UniGroup Middle Mile Hub Project

SCMA-6350

Dr. Shakiba Enayati

James Tao, Alissa Oprisoni

Table of Contents

[Problem Description 2](#_Toc1942602954)

[Baseline Model (1-Hub) 3](#_Toc2026386232)

[Numerical Results 5](#_Toc2006587108)

[1-Hub Results: 5](#_Toc1100711156)

[Sensitivity Analysis: 2-Hub Location Model and Numeric Results 6](#_Toc973265488)

[What-if: No-Hub Scenario (Point-to-Point) 9](#_Toc702272903)

[Conclusion: 10](#_Toc1759845167)

[Reference: 11](#_Toc1222356031)

[Appendix A: One Hub Python Code 11](#_Toc1817102350)

[Appendix B: Two Hubs Python Code 13](#_Toc743610776)

[Appendix C: No Hub Calculation Python Code 15](#_Toc2040399430)

# Problem Description

The main goal is for UniGroup to find the best possible location to locate a middle-mile-hub between order origins and destination to minimize the transportation cost. To ensure the efficiency of the transportation, Hub-and-Spoke could be a better methodology other than Point-to-Point to achieve the goal as shown in Figure 1. (Hub & Spoke vs Point-To-Point, Which Is Better for Roadways Delivery? 2019) In this case, all orders will go through the middle-mile-hub from origins to destinations with cheaper unit costs instead of shipping directly from origins to destinations. The objective of the project is to form a model and determine which city will be the best location option to locate the middle-mile-hub. Several sensitivity analyses including 2-Hub Scenario and No-Hub Scenario will be discussed in the report.

Diagram

Description automatically generated

Figure 1 Point-to-Point and Hub & Spoke Models

# Baseline Model (1-Hub)

The following linear model is based on Multiple Allocation P-Hub Location Model (P-Hub Median Location Model) illustrated by Campbell (1991). Capacity Limitation of Hub Location Problem is considered in this model as well. (James F. Campbell, 1996)

**Model Assumptions**

* The objective function is to minimize the total cost.
* Each non-hub node is assigned to only one hub.
* The installation cost of the hub nodes is not considered.

**Model Parameters**

* hij: The load of each order from origin i to destination j through hub k.
* Dkij: The distance from i to j through the hub k.
* Ckij: Unit cost of local (non-hub to hub k) movement between nodes i and j.
* qk: The capacity of the Hub node k.

**Model Decision Variables**

* Xk: A hub is located at a candidate node k.
* Zkij: Fraction of flow from i to j that is routed via hub k.

**Objective Function**

Min Si Sj Sk Ckij hij Dkij Zkij (A)

**Constraints**

1. SkZkij = 1 " i, j,
2. SkXk= 1,
3. Zkij <= xk " i, j, k,
4. Si Sj hij Zkij  <= qk Xk " k,
5. Zkij >= 0 " i, j, k,
6. xk = 0, 1 " k.

Equation (A) minimizes the maximum cost of transportation between each origin/

destination pair. (1) ensures that each origin-destination pair (i, j) must be assigned to exactly one hub pair. Equation (2) limits there will be only one hub (3) limits that flow from origin i to destination j cannot be assigned to a hub at location k unless a hub is located at these candidate nodes (when we travel from one node to another node via one hub node, k is coincided with each other). Equation (4) indicates that the total flows going through the hub k cannot exceed the capacity of the node k. Equation (5) is a standard integrality constraint. Equation (6) is relaxed decision variable constraint(binary).

# Numerical Results

## 1-Hub Results:

Two example files with 150 order records and distances matrix have been received for analysis. 20 MSAs in 5 USA continental regions are chosen (4 cities in each region) as hub candidates based on the origins and destinations in the example files. Another file is created to store City names, State names, and their corresponding Zip codes. The order records file contains zip codes of all order pairs (origins and destinations) and the route distance between each pair. In this case, assumption with Unit Cost = 0.36 and Hub Capacity = 10,000 have been made.

