COMP 5413

Topics in Smart Health Informatics

Assignment 1

Name: Bhargavkumar Patel

Student Id: 1106925

**Problem 1:**

**Solution:**

**Number of doctors =6**

**Number of patients=6**

Each patient needs to be addressed by any one of its six available doctors. Because of the differences in symptoms of the patients as well as the expertise and experience of the doctors, the doctors require spending varying amounts of time on attending each patient.

Optimal Assignment for Patient 1 and 2 to doctor 4 and 5 would be:

There is a possibility:

1. There can only be one patient attended by the doctor at a time.
2. If patient1 visits doctor 4 , patient 2 is allotted to doctor5 else, if patient 2 is treated by doctor4 , the patient 1 has to be given to doctor 5 for treatment.

So if we see at the constraints:

d4p1 --> doctor4 and patient 1

For constraint to function properly , we have set an objective function that would set the limit to 1 patient per doctor.

Objective function:

This would result in checking all the possibilities for the patient -doctor combinations and would give the minimal time spent on particular patient.

The Optimal Solution Count comes out to be: 213

A screenshot of a social media post

Description automatically generated

Visual Representation:

A picture containing clock, sitting, large, water

Description automatically generated

This representation justifies the constraints .

**Problem 2:**

**Solution:**

In this problem, we have tried using random generation of values into the matrix.

And for this the numpy and random library has been used from python.

Considering the cost matrix formed by first input of values:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 115 | 97 | 105 | 133 | 69 | 93 |
| 114 | 95 | 117 | 53 | 86 | 70 |
| 96 | 62 | 92 | 83 | 134 | 139 |
| 132 | 54 | 110 | 50 | 127 | 77 |
| 62 | 67 | 69 | 131 | 55 | 87 |
| 144 | 113 | 122 | 81 | 88 | 133 |

**Following the Row-echlon method:**

We subtract the row minimum from each row of the matrix.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 46 | 28 | 36 | 64 | 0 | 24 | (-69) |
| 61 | 42 | 64 | 0 | 33 | 17 | (-53) |
| 34 | 0 | 30 | 21 | 72 | 77 | (-62) |
| 82 | 4 | 60 | 0 | 77 | 27 | (-50) |
| 7 | 12 | 14 | 76 | 0 | 32 | (-55) |
| 63 | 32 | 41 | 0 | 7 | 52 | (-81) |

Next step: Remove the column minima from each column from the matrix.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 | 28 | 22 | 64 | 0 | 7 |
| 54 | 42 | 50 | 0 | 33 | 0 |
| 27 | 0 | 16 | 21 | 72 | 60 |
| 75 | 4 | 46 | 0 | 77 | 10 |
| 0 | 12 | 0 | 76 | 0 | 15 |
| 56 | 32 | 27 | 0 | 7 | 35 |

**Cover all zeros with a minimum number of lines**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 39 | 28 | 22 | 64 | 0 | 7 | **X** |
| 54 | 42 | 50 | 0 | 33 | 0 | **X** |
| 27 | 0 | 16 | 21 | 72 | 60 | **X** |
| 75 | 4 | 46 | 0 | 77 | 10 |  |
| 0 | 12 | 0 | 76 | 0 | 15 | **X** |
| 56 | 32 | 27 | 0 | 7 | 35 |  |
|  |  |  | **X** |  |  |  |

Similarly ,

We create additional zeroes and cover all zeroes with a minimum number of lines until the optimal solution is gained.

Final cost matrix with the minimum cost (cost in terms of time consumed) is:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 115 | 97 | 105 | 133 | 69 | 93 |
| 114 | 95 | 117 | 53 | 86 | 70 |
| 96 | 62 | 92 | 83 | 134 | 139 |
| 132 | 54 | 110 | 50 | 127 | 77 |
| 62 | 67 | 69 | 131 | 55 | 87 |
| 144 | 113 | 122 | 81 | 88 | 133 |

The optimal solution count comes out to be : 428

A screenshot of a social media post

Description automatically generated

**Problem 3:**

**Solution:**

The optimal formulation made over problem 1 and 2 have been scripted through Gurobi using python.

