ORIE 5380, CS 5277: Optimization Methods

Cargo Operations of Express Air

Final Project, Fall 2018

Ang Zhou

az493

1. Summary

The objective of the project is to help Express Air company to find the best weekly aircraft routine by developing an optimization model, thereby minimizing the cost for shifting empty aircraft and the cost for postpone the cargo transportation, which is equivalent to holding a cargo at airport. I am going to convert this problem into a linear programing problem and use python with Gurobi tool for solving this problem.
In this report, I am going to cover the problem overview, data description, optimization model, and the final result.

2. Problem Overview

The overall problem is basically a minimum cost problem. Express Air runs a cargo transportation business. There are three airports A, B, and C, and every day there are a certain amount of new cargo transportation tasks (transport one cargo is one cargo transportation task) added to path A to B, A to C, B to A, B to C, C to A, and C to B. Each path and each day have different certain amount of new cargo transportation tasks. There are also a fixed amount of aircrafts in this business system and each aircraft can carry one unit load of cargo. Aircrafts are distributed to airport A, B, and C, and the total number of the cargo transported out from an airport depends on the totally number of the available aircrafts at that airport on that day. As we can see, since the total number of aircraft for an airport on a certain day is limited, there are cases that it is impossible for some cargo transportation tasks to be complete in one day. In that case, we allow tasks to be postponed for several days but need to pay an extra cost every day. Also, in order to make some airports to have enough or as many as available aircrafts for doing the transportation tasks in the next day, sometimes empty aircrafts need to be flied from some airports to others, which also cost an extra fee.

Furthermore, the problem assumes that it takes an aircraft a whole day to travel from one airport to another, no matter it carries cargo or not. And also, that aircraft become an available aircraft for the new airport at the beginning of the next day.

Our challenge is to find a weekly logistic route to minimize the extra cost, which includes the cost for postponing of the cargo transportation tasks, and the cost for flying empty aircrafts.

3. Data Description

3.1 Total amount of the aircraft

Express Air has 1200 aircraft for this business, which means the sum of the available aircraft in all three airports equals 1200 every day.

3.2 The amount of the new cargo that need to be transport

On each day, each path, there are a certain amount of new cargo transportation tasks added into the system. The demands for each path on each day are shown as follows:

Origin-destination \ Day	Monday	Tuesday	Wednesday	Thursday	Friday
A-B	100	200	100	400	300
A-C	50	50	50	50	50
B-A	25	25	25	25	25
B-C	25	25	25	25	25
C-A	40	40	40	40	40
C-B	400	200	300	200	400

Table 1: Amounts of cargo (in aircraft loads) arriving into the system on each day that need to be carried between each origin-destination airport

3.3 Extra operation cost

There are two different extra cost:

1) Postpone cargo transportation task:
   Holding one full aircraft loaf of cargo at an airport generates 10 unit cost per day.
2) Empty repositioning aircrafts:
   Shifting an empty aircraft from one airport to another cost differently depends on from which airport to which airport.

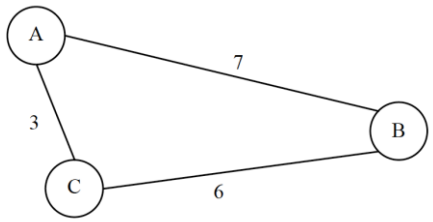


Figure 1: Empty repositioning costs among different airports

For example, the cost for flying an empty aircraft from A to C is 3.

4. Optimization model

4.1 Decision Variables

I defined four types of variables for this problem:

1) X_{ki}: # available aircraft at airport K on day i in the morning

2) Y_{k1k2i}: # aircraft transfer from airport K1 to airport K2 on day i
3) N_{k1k2i}: # cargo transportation tasks left from K1 to airport K2 on day i in the morning (before new cargo transportation tasks added into the system), which is the amount of cargo left from yesterday night
4) Mk1k2i: # cargo transported from airport K1 to airport K2 on day i

Where k1, k2 ∈ $\{A, B, C\}$; k1!=k2; i ∈ {1,2,3,4,5};

4.2 Coefficients/Constants:

Zk1k2i: # new cargo transportation tasks added to K1-K2 path. (refer to the entries in Table 1 in section 3.2)

