THE AMAZOFF COMPANY

Logistics Plan
MA424 Modeling in Operations Research
Final Project 2019-2020

Candidate Number: 33815

Executive Summary

In order to help the Amazoff Company make better logistics plan and find reasonable operational policy that can minimize their potential cost / increase their potential revenue, I developed serval mathematical programming and simulation models aiming at solving warehousing, routing and e-commerce problems of the company.

For question 1, given the generated data of 15 potential warehouses' location and 60 clients' location, my *basic warehousing model* suggests that the company should only open **warehouse 8** (starts from 0) and assign all the clients to this warehouse only considering the cost of distance. In this way, the company are expected to minimize their total warehousing cost, which is **57.049.**

Based on the *basic warehousing model*, I take the capacity of each warehouse into account and built a *capacitated warehousing model*. The modified model gives us an optimal solution that we should open **warehouse 0,8 and 9** (the number of clients assigned to them is **20, 27, 13** respectively), in this case, the total cost of warehousing is **80.768**.

Moreover, in order to maintain the difference between the warehouses facing the highest and lowest demand not exceeding a particular threshold (set to 10 in my implement), a *balanced warehousing model* was set up to meet this requirement. According to the optimal solution of this model, I advise that the company should still open **warehouse 0,8 and 9** but do a slight change of assignment between warehouse 8 and 9 to deal with this situation, that is, assign the same **20 clients to warehouse 0 but assign 25 and 15 clients to warehouse 8 and 9** respectively. With this assignment, the total warehousing cost is minimized to **81.436**.

In question 2, a *single vehicle routing model* was firstly designed to produce the optimal routing plan aiming at delivering products from depot to 25 clients with highest demand in the cheapest way. The result of this basic model indicates that the vehicle can serve all the 25 clients with the total routing cost of **8.4797** following the optimal route. Detailed routing can be found in the 2nd sheet of 2.0 Vehicle Routing_result.xlsx, the position variables u have been sorted to indicate the order of the routing point.

The single vehicle routing model can be generalized to *multiple vehicles routing model* given the company has 3 vehicles to perform the dispatching operation. The total routing cost will slightly increase to **10.7164** but the routing plan will be more efficient (time-saving etc.) because 3 vehicles will serve a subset of the 25 clients in this situation. Detailed routing can be found in the 2nd sheet of 2.1 Multiple Vehicle Routing_result.xlsx, the first index of u indicates the clients and the second indicates the vehicles number, u have been sorted to show the routing order.

Taking the results of question 1 into account (how the clients are assigned to different warehouses in the balanced situation), assignment results can serve as a part of the input data (2.2 input data from 1.2.xlsx) to the previous model. In this case, vehicles belong to different warehouses will only visit the clients that are assigned to its warehouse in question 1 in the most efficient way. Total cost will increase to **14.7057** in this setting and the detailed routing can be found in2.2 Multiple Vehicles and Warehousing Routing_result.xlsx sheet 2 in in the same way.

We are supposed to consider two new clients that should be routed with priority based on question 2 but with different distance measurement. *Single vehicle routing with special orders model* was built to solve this problem and the optimal solution suggests that the vehicles departed from depot should firstly visit the new client 1 and then the new client 2 before visiting the other 25 clients in question 2, details can be checked in 3 Routing Special Orders_result.xlsx. In addition, the total routing cost is **11.868** under this setting.

For question 4, a mathematical model was built to find a matching between 10 internships position and 30 applicants that can cover each position in an ideal way (i.e. taking both the positions' and applicants' preference into account and ensure the matching is stable.). The total cost is **10** indicating that there are 10 pairs (internship and applicant) in the ideal matching, which fulfill the company's requirement. Matching result can be found in the next part (management report) or in the 4 Internships_result.xlsx file.

In the last question, I propose that the company should make full use of the historical data and make reasonable assumptions to estimate the uncertain demand and total revenue (maximize the total revenue is our goal) in the upcoming months. Basically, the result will mostly depend on the samples of the demand as well as other data we have and the assumptions we make. The most important assumption I've made is

that the historical demand is monthly data (i.e. we have previous demand data of each month in years), with this assumption, I also implement a sample model assuming the company have 3 different products and unlimited capacity to store all the orders to help us analyze how the total revenue changes. In my setting, the total revenue is found to fluctuate between **6500 and 8000**, approximately 23%. Details will be discussed in the following sections.

Management Report

In the warehousing part, we are supposed to consider the facility location problem which takes assignment cost and facility open cost into account. Our *basic warehousing model* suggests us to only open warehouse 8 without considering capacity and other factors. This is because the opening cost of a facility ($100 \times 3^{-\|f\|_1}$) is relatively large compared with the assignment cost (ℓ_1 distance between two points in the space of $[-1,1] \times [-1,1]$, therefore, our model will tend to choose a facility with lowest opening cost. Actually, if we check the opening cost matrix of 15 potential facilities, it is not surprisingly to find that warehouse 8 has the lowest opening cost, which is 17.78 based on the generated data.

```
array([[22.9473303 ],
       [27.70949013],
       [57.154513  ],
       [42.48354031],
       [24.59518131],
       [23.79132883],
       [27.74663096],
       [26.96291024],
       [17.77940099],
       [18.81242089],
       [25.28579539],
       [53.22285998],
       [36.81753143],
       [39.26519872],
       [41.81795324]])
```

Figure 1 Opening cost of 15 potential facilities

Furthermore, I've tried to set 10 different seed to generate the location data and find our model will always allocate the warehouse with lowest opening cost to all the clients in order to minimize the total warehousing cost.

