appendix1

December 5, 2022

1 Appendix 1 - Original Formulation of Optimization Problem

[ ]:

```
# import gurobi
from gurobipy import *
import numpy as np

# number of aircraft
planes = 1200

# number of days
days = 5

# number of airports
noairports = 3

# list of airports
airports = ['A', 'B', 'C']

# origin-destination pairs
odpairs = ['AB', 'AC', 'BA', 'BC', 'CA', 'CB']

# number of origin-destination pairs
pairs = 6

# initialize cargo amounts
cargo_amounts = np.array([
    [100, 200, 100, 400, 300],
    [50, 50, 50, 50, 50],
    [25, 25, 25, 25, 25],
    [25, 25, 25, 25, 25],
    [40, 40, 40, 40, 40],
    [400, 200, 300, 200, 400]
])

# holding costs
holdingcost = 10
```

```
# repositioning costs
ABcost = 7
BCcost = 6
ACcost = 3

# create new model
myModel = Model("Cargo_Operations")

# create decision variables for cargo supply
xVars = [[0 for i in range(days)] for j in range(pairs)]

for i in range(pairs):
    for j in range(days):
        curVar = myModel.addVar(vtype = GRB.INTEGER, name = "x" + odpairs[i] +␣
 ↪str(j+1))
        xVars[i][j] = curVar

# create decision variables for airplane shipments
yVars = [[0 for i in range(days)] for j in range(pairs)]

for i in range(pairs):
    for j in range(days):
        curVar = myModel.addVar(vtype = GRB.INTEGER, name = "y" + odpairs[i] +␣
 ↪str(j+1))
        yVars[i][j] = curVar

# create decision variables for airplane repositioning
zVars = [[0 for i in range(days)] for j in range(pairs)]

for i in range(pairs):
    for j in range(days):
        curVar = myModel.addVar(vtype = GRB.INTEGER, name = "z" + odpairs[i] +␣
 ↪str(j+1))
        zVars[i][j] = curVar

# create decision variables for airplanes that are grounded at the airport
sVars = [[0 for i in range(days)] for j in range(noairports)]

for i in range(noairports):
    for j in range(days):
        curVar = myModel.addVar(vtype = GRB.INTEGER, name = "s" + airports[i] +␣
 ↪str(j+1))
        sVars[i][j] = curVar

# integrate decision variables into the model
myModel.update()
```

```
# create a linear expression for the objective
objExpr = LinExpr()

# holding costs (number of cargo available minus number actually shipped)
for i in range(pairs):
    for j in range(days):
        curVar1 = xVars[i][j]
        curVar2 = yVars[i][j]
        objExpr += holdingcost * (curVar1 - curVar2)

# repositioning costs for A to B
for j in range(days):
    curVar = zVars[0][j]
    objExpr += ABcost * curVar

# repositioning costs for A to C
for j in range(days):
    curVar = zVars[1][j]
    objExpr += ACcost * curVar

# repositioning costs for B to A
for j in range(days):
    curVar = zVars[2][j]
    objExpr += ABcost * curVar

# repositioning costs for B to C
for j in range(days):
    curVar = zVars[3][j]
    objExpr += BCcost * curVar

# repositioning costs for C to A
for j in range(days):
    curVar = zVars[4][j]
    objExpr += ACcost * curVar

# repositioning costs for C to B
for j in range(days):
    curVar = zVars[5][j]
    objExpr += BCcost * curVar

myModel.setObjective(objExpr, GRB.MINIMIZE)

# create constraints for flow of cargo
for i in range(pairs):
    for j in range(days):
        constExpr = LinExpr()
        xVar1 = xVars[i][j]
        yVar1 = yVars[i][j]

        # last day should have zero left-over in cargo
        if j != 4:
            xVar2 = xVars[i][j+1]
            constExpr += xVar1 + cargo_amounts[i][j+1] - yVar1
            myModel.addConstr(lhs = constExpr, sense=GRB.EQUAL, rhs = xVar2,␣
 ↪name = "CargoSupplyConstraints")
        else:
            constExpr += xVar1 - yVar1
            myModel.addConstr(lhs = constExpr, sense=GRB.EQUAL, rhs = 0, name =␣
 ↪"CargoSupplyConstraints")

# create constraints to initialize initial demand of cargo
for i in range(pairs):
    constExpr = LinExpr()
    curVar = xVars[i][0]
    constExpr += curVar
    myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs =␣
 ↪cargo_amounts[i][0])

# create constraints to ensure shipment doesn't exceed available supply
for i in range(pairs):
    for j in range(days):
        constExpr = LinExpr()
        xVar1 = xVars[i][j]
        yVar1 = yVars[i][j]
        constExpr += yVar1
        myModel.addConstr(lhs = constExpr, sense=GRB.LESS_EQUAL, rhs =xVar1,␣
 ↪name = "ShipmentConstraints")

# create constraints for flow of airplanes for airport A
for i in range(days):
    constExpr = LinExpr()
    # last day should reset the number of airplanes for following week
    if i != 4:
        constExpr = sVars[0][i] + yVars[2][i] + yVars[4][i] + zVars[2][i] +␣
 ↪zVars[4][i] - sVars[0][i+1] - yVars[0][i+1] - yVars[1][i+1] - zVars[0][i+1]␣
 ↪- zVars[1][i+1]
    else:
        constExpr = sVars[0][i] + yVars[2][i] + yVars[4][i] + zVars[2][i] +␣
 ↪zVars[4][i] - sVars[0][0] - yVars[0][0] - yVars[1][0] - zVars[0][0] -␣
 ↪zVars[1][0]
    myModel.addConstr(lhs = constExpr, sense=GRB.EQUAL, rhs = 0, name =␣
 ↪"AirportAConstraints")
```

```python
# create constraints for flow of airplanes for airport B
for i in range(days):
    constExpr = LinExpr()
    # last day should reset the number of airplanes for following week
    if i != 4:
        constExpr = sVars[1][i] + yVars[0][i] + yVars[5][i] + zVars[0][i] + zVars[5][i] - sVars[1][i+1] - yVars[2][i+1] - yVars[3][i+1] - zVars[2][i+1] - zVars[3][i+1]
    else:
        constExpr = sVars[1][i] + yVars[0][i] + yVars[5][i] + zVars[0][i] + zVars[5][i] - sVars[1][0] - yVars[2][0] - yVars[3][0] - zVars[2][0] - zVars[3][0]
    myModel.addConstr(lhs = constExpr, sense=GRB.EQUAL, rhs = 0, name = "AirportBConstraints")

# create constraints for flow of airplanes for airport C
for i in range(days):
    constExpr = LinExpr()
    # last day should reset the number of airplanes for following week
    if i != 4:
        constExpr = sVars[2][i] + yVars[1][i] + yVars[3][i] + zVars[1][i] + zVars[3][i] - sVars[2][i+1] - yVars[4][i+1] - yVars[5][i+1] - zVars[4][i+1] - zVars[5][i+1]
    else:
        constExpr = sVars[2][i] + yVars[1][i] + yVars[3][i] + zVars[1][i] + zVars[3][i] - sVars[2][0] - yVars[4][0] - yVars[5][0] - zVars[4][0] - zVars[5][0]
    myModel.addConstr(lhs = constExpr, sense=GRB.EQUAL, rhs = 0, name = "AirportCConstraints")

# create constraint to bound the total number of planes
constExpr = LinExpr()
for i in range(pairs):
    constExpr += yVars[i][0]
    constExpr += zVars[i][0]
for i in range(noairports):
    constExpr += sVars[i][0]
myModel.addConstr(lhs = constExpr, sense=GRB.EQUAL, rhs = planes, name = "TotalPlaneConstraints")

# integrate objective and constraints into the model
myModel.update()

# write the model in a file to make sure it is constructed correctly
myModel.write(filename = "CargoProject.lp")
```

```python
# optimize the model
myModel.optimize()
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2023-10-08
Warning: linear constraint 0 and linear constraint 1 have the same name
"CargoSupplyConstraints"
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 82 rows, 105 columns and 315 nonzeros
Model fingerprint: 0xbfb441be
Variable types: 0 continuous, 105 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [3e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+01, 1e+03]
Presolve removed 48 rows and 30 columns
Presolve time: 0.00s
Presolved: 34 rows, 75 columns, 225 nonzeros
Variable types: 0 continuous, 75 integer (0 binary)

Root relaxation: infeasible, 25 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 infeasible    0               - infeasible      -     -    0s

Explored 1 nodes (25 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 0

Model is infeasible or unbounded
Best objective -, best bound -, gap -
```

