assignment5_v2

December 5, 2022

[228]:

```
%reset
```

[229]:

```
#!/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())
```

```
Version identifier: 22.1.0.0 | 2022-03-27 | 54982fbec
CPXPARAM_Read_DataCheck                           1
Infeasible column 'x1'.
Presolve time = 0.00 sec. (0.00 ticks)

CPLEX Error  1217: No solution exists.
```

```
---------------------------------------------------------------------------
```

```
CplexSolverError                          Traceback (most recent call last)
/Users/Matt1/Library/CloudStorage/Box-Box/UT/Course_Materials/ORI_391Q/
 ↪programming_assignments/assignment5_v2.ipynb Cell 2 in <cell line: 105>()
    <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#W2sZmlsZQ%3D%3D?line=99'>100</a> model.linear_constraints.add(lin_expr=␣
 ↪new_constraints,
    <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#W2sZmlsZQ%3D%3D?line=100'>101</a>                                senses =␣
 ↪constraint_senses,
    <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#W2sZmlsZQ%3D%3D?line=101'>102</a>                                rhs=␣
 ↪rhs,
    <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#W2sZmlsZQ%3D%3D?line=102'>103</a>                                names=␣
 ↪constraint_names)
    <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#W2sZmlsZQ%3D%3D?line=103'>104</a> model.solve()
--> <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#W2sZmlsZQ%3D%3D?line=104'>105</a> print("Objective Function Value:
 ↪",model.solution.get_objective_value())
    <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#W2sZmlsZQ%3D%3D?line=105'>106</a> print("Decision Variables Values:
 ↪",model.solution.get_values())

File ~/opt/miniconda3/lib/python3.8/site-packages/cplex/_internal/_subinterfaces.
 ↪py:7211, in SolutionInterface.get_objective_value(self)
   7197 def get_objective_value(self):
   7198     """Returns the value of the objective function.
   7199
   7200     Example usage:
   (…)
   7209     -202.5
   7210     """
-> 7211     return CPX_PROC.getobjval(self._env._e, self._cplex._lp)

File ~/opt/miniconda3/lib/python3.8/site-packages/cplex/_internal/_procedural.py:
 ↪2055, in getobjval(env, lp)
   2053 objval = CR.doublePtr()
   2054 status = CR.CPXXgetobjval(env, lp, objval)
-> 2055 check_status(env, status)
   2056 return objval.value()

File ~/opt/miniconda3/lib/python3.8/site-packages/cplex/_internal/_procedural.py:
 ↪249, in StatusChecker.__call__(self, env, status, from_cb)
```

```
    247     else:
    248         error_string = geterrorstring(env, status)
--> 249 raise CplexSolverError(error_string, env, status)

CplexSolverError: CPLEX Error  1217: No solution exists.
```

[230]:

```
#########################################
###############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())
```

```
---------------------------------------------------------------------------
CplexError                                Traceback (most recent call last)
/Users/Matt1/Library/CloudStorage/Box-Box/UT/Course_Materials/ORI_391Q/
 ↪programming_assignments/assignment5_v2.ipynb Cell 3 in <cell line: 36>()
     <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#X30sZmlsZQ%3D%3D?line=33'>34</a> model.objective.set_sense(model.
 ↪objective.sense.minimize)
     <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#X30sZmlsZQ%3D%3D?line=34'>35</a> rhs = list(b[0])
---> <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#X30sZmlsZQ%3D%3D?line=35'>36</a> model.linear_constraints.add(lin_expr=␣
 ↪new_constraints,
     <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#X30sZmlsZQ%3D%3D?line=36'>37</a>                               senses=␣
 ↪constraint_senses,
     <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#X30sZmlsZQ%3D%3D?line=37'>38</a>                               rhs= rhs,
     <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#X30sZmlsZQ%3D%3D?line=38'>39</a>                               names=␣
 ↪constraint_names)
     <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#X30sZmlsZQ%3D%3D?line=39'>40</a> model.solve()
     <a href='vscode-notebook-cell:/Users/Matt1/Library/CloudStorage/Box-Box/UT/
 ↪Course_Materials/ORI_391Q/programming_assignments/assignment5_v2.
 ↪ipynb#X30sZmlsZQ%3D%3D?line=40'>41</a> print("Objective Function Value:",model.
 ↪solution.get_objective_value())

File ~/opt/miniconda3/lib/python3.8/site-packages/cplex/_internal/_subinterfaces.
 ↪py:1273, in LinearConstraintInterface.add(self, lin_expr, senses, rhs,␣
 ↪range_values, names)
   1217 """Adds linear constraints to the problem.
   1218
   1219 linear_constraints.add accepts the keyword arguments lin_expr,
   (…)
   1269 [0.0, 1.0, -1.0, 2.0]
   1270 """
   1271 lin_expr, senses, rhs, range_values, names = init_list_args(
   1272     lin_expr, senses, rhs, range_values, names)
-> 1273 return self._add_iter(self.get_num, self._add,
   1274                       lin_expr, senses, rhs, range_values, names)
```

```
File ~/opt/miniconda3/lib/python3.8/site-packages/cplex/_internal/_baseinterface.
 ↪py:41, in BaseInterface._add_iter(getnumfun, addfun, *args, **kwargs)
     39 """non-public"""
     40 old = getnumfun()
---> 41 addfun(*args, **kwargs)
     42 return range(old, getnumfun())

File ~/opt/miniconda3/lib/python3.8/site-packages/cplex/_internal/_subinterfaces.
 ↪py:1193, in LinearConstraintInterface._add(self, lin_expr, senses, rhs,␣
 ↪range_values, names)
   1191 arg_list = [rhs, senses, range_values, names, lin_expr]
   1192 num_new_rows = max_arg_length(arg_list)
-> 1193 validate_arg_lengths(
   1194     arg_list,
   1195     extra_msg=": lin_expr, senses, rhs, range_values, names"
   1196 )
   1197 num_old_rows = self.get_num()
   1198 if lin_expr:

File ~/opt/miniconda3/lib/python3.8/site-packages/cplex/_internal/_aux_functions.
 ↪py:99, in validate_arg_lengths(arg_list, allow_empty, extra_msg)
     97 for arg_length in arg_lengths:
     98     if arg_length != max_length:
---> 99         raise CplexError("inconsistent argument lengths" + extra_msg)

CplexError: inconsistent argument lengths: lin_expr, senses, rhs, range_values,␣
 ↪names
```

```
[ ]:
```