The situation maintains even when we need to take the warehouses' capacity and clients' demand into account. As our *capacitated warehousing model* suggests, warehouse 0, 8 and 9 should be chosen to meet the total demand of the clients. After checking the same opening cost in figure 1, we can easily find that they are three warehouses with lowest opening cost. It is thus reasonable to think the model will firstly pick as least warehouses as possible, then based on the warehouses with cheapest opening cost it chose, the model allocates their neighbor clients to them to minimize the total cost. This also makes sense in practice where the company will consider the main part of the total cost with priority when making decision, e.g. factory will be built close to places that can easily access to raw materials.

Then balance between warehouses is taken into account. From the previous part we know that warehouse 8 has the highest demand (27 clients are allocated to it) and warehouse 9 has the lowest demand (13 clients) and the *balanced warehousing model* aims at maintain the difference of demand (total demand of the clients assigned to each warehouse) under a certain value. Therefore, the optimal solution indicate that we should allocate 25 clients to warehouse 8 (2 less) and allocate 15 clients to warehouse 9 (2 more) to satisfy the requirement with threshold 10 (units). It is worth noting that the total cost increase from 80.768 to 81.436, this is because we add an extra constraints to the previous *capacitated*

warehousing model whose objective value is the lower bound of the objective value of *balanced warehousing model*. The total cost only change 0.8% because the result does not change a lot and the reallocation cost is relatively small.

Allocation results of *capacitated warehousing model* and *balanced warehousing model* are shown below.

Warehous 0			Warehous 8			Warehous 9		
	x[0, 5]	1		x[8, 1]	1		x[9, 0]	1
	x[0, 10]	1		x[8, 2]	1		x[9, 7]	1
	x[0, 11]	1		x[8, 3]	1		x[9, 17]	1
	x[0, 15]	1		x[8, 4]	1		x[9, 18]	1
	x[0, 16]	1		x[8, 6]	1		x[9, 24]	1
	x[0, 19]	1		x[8, 8]	1		x[9, 28]	1
	x[0, 21]	1		x[8, 9]	1		x[9, 37]	1
	x[0, 23]	1		x[8, 12]	1		x[9, 38]	1
	x[0, 25]	1		x[8, 13]	1		x[9, 39]	1
	x[0, 26]	1		x[8, 14]	1		x[9, 44]	1
	x[0, 27]	1		x[8, 20]	1		x[9, 45]	1
	x[0, 30]	1		x[8, 22]	1		x[9, 48]	1
	x[0, 31]	1		x[8, 29]	1		x[9, 53]	1
	x[0, 35]	1		x[8, 32]	1			
	x[0, 36]	1		x[8, 33]	1			
	x[0, 41]	1		x[8, 34]	1			
	x[0, 43]	1		x[8, 40]	1			
	x[0, 50]	1		x[8, 42]	1			
	x[0, 56]	1		x[8, 46]	1			
	x[0, 58]	1		x[8, 47]	1			
				x[8, 49]	1			
				x[8, 51]	1			
				x[8, 52]	1			
				x[8, 54]	1			
				x[8, 55]	1			
				x[8, 57]	1			
				x[8, 59]	1			

Figure 2 Results of capacitated model

Warehous 0			Warehous 8			Warehous 9		
	x[0, 5]	1		x[8, 1]	1		x[9, 0]	1
	x[0, 10]	1		x[8, 2]	1		x[9, 6]	1
	x[0, 11]	1		x[8, 3]	1		x[9, 17]	1
	x[0, 15]	1		x[8, 4]	1		x[9, 18]	1
	x[0, 16]	1		x[8, 7]	1		x[9, 28]	1
	x[0, 19]	1		x[8, 8]	1		x[9, 32]	1
	x[0, 21]	1		x[8, 9]	1		x[9, 37]	1
	x[0, 23]	1		x[8, 12]	1		x[9, 38]	1
	x[0, 25]	1		x[8, 13]	1		x[9, 39]	1
	x[0, 26]	1		x[8, 14]	1		x[9, 44]	1
	x[0, 27]	1		x[8, 20]	1		x[9, 45]	1
	x[0, 30]	1		x[8, 22]	1		x[9, 48]	1
	x[0, 31]	1		x[8, 24]	1		x[9, 51]	1
	x[0, 35]	1		x[8, 29]	1		x[9, 53]	1
	x[0, 36]	1		x[8, 33]	1		x[9, 54]	1
	x[0, 41]	1		x[8, 34]	1			
	x[0, 43]	1		x[8, 40]	1			
	x[0, 50]	1		x[8, 42]	1			
	x[0, 56]	1		x[8, 46]	1			
	x[0, 58]	1		x[8, 47]	1			
				x[8, 49]	1			
				x[8, 52]	1			
				x[8, 55]	1			
				x[8, 57]	1			
				x[8, 59]	1			

Figure 3 Results of balanced model

In the vehicle routing part, we firstly need to select the 25 clients with highest demand from the 60 clients in the previous part, which have been labeled into red in figure 3. In the one vehicle setting, the vehicle will depart from the depot (whose index is 0, coordinate is (1,1)) and visit all the 25 clients and lastly come back to the depot in the most efficient way to minimize the total traveling cost. Result can be found in figure 4 and more modelling detail will be discussed in the technical appendices. (Notice: the indices of the 25 clients do not follow the original index of the 60 clients, to find their mapping, you can check the *HDC_ind* variable in 2.0 Vehicle Routing.py)

	A	B
1	u[0]	1
2	u[21]	2
3	u[18]	3
4	u[23]	4
5	u[13]	5
6	u[25]	6
7	u[1]	7
8	u[17]	8
9	u[9]	9
10	u[11]	10
11	u[14]	11
12	u[6]	12
13	u[4]	13
14	u[12]	14
15	u[20]	15
16	u[19]	16
17	u[5]	17
18	u[16]	18
19	u[10]	19
20	u[3]	20
21	u[8]	21
22	u[15]	22
23	u[7]	23
24	u[22]	24
25	u[2]	25
26	u[24]	26

Figure 4 Results of single vehicle routing

As figure4 shows, the vehicle will depart from point 0 (depot) and visit client 21 → client18 … , after visiting the last client, client 24, it will be back to point0.

