

where the diagnals are set to 999 to prevent the optimizer go to the point itself.

The python code shown:

```
import cvxpy as cp
import numpy as np

x = cp.Variable((5,5),integer = True)
y = np.array([[999,10,12,17,34],
                [10,999,18,8,46],
                [12,18,999,9,27],
                [17,8,9,999,20],
                [34,46,27,20,999]])


prob = cp.Problem(cp.Minimize(cp.sum(cp.sum(cp.multiply(y,x),axis = 1), axis =
0)),
                  [
                     -cp.sum(x,axis = 1)[0] + cp.sum(x,axis = 0)[0] >= 200 -
130,
                     -cp.sum(x,axis = 1)[1] + cp.sum(x,axis = 0)[1] >= 400 -
385,
                     -cp.sum(x,axis = 1)[2] + cp.sum(x,axis = 0)[2] >= 600 -
400,
                     -cp.sum(x,axis = 1)[3] + cp.sum(x,axis = 0)[3] >= 200 -
480,
                     -cp.sum(x,axis = 1)[4] + cp.sum(x,axis = 0)[4] >= 300 -
610,
                     cp.diag(x) == 0,
                     cp.min(x) >= 0
                  ])
prob.solve()

# Print result.
print("\nThe optimal value is", prob.value)
print("A solution x is")
print(x.value)
```

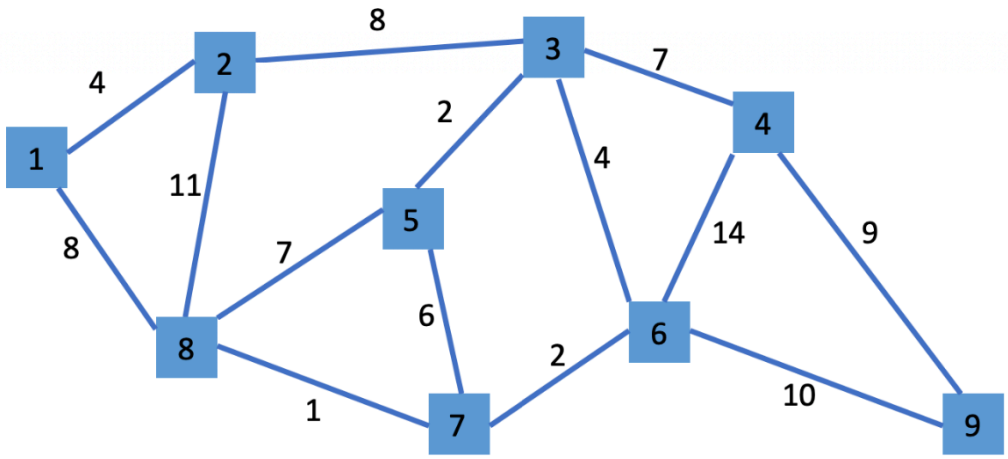
```
ccjeff@ccjeffdeMacBook-Pro ~/Desktop/3007/codes master ● python3 assignment\ 1/q2.py
Long-step dual simplex will be used

The optimal value is 3195.0
A solution x is
[[  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.]
 [ 65.  15. 200.   0.   0.]
 [  5.   0.   0.   0.   0.]]
```

The optimal value is 3195, and the corresponding $X_{41} = 65, X_{42} = 15, X_{43} = 200, X_{51} = 5$

This means we should move 65 cars from 4 to 1, 15 cars from 4 to 2, 200 cars from 4 to 3, and 5 cars from 5 to 1

Question 4

> Write a MATLAB code to use linear optimization to solve the shortest path problem. Suppose the input of the problem will be an n × n matrix of W , where wij is the length of the path from i to j (in general, wij does not necessarily equal wji). In our implementation, we always use 1 to denote the source node (the s node in the lecture slides), and n to denote the terminal node (the t node in the lecture slides). In addition, we assume for any i and j, there is a path, i.e., the set of E is all pairs of nodes. This is without loss of generality since one can set wij to be an extremely large number if there is no edge between i and j, effectively eliminating it from consideration. Note that you are asked to solve the problem of the shortest path from node 1 to node 9.
>
> After writing the code, you are asked to solve the concrete problem in the lecture slides, with the given labeling shown in Figure 1. Basically, you need to input the W matrix for this case, then solve it, and then report your solution (the optimal path).




```
import cvxpy as cp
import numpy as np

y = np.array([
    [999,4,999,999,999,999,999,8,999],
    [4,999,8,999,999,999,999,11,999],
    [999,8,999,7,2,4,999,999,999],
    [999,999,7,999,999,14,999,999,9],
    [999,999,2,999,999,999,6,7,999],
    [999,999,4,14,999,999,2,999,10],
    [999,999,999,999,6,2,999,1,999],
    [8,11,999,999,7,999,1,999,999],
    [999,999,999,9,999,10,999,999,999]
    ])

x = cp.Variable((9,9),integer = True)
```

```
prob = cp.Problem(cp.Minimize(cp.sum(cp.sum(cp.multiply(y,x),axis = 1), axis =
0)),
                  [
                     cp.sum(x,axis = 0)[0] == 1,
                     cp.sum(x,axis = 1)[8] == 1,
                     cp.sum(x,axis = 0)[1] - cp.sum(x,axis = 1)[1] == 0,
                     cp.sum(x,axis = 0)[2] - cp.sum(x,axis = 1)[2] == 0,
                     cp.sum(x,axis = 0)[3] - cp.sum(x,axis = 1)[3] == 0,
                     cp.sum(x,axis = 0)[4] - cp.sum(x,axis = 1)[4] == 0,
                     cp.sum(x,axis = 0)[5] - cp.sum(x,axis = 1)[5] == 0,
                     cp.sum(x,axis = 0)[6] - cp.sum(x,axis = 1)[6] == 0,
                     cp.sum(x,axis = 0)[7] - cp.sum(x,axis = 1)[7] == 0,
                     cp.min(x) >= 0,
                     cp.max(x) <= 1
                  ])
prob.solve()

# Print result.
print("\nThe optimal value is", prob.value)
print("A solution x is")
print(x.value)
```

The above code gives:

```
TypeError: sum() takes no keyword arguments
✘ ccjeff@ccjeffdeMacBook-Pro ~/Desktop/3007/codes  master ● python3 assignment\ 1/q3.py
Long-step dual simplex will be used

The optimal value is 21.0
A solution x is
[[0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0.]]
ccjeff@ccjeffdeMacBook-Pro ~/Desktop/3007/codes  master ●
```

And thus, we can get the shortest path going from node 1: $1 \rightarrow 8 \rightarrow 7 \rightarrow 6 \rightarrow 9$, while summing up all the weights along the path gives the optimal value 21

The 999 in the code denotes infinite weight.