DHL Supply Chain Design

Research Proposal: Introduction and Method

Team Name: A Plus

Instructor: Michael Silas and Adrian Kumar

Introduction

Business Understanding

DHL is a German logistics company providing courier, package delivery, and express mail services, which is a division of the German logistics firm Deutsche Post. The company group delivers over 1.6 billion parcels per year.

The competition in the courier industry is very intense. The major competitors DHL is facing are FedEx, USPS, UPS and DSV. Customers of DHL are the companies from various industries, government organizations and individual customers who are looking for various logistics services.

Project goals and Sensitive Analysis

Customers get a competitive edge from working with DHL because the company offers individualized logistics solutions that are constructed from globally standardized warehouse, transportation, and integrated service components. When it comes to designing and managing supply chains, we bring industry expertise, global size, and local understanding to the table. It does this by establishing connections from suppliers to warehouses via truck loads as the mode of transport and by establishing connections from warehouses to customers via parcels as the mode of transport.

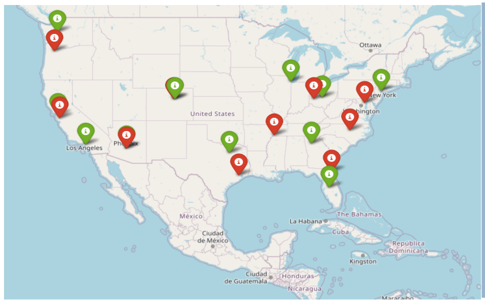
Our project goal is to create a robust network design model for Juice2U by selecting locations for manufacturing and distribution warehouses to fulfill the goal of minimizing the costs (fixed, variable, inbound, and outbound) under several constraints.

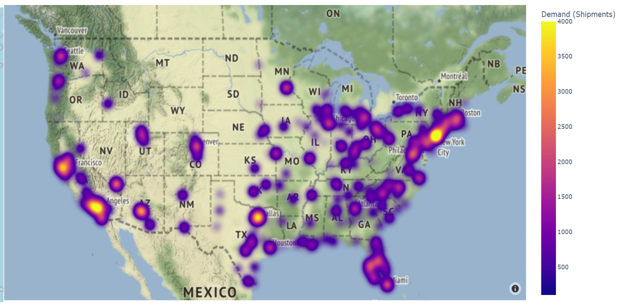
Sensitive Analysis will provide insight into the variables' relationships and determine the extent to which changes in the input values for a specific variable influence the outcomes of our mathematical model. According to our project, we will further investigate and analyze the nine sensitive analysis scenarios and conduct the results.

Data preparation and Exploratory Data Analysis

In the network design input data Excel, there are parameters, suppliers, WHs, customers, supplier-WHs-Dist, supplier-WH-Rates, WHs-customers-Dist and WH-customers-Rates, total in nine sheets. In the nine sheets, we focus on supplier_ID, Variable Cost / Shipment, Fixed cost, WH_ID, Variable Processing Cost / Shipment, customer_ID, Demand (shipments) variables. Also, we apply supplier to WHs: Distance Matrix, Supplier to WHs: Cost Matrix Per Shipment (Truck Loads), WHs to Customers: Distance Matrix, WHs to Customers: Distance Matrix (Per Shipment Parcels) in the below function. Having accurate information is important before beginning a project, so we check if the data has been incomplete and improperly formatted. After exploring the data set, there is no missing value. There are 10 suppliers, 11 warehouses and 988 customers in the data.