In the three vehicles setting, three vehicles will all depart from depot and each of them is going to visit a part of the clients. It is notable that the total routing cost of three vehicles is 10.7164, larger than the total cost in the one vehicle setting, which is 8.4797. This can be caused by the extra routing cost from depot to the first client and from the last client back to the depot since here we write constraints to force all of the three vehicles have to visit part of the clients. Routing result in this situation is shown below.

22	Vehicle 0		Vehicle 1		Vehicle 2	
23		u[5, 0]		u[24, 1]		u[2, 2]
24		u[16, 0]				u[22, 2]
25		u[19, 0]				u[7, 2]
26		u[20, 0]				u[15, 2]
27		u[12, 0]				u[3, 2]
28		u[10, 0]				u[8, 2]
29		u[4, 0]				
30		u[6, 0]				
31		u[14, 0]				
32		u[11, 0]				
33		u[9, 0]				
34		u[17, 0]				
35		u[1, 0]				
36		u[25, 0]				
37		u[13, 0]				
38		u[23, 0]				
39		u[18, 0]				
40		u[21, 0]				

Figure 5 Results of multiple vehicles routing

Multiple vehicles and warehousing routing model takes the results of warehousing into account, on the basis of *multiple vehicles routing model*, the new model runs the routing for each vehicle based on the assignment of clients to its warehouse (how we input the data from previous part will be illustrated in technical appendices). Result is shown in figure 6.

	A	B	C	D	E	F
1	Warehouse 0		Warehouse 8		Warehouse 9	
2		u[21, 0]		u[3, 1]		u[13, 2]
3		u[18, 0]		u[1, 1]		u[25, 2]
4		u[23, 0]		u[6, 1]		u[17, 2]
5		u[8, 0]		u[4, 1]		u[9, 2]
6		u[15, 0]		u[5, 1]		u[11, 2]
7		u[7, 0]				u[14, 2]
8		u[22, 0]				u[12, 2]
9		u[2, 0]				u[10, 2]
10		u[24, 0]				u[20, 2]
11						u[19, 2]
12						u[16, 2]

Figure 6 Results of multiple vehicles and warehousing routing

It is necessary to notice that the second index of variable u indicate the number of vehicle and the mapping between it and warehouse is: vehicle 0 --- warehouse 0, vehicle 1 --- warehouse 8, vehicle 2 --- warehouse 9. And as previous part, the first index points to the 25 clients and values of u(s) have been sorted to indicate the routing order (we'll talk about the meaning of u in the next section). The total cost increase to 14.7057 in this case because some of the variables are pre-set instead of being found to be optimal as in the previous model. But the result is more reasonable since practically the company is more likely to perform routing operations on the basis of warehousing and it is even impossible to only assign one client to a vehicle as results shown in figure5, which informs that we will consider more factors and complicated situation in the real world.

When some special orders (new clients) are supposed to be routed before visiting the remaining 25 clients in the last part, new constraints should be added to our *single vehicle routing model*. In this situation, we can consider that the vehicle should firstly find the cheapest route in the graph of depot and special clients, and then find the cheapest route in the graph of the normal clients, meanwhile, the model should also take the connection cost between these two graphs into account. The idea is clear while we will find that the model with these constraints cannot be solved directly (should be talk in detail in technical appendices). Then Heuristic method can be considered as an alternative. Specifically, routing cost between depot and special clients and cost among special clients could be set to relatively small (e.g. the original cost / 100) manually and we restore the original cost to the objective value after getting the result. With this idea, it is easy to force the *single vehicle routing model* visit the special clients with priority in order to minimize the total routing cost. The model output is shown in figure 7.

	A	B	C	D	E	F	G	H	I
1	u[0]	1							
2	u[1]	2							
3	u[2]	3		The original 25 clients indices		Result of question 2		Change the measurement in Q3 to 12 norm	
4	u[15]	4			u[13]		u[21]		u[13]
5	u[3]	5			u[1]		u[18]		u[1]
6	u[19]	6			u[17]		u[23]		u[17]
7	u[11]	7			u[9]		u[13]		u[9]
8	u[13]	8			u[11]		u[25]		u[11]
9	u[16]	9			u[14]		u[1]		u[14]
10	u[8]	10			u[6]		u[17]		u[6]
11	u[6]	11			u[4]		u[9]		u[4]
12	u[12]	12			u[10]		u[11]		u[10]
13	u[14]	13			u[12]		u[14]		u[12]
14	u[21]	14			u[19]		u[6]		u[20]
15	u[22]	15			u[20]		u[4]		u[19]
16	u[18]	16			u[16]		u[12]		u[5]
17	u[7]	17			u[5]		u[20]		u[16]
18	u[24]	18			u[22]		u[19]		u[22]
19	u[4]	19			u[2]		u[5]		u[2]
20	u[9]	20			u[7]		u[16]		u[7]
21	u[17]	21			u[15]		u[10]		u[15]
22	u[5]	22			u[3]		u[3]		u[3]
23	u[10]	23			u[8]		u[8]		u[8]
24	u[27]	24			u[25]		u[15]		u[25]
25	u[23]	25			u[21]		u[7]		u[21]
26	u[20]	26			u[18]		u[22]		u[18]
27	u[25]	27			u[23]		u[2]		u[23]
28	u[26]	28			u[24]		u[24]		u[24]

