appendix2

December 5, 2022

1 Appendix 2 - Updated Formulation with Number of Aircraft as Decision Variable

[ ]:
```
# import gurobi
from gurobipy import *
import numpy as np

# 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

# create decision variable for total number of airplanes
aircraftVar = myModel.addVar(vtype = GRB.INTEGER, name = "aircraft")

# 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

# add a new cost for each aircraft (so to minimize total number required)
objExpr += aircraftVar

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")

# 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 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 = aircraftVar, 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")

# 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, 106 columns and 316 nonzeros
Model fingerprint: 0x4ad0dab4
Variable types: 0 continuous, 106 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+01, 4e+02]
Presolve removed 49 rows and 31 columns
Presolve time: 0.00s
Presolved: 33 rows, 75 columns, 210 nonzeros
Variable types: 0 continuous, 75 integer (0 binary)

Root relaxation: objective 1.651500e+04, 36 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               0    16515.000000 16515.0000  0.00%     -    0s

Explored 1 nodes (36 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 1: 16515

Optimal solution found (tolerance 1.00e-04)
Best objective 1.651500000000e+04, best bound 1.651500000000e+04, gap 0.0000%
```

[ ]:
```
# 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))
```

```
Optimal Objective: 16515.0

Optimal Solution:
xAB1 100.0
xAB2 200.0
xAB3 100.0
xAB4 400.0
xAB5 300.0
xAC1 50.0
xAC2 50.0
xAC3 50.0
xAC4 50.0
xAC5 50.0
xBA1 25.0
xBA2 25.0
xBA3 25.0
xBA4 25.0
xBA5 25.0
xBC1 25.0
xBC2 25.0
xBC3 25.0
xBC4 25.0
xBC5 25.0
xCA1 40.0
xCA2 40.0
xCA3 40.0
xCA4 40.0
xCA5 40.0
xCB1 400.0
xCB2 200.0
xCB3 300.0
xCB4 200.0
xCB5 400.0
yAB1 100.0
yAB2 200.0
yAB3 100.0
yAB4 400.0
yAB5 300.0
yAC1 50.0
yAC2 50.0
yAC3 50.0
yAC4 50.0
yAC5 50.0
```

```
yBA1 25.0
yBA2 25.0
yBA3 25.0
yBA4 25.0
yBA5 25.0
yBC1 25.0
yBC2 25.0
yBC3 25.0
yBC4 25.0
yBC5 25.0
yCA1 40.0
yCA2 40.0
yCA3 40.0
yCA4 40.0
yCA5 40.0
yCB1 400.0
yCB2 200.0
yCB3 300.0
yCB4 200.0
yCB5 400.0
zAB1 -0.0
zAB2 -0.0
zAB3 -0.0
zAB4 -0.0
zAB5 -0.0
zAC1 -0.0
zAC2 -0.0
zAC3 -0.0
zAC4 -0.0
zAC5 -0.0
zBA1 320.0
zBA2 450.0
zBA3 170.0
zBA4 -0.0
zBA5 85.0
zBC1 430.0
zBC2 -0.0
zBC3 180.0
zBC4 350.0
zBC5 365.0
zCA1 -0.0
zCA2 -0.0
zCA3 -0.0
zCA4 -0.0
zCA5 -0.0
zCB1 -0.0
zCB2 -0.0
zCB3 -0.0
```

```
zCB4 -0.0
zCB5 -0.0
sA1 -0.0
sA2 135.0
sA3 500.0
sA4 285.0
sA5 -0.0
sB1 -0.0
sB2 -0.0
sB3 -0.0
sB4 -0.0
sB5 100.0
sC1 -0.0
sC2 265.0
sC3 -0.0
sC4 15.0
sC5 -0.0
aircraft 1390.0
```