[ ]:

```python
# print optimal objective and optimal solution
print("\nOptimal Objective: " + str(myModel.ObjVal))
print("\nOptimal Solution: " )
allVars = myModel.getVars()
for curVar in allVars:
    print(curVar.varName + " " + str(curVar.x))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
c:\Users\jchu1\Desktop\FileLibrary\ORIE5380\project\project.ipynb Cell 3 in␣
  ↪<cell line: 2>()
```

```
      <a href='vscode-notebook-cell:/c%3A/Users/jchu1/Desktop/FileLibrary/
  ↪ORIE5380/project/project.ipynb#W1sZmlsZQ%3D%3D?line=0'>1</a> # print optimal␣
  ↪objective and optimal solution
----> <a href='vscode-notebook-cell:/c%3A/Users/jchu1/Desktop/FileLibrary/
  ↪ORIE5380/project/project.ipynb#W1sZmlsZQ%3D%3D?line=1'>2</a> print("\nOptimal␣
  ↪Objective: " + str(myModel.ObjVal))
      <a href='vscode-notebook-cell:/c%3A/Users/jchu1/Desktop/FileLibrary/
  ↪ORIE5380/project/project.ipynb#W1sZmlsZQ%3D%3D?line=2'>3</a> print("\nOptimal␣
  ↪Solution: " )
      <a href='vscode-notebook-cell:/c%3A/Users/jchu1/Desktop/FileLibrary/
  ↪ORIE5380/project/project.ipynb#W1sZmlsZQ%3D%3D?line=3'>4</a> allVars = myMode.
  ↪getVars()

File src\gurobipy\model.pxi:353, in gurobipy.Model.__getattr__()

File src\gurobipy\model.pxi:1884, in gurobipy.Model.getAttr()

File src\gurobipy\attrutil.pxi:100, in gurobipy.__getattr()

AttributeError: Unable to retrieve attribute 'ObjVal'
```