**Gurobi script for Problem 1:**

**Appendix:**

#!/usr/bin/env python

# coding: utf-8

# In[2]:

from gurobipy import \*

# In[4]:

#INITIALIZING MODEL

m = Model("Problem3")

# In[5]:

#INITIALIZING VARIABLES

d4p1 = m.addVar(vtype="Binary", name="d4p1")

d4p2 = m.addVar(vtype="Binary", name="d4p2")

d5p1 = m.addVar(vtype="Binary", name="d5p1")

d5p2 = m.addVar(vtype="Binary", name="d5p2")

# In[6]:

#SETTING OBJECTIVE FUNCTION FOR MINIMIZATION

m.setObjective(130\*d4p1+ 95\*d4p2 + 118\*d5p1 + 83\*d5p2, GRB.MINIMIZE)

# In[7]:

#DEFINING CONSTRAINTS

m.addConstr(d4p1+d4p2== 1,"c1")

m.addConstr(d4p1+d5p1 == 1,"c2")

m.addConstr(d4p2+d5p2 == 1,"c3")

m.addConstr(d5p2+d5p1 == 1,"c4")

# In[8]:

m.optimize()

# In[9]:

for v in m.getVars():

print(v.varName,v.x)

**Appendix for problem 2 and its gurobi script:**

#!/usr/bin/env python

# coding: utf-8

# In[1]:

from gurobipy import \*

import numpy as np

import random

# In[2]:

D = [0,1,2,3,4,5]

P = [0,1,2,3,4,5]

# In[3]:

availableDataMapping = {}

for d in range(len(D)):

for p in range(len(P)):

availableDataMapping[d,p]=np.random.randint(50,150)

availableDataMapping

# In[4]:

combinations, values = multidict(availableDataMapping)

values

# In[5]:

m = Model("Assignment1 - Problem3")

# In[6]:

X = m.addVars(combinations,vtype=GRB.BINARY,name="Selected Match")

# In[7]:

PatientsAllocation = m.addConstrs((X.sum(p,'\*')==1 for p in P),'PatientsAllocation')

doctorsAllocation = m.addConstrs((X.sum('\*',d)==1 for d in D),'doctorsAllocation')

# PatientsAllocation = m.addConstrs((X.sum('\*',p)==1 for p in P),'PatientsAllocation')

# In[8]:

m.setObjective(X.prod(values),GRB.MINIMIZE)

# In[9]:

m.optimize()

# In[10]:

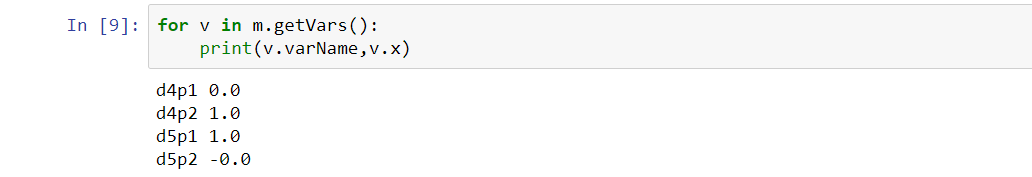
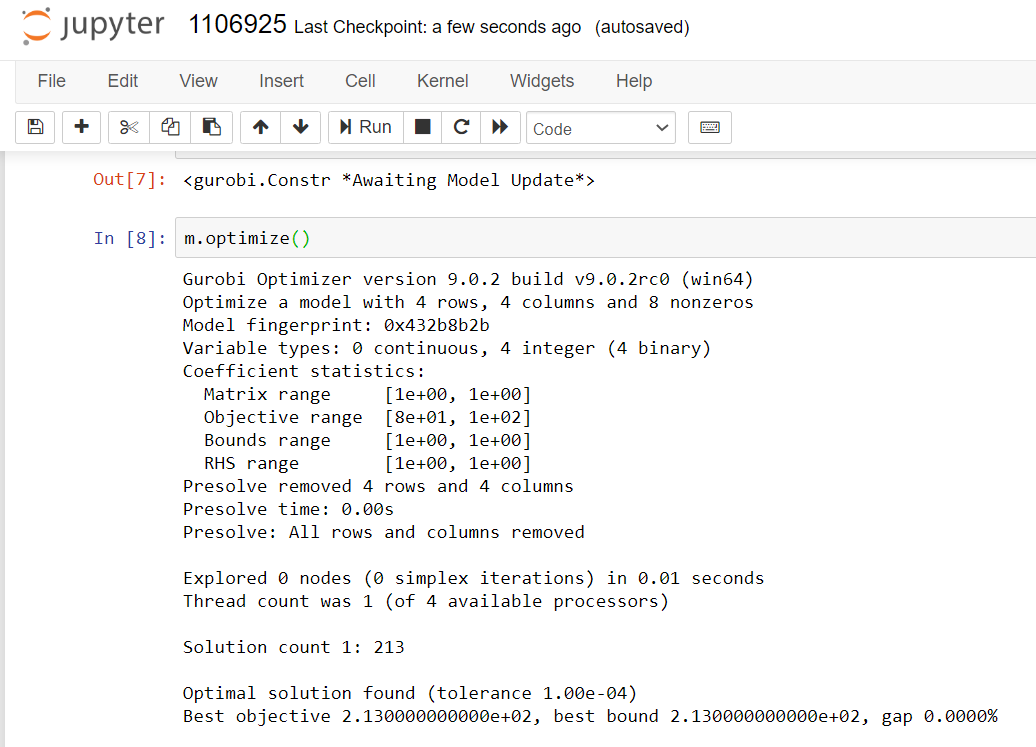
for v in m.getVars():