Figure 7 Results of routing special orders

It is shown that the vehicle will firstly visit special client 1 and then special client 2 after departure from depot, then it will visit client 15 (which is client 13 in the 25 clients set, since here we add 2 clients into the combined client set), client 3…client 26 one by one before coming back to depot. If we compare the routing path with the previous *single vehicle routing model*, it is worth noting the two model choose to visit some series of clients in the same order (colored in blue in the figure, e.g. 1→17→9→11→6→4) while the two routes are not exactly the same in these two models, this can result from two reasons:

1) Starting point of visiting the 25 clients in special orders is one of the special clients' location instead of the depot, which can influence the route.
2) Distance measurements are different in the two models, ℓ_2 in *single vehicle routing model* but ℓ_1 in special orders model. Actually, I've tried to change the distance norm back to ℓ_2 in this part and compared their result then I found that the route is closer to *single vehicle routing model* in this case (colored in red, if we use ℓ_1 norm the vehicle will visit 19→20→ 16→5 while the solution changes to 20→19→ 5→16 with ℓ_2 norm).

In the part of Internships, stable matchings model is designed to ensure 'matching' and 'stability' between 10 internships position and 30 applicants while respecting both internships' and applicants' preference (full ranking list that is generated randomly) and covering all the internships position. Specifically, a matching is stable when there does not exist any match (*Internship*, *Applicant*) which both prefer each other to their current partner under the matching. Modelling detail will also be discussed in the next section and result of the model is shown below, it also shows that there's one unique stable matching under this situation.

Table 1 Result of stable matchings

Internship	Applicant
0	26
1	18
2	15
3	24
4	2
5	29
6	25
7	6
8	14
9	9

E-commerce company does not always know how much demand it will face in the future but at the same time it has to have an approximation about how much stock they should prepare in advance or the revenue variability to manage the demand uncertainty and reduce the supply chain risk. Aiming at helping the company maximize their potential total revenue and giving optimal plan for placing orders and inventory in the following months under the uncertain demand, I make the following suggestions based on serval assumptions and stochastic programming (will be discuss in technical appendices):

1) Take the full advantage of the historical data. Historical data can be utilized to approximate the stochastic optimization model (https://en.wikipedia.org/wiki/Stochastic_programming)and to do demand forecasting (https://en.wikipedia.org/wiki/Stochastic_programming).
2) Try to use resampling method and simulation (e.g. bootstrap sampling) to have better approximation on the demand distribution which can be beneficial to our stochastic optimization model.
3) Try to make use of different methods to model the demand uncertainty and total revenue and compare their result to get a more comprehensive approximation results. And it is advisable to utilize multivariate analysis, time series analysis etc. to figure out the most important factors that affecting the future demand.

4) Based on 1) 2) and 3), evaluate the range of optimal amount of placing orders and inventory in the following months and reserve reasonable capital to cope with the uncertain demand.

   Detailed assumptions and modelling method will be discuss in technical appendices

Technical Appendices

Specification

Programming language and version:
Python 3.6.8

Used libraries:
Gurobi 9.0.0
NumPy 1.16.5
Pandas 0.23.3

Notice:

- The first index of any data structure in python is 0 instead of 1.
- The python code files include reading and writing the data from/to excel file based on my computer file path, if you want to read the excel file or run the python code files successfully, please remember to modify the file path in the code files and make sure your environment meet the libraries' requirement.
- Data is generated by python *random* module and I've set the same seed to the same data in different code files for reproducibility. All the data are stored in array type in python environment for convenience in computation. You can check the data by simply print it in python console.

1 Warehousing

Basic facility location model is developed, the formulas are:

$$\boldsymbol{min} \quad \sum_{i\in F} \boldsymbol{f_i y_i} + \sum_{i\in F}\sum_{j\in C} \boldsymbol{c_{ij} x_{ij}} \qquad \textbf{(1.0)}$$

$$\boldsymbol{s.t.} \quad \sum_{i\in F} \boldsymbol{x_{ij}} = 1 \ \ \boldsymbol{for\ every\ j \in C} \qquad \textbf{(1.1)}$$

$$\boldsymbol{x_{ij}} \leq \boldsymbol{y_i} \ \ \boldsymbol{for\ every\ (i,j) \in F \times C} \qquad \textbf{(1.2)}$$

$$\boldsymbol{x_{ij}} \in \{\mathbf{0},\mathbf{1}\} \ \ \boldsymbol{for\ every\ (i,j) \in F \times C} \qquad \textbf{(1.3)}$$

$$\boldsymbol{y_i} \in \{\mathbf{0},\mathbf{1}\} \ \ \boldsymbol{for\ every\ i \in F} \qquad \textbf{(1.4)}$$

Assumption(s):

- The capacity of each warehouse is unlimited, i.e. we do not consider the capacity in this setting.

