A Shipment Problem

Problem Statement

Apple is releasing several new products: the iThing1, iThing2, ..., iThing15. Pre-orders have come flooding in from across the world, and Apple wants to fulfill these pre-orders in the most cost-effective way possible. Apple has multiple warehouses worldwide (W1, W2, ..., W10) where the products are stocked. Apple needs to determine the optimal shipping plan - i.e., the number of units of each product to be shipped from each warehouse to each customer.

Data

The following data are given:

- The number of pre-ordered units of each product by each customer (*pre_order_matrix.xlsx*)
- The stock of each product at each warehouse (*stock_matrix.xlsx*)
- The distance from each warehouse to each customer (*distance_matrix.xlsx*)

For the purposes of our model, we'll consider the cost of shipping to be directly proportional to the distance from the warehouse to the customer. Thus, we can consider the distance as our cost matrix.

Please formulate this model and help Apple find the optimal shipment scenario using Gurobi.

Linear Program

Sets and parameters:

$W = \{1, 2, ..., 10\}$: Set of warehouses.

$C = \{1, 2, ..., 859\}$: Set of customers.

$P = \{1, 2, ..., 15\}$: Set of products.

s_{ij} : stock of product j at warehouse i.

d_{ik} : distance between warehouse i to customer k.

q_{kj} : quantity of product j pre-ordered by customer k.

Decision Variables:

x_{ijk} : quantity of product j to be shipped from warehouse i to customer k

Objective Function:

Minimize the total shipping cost. The cost of shipping a product from a warehouse to a customer is assumed to be directly proportional to the distance from the warehouse to the customer.

$$Minimize \sum_{i \in W} \sum_{k \in C} \sum_{j \in P} d_{ik} \cdot x_{ijk}$$

A Shipment Problem

Constraints:

1. **Stock Constraints:** These constraints ensure that we do not ship more units of a product than are available in each warehouse.

$$\sum_{k \in C} x_{ijk} \leq s_{ij}, \forall i \in W, j \in P$$

2. **Pre-order constraints:** These constraints ensure that the total quantity of each product shipped to each customer matches their pre-order quantity.

$$\sum_{i \in W} x_{ijk} = q_{kj}, \forall k \in C, j \in P$$

3. **Non-negativity Constraints:** These constraints ensure that we cannot ship a negative number of units.

$$x_{ijk} \geq 0, \forall i \in W, k \in C, j \in P$$

The decision variables (x_{ijk}) can also be constrained to be integers if you wish the solution to specify whole numbers of food units. This would make the problem an integer linear program (ILP).

Gurobi Solution

Import libraries

```
import gurobipy as gp
from gurobipy import GRB
import pandas as pd
import numpy as np
import random

import re

import seaborn as sns
import matplotlib.pyplot as plt
```

Read data from Excel:

```
pre_order_matrix = pd.read_excel("pre_order_matrix.xlsx", index_col=0)
stock_matrix = pd.read_excel("stock_matrix.xlsx", index_col=0)

# Dynamically find the number of warehouses based on the number of
    distance_matrix files
warehouses = ["W" + str(i) for i in range(1, len(stock_matrix.index) + 1)]
distance_matrix = {w: pd.read_excel("distance_matrix_" + w + ".xlsx",
    index_col=0) for w in warehouses}

# Dynamically find the number of customers and products based on
    pre_order_matrix's size
num_customers, num_products = pre_order_matrix.shape
print("Number of customers: ", num_customers)
print("Number of products: ", num_products)

# Dynamically find the number of warehouses based on cost_matrix's size
num_warehouses = len(distance_matrix)
print("Number of warehouses: ", num_warehouses)
```

A Shipment Problem

Convert data frames to dictionaries for Gurobi model:

```
pre_order_dict = pre_order_matrix.stack().to_dict()
stock_dict = stock_matrix.stack().to_dict()
distance_dict = {w: distance_matrix[w].stack().to_dict() for w in
    warehouses}
```

Model the problem and optimize it:

```
# Create a new model
m = gp.Model("Apple_preorder_distribution")

# Create variables
x = m.addVars(warehouses, pre_order_matrix.index, pre_order_matrix.columns,
     vtype=GRB.INTEGER, name="x")

# Set objective
m.setObjective(gp.quicksum(x[w, c, p]*distance_dict[w][(c, p)] for w in
    warehouses for c in pre_order_matrix.index for p in pre_order_matrix.
    columns), GRB.MINIMIZE)

# Add stock constraints
m.addConstrs((gp.quicksum(x[w, c, p] for c in pre_order_matrix.index) <=
    stock_dict[(w, p)] for w in warehouses for p in pre_order_matrix.columns
    ), "stock")

# Add pre-order constraints
m.addConstrs((gp.quicksum(x[w, c, p] for w in warehouses) == pre_order_dict
    [(c, p)] for c in pre_order_matrix.index for p in pre_order_matrix.
    columns), "preorder")

# Optimize model
m.optimize()
```

Create a dictionary to hold results:

```
results = {(w, p): 0 for w in warehouses for p in pre_order_matrix.columns}

# Iterate over the variables and collect the results
for v in m.getVars():
    if v.x > 0:
        w, c, p = v.varName.split('[')[1].split(']')[0].split(',')
        w = w.strip('\'')
        p = p.strip('\'')
        results[(w, p)] += v.x
```

Transform results into a DataFrame:

```
results_df = pd.DataFrame.from_dict(results, orient='index', columns=['
    Shipment'])
results_df.index = pd.MultiIndex.from_tuples(results_df.index)
results_df.index.names = ['Warehouse', 'Product']

# Create pivot table
pivot_df = results_df.reset_index().pivot(index='Warehouse', columns='
    Product', values='Shipment').fillna(0)

# Print table
display(pivot_df)
```

A Shipment Problem

A *"fancier"* way of presenting your results:

```
def fmt(x):
    return "{:03d}".format(int(x))

plt.figure(figsize=(20, 10))

# Create a 2D array of annotations
annotations = np.vectorize(fmt)(pivot_df.values)

sns.heatmap(pivot_df, cmap='YlGnBu', linewidths=.5, annot=annotations, fmt=
    "", annot_kws={"size": 10})

plt.title('Product Distribution from Each Warehouse', fontsize=20)
plt.xlabel('Product', fontsize=15)
plt.ylabel('Warehouse', fontsize=15)

plt.show()
```