if(abs(v.x)>1e-6):

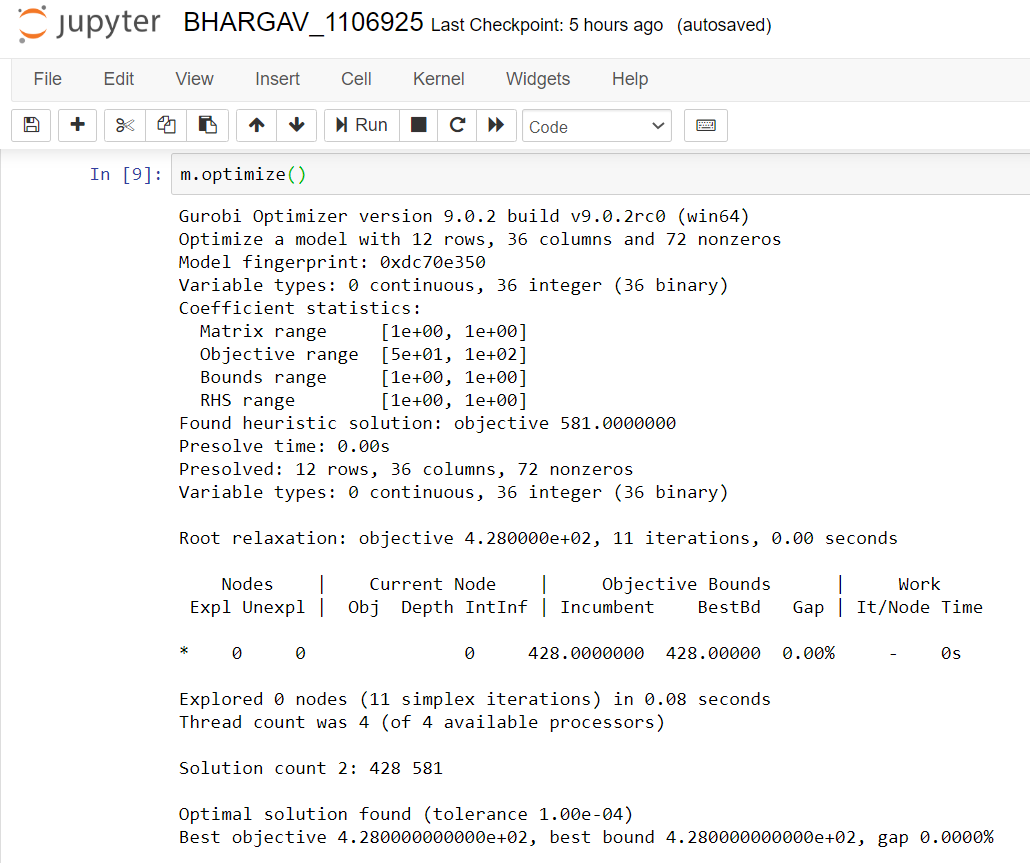
print(v.varName,v.x)

**Screenshot for the outputs:**

**Problem 1:**



**Problem 2:**



A screenshot of a social media post

Description automatically generated