Parameters:
f_i: the cost of opening warehouse i (given by $100 \times 3^{-\|f\|_1}$)
c_{ij}: cost of assigning client j to facility i (ℓ_1 distance)

Variables:
y_i: binary, whether open warehouse i or not
x_{ij}: binary, whether assign client j to facility i or not

1.1 Capacitated Model

Clients' demand and facilities' capacity are considered in this setting, formulas are modified to:

$$\boldsymbol{min}\ \sum_{i\in F} \boldsymbol{f_i y_i} + \sum_{i\in F}\sum_{j\in C} \boldsymbol{c_{ij} x_{ij}} \qquad \textbf{(1.0)}$$

$$\boldsymbol{s.t.}\ \sum_{i\in F} \boldsymbol{x_{ij}} = 1\ \ \boldsymbol{for\ every\ j \in C} \qquad \textbf{(1.1)}$$

$$\sum_{j\in C} \boldsymbol{d_j x_{ij}} \leq\ \boldsymbol{u_i y_i}\ \ \boldsymbol{for\ every}\ (\boldsymbol{i,j}) \in \boldsymbol{F \times C} \qquad \textbf{(1.2.1)}$$

$$\boldsymbol{x_{ij}} \in \{\mathbf{0,1}\}\ \ \boldsymbol{for\ every}\ (\boldsymbol{i,j}) \in \boldsymbol{F \times C} \qquad \textbf{(1.3)}$$

$$\boldsymbol{y_i} \in \{\mathbf{0,1}\}\ \ \boldsymbol{for\ every\ i} \in \boldsymbol{F} \qquad \textbf{(1.4)}$$

Assumption(s):

- Each client has a demand that is uniformly distributed between [0,25].

Parameters:
f_i: the cost of opening warehouse i (given by $100 \times 3^{-\|f\|_1}$)
c_{ij}: cost of assigning client j to facility i (ℓ_1 distance)
u_i: capacity of warehouse i (given by $100 \times 2^{\|f\|_1}$)
d_j: demand of client j

Variables:
y_i: binary, whether open warehouse i or not
x_{ij}: binary, whether assign client j to facility i or not

1.2 Balanced Model

Balanced loads added into the constraints in this setting, formulas are modified to:

$$\boldsymbol{min}\ \sum_{i\in F} \boldsymbol{f_i y_i} + \sum_{i\in F}\sum_{j\in C} \boldsymbol{c_{ij} x_{ij}} \qquad \textbf{(1.0)}$$

$$\boldsymbol{s.t.}\ \sum_{i\in F} \boldsymbol{x_{ij}} = 1\ \ \boldsymbol{for\ every\ j \in C} \qquad \textbf{(1.1)}$$

$$\sum_{j\in C} \boldsymbol{d_j x_{ij}} \leq\ \boldsymbol{u_i y_i}\ \ \boldsymbol{for\ every}\ (\boldsymbol{i,j}) \in \boldsymbol{F \times C} \qquad \textbf{(1.2.1)}$$

$$\boldsymbol{y_k} \sum_{j\in C} \boldsymbol{d_j x_{ij}} \leq\ \boldsymbol{diff} +\ \boldsymbol{y_i} \sum_{j\in C} \boldsymbol{d_j x_{kj}}$$
$$\boldsymbol{for\ every}\ (\boldsymbol{i,j}), (\boldsymbol{k,j}) \in \boldsymbol{F \times C} \qquad \textbf{(1.5)}$$

$$\boldsymbol{x_{ij}} \in \{\mathbf{0,1}\}\ \ \boldsymbol{for\ every}\ (\boldsymbol{i,j}) \in \boldsymbol{F \times C} \qquad \textbf{(1.3)}$$

$$\boldsymbol{y_i} \in \{\mathbf{0,1}\}\ \ \boldsymbol{for\ every\ i} \in \boldsymbol{F} \qquad \textbf{(1.4)}$$

Assumption(s):

- Each client has a demand that is uniformly distributed between [0,25].

Parameters:
f_i: the cost of opening warehouse i (given by $100 \times 3^{-\|f\|_1}$)
c_{ij}: cost of assigning client j to facility i (ℓ_1 distance)
u_i: capacity of warehouse i (given by $100 \times 2^{\|f\|_1}$)
d_j: demand of client j

Variables:
y_i: binary, whether open warehouse i or not
x_{ij}: binary, whether assign client j to facility i or not

In order to maintain the model as linear programming, I try to avoid using max and min function. Instead, an equivalent constraints can be placed to meet the requirement, it comes from an intuitive idea: if the loads' difference between the warehouses facing the highest demand and lowest demand is under a certain threshold, then this difference within each pairs of opening warehouses will also under the threshold (range is the upper bound of difference in a set of number). So what we can add a constraints to prevent the loads' difference within every pairs of opening warehouses exceeding a certain value.

So how can formula 1.5 realize this idea? Consider the following 3 situations:

1) if $\boldsymbol{y_k = y_i = 1}$, then $\sum_{j \in C} \boldsymbol{d_j x_{ij}} \leq \boldsymbol{diff} + \sum_{j \in C} \boldsymbol{d_j x_{kj}}$

2) if $\boldsymbol{y_k = 0, y_i = 1}$, then $\boldsymbol{diff} + \sum_{j \in C} \boldsymbol{d_j x_{kj}} = \boldsymbol{diff} \geq 0$

3) if $\boldsymbol{y_i = 0, y_k = 1}$, then still $\boldsymbol{diff} \geq 0$

The constraints will impose if and only if facility i and k are both in the optimal facility opening set, otherwise, it will only impose the threshold is greater and equal than 0, which is always satisfied. In the implement, we set the threshold to 10 for computation.

