assignment5_v4

December 5, 2022

[ ]:
```
#ORI391 Programming Assignment 5
#Matt Skiles ms82657
#Alexandar Mills adm5547

#!/usr/bin/env python3

from docplex.mp.model import Model
import numpy as np
import random
from scipy.sparse import rand
import pandas as pd
from scipy.linalg import lu_factor, lu_solve
import cplex
from random import randrange
import random

# TODO, use either cplex library or docplex library
# probably not both:
# here is API for cplex lib: https://courses.ie.bilkent.edu.tr/ie400/wp-content/
 ↪uploads/sites/8/2021/12/IBM-ILOG-CPLEX-PYTHON-API.pdf

# Assignment # 5. Using a commercial LP code; Due Dec 05, 2022
# Part 1.  Use CPLEX, XPRESS, Gurobi or CLP (COIN-OR) to solve the same set of␣
 ↪simultaneous
# equations that you solved in Assignment #1. CPLEX is on the ME Server.  You␣
 ↪will have to download
# CLP to your computer to use it.  On a Windows machine, sample CPLEX programs␣
 ↪in C, C++, java,
# Phython and perhaps other languages are available at

# C:\Program Files\IBM\ILOG\CPLEX_Studio1261\cplex\examples\src

# Part 2.  Also, solve a 10   20 LP.  Use your random matrix generator with the␣
 ↪same parameters from
# Assignment #1 to generate the LP.  The direction of the inequality for each␣
 ↪constraint should have a 0.7
```

```
# chance of being  and a 0.3 chance of being  (no equality constraints).  The␣
 ↪objective function is to be
# minimized and should have coefficients cj randomly distributed between -10␣
 ↪and +5.  You might want to
# add an upper bound on each variable to ensure that the problem has a finite␣
 ↪solution.  If you are having
# difficulty generating a feasible problem, you can construct one by selecting␣
 ↪nonnegative values for the
# decision variables (say,   ˆx j =1, for all j), and then fix the vector b so␣
 ↪that  Aˆx =b .

def A_matrix (U, L, density, m, n):
    #define a matrix of random values between 0 and 1 of specific density and␣
 ↪size
    matrix=rand(m,n,density)
    #interpolate between upper and lower bounds with randomly generated number
    matrix = (matrix.toarray()*(U-L))
    #convert array to dataframe
    matrix_df=pd.DataFrame(matrix)
    # cycle through rows and check if all values in row are zero

    for row in matrix_df.index:
        if (matrix_df.loc[row,:]==0).all():
            #if all values in row are zero, then recurse
            return A_matrix(U,L,density,m,n)
    #cycle through columns and check if all values in column are zero
    for col in matrix_df.columns:
        if (matrix_df.loc[:,col]==0).all():
            #if all values in column are zero, then recurse
            return A_matrix(U,L,density,m,n)

    return matrix_df




def b_matrix (U, L, density, m, n):
    #define a matrix of random values between 0 and 1 of specific density and␣
 ↪size
    matrix=rand(m,n,density)
    #interpolate between upper and lower bounds with randomly generated number
    matrix = matrix.toarray()*(U-L)+L
    #convert array to dataframe
    matrix_df=pd.DataFrame(matrix)
    #cycle through rows and check if all values in row are zero
```

```
    return matrix_df


##########################################
###############PART 1##################
##########################################
A=A_matrix(30,-10,0.6,10,10)
b=b_matrix(50,0,0.8,10,1)
model=cplex.Cplex()
objective= []
vars=[]
var_types=[]
constraint_names=[]
constraint_senses=[]

for col in A.columns:
    vars.append('x' + str(col))
    var_types.append('C')
    constraint_senses.append('E')
    objective.append(1)
constraints={}
for row in A.index:
    constraint_names.append('c' + str(row))
    constraints[str(row)]=[vars,list(A.loc[row,:])]
new_constraints=[]
for key in constraints:
    new_constraints.append(constraints[key])

variable_names = vars
variable_types = var_types
model.variables.add(obj=objective,
                   names= variable_names)
model.objective.set_sense(model.objective.sense.maximize)
rhs = list(b[0])
model.linear_constraints.add(lin_expr= new_constraints,
                             senses= constraint_senses,
                             rhs= rhs,
                             names= constraint_names)
model.solve()
print("Objective Function Value:",model.solution.get_objective_value())
print("Decision Variables Values:",model.solution.get_values())
##########################################
###############PART 2##################
##########################################
A=A_matrix(30,-10,0.6,10,20)
b=b_matrix(50,0,0.8,10,1)
model=cplex.Cplex()
```

```
objective= []
vars=[]
var_types=[]
constraint_names=[]
constraint_senses=[]

for col in A.columns:
    vars.append('x' + str(col))
    var_types.append('C')
    num=random.random()
    objective.append(randrange(-10,5))
    if num > 0.3:
        constraint_senses.append('G')
    else:
        constraint_senses.append('L')
constraints={}
for row in A.index:
    constraint_names.append('c' + str(row))
    constraints[str(row)]=[vars,list(A.loc[row,:])]
new_constraints=[]
for key in constraints:
    new_constraints.append(constraints[key])

variable_names = vars
variable_types = var_types
model.variables.add(obj=objective,
                   names= variable_names)
model.objective.set_sense(model.objective.sense.minimize)
rhs = list(b[0])
model.linear_constraints.add(lin_expr= new_constraints,
                               senses= constraint_senses,
                               rhs= rhs,
                               names= constraint_names)
model.solve()
print("Objective Function Value:",model.solution.get_objective_value())
print("Decision Variables Values:",model.solution.get_values())
```