4.3 Objective Functions:

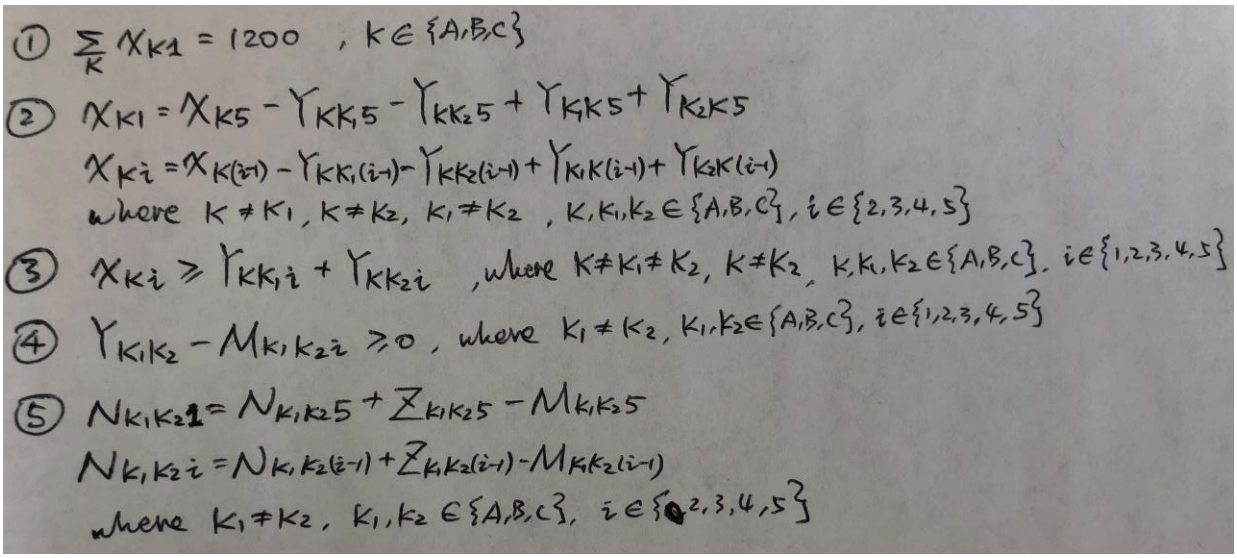
Minimize

$10\sum_{k1}\sum_{k2}\sum_{i,i\leq 4} N_{k1k2i} + 30\sum_{k1}\sum_{k2}\sum_{i,i=5} N_{k1k2i} + 3\sum_i(Y_{ABi} - M_{ABi} + Y_{BAi} - M_{BAi}) + 7\sum_i(Y_{ACi} - M_{ACi} + Y_{CAi} - M_{CAi}) + 6\sum_i(Y_{BCi} - M_{BCi} + Y_{CBi} - M_{CBi})$

Where k1, k2 ∈ $\{A, B, C\}$; k1!=k2; i ∈ {1,2,3,4,5};

Note that I assume the postpone of the cargo on Friday takes the postpone cost three times the regular postpone cost, which is 30, since the cost keep adding during the weekend.

4.3 Constraints:



Constraint 1: total amount of the aircraft is 1200. Since the totally amount of the aircraft never changed in the system, I only need to guarantee the amount of the aircraft on the first day (Monday)

Constraint 2: # available aircrafts in this airport today = # available aircrafts in this airport yesterday - # aircraft left from this airport yesterday + # aircraft arrived at this airport yesterday

Constraint 3: # available aircrafts in this airport today >= # aircraft leave from this airport today

Constraint 4: # aircrafts shift today on one path >= # units of cargo transported today on one path

Constraint 5: # units of cargo left today morning on this path= # units of cargo left yesterday morning on this path + # units of new cargo added yesterday into this path - # units of cargo transported yesterday on this path

5. Results

By using Gurobi, we can got the optimized value with the minimum total extra cost 21725.