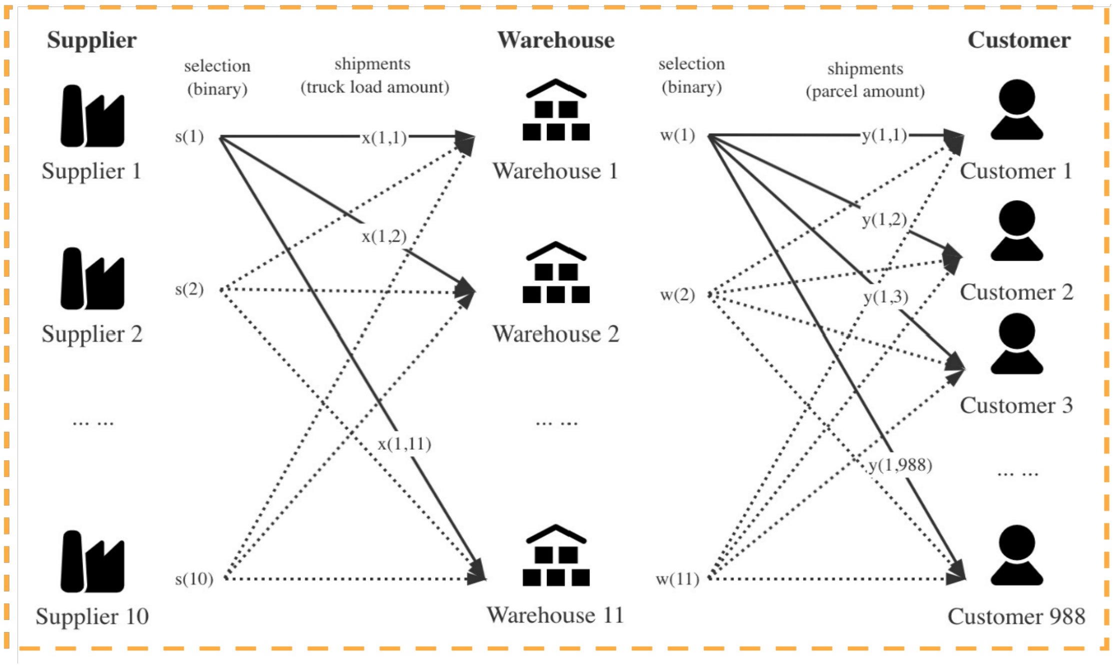
We use Exploratory data analysis to analyze datasets to summarize main characteristics. There are two graphs shown below. Using the folium library, we visualize the geospatial data that's been manipulated in Python on an interactive leaflet map. The first graph shows the distribution of suppliers and warehouses. It interprets that the suppliers and warehouses are mainly distributed in the big cities on the east coast, southeast, and west coast. Some of the suppliers and warehouses are relatively close to each other. We also visualized customer demands using Mapbox, a density heatmap in Python. In the second graph, there is a high customer demand density in the middle east and west. We especially discovered that high demand is focused on the east and west coasts, with New York, California, and Texas being the top three states in the United States with the highest customer demand.





Method

Variables definition



In our model, we have 4 variable matrices, including two shipment amount matrices and two binary matrices. For the shipment amount matrix from suppliers to warehouses, we set the matrix to be X and denote the i-th supplier to j-th warehouse to be x_{i,j}, which means the dimension of matrix X is 10*11. The other shipment amount matrix is from warehouse to customers and we set it to be Y. Similarly denote the j-th warehouse to k-th customer to be y_{j,k}, which means the dimension of matrix Y is 988*11. For the binary matrices, we use them to select the supplier and warehouse. We denote the supplier selection binary matrix to be S with dimension of 10*1 and all elements are either 0 or 1. If s_{i} is 1, it means we select this supplier to send shipment to the warehouse. Otherwise, we will not count the fixed cost of this supplier. We denote the warehouse selection binary matrix to be W with dimension of 11*1 and all elements are either 0 or 1. If w_{j} is 1, it means we select this warehouse to send shipment to customers. Otherwise, we will not count the fixed cost of this warehouse.

Then we have 7 constant matrices including two fixed cost matrices, two variable cost matrices, two rate matrices and one demand matrix:

- The first one is **supplier fixed cost** that grabs them from the supplier tab (column B) in the Excel file: **FC(i) where i∈[1, 10]**.
- The second parameter is **supplier variable cost per shipment** from the supplier sheet (column C): **VC(i) where i∈[1, 10]**.

- Then, our function has **supplier-to-warehouse rates per shipment** from Supplier-WHs -Rates sheet: **R(i,j) where i∈[1, 10], j∈[1, 11]**.
- The fourth one is the warehouse **fixed cost** from the warehouse tab(column B): **FC(j) where j∈[1,11]**.
- The fifth parameter is the **warehouse-to-customer rates per shipment** from WHs- Customers-Rates sheet: R(j,k) where j∈[1, 11], k∈[1, 988]
- Next, our function contains **warehouse-to-customer rates per shipment** from WHs- Customers-Rates tab: R(j,k) where j∈[1, 11], k∈[1, 988]
- Our last parameter is **customer demand** from the Customers tab (Column D): D(k) where k∈[1, 988]

All data required for constant matrices are given in excel.

Objective function:

The basic concept of our objective function is the sum of demand times variable cost plus the fixed cost. So the first thing we need to figure out is the variable cost. We started on the first half trip: supplier to warehouse part. The variable cost for this trip includes the variable cost in supplier and in the shipment trip. So VC(i)+R(i,j) is the total amount of variable cost from i-th supplier to j-th warehouse per shipment. Since the matrix X is the demand matrix explained above, the total variable cost in this part is . For the warehouse to customer part, we have the similar steps, and the total variable cost is.

