Internship Assignment NITIE

- Submitted by Adnan Hassan

Problem Statement
The given task is about the execution of a capacitated vehicle routing problem (CVRP) with some additional optimizations that are not covered in the standard vehicle routing problem (VRP).

The *capacitated vehicle routing problem* (**CVRP**) is a vehicle routing problem in which vehicles with limited carrying capacity need to pick up or deliver items at various locations. The items have a quantity, such as weight or volume, and the vehicles have a maximum capacity that they can carry. The problem is to pick up or deliver the items for the least cost, while never exceeding the capacity of the vehicles.

There are many variants of vehicle routing problems. We will try to solve three variants of it. These are as follows:-

1. **CVRP** (Capacitated Vehicle Routing Problem) : Vehicles have a limited carrying capacity of the goods that must be delivered.

2. **CVRPTW** (Capacitated Vehicle Routing Problem with Time Windows) : The delivery locations have time windows within which deliveries (or visits) must be made.

2. **CVRP_with_greedy_approach** :The aim is to provide the customers with their demand by minimising the total distance travelled by all the vehicles. (Brute force solving through functional programming and naive greedy approach)

CVRP (Capacitated Vehicle Routing Problem)

1. *Definition of the problem*

The capacitated vehicle routing problem (CVRP) is a VRP in which vehicles with limited carrying capacity need to pick up or deliver items to various locations. The items have a quantity, such as weight or volume, and each vehicle has a maximum capacity that they can carry. The problem is to pick up or deliver the items with the least cost, while never exceeding the capacity of the vehicles.

So, I will set a customer number, vehicle number, demand of each customer and capacity of vehicles as follows.

- Customer number : 9 (plus one depot)
- Vehicle number: 4
- Demand of each customer: random values between 10 and 20
- Vehicle capacity: 50 (assume all vehicles have the same capacity)

2. *Model and data to be used*

Graph theory is often used (numerical data is added to nodes and edges of graphs) for the vehicle routing problems. This is because it is possible to think in a way that selects (or not selecting) routes by passing (or not passing) edges. In this context, assigning ‘1’ to an edge means that a route is being selected. It can be said that this corresponds to the integer optimization problem. This post will also explain the CVRP problem using a graph.

In this problem, we will generate the data in the code itself. Firstly, the latitude and longitude are used to derive the location and distance is formulated by use of that only through the function defined in the code. For other values the data is created random module by keeping constraints in mind.

Definition of network

① $G = (V, E)$: A graph showing the location and route of the customer

② $V = \{0, 1, \ldots, n\}$: A collection of nodes. The node that represents the Depot (vehicle warehouse) is 0, and the nodes that represent each customer are 1 to n

③ E : Set of Edges connecting each node $e_{ij} = (i, j)$

④ $K = \{1, 2, \ldots, |K|\}$: Set of vehicles

⑤ $d_i \geq 0$: demand of customer i

⑥ $Q \geq 0$: Maximum vehicle capacity (all vehicles have the same maximum capacity Q)

⑦ c_{ij} : Travel costs between customer i and j (distance between i and j)

Definition of decision variables

A decision variable that indicates whether the vehicle k has taken the route e_{ij} $\quad x_{ij}^{k} = \begin{cases} 1 & \text{taken} \\ 0 & \text{Not taken} \end{cases}$

Definition of objective function

Minimize the sum of travel costs (travel distance) of all vehicles

Description of constraints

- Only one vehicle visits each customer's location
- Depot starts and returns (and no subtours)
- The deliverable capacity of each vehicle is less than or equal to the maximum capacity K

3. *Formulate according to given constraints and model*

$$(1) \quad \min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ij}^k$$

$$(2) \quad \sum_{k \in K} \sum_{i \in V, i \neq j} x_{ij}^k = 1 \qquad \forall j \in V \setminus \{0\}$$

$$(3) \quad \sum_{j \in V \setminus \{0\}} x_{0j}^k = 1 \qquad \forall k \in K$$

$$(4) \quad \sum_{i \in V, i \neq j} x_{ij}^k - \sum_{i \in V} x_{ji}^k = 0 \qquad \forall j \in V, \forall k \in K$$

$$(5) \quad \sum_{i \in V} \sum_{j \in V \setminus \{0\}, i \neq j} q_j x_{ij}^k \leq Q \qquad \forall k \in K$$

$$(6) \quad \sum_{k \in K} \sum_{(i,j) \in S, i \neq j} x_{ij}^k \leq |S| - 1 \qquad S \subseteq V \setminus \{0\}$$

$$(7) \quad x_{ij}^k \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in E$$

Based on the definition of parameters in the figure of the second section, I formulated constraints for the decision variables and the objective function of CVRP.