The logistic is as following:

X_{ki}	Mon	Tue	Wen	Thu	Fri
A	65	335	150	450	350
B	510	620	485	400	600
C	625	245	565	350	250

Y_{k1k2i}	Mon	Tue	Wen	Thu	Fri
A-B	15	285	100	400	300
A-C	50	50	50	50	50
B-A	300	105	410	310	25
B-C	195	515	75	90	575
C-A	35	45	40	40	40
C-B	590	200	300	200	210

N_{k1k2i}	Mon	Tue	Wen	Thu	Fri
A-B	0	85	0	0	0
A-C	0	0	0	0	0
B-A	0	0	0	0	0
B-C	0	0	0	0	0
C-A	0	5	0	0	0
C-B	190	0	0	0	0

M_{k1k2i}	Mon	Tue	Wen	Thu	Fri
A-B	15	285	100	400	300
A-C	50	50	50	50	50
B-A	25	25	25	25	25
B-C	25	25	25	25	25
C-A	35	45	40	40	40
C-B	590	200	300	200	210

Appendix

Code:

```
from gurobipy import *
import pandas as pd
import numpy as np
# create a new model
myModel = Model("Final_Project")

# create decision variables and integrate them into the model
k_s = 3
i_s = 5

# vars storage
x_s = [[0 for i in range(i_s)] for k in range(k_s)]
y_s = [[[0 for i in range(i_s)] for k in range(k_s)] for k in range(k_s)]
n_s = [[[0 for i in range(i_s)] for k in range(k_s)] for k in range(k_s)]
m_s = [[[0 for i in range(i_s)] for k in range(k_s)] for k in range(k_s)]
z_s = [[[0 for i in range(i_s)] for k in range(k_s)] for k in range(k_s)]

# z_s (zk1k2i) #cargo tasks to K1-K2 path.
z_s[0][1]=[100,200,100,400,300]
z_s[0][2]=[50,50,50,50,50]
z_s[1][0]=[25,25,25,25,25]
z_s[1][2]=[25,25,25,25,25]
z_s[2][0]=[40,40,40,40,40]
z_s[2][1]=[400,200,300,200,400]
# x_s (xki) avaliable #aircraft at airport K at day i (morning)
for k in range(k_s):
    for i in range(i_s):
        xVar = myModel.addVar(vtype=GRB.INTEGER, name='x' + str(k+1) + ',' + str(i+1))
        x_s[k][i] = xVar

myModel.update()

# y_s (yk1k2i) #aircraft transfer from K1 to K2 in day i
for k1 in range(k_s):
    for k2 in range(k_s):
        if k1 != k2:
            for i in range(i_s):
                yVar = myModel.addVar(vtype=GRB.INTEGER, name='y' + str(k1+1) + ',' + str(k2+1) + ',' + str(i+1))
                y_s[k1][k2][i] = yVar

myModel.update()

# n_s (nk1k2i) #cargo tasks left from K1 to K2 at day i (morning) before adding new tasks
for k1 in range(k_s):
    for k2 in range(k_s):
        if k1 != k2:
            for i in range(i_s):
                nVar = myModel.addVar(vtype=GRB.INTEGER, name='n' + str(k1+1) + ',' + str(k2+1) + ',' + str(i+1))
                n_s[k1][k2][i] = nVar

myModel.update()

# m_s (mk1k2i) #cargo tasks left from K1 to K2 at day i (morning) before adding new tasks
for k1 in range(k_s):
    for k2 in range(k_s):
        if k1 != k2:
            for i in range(i_s):
                mVar = myModel.addVar(vtype=GRB.INTEGER, name='m' + str(k1+1) + ',' + str(k2+1) + ',' + str(i+1))
                m_s[k1][k2][i] = mVar
```

```
myModel.update()


# create a linear expression for the objective
objExpr = LinExpr()
for k1 in range(k_s):
    for k2 in range(k_s):
        if k1 != k2:
            for i in range(i_s):
                nVar = n_s[k1][k2][i]
                objExpr += 10*nVar
                myModel.setObjective(objExpr, GRB.MINIMIZE)

myModel.update()

# create a linear expression for the objective
objExpr = LinExpr()
for k1 in range(k_s):
    for k2 in range(k_s):
        if k1 != k2:
            nVar = n_s[k1][k2][0]
            objExpr += 30*nVar
            for i in range(1,i_s):
                nVar = n_s[k1][k2][i]
                objExpr += 10*nVar
            myModel.setObjective(objExpr, GRB.MINIMIZE)