For the fixed cost, we time the supplier fixed cost matrix and the binary supplier selection matrix and get the sum of the product, which is the total fixed cost of suppliers. Then the sum of warehouse fixed cost is the sum of the warehouse fixed cost matrix times the binary warehouse selection matrix. To write in a mathematical way, we say the fixed cost is: +

Therefore since our research is to minimize the total cost, we have the objective function to be:

$$\min \sum_{i=1}^{10} \sum_{j=1}^{11} x(i,j) * (VC(i) + R(i,j)) + \sum_{j=1}^{11} \sum_{k=1}^{988} y(j,k) * (VC(j) + R(j,k)) + \sum_{i=1}^{10} FC(i) * s(i) + \sum_{j=1}^{11} FC(j) * w(j)$$

Constraints:

1. *Customer demand constraint*

We need to guarantee each demand for 988 customers. Then we have the first constraint to make sure the goods each customer received from all warehouses($\sum_{\square=\square}^{\square\square}$ **y(j,k)** where j is the number of warehouse) should be equal or larger than its demand (D_k where k is the number of customers). Based on this constraint, we have 988 formulas which can be concluded as:

$\sum_{j=1}^{11}$ **y(j,k)** $\geqq$ **D(k)**, where k$\in$ [1,2, … ,988]

2. *Warehouse supply constraint*

This constraint is meant to guarantee that each warehouse has enough supply to satisfy needs.
So for each warehouse out of 11, goods received from all suppliers($\sum_{\square=\square}^{\square\square}$ **x(i,j)** where i is the number of suppliers) should be equal or larger than the goods delivered to all customers($\sum_{\square=\square}^{\square\square\square}$ **y(j,k)/1000** where k is the number of customers). The second term should be divided by 1000 because the parcel to shipment ratio is 1000. Based on this constraint, we have 11 sub constraints which can be concluded as:

$\sum_{i=1}^{10}$ **x(i,j)** $\geqq$ $\sum_{k=1}^{988}$ **y(j,k)/1000**, where j$\in$ [1,11]

3. *Supplier selection constraint*

If a supplier has a shipment out of it ($x \neq 0$), then this supplier must be selected (s =1), which means the fixed cost for this supplier must occur. Based on this constraint, we have 10 sub constraints which can be concluded as:

0.00001* S(i) $\leq$ $\sum_{\square=\square}^{\square\square}$ **x(i,j)** $\leq$ **100000* S(i)**,where i$\in$ [1,10]

We use parameters 0.00001 and 100000 to guarantee the interactive relationship between total shipment amount from supplier to warehouse ***x*** and selection choice of supplier ***s***. When total shipment amount from one supplier to all warehouse is 0 ($\sum_{\square=\square}^{\square\square}$ **x(i,j)= 0)** , the inequation will force S(i) = 0. It means for this specific supplier, if there is no outbound shipment amount, we will not choose this supplier and fixed cost for this supplier will not occur either. Likewise, when total shipment amount from one supplier to all warehouse is not 0 ($\sum_{\square=\square}^{\square\square}$ **x(i,j)**$\neq$0) , the inequation will force S(i) = 1.

4. *Warehouse selection constraint*

If a warehouse has a shipment out of it ($y \neq 0$), then this warehouse must be selected ($w = 1$), which means the fixed cost for this supplier must occur. Based on this constraint, we have 10 formulas which can be concluded as:

0.00001* W(j) $\leq \sum_{\square=\square}^{\square\square\square}$ y(j,k) $\leq$ **100000* W(j) ,** where $j \in [1,11]$

Similar to constraint 3, we use parameters 0.00001 and 100000 to guarantee the interactive relationship between total shipment amount from warehouse customer ***y*** and selection choice of warehouse ***w***.

5. *Integer constraint*

Total shipments and total parcels should be integer.

x(i,j), y(j,k) are integer ,where i[1,2, … ,10], j[1,2, … ,11], k[1,2, … ,988]

6. *Binary constraint*

Selection of suppliers and selection of warehouses are binary.

s(i), w(j) are binary, where

$$s(i) = \begin{cases} 1, \text{ if this supplier is selected} \\ 0, \text{ if this supplier is not selected} \end{cases}$$

$$w(j) = \begin{cases} 1, \text{ if this warehouse is selected} \\ 0, \text{ if this warehouse is not selected} \end{cases}$$