Python and Gurobi are used as the main tool to solve the problem based on the model demonstrated before including Decision Variables, Objective function, and several Constraints (Appendix A: One Hub Python Code). After running the code, the console shows the optimal solution is the location with the zip code [37922] [‘Knoxville, Tennessee’] with total cost $1,165,720,000. A map has been generated with ArcGIS to virtualized orders and the optimal solution with other hub candidates according to example files. (Figure 2) The green lines show the orders form origins to destinations going through the hub chosen in Knoxville, Tennessee. All the other dots represent hub candidates which are not chosen.

A picture containing chart

Description automatically generated

Figure 2 1-hub Optimal Solution Virtualization

## Sensitivity Analysis: 2-Hub Location Model and Numeric Results

**Model Assumptions**

* The objective function is to minimize the total cost.
* Each non-hub node is assigned to only one hub.
* The installation cost of the hub nodes will not be considered.
* All the variables are binary variables (0—1).

**Model Parameters**

* hij: The order volume of each route from origin i to destination j through hub k.
* Dkmij: The distance from i to j through the hub k and hub m.
* Ckmij: Unit cost travel between nodes i and j when going via hubs at nodes k and m.

Ckmij = Cik + aCkm + Cmj (a is the discount rate between hub k and m)

* qk: The capacity of the Hub node k.

**Model Decision Variables**

* Xk: A hub is located at a candidate node k.
* Zkmij: Fraction of flow from i to j that is routed via hubs k and m.

**Objective Function**

Min Si Sj Sk Sm Ckmij hij Dkmij Zkmij  (B)

**Constraints**

1. SkSm Zkmij = 1 " i, j,
2. Sk Xk = p
3. Zkmij <= Xk " i, j, k,
4. Zkmij <= Xm " i, j, m,
5. Si Sj hij Zkmij <= qkm Xkm " k, m
6. Zkmij >= 0 " i, j, k, m
7. xk = 0, 1 " k.

Equation (B) minimizes the maximum cost of transportation between each origin/

destination pair. (1) ensures that each origin-destination pair (i, j) must be assigned to exactly one hub pair. Equation (2) limits there will be only p hub(s) (3) limits that flow from origin i to destination j cannot be assigned to a hub at location k unless a hub is located at these candidate nodes (when we travel from one node to another node via one hub node, k is coincided with each other). (4) limits that flow from origin i to destination j cannot be assigned to a hub at location m unless a hub is located at these candidate nodes. Equation (5) indicates that the total flows going through the hub k and hub m cannot exceed the capacity of the nodes k and m. Equation (6) is a standard integrality constraint. Equation (7) is relaxed decision variable constraint(binary).

Similarly, two example files and the hub candidates file are imported into the python code (Appendix B: Two Hubs Python Code). Instead of 1 hub, there are 2 hubs chosen in this case. Assumptions of Unit Cost = 0.36 and Capacity = 10,000 stay the same. Since there is discount (a) on unit cost between hub k and hub m, a = 0.6 is another additional assumption to calculate the cost between hub k and hub m. the result of the 2-hub case model shows that 2 locations with zip codes [37922] [‘Knoxville, Tennessee’] and [43223] [‘Columbus, Ohio’]. The total cost of orders going through hub k and hub m according to the example files is $1,281,650,000. As shown in the map in ArcGIS (Figure 3), locations of Knoxville, Tennessee and Columbus, Ohio have been selected as hub locations. The cost between 2 hubs will stay still if there is not any change on gas price. All the orders will go through either origin - Knoxville, Tennessee - Columbus, Ohio – destination, or origin - Columbus, Ohio -Knoxville, Tennessee – destination. The blue lines represent all the flows going in and out from hub located at Columbus, Ohio and green lines show all the orders/flows going in and out from the hub located in Knoxville, Tennessee. The other dots on the map represent other hub candidates which are not selected.