for i in range(i_s):
    yVar1 = y_s[0][1][i]
    mVar1 = m_s[0][1][i]
    yVar2 = y_s[1][0][i]
    mVar2 = m_s[1][0][i]
    objExpr += 7*(yVar1-mVar1+yVar2-mVar2)
    myModel.setObjective(objExpr, GRB.MINIMIZE)
for i in range(i_s):
    yVar1 = y_s[0][2][i]
    mVar1 = m_s[0][2][i]
    yVar2 = y_s[2][0][i]
    mVar2 = m_s[2][0][i]
    objExpr += 3*(yVar1-mVar1+yVar2-mVar2)
    myModel.setObjective(objExpr, GRB.MINIMIZE)
for i in range(i_s):
    yVar1 = y_s[1][2][i]
    mVar1 = m_s[1][2][i]
    yVar2 = y_s[2][1][i]
    mVar2 = m_s[2][1][i]
    objExpr += 6*(yVar1-mVar1+yVar2-mVar2)
    myModel.setObjective(objExpr, GRB.MINIMIZE)

myModel.update()

# Type 1 constraint: Constraint for number of aircraft on the first day
constExpr = LinExpr()
for k in range(k_s):
    xVar = x_s[k][0]
    constExpr += xVar
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 1200, name="total_aircraft")

# Type 2 constraint: Constraint for flight balance
# 5-1-A
constExpr = LinExpr()
constExpr += x_s[0][0] - x_s[0][4]+y_s[0][1][4]+y_s[0][2][4]-y_s[1][0][4]-y_s[2][0][4]
```

```
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_5_1_A")
# 5-1-B
constExpr = LinExpr()
constExpr += x_s[1][0] - x_s[1][4]+y_s[1][0][4]+y_s[1][2][4]-y_s[0][1][4]-y_s[2][1][4]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_5_1_B")
# 5-1-C
constExpr = LinExpr()
constExpr += x_s[2][0] - x_s[2][4]+y_s[2][0][4]+y_s[2][1][4]-y_s[0][2][4]-y_s[1][2][4]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_5_1_C")

# 2-1-A
constExpr = LinExpr()
constExpr += x_s[0][1] - x_s[0][0]+y_s[0][1][0]+y_s[0][2][0]-y_s[1][0][0]-y_s[2][0][0]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_2_1_A")
# 2-1-B
constExpr = LinExpr()
constExpr += x_s[1][1] - x_s[1][0]+y_s[1][0][0]+y_s[1][2][0]-y_s[0][1][0]-y_s[2][1][0]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_2_1_B")
# 2-1-C
constExpr = LinExpr()
constExpr += x_s[2][1] - x_s[2][0]+y_s[2][0][0]+y_s[2][1][0]-y_s[0][2][0]-y_s[1][2][0]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_2_1_C")

# 3-2-A
constExpr = LinExpr()
constExpr += x_s[0][2] - x_s[0][1]+y_s[0][1][1]+y_s[0][2][1]-y_s[1][0][1]-y_s[2][0][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_3_2_A")
# 3-2-B
constExpr = LinExpr()
constExpr += x_s[1][2] - x_s[1][1]+y_s[1][0][1]+y_s[1][2][1]-y_s[0][1][1]-y_s[2][1][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_3_2_B")
# 3-2-C
constExpr = LinExpr()
constExpr += x_s[2][2] - x_s[2][1]+y_s[2][0][1]+y_s[2][1][1]-y_s[0][2][1]-y_s[1][2][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_3_2_C")

# 4-3-A
constExpr = LinExpr()
constExpr += x_s[0][3] - x_s[0][2]+y_s[0][1][2]+y_s[0][2][2]-y_s[1][0][2]-y_s[2][0][2]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_4_3_A")
# 4-3-B
constExpr = LinExpr()
constExpr += x_s[1][3] - x_s[1][2]+y_s[1][0][2]+y_s[1][2][2]-y_s[0][1][2]-y_s[2][1][2]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_4_3_B")
# 4-3-C
constExpr = LinExpr()
constExpr += x_s[2][3] - x_s[2][2]+y_s[2][0][2]+y_s[2][1][2]-y_s[0][2][2]-y_s[1][2][2]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_4_3_C")

# 5-4-A
constExpr = LinExpr()
constExpr += x_s[0][4] - x_s[0][3]+y_s[0][1][3]+y_s[0][2][3]-y_s[1][0][3]-y_s[2][0][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_5_4_A")
# 5-4-B
constExpr = LinExpr()