2 Vehicle Routing

Basic Traveling Salesman Problem (TSP) formulas can be used in the single vehicle setting:

$$\boldsymbol{min} \ \sum_{(i,j) \in E} \boldsymbol{c_{ij} x_{ij}} \qquad \textbf{(2.0)}$$

$$\boldsymbol{s.t.} \ \sum_{j=1}^{n} \boldsymbol{x_{ij}} = 1 \ \boldsymbol{for\ every\ i} \in \boldsymbol{V} \qquad \textbf{(2.1)}$$

$$\sum_{j=1}^{n} \boldsymbol{x_{ji}} = 1 \ \boldsymbol{for\ every\ i} \in \boldsymbol{V} \qquad \textbf{(2.2)}$$

$$\boldsymbol{u_1 = 1} \qquad \textbf{(2.3.1)}$$

$$\boldsymbol{u_i - u_j + 1} \leq \boldsymbol{n(1 - x_{ij})} \ \boldsymbol{for\ every} \ (\boldsymbol{i,j}) \in \boldsymbol{E, j \neq 1} \qquad \textbf{(2.3.2)}$$

$$\boldsymbol{x_{ij}} \in \{\boldsymbol{0,1}\} \ \boldsymbol{for\ every} \ (\boldsymbol{i,j}) \in \boldsymbol{E} \qquad \textbf{(2.4)}$$

Assumption(s):

- The capacity of the vehicle is unlimited (or at least greater than or equal to the total demand of the 25

clients), i.e. it can visit all the 25 clients in one continuous route.

- The vehicle starts its routing at depot (point (1,1)) and finish there.

Parameters:
c_{ij}: routing cost from vertex i to vertex j in graph G (ℓ_2 distance)
n: number of vertices in the graph, 26 in this case
V: vertex set in graph G, {0, 1, …, 25}
E: every possible edge in graph G

Variables:
x_{ij}: binary, whether the vehicle will go from vertex i to vertex j or not
u_i: integer, the location index of vertex i

2.1 Multiple Vehicles

Basic Vehicle Routing Problem (VRP) formulas can be used here:

$$\boldsymbol{min} \ \sum_{(i,j)\in E} \boldsymbol{c_{ij}x_{ij}} \qquad \textbf{(2.0)}$$

$$\boldsymbol{s.t.} \ \sum_{k=1}^{K} \boldsymbol{y_{ik}} = 1 \ \boldsymbol{for\ every\ i} \in \boldsymbol{V}\backslash\{\boldsymbol{0}\} \qquad \textbf{(2.5)}$$

$$\sum_{k=1}^{K} \boldsymbol{y_{0k}} = K \qquad \textbf{(2.6)}$$

$$\sum_{i\neq j}^{n} \boldsymbol{x_{ijk}} = \sum_{j\neq i}^{n} \boldsymbol{x_{jik}} = \boldsymbol{y_{ik}}$$
$$\boldsymbol{for\ every\ i,j} \in \boldsymbol{V}, \boldsymbol{i \neq j}, \boldsymbol{k = 1 \dots K} \qquad \textbf{(2.7)}$$

$$\boldsymbol{u_{ik} - u_{jk} + d_j} \leq \boldsymbol{C_k}\left(\boldsymbol{1 - x_{ijk}}\right)$$
$$\boldsymbol{for\ every\ i} \in \boldsymbol{V}, \boldsymbol{j} \in \boldsymbol{V}\backslash\{\boldsymbol{0}\}\ \boldsymbol{i \neq j}, \boldsymbol{k = 1 \dots K} \qquad \textbf{(2.8)}$$

$$\boldsymbol{u_{ik}} \geq \boldsymbol{d_i y_{ik}} \quad \boldsymbol{for\ every\ i} \in \boldsymbol{V}\backslash\{\boldsymbol{0}\}, \boldsymbol{k = 1 \dots K} \qquad \textbf{(2.9)}$$

$$\boldsymbol{u_{ik}} \leq \boldsymbol{C_k y_{ik}} \quad \boldsymbol{for\ every\ i} \in \boldsymbol{V}\backslash\{\boldsymbol{0}\}, \boldsymbol{k = 1 \dots K} \qquad \textbf{(2.10)}$$

Assumption(s):

- The capacity of the vehicles is unlimited (or at least greater than or equal to the total demand of the clients they are going to serve).
- All of the vehicles start their routing at depot (point (1,1)) and finish there.

Parameters:
c_{ij}: routing cost from vertex i to vertex j in graph G (ℓ_2 distance)
d_i: demand of client i
C_k: capacity of vehicle k, does not state in the problem, but can be set to the total demand of the 25 clients
V: vertex set in graph G, {0, 1, …, 25}, where 0 is depot
E: every possible edge in graph G
K: number of vehicles, is 3 in this setting

Variables:

y_{ik}: integer, whether assign client i to vehicle k or not
x_{ijk}: binary, whether the vehicle k will go from vertex i to vertex j or not
u_{ik}: continuous, the load of vehicle k after visiting client i (ascending, minimum at starting point and maximum at the last client it is going to visit)

