UCS1403
Design and Analysis of Algorithms
Ex 9
Simplex Tableau Method
Sabarivasan V
205001085

Source Code

```
def Diff(li1, li2):
    li_dif = [i for i in li1 + li2 if i not in li1 or i not in li2]
    return li_dif

def simplex(mat):
    res = list()
    ratios = []

    rows = len(mat)
    cols = len(mat[0])

    temp = [-1] * (rows-1)

    print("\nReceived matrix is: ")
    for i in range(rows):
        for j in range(cols):
            print(f"{mat[i][j]}",end = '\t')
        print("\n")

    min_col = 0

    for j in range(cols):
        if(mat[rows-1][min_col]>mat[rows-1][j]):
            min_col = j

    if(mat[rows-1][min_col]>=0):
        for i in range(rows):
            res.append(mat[i][cols-1])

    else:
        while (any(x < 0 for x in mat[rows-1])) :
            ratios = []

            for j in range(cols):
```

```
            if(mat[rows-1][min_col]>mat[rows-1][j]):
                min_col = j

        for i in range(rows-1):
            ratios.append(mat[i][cols-1]/mat[i][min_col])

        min_row = ratios.index(min(ratios))
        pivot = mat[min_row][min_col]
        temp[min_row] = min_col

        for j in range(cols):
            mat[min_row][j]/=pivot #Making the pivot as 1

        #Making the values above and below pivot as 0

        for i in range(min_row):
            factor = mat[i][min_col]

            for j in range(cols):
                mat[i][j] -= factor * mat[min_row][j]

        for i in range(min_row + 1,rows):
            factor = mat[i][min_col]

            for j in range(cols):
                mat[i][j] -= factor * mat[min_row][j]

        print("\nIntermediate matrix")
        for i in range(rows):
            for j in range(cols):
                print(f"{mat[i][j]}",end = '\t')
            print("\n")



    '''for element in temp:
        res.append(element)

    if(len(temp)!=(cols-2)/2):
        for i in range((cols-2)//2 - len(temp)):
            res.append(0)'''

print(f"Optimal value = {mat[rows-1][cols-1]}")
```

```
    return temp

#Driver Code
var = []
found = []

n = int(input("\nEnter the no of parameters: "))
c = int(input("\nEnter the no of constraints: "))

cols = 2*n+2
rows = c+1

print("\nEnter the matrix: ")

mat = list()

for i in range(rows):
    mat.append([int(x) for x in input().split()])

res = simplex(mat)

for i in range(rows-1):
  var.append(chr(120+i))

print("Paramater values: ")
for i in range(len(res)):
    if res[i]!=-1:
        print(f"{chr(res[i] + 120)}={mat[i][-1]}",end=" ")
        found.append(chr(res[i] + 120))


print(f"{Diff(var,found)[0]}=0",end=" ")

print()
```

1. ABC telecom services offers internet services in two bandwidths, 20 Gbps and 10Gbps.

Electricity cost of the Routers is rounded off to Rs 10/hour for both the cases. a 400 (connections) port switch is used to offer 20 Gbps and 300 (connections) port switch is used to

offer 10 Gbps. A maximum speed of 160 Gbps can be offered and a maximum of 120 Rs/hour can be paid. Maximize the number of connections.

Answer

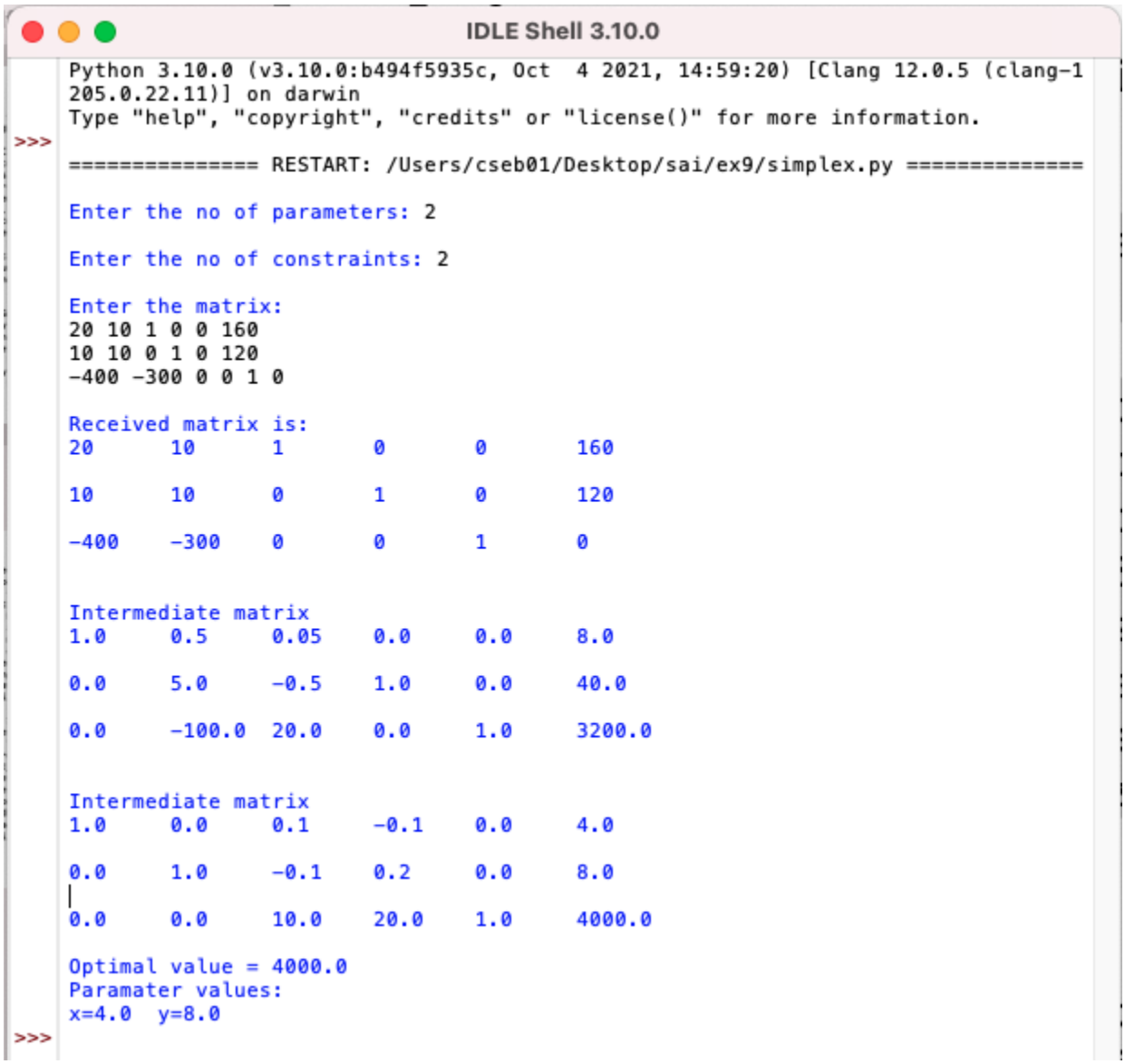
Constraints:

20x + 10y <= 160

10x + 10y <= 120

Optimisation function:

Z = 400x + 300y



2. A Green switch is defined with 3 energy saving policies. Policy 1 puts the Switch in sleep mode for 2 seconds per minute, queues 4 packets per second and saves 1 Re in the EB bill per hour. Policy 2 puts the Switch in sleep mode for 1 second per minute, queues 2 packets

per second and saves Rs 2 in the EB bill per hour. Policy 3 makes the switch to sleep for 1 second per minute, queues 3 packets per second and saves 1 Re per hour. Policy 1, Policy 2 and Policy 3 cause an additional delay of 2 ms, 5 ms and 5 ms for every tranmission of bursts. In an organization, a total of 14 seconds of sleep time, Maximum of 28 packets waiting in queue and 30 ms delay could be tolerated per hour. How does the organization maximize the energy savings?

Answer:

Constraints:

$2x + y + z <= 14$

$4x + 2y + 3z <= 28$

$2x + 5y + 5z <= 30$

Optimisation function:

$Z = x + 2y + z$

```
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:20) [Clang 12.0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.

=============== RESTART: /Users/cseb01/Desktop/sai/ex9/simplex.py ==============

Enter the no of parameters: 3

Enter the no of constraints: 3

Enter the matrix:
2 1 1 1 0 0 0 14
4 2 3 0 1 0 0 28
2 5 5 0 0 1 0 30
-1 -2 -1 0 0 0 1 0

Received matrix is:
2       1       1       1       0       0       0       14

4       2       3       0       1       0       0       28

2       5       5       0       0       1       0       30

-1      -2      -1      0       0       0       1       0


Intermediate matrix
1.6     0.0     0.0     1.0     0.0     -0.2    0.0     8.0

3.2     0.0     1.0     0.0     1.0     -0.4    0.0     16.0

0.4     1.0     1.0     0.0     0.0     0.2     0.0     6.0

-0.19999999999999996    0.0     1.0     0.0     0.0     0.4     1.0     12.0


Intermediate matrix
1.0     0.0     0.0     0.625   0.0     -0.125  0.0     5.0

0.0     0.0     1.0     -2.0    1.0     0.0     0.0     0.0

0.0     1.0     1.0     -0.25   0.0     0.25    0.0     4.0

0.0     0.0     1.0     0.12499999999999997     0.0     0.375   1.0     13.0

Optimal value = 13.0
Paramater values:
x=5.0 y=4.0 z=0
```

3. An SDN controller is defined with 2 schemes. 10 flows are installed in Scheme 1 and 7 flows are installed in scheme 2. On a request, 1 Packet_In message and 2 Packet_In messages are processed via scheme 1 and scheme 2 respectively. Scheme 1 takes a round trip time (RTT) of 4 seconds and Scheme 2 take an RTT of 3 seconds. A total RTT of 500 seconds and 200 Packet_In messages can be processed. maximize the the installation of flows.

Constraints:

$4x + 3y <= 500$

$X + 2y <= 200$

Optimisation function:

$Z = 10x + 7y$

IDLE Shell 3.10.0

```
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:20) [Clang 12.0.5 (clang-1
205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: /Users/cseb01/Desktop/sai/ex9/simplex.py ==========

Enter the no of parameters: 2

Enter the no of constraints: 2

Enter the matrix:
4 3 1 0 0 500
1 2 0 1 0 200
-10 -7 0 0 1 0

Received matrix is:
4       3       1       0       0       500

1       2       0       1       0       200

-10     -7      0       0       1       0


Intermediate matrix
1.0     0.75    0.25    0.0     0.0     125.0

0.0     1.25    -0.25   1.0     0.0     75.0

0.0     0.5     2.5     0.0     1.0     1250.0

Optimal value = 1250.0
Paramater values:
x=125.0  y=0
>>>
```