constExpr += x_s[1][4] - x_s[1][3]+y_s[1][0][3]+y_s[1][2][3]-y_s[0][1][3]-y_s[2][1][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_5_4_B")
# 5-4-C
constExpr = LinExpr()
constExpr += x_s[2][4] - x_s[2][3]+y_s[2][0][3]+y_s[2][1][3]-y_s[0][2][3]-y_s[1][2][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="aircraft_5_4_C")
```

```
# Type 3 constraint: #avaliable aircraft >= #aircraft out
for i in range(i_s):
    constExpr = LinExpr()
    constExpr += x_s[0][i] - y_s[0][1][i] - y_s[0][2][i]
    myModel.addConstr(lhs = constExpr, sense = GRB.GREATER_EQUAL, rhs = 0,
name="airport_A_aircraft_requirement_"+str(i+1))
    constExpr = LinExpr()
    constExpr += x_s[1][i] - y_s[1][0][i] - y_s[1][2][i]
    myModel.addConstr(lhs = constExpr, sense = GRB.GREATER_EQUAL, rhs = 0,
name="airport_B_aircraft_requirement_"+str(i+1))
    constExpr = LinExpr()
    constExpr += x_s[2][i] - y_s[2][0][i] - y_s[2][1][i]
    myModel.addConstr(lhs = constExpr, sense = GRB.GREATER_EQUAL, rhs = 0,
name="airport_C_aircraft_requirement_"+str(i+1))

# Type 4 constraint: #aircraft our >= #cargo out
for k1 in range(k_s):
    for k2 in range(k_s):
        if k1 != k2:
            for i in range(i_s):
                constExpr = LinExpr()
                constExpr += y_s[k1][k2][i] - m_s[k1][k2][i]
                myModel.addConstr(lhs = constExpr, sense = GRB.GREATER_EQUAL, rhs = 0,
name="aircraft_cargo_requirement_"+str(k1+1)+str(k2+1)+str(i+1))

# Type 5 constraint: #left cargo balance
# AB_5_1
constExpr = LinExpr()
constExpr += n_s[0][1][0] - n_s[0][1][4] - z_s[0][1][4] + m_s[0][1][4]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AB_5_1")
# AC_5_1
constExpr = LinExpr()
constExpr += n_s[0][2][0] - n_s[0][2][4] - z_s[0][2][4] + m_s[0][2][4]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AC_5_1")
# BA_5_1
constExpr = LinExpr()
constExpr += n_s[1][0][0] - n_s[1][0][4] - z_s[1][0][4] + m_s[1][0][4]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BA_5_1")
# BC_5_1
constExpr = LinExpr()
constExpr += n_s[1][2][0] - n_s[1][2][4] - z_s[1][2][4] + m_s[1][2][4]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BC_5_1")
# CA_5_1
constExpr = LinExpr()
constExpr += n_s[2][0][0] - n_s[2][0][4] - z_s[2][0][4] + m_s[2][0][4]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CA_5_1")
# CB_5_1
constExpr = LinExpr()
constExpr += n_s[2][1][0] - n_s[2][1][4] - z_s[2][1][4] + m_s[2][1][4]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CB_5_1")

# AB_2_1
constExpr = LinExpr()
constExpr += n_s[0][1][1] - n_s[0][1][0] - z_s[0][1][0] + m_s[0][1][0]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AB_2_1")
# AC_2_1
constExpr = LinExpr()
constExpr += n_s[0][2][1] - n_s[0][2][0] - z_s[0][2][0] + m_s[0][2][0]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AC_2_1")
# BA_2_1
constExpr = LinExpr()
constExpr += n_s[1][0][1] - n_s[1][0][0] - z_s[1][0][0] + m_s[1][0][0]
```

```
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BA_2_1")
# BC_2_1
constExpr = LinExpr()
constExpr += n_s[1][2][1] - n_s[1][2][0] - z_s[1][2][0] + m_s[1][2][0]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BC_2_1")
# CA_2_1
constExpr = LinExpr()
constExpr += n_s[2][0][1] - n_s[2][0][0] - z_s[2][0][0] + m_s[2][0][0]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CA_2_1")
# CB_2_1
constExpr = LinExpr()