So the linear programming formula is:

$$
\begin{array}{ll}
\text{Minimize} & \sum_{i=1}^{10}\sum_{j=1}^{11} x(i,j) * (VC(i) + R(i,j)) + \sum_{j=1}^{11}\sum_{k=1}^{988} y(j,k) * (VC(j) + R(j,k)) + \\
& \sum_{i=1}^{10} FC(i) * s(i) + \sum_{j=1}^{11} FC(j) * w(j) \\
\text{Subject to} & x(i,j), y(j,k) \in Z, \qquad \forall i \in 1,2,\dots,10,\ \ \forall j \in 1,2,\dots,11,\ \ \forall k \in 1,2,\dots,988 \\
& s(i), w(j) \in (0,1), \qquad \forall i \in 1,2,\dots,10,\ \ \forall j \in 1,2,\dots,11 \\
& \sum_{i=1}^{10} x(i,j) \geqq \sum_{k=1}^{988} y(j,k)/1000, \qquad \forall j \in 1,2,\dots,11 \\
& \sum_{j=1}^{11} y(j,k) \geqq D(k), \qquad \forall k \in 1,2,\dots,988 \\
& 100000 * s(i) \geqq \sum_{j=1}^{11} x(i,j) \geqq 0.00001 * s(i), \qquad \forall i \in 1,2,\dots,10 \\
& 100000 * w(j) \geqq \sum_{k=1}^{988} y(j,k) \geqq 0.00001 * w(j), \qquad \forall j \in 1,2,\dots,11
\end{array}
$$

Implementation

We plan to implement our objective function and solution in two approaches including Excel Solver and Python and figure out the most efficient and pragmatic method.

Excel Solver

Excel solver is one of the common tools of solving optimization problems. We have also tried to implement our objective function in Excel Solver. We have reduced the number of suppliers, warehouses and customers to simplify the problem and tested it in Solver and got the solution successfully. Besides, even for a simplified version, it takes Solver over 10 minutes to finish computing which means it's almost impossible to scale the problem to its original version.
In addition, there are too many intermediate variables (over 10,000) involved in the computing process which cannot be handled by Solver due to software functionality limitations. Thus, barriers including excessive variable number and computational complexity push us to new implementation approaches like Python.

Use IBM Decision Optimization CPLEX Modeling for Python

***i. Dataset Set-up*¶**

To extract the datasets from the raw data file, we use pandas to convert each chart into data frames. We reorganize data structure, tidy data, and customize data for further analysis and computing.
First, the total variable cost consists of the shipping cost from the supplier to the warehouse, the variable cost of the supplier, the shipping cost from the warehouse to the customer, and the

variable cost of the warehouse. The resulting dataset for the shipping cost from the supplier to the warehouse is noted as a 10x11 matrix, where columns are suppliers and rows are warehouses. In order to add the variable cost of suppliers together with the shipping cost, we convert the variable cost of suppliers into a 10x11 matrix as well, where rows are suppliers and the values are the same for each row. The same operations are applied to the shipping cost from the warehouse to the customer and the variable cost of the warehouse. They are both 988x11 matrices, where columns are warehouses and rows are customers. Second, the total fixed cost consisted of the fixed cost of the supplier and the fixed cost of the warehouse. The fixed cost of the supplier is a 1x10 matrix while the fixed cost of the warehouse is a 1x11 matrix, where each value is the fixed cost for a specific supplier/warehouse.
We used cplex to solve this linear optimization problem, and in total, we add 10999 decision variables, 12019 constraints, and an objective function into the model, and ask the computer to give out the final result.

ii. Optimization Set-up¶

Given the predefined objective functions with constraints, the prescription model is set up using the Python cplex library, with the imported Model class from docplex.mp. Then four decision variables are generated, with the range of the supplies from 1 to 11, warehouses from 1 to 12, and the customers from 1 to 988. Suppliers selection s(i) and warehouses selection w(j) are generated as two binary variables with opt_model.binary_var; and quantity of truckloads from suppliers to warehouses x(i, j), and the parcel amount from warehouses to customers y(j, k) are generated as two integers with opt_model.integer_var. Next, the constraints are added with the customer demand that "S-WH shipment >= 0", "WH-C shipment >= 0", "WH: 0.00001* w(j) <= sum(y(j, k)) <= 1000000 * w(j)", "S: 0.00001* s(i) <= sum(x(i, j)) <= 1000000 * s(i)", "Warehouse inbound > outbound", "Customer inbound >= demand". Finally, the objective function is computed for minimization by opt_model.minimize, and the model is solved by opt_model.solve().
The best optimal result for supplier selection is supplier 2 (Phoenix AZ) and supplier 5 (Columbus OH). The result for warehouse selection is warehouse 1 (Newark), warehouse 2 (Atlanta), warehouse 3 (Dallas), warehouse 5 (Los Angeles), and warehouse 6 (Chicago). There are five combinations of suppliers and warehouses, which are (2, 5), (5, 1), (5, 2), (5, 3), and (5, 6). There are 991 combinations of warehouse and customer shipment, which are (1, 1), (1, 6), (1, 9), (1, 10), (1, 19) … (6, 965), (6, 966), (6,

967), (6, 973), (6, 979). With the optimized solution of DHL network design, the total minimized price is $16,037,400.