Diagram

Description automatically generated

Figure 3 2-hub Optimal Solution Virtualization

## What-if: No-Hub Scenario (Point-to-Point)

This is the scenario of calculation Total cost with No-Hub existing based on the order information provided. The formulation of the No-Hub case is significantly similar as the Objective Function of Baseline 1-Hub model:

For each order, Cost = Load \* Distance \* Unit\_Cost

Total cost will be the sum of the cost of each order (Appendix C: No Hub Calculation Python Code). The Unit Cost in Point-to-Point case is higher than Hub-and-Spoke case as described previously(*Cost Structure of Point-to-Point and Hub-and-Spoke Networks*, n.d.). In order to compare the total cost of No-Hub with 1-Hub and 2-Hub case, different Unit Cost from $0.5 to $1.5 in No-Hub case will be discussed. The following equation calculates the No-Hub case percentage total cost comparing with 1-Hub Case and 2-Hub Case:

As shown in Figure 4, Unit costs from $0.5 to $1.5 have been listed in the first column. The second column concludes the No-Hub Total Cost changing along with the Unit Cost. The Third Column values are from the % Total Cost formulation with 1-hub Total Cost ($1,165,720,000), and the fourth column values are the % Total Cost of No-Hub and 2-hub Total Cost ($1,281,650,000). A chart of No-Hub vs. Hub-and-Spoke % Total Cost is created to observe the difference between these 2 cases. The X axis represents the Unit Cost Changes in No-Hub case, and Y axis represents % Total Cost. The graph significantly shows that % total cost is going up along with the Unit Cost increasing in no hub case. If unit cost = $1 in no hub case, the 2-Hub case will save 35% of the Total Cost, and the 1-Hub case will save nearly 40% of the Total Cost. The chart has shown Hub-and-Spoke could save a large amount of money for the company under some conditions.

A picture containing graphical user interface

Description automatically generated

Figure 4 % Total Cost

# Conclusion:

According to the dataset provided, the 1-hub model should be the one to explore further or even implement for UniGroup. We suggest the hub location should be in Knoxville, Tennessee but that UniGroup use their entire dataset for better results to confirm where that 1-hub location should be. The 1-hub model will help UniGroup to minimize their total costs while maximizing their number of trucks and containers. The total cost for the 1-hub model is $1,165,720,000. The fuel price would be $0.36 without taking into consideration inflation. But further analysis should be considered for inflation for the year 2022 for changes in the fuel price. We also considered a 2-hub model, and we suggest the hubs to be located in Knoxville, Tennessee and Columbus, Ohio. The total cost for running a 2-hub model is $1,281,650,000 which is higher than the 1-hub model. Furthermore, considering the economic inflation, the total cost might be higher as the prices keep rising. Lastly, we considered a No-Hub model, it can be observed that No-Hub case total cost could be higher than total cost in 1-Hub and 2-Hub case.

In all consideration, the 1-hub model located in Knoxville, Tennessee would be the best option for UniGroup to implement in their business to run an efficient and low-cost operation according to the dataset provided. Also, deliveries and pick-ups will be more calculated and on time, which will yield a lower cost.

# Reference:

*Cost Structure of Point-to-Point and Hub-and-Spoke Networks*. (n.d.). Retrieved December 13, 2021, from https://transportgeography.org/contents/chapter2/geography-of-transportation-networks/point-to-point-hub-and-spoke-network-cost/

*Hub & Spoke vs Point-To-Point, Which Is Better For Roadways Delivery?* (2019, May 24). https://www.abivin.com/post/hub-spoke-vs-point-to-point-which-is-better-for-roadways-delivery

James F. Campbell. (1996). Hub Location and the p-Hub Median Problem. *INFORMS*, *44*(6), 923–935.