constExpr += n_s[2][1][1] - n_s[2][1][0] - z_s[2][1][0] + m_s[2][1][0]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CB_2_1")

# AB_3_2
constExpr = LinExpr()
constExpr += n_s[0][1][2] - n_s[0][1][1] - z_s[0][1][1] + m_s[0][1][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AB_3_2")
# AC_3_2
constExpr = LinExpr()
constExpr += n_s[0][2][2] - n_s[0][2][1] - z_s[0][2][1] + m_s[0][2][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AC_3_2")
# BA_3_2
constExpr = LinExpr()
constExpr += n_s[1][0][2] - n_s[1][0][1] - z_s[1][0][1] + m_s[1][0][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BA_3_2")
# BC_3_2
constExpr = LinExpr()
constExpr += n_s[1][2][2] - n_s[1][2][1] - z_s[1][2][1] + m_s[1][2][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BC_3_2")
# CA_3_2
constExpr = LinExpr()
constExpr += n_s[2][0][2] - n_s[2][0][1] - z_s[2][0][1] + m_s[2][0][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CA_3_2")
# CB_3_2
constExpr = LinExpr()
constExpr += n_s[2][1][2] - n_s[2][1][1] - z_s[2][1][1] + m_s[2][1][1]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CB_3_2")

# AB_4_3
constExpr = LinExpr()
constExpr += n_s[0][1][3] - n_s[0][1][2] - z_s[0][1][2] + m_s[0][1][2]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AB_4_3")
# AC_4_3
constExpr = LinExpr()
constExpr += n_s[0][2][3] - n_s[0][2][2] - z_s[0][2][2] + m_s[0][2][2]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AC_4_3")
# BA_4_3
constExpr = LinExpr()
constExpr += n_s[1][0][3] - n_s[1][0][2] - z_s[1][0][2] + m_s[1][0][2]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BA_4_3")
# BC_4_3
constExpr = LinExpr()
constExpr += n_s[1][2][3] - n_s[1][2][2] - z_s[1][2][2] + m_s[1][2][2]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BC_4_3")
# CA_4_3
constExpr = LinExpr()
constExpr += n_s[2][0][3] - n_s[2][0][2] - z_s[2][0][2] + m_s[2][0][2]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CA_4_3")
# CB_4_3
constExpr = LinExpr()
constExpr += n_s[2][1][3] - n_s[2][1][2] - z_s[2][1][2] + m_s[2][1][2]
```

```
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CB_4_3")

# AB_5_4
constExpr = LinExpr()
constExpr += n_s[0][1][4] - n_s[0][1][3] - z_s[0][1][3] + m_s[0][1][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AB_5_4")
# AC_5_4
constExpr = LinExpr()
constExpr += n_s[0][2][4] - n_s[0][2][3] - z_s[0][2][3] + m_s[0][2][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_AC_5_4")
# BA_5_4
constExpr = LinExpr()
constExpr += n_s[1][0][4] - n_s[1][0][3] - z_s[1][0][3] + m_s[1][0][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BA_5_4")
# BC_5_4
constExpr = LinExpr()
constExpr += n_s[1][2][4] - n_s[1][2][3] - z_s[1][2][3] + m_s[1][2][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_BC_5_4")
# CA_5_4
constExpr = LinExpr()
constExpr += n_s[2][0][4] - n_s[2][0][3] - z_s[2][0][3] + m_s[2][0][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CA_5_4")
# CB_5_4
constExpr = LinExpr()
constExpr += n_s[2][1][4] - n_s[2][1][3] - z_s[2][1][3] + m_s[2][1][3]
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 0, name="left_cargo_balance_CB_5_4")

#############################################################################

# 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="Project.lp")

# optimize the model
myModel.optimize()

allVars = myModel.getVars()
```