2.2 Multiple Vehicles and Warehousing

Formulas in this part are almost the same as 2.1 except we ignore formula (2.5) and (2.6), because here we will choose to input y_{ik} from the results of 1.2, which indicate the matching (assignment) between warehouse and clients:

$$\boldsymbol{min} \ \sum_{(i,j)\in E} \boldsymbol{c_{ij}x_{ij}} \qquad \textbf{(2.0)}$$

$$\boldsymbol{s.t.} \ \sum_{i\neq j}^{n} \boldsymbol{x_{ijk}} = \sum_{j\neq i}^{n} \boldsymbol{x_{jik}} = \boldsymbol{y_{ik}}$$
$$\boldsymbol{for\ every\ i,j\ \in V, i\neq j, k = 1 \dots K} \qquad \textbf{(2.7)}$$

$$\boldsymbol{u_{ik} - u_{jk} + d_j \ \leq C_k(1 - x_{ijk})}$$
$$\boldsymbol{for\ every\ i \in V, j \in V\backslash\{0\}\ i\neq j, k = 1 \dots K} \qquad \textbf{(2.8)}$$

$$\boldsymbol{u_{ik} \geq d_i y_{ik} \quad for\ every\ i \in V\backslash\{0\}, k = 1 \dots K} \qquad \textbf{(2.9)}$$

$$\boldsymbol{u_{ik} \leq C_k y_{ik} \quad for\ every\ i \in V\backslash\{0\}, k = 1 \dots K} \qquad \textbf{(2.10)}$$

Assumption(s):

- The capacity of the vehicles is unlimited (or at least greater than or equal to the total demand of the clients they are going to serve).
- All of the vehicles start their routing at depot (point (1,1)) and finish there.

Parameters:
c_{ij}: routing cost from vertex i to vertex j in graph G (ℓ_2 distance)
y_{ik}: whether assign client i to vehicle k (warehouse k) or not, input from the results of 1.2
d_i: demand of client i
C_k: capacity of vehicle k, does not state in the problem, but can be set to the total demand of the 25 clients
V: vertex set in graph G, {0, 1, …, 25}, where 0 is depot
E: every possible edge in graph G
K: number of vehicles, is 3 in this setting

Variables:
x_{ijk}: binary, whether the vehicle k will go from vertex i to vertex j or not
u_{ik}: continuous, the load of vehicle k after visiting client i (ascending, minimum at starting point and maximum at the last client it is going to visit)

The process of data input follows the steps below:

1) Identify how the 25 clients with highest demand are assigned to the 3 warehouses.
2) Follow the matching and construct a integer matrix as shown below:

	A	B	C	D
1		y[,0]	y[,1]	y[,2]
2	y[0,]	1	1	1
3	y[1,]	0	1	0
4	y[2,]	1	0	0
5	y[3,]	0	1	0
6	y[4,]	0	1	0
7	y[5,]	0	1	0
8	y[6,]	0	1	0
9	y[7,]	1	0	0
10	y[8,]	1	0	0
11	y[9,]	0	0	1
12	y[10,]	0	0	1
13	y[11,]	0	0	1
14	y[12,]	0	0	1
15	y[13,]	0	0	1
16	y[14,]	0	0	1
17	y[15,]	1	0	0
18	y[16,]	0	0	1
19	y[17,]	0	0	1
20	y[18,]	1	0	0
21	y[19,]	0	0	1
22	y[20,]	0	0	1
23	y[21,]	1	0	0
24	y[22,]	1	0	0
25	y[23,]	1	0	0
26	y[24,]	1	0	0
27	y[25,]	0	0	1

Figure 8 Integer matrix from results of 1.2

Vehicle 0, 1, 2 belongs to warehouse 0, 8, 9 respectively. You can check more details in [2.2 input data from 1.2.xlsx](#).

3 Routing Special Orders

Formulas are almost the same as vehicle routing in this part, except we add the points of the 2 new clients with priority in the vertex set (V) and deduce some of the traveling cost manually.

$$\boldsymbol{min} \ \sum_{(i,j)\in E'} c'_{ij}x_{ij} \qquad \textbf{(2.0)}$$

$$\boldsymbol{s.t.} \ \sum_{j=1}^{n} x_{ij} = 1 \ \ \boldsymbol{for\ every\ i} \in \boldsymbol{V'} \qquad \textbf{(2.1)}$$

$$\sum_{j=1}^{n} x_{ji} = 1 \ \ \boldsymbol{for\ every\ i} \in \boldsymbol{V'} \qquad \textbf{(2.2)}$$

$$u_1 = 1 \qquad \textbf{(2.3.1)}$$

$$u_i - u_j + 1 \leq n(1 - x_{ij}) \ \ \boldsymbol{for\ every}\ (i,j) \in E', j \neq 1 \qquad \textbf{(2.3.2)}$$

$$x_{ij} \in \{0,1\} \ \ \boldsymbol{for\ every}\ (i,j) \in E' \qquad \textbf{(2.4)}$$

$$c'_{01}, c'_{02}, c'_{12}, c'_{21}\ /100$$

Assumption(s):

- The capacity of the vehicle is unlimited (or at least greater than or equal to the total demand of the 25 clients), i.e. it can visit all the 25 clients in one continuous route.
- The vehicle starts its routing at depot (point (1,1)) and finish there.

Parameters:

c'_{ij}: routing cost from vertex i to vertex j in graph G' (ℓ_2 distance)

n: number of vertices in the graph, 28 in this case

V: vertex set in graph G', {0, 1, 2, …, 25, 26, 27}
E: every possible edge in graph G'

Variables:
x_{ij}: binary, whether the vehicle will go from vertex i to vertex j or not
u_i: integer, the location index of vertex i

Actually, this problem can be seen as VRP with Backhauls (VRPB), but there's no easily implemented VRPB formulation can be solved directly, so we use a heuristic method to get the result in this part. While we should add the original cost to our objective value when we get it.