# Appendix A: One Hub Python Code

import pandas as pd

import gurobipy as gp

from gurobipy import GRB

distance = pd.read\_csv('dist\_matrix.csv')

orders = pd.read\_csv('toy\_dataset.csv')

hubs = pd.read\_csv('Hubs.csv')

#assume $0.36 per pound per mile constant cost in the middle mile problem

cost = 0.37

#open 1 hub

p = 1

#assume capacity of the hub is 100000

capacity = 100000

hub\_candidate = list(hubs.hubs)

#fixed cost

#fixed\_list = list(hubs.fixed\_cost)

origin\_locs\_list = list(orders.ORIGIN\_POSTAL\_CODE)

dest\_locs\_list = list(orders.DESTINATION\_POSTAL\_CODE)

order\_pairs = list(zip(origin\_locs\_list,dest\_locs\_list))

Load\_list = list(orders.REGISTERED\_TOTAL\_WEIGHT)

m = gp.Model("1\_hub\_larger")

#Decision Variables

x = m.addVars(hub\_candidate, vtype=GRB.BINARY, name="hub")

y = m.addVars(order\_pairs, hub\_candidate, vtype=GRB.CONTINUOUS, name='Assign')

m.update()

#Objective Function

objexpr1=0

for (i,j) in order\_pairs:

for k in hub\_candidate:

C\_i\_k = round(cost\*list(distance.loc[distance['From']==i][str(k)])[0],2)

C\_k\_j = round(cost\*list(distance.loc[distance['From']==j][str(k)])[0],2)

C\_i\_j\_k = round(C\_i\_k + C\_k\_j)

objexpr1+=Load\_list[origin\_locs\_list.index(i)] \* C\_i\_j\_k \* y[i,j,k]

m.setObjective(objexpr1, GRB.MINIMIZE)

#if oncludes fixed cost

#m.setObjective(objexpr1+y.prod(fixed\_list), GRB.MINIMIZE)

#Constraints

#con1

for (i,j) in order\_pairs:

expr\_1=0

for k in hub\_candidate:

expr\_1+=y[i,j,k]

m.addConstr(expr\_1==1 ,name=f"cons1[{i,j}]")

#con2

expr2=0

for k in hub\_candidate:

expr2+=x[k]

m.addConstr(expr2==p ,name="cons2")

#con3

for (i,j) in order\_pairs:

for k in hub\_candidate:

m.addConstr(y[i,j,k]<=x[k] ,name=f"cons3[{i,j,k}]")

m.addConstr(y[i,j,k]>=0 ,name=f"cons3\_2[{i,j,k}]")

#con4

for (i,j) in order\_pairs:

for k in hub\_candidate:

m.addConstr(Load\_list[origin\_locs\_list.index(i)] \* y[i,j,k] <= capacity \* x[k], name="cons4")

m.write("1\_hub\_larger.lp")

m.optimize()

if m.status == GRB.OPTIMAL:

print('Optimal objective: %g' % m.objVal)

elif m.status == GRB.INF\_OR\_UNBD:

sys.exit(0)

elif m.status == GRB.INFEASIBLE:

sys.exit(0)

elif m.status == GRB.UNBOUNDED:

sys.exit(0)

else:

print('Optimization ended with status %d' % m.status)

sys.exit(0)

for v in m.getVars():

if v.x!=0:

print('%s %g' % (v.varName, v.x))

# Appendix B: Two Hubs Python Code

import pandas as pd

import gurobipy as gp

from gurobipy import GRB

import sys

Rate=0.36

alpha=0.6

capacity=100000

p=2#number of open hubs

orders=pd.read\_csv('toy\_dataset.csv')

Hubs\_info=pd.read\_csv('Hubs.csv')

distanceMat=pd.read\_csv('dist\_matrix.csv')

#converting a column to a list

hub\_candidate\_locs=list(Hubs\_info.hubs)

hub\_pairs=[]

for hub1 in hub\_candidate\_locs:

for hub2 in hub\_candidate\_locs:

if hub1!=hub2:

hub\_pairs.append((hub1,hub2))

#hub\_fixedcost=list(Hubs\_info.fixed\_cost)

origin\_locs\_list=list(orders.ORIGIN\_POSTAL\_CODE)

dest\_locs\_list=list(orders.DESTINATION\_POSTAL\_CODE)

order\_pairs=list(zip(origin\_locs\_list,dest\_locs\_list))

Load\_W=list(orders.REGISTERED\_TOTAL\_WEIGHT)

#model implementation

model = gp.Model("UG\_Hub")

#define decision variables

x = model.addVars(order\_pairs,hub\_pairs,vtype=GRB.CONTINUOUS, name="x")

y= model.addVars(hub\_candidate\_locs,vtype=GRB.BINARY, name="y")

model.update()

#define your objective function

objexpr1=0

for (i,j) in order\_pairs:

for (k,m) in hub\_pairs:

C\_i\_k=round(Rate\*list(distanceMat.loc[distanceMat['From']==i][str(k)])[0],2)

C\_m\_j=round(Rate\*list(distanceMat.loc[distanceMat['From']==j][str(m)])[0],2)

C\_k\_m=round(alpha\*Rate\*list(distanceMat.loc[distanceMat['From']==k][str(m)])[0],2)

C\_i\_j\_k\_m=round(C\_i\_k+C\_m\_j+ C\_k\_m,2)

objexpr1+=Load\_W[origin\_locs\_list.index(i)]\*C\_i\_j\_k\_m\*x[i,j,k,m]

model.setObjective(objexpr1, GRB.MINIMIZE)

#define your constraints

#cons(1)

for (i,j) in order\_pairs:

expr\_1=0

for (k,m) in hub\_pairs:

expr\_1+=x[i,j,k,m]

model.addConstr(expr\_1==1 ,name=f"cons1[{i,j}]")

#cons(2)

expr2=0

for k in hub\_candidate\_locs:

expr2+=y[k]

model.addConstr(expr2==p ,name="cons2")

#cons(3)

for (i,j) in order\_pairs:

for (k,m) in hub\_pairs:

model.addConstr(x[i,j,k,m]<=y[k] ,name=f"cons3[{i,j,k,m}]")

model.addConstr(x[i,j,k,m]>=0 ,name=f"cons3\_2[{i,j,k,m}]")

#cons(4)

for (i,j) in order\_pairs:

for (k,m) in hub\_pairs:

model.addConstr(x[i,j,k,m]<=y[m] ,name=f"cons4[{i,j,k,m}]")

#capacity constraint

for (i,j) in order\_pairs:

for (k, m) in hub\_pairs:

model.addConstr(Load\_W[origin\_locs\_list.index(i)] \* x[i,j,k,m] <= capacity \* y[k], name="capacity\_k")

model.addConstr(Load\_W[origin\_locs\_list.index(i)] \* x[i,j,k,m] <= capacity \* y[m], name="capacity\_m")

model.write("UG\_Hub.lp")

model.optimize()

if model.status == GRB.OPTIMAL:

print('Optimal objective: %g' % model.objVal)

elif model.status == GRB.INF\_OR\_UNBD:

sys.exit(0)

elif model.status == GRB.INFEASIBLE:

sys.exit(0)

elif model.status == GRB.UNBOUNDED:

sys.exit(0)

else:

print('Optimization ended with status %d' % m.status)

sys.exit(0)

for v in model.getVars():

if v.x!=0:

print('%s %g' % (v.varName, v.x))

# Appendix C: No Hub Calculation Python Code

orders = pd.read\_csv('toy\_dataset.csv')

#assume $0.5 per pound per mile constant cost without going through a hub(cheaper)

cost = 0.5

Load\_list = list(orders.REGISTERED\_TOTAL\_WEIGHT)

Distance = list(orders.distance)

x = 0

for y in Distance:

x += round(Load\_list[Distance.index(y)] \* y \* cost)

print('No Hub Total Cost: ' + str(x) + ' (Unit cost:' + str(cost) + ')')