4 Internships

Modified Stable Matchings formulas can be applied to tackle the problem:

$$\boldsymbol{max} \ \sum_{\mathbf{i=1}}^{\mathbf{I}} \sum_{\mathbf{j=1}}^{\mathbf{A}} \mathbf{x_{ij}} \qquad \textbf{(4.0)}$$

$$\boldsymbol{s.t.} \ \sum_{\mathbf{j=1}}^{\mathbf{A}} \mathbf{x_{ij}} = \mathbf{1} \qquad \textbf{(4.1)}$$

$$\boldsymbol{x_{ij}} + \sum_{\boldsymbol{q} \in \boldsymbol{A_j^{<}(i)}} \boldsymbol{x_{iq}} + \sum_{\boldsymbol{p} \in \boldsymbol{I_i^{<}(j)}} \boldsymbol{x_{pj}} \geq \mathbf{1} \qquad \textbf{(4.2)}$$

$$\boldsymbol{x_{ij}} \in \{\mathbf{0}, \mathbf{1}\} \ \ \boldsymbol{i} = \mathbf{1} \ldots \boldsymbol{I}, \boldsymbol{j} = \mathbf{1} \ldots \boldsymbol{A} \qquad \textbf{(4.3)}$$

Assumption(s):

- Each applicant and each internship both have a full ranking list over the other (i.e. all options are acceptable but with different preference), ranking of each agent involved follows the discrete uniform distribution.
- Every internships should be covered by one applicant while we do not guarantee that all applicants can get an internship.

Parameters:
$A_j^{<}(i)$: the set of applicants that internship i ranks better than applicant j
$I_i^{<}(j)$: the set of internships that applicant j ranks better than internship i

These two sets are called the successors of $\boldsymbol{x_{ij}}$.

Variables:
x_{ij}: binary, whether there is a matching between internship i and applicant j

Actually, the objective function is not necessary in this setting, we can simply change it to constant 1 and the model will give us the exactly same result.

5 Uncertainty on the Demand

Firstly, we consider the deterministic situation, i.e., we know the exact demand in the upcoming months. Then the problem can be formulated as a minimum cost flow model:

$$max \ \sum_{i=1}^{T}\sum_{j=1}^{N}(p_{ij} \cdot d_{ij} - o_{ij} \cdot x_{ij} - h_{ij} \cdot y_{ij}) \quad (5.0)$$

$$s.t. \ x_{0j} - y_{0j} = d_{0j} \ for \ j = 1, \dots N \quad (5.1)$$

$$x_{ij} - y_{ij} + y_{i-1j} = d_{ij} \ for \ i \neq 0, \ j = 1, \dots N \quad (5.2)$$

$$x_{ij} \in \{0, 1\} \ i = 1 \dots T, j = 1 \dots N \quad (5.3)$$

Assumption(s):

- The number of upcoming months with uncertain demand in this year is 10, and the company have 3 different products.
- Samples of the demand are from different months, or specifically, we have 10 demand samples of each month from the historical data (so 100 in total).
- Capacity of the warehouse can meet the demand and we do not consider the warehouse renting cost here, also, do not need to consider the capacity allocation for different products, instead, we can see them as independent. Actually, to consider the capacitated situation we can simply add a new constraints to the above model:

$$\sum_{j=1}^{N} x_{ij} \leq C \quad (5.4)$$

- The supply should meet the demand, i.e. the amount of products we sell should be equal to the demand of that product in a certain month
- All the cost will not be affected by the amount of placing orders and inventory.

Parameters:

p_{ij}: selling price of product j in month i
d_{ij}: demand of product j in month i
o_{ij}: the cost of placing an order of product j in month i
h_{ij}: holding cost of product j in month i

Variables:

x_{ij}: continuous, units of placing orders for product j in month i
y_{ij}: continuous, units of inventory for product j in month i

When the demand (one of the parameters) become uncertain, then the above model can be modified to a stochastic programming model with random variable d_{ij} and the following techniques can be utilized to cope with this situation:

- Sample Average Approximation (SAA)

We can rewrite the objective function as

$$\hat{f}_N(x) = \frac{1}{N}\sum_{s=1}^{N} F(x_{ij}, d_{ij}^s)$$

where d_{ij}^s is the historical demand of product j in month i (in previous year). This method sees d_{ij} as a random variable and use the sample average to approximate the true unknown objective function. The assumption behind this method is that all the samples are iid.

- Simulation with Bootstrap Sampling

Though the Sample Average Approximation can give us an approximation about the objective function, it may not be too accurate because we only have 10 samples for each d_{ij} and the consistency of this method lies on Law of Large Numbers (LLN).

Therefore, it is advisable to utilize some resampling method to get a higher accuracy of the approximation. For example, bootstrapping is a suitable choice to do that which resample the data from the original samples with replacement. Resampling and simulation can use together with sample average approximation to get a better approximation.

- Predictive Analytics (e.g. time series forecasting)

Unlike the previous methods, demand forecasting or predictive analytics input the predicted $\hat{d}_{ij}$ instead of the historical samples into the model, while $\hat{d}_{ij}$ is still estimated by the historical data or other factors that we consider to have higher correlation to the future demand. Serval predictive method can be used to tackle this problem such as time series analysis and multiple linear regression etc., we can compared the results from different models and do further analysis to model the uncertainty, we do not extend this part here.

Sampled implement of SAA method can be found in 5 Uncertainty on the Demand (implement example).py.