- Objective function which means “minimise the travel cost (sum of travelling distance) of all vehicles” - (1)
- Constraint which means “only one visit per vehicle per customer’s location”-(2)
- Constraint which means “depart from depot” - (3)
- Constraint which means “the number of vehicles coming in and out of a customer’s location is the same” - (4)
- Constraint which means “ the delivery capacity of each vehicle should not exceed the maximum capacity” - (5)
- Constraint for “removal of subtours” - (6)
- Constraint for decision variables - (7)

4. *Coding*

Based on the given formulas, I wrote python code for solving CVRP with pulp, which is an open-source package that allows mathematical programs to be described in Python. The code is attached in the repository.

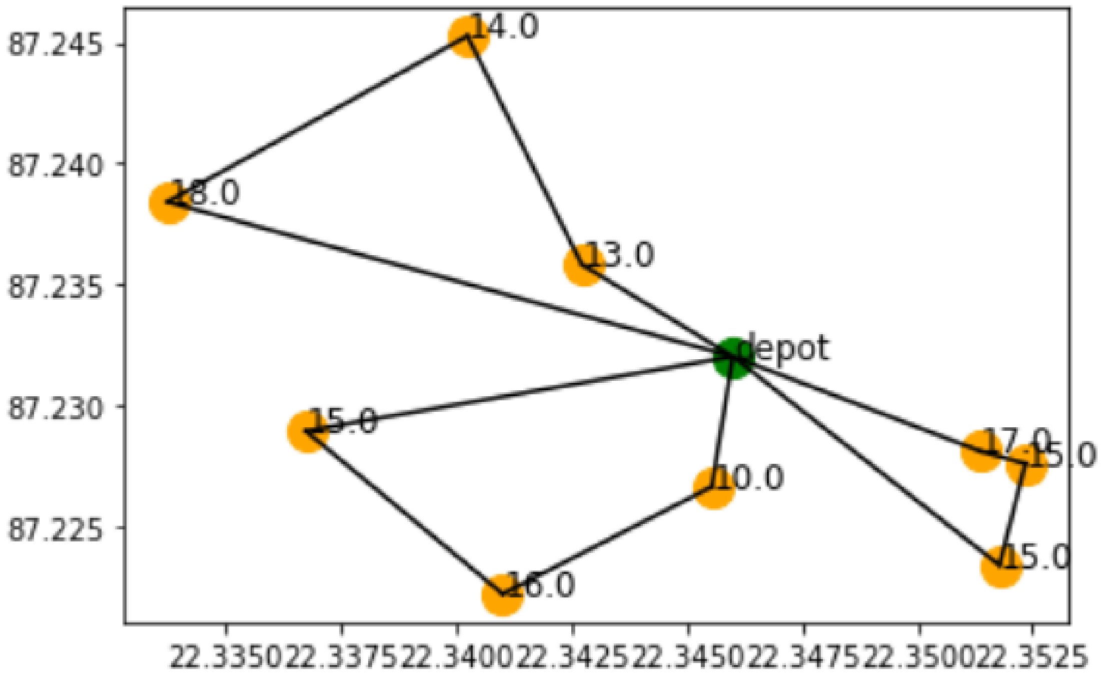
5. Plots and results obtained

	latitude	longitude	demand
0	22.346000	87.232000	0.0
1	22.342723	87.235814	13.0
2	22.340240	87.245266	14.0
3	22.345542	87.226614	10.0
4	22.341006	87.222178	16.0
5	22.352344	87.227573	15.0
6	22.351364	87.228088	17.0
7	22.351782	87.223367	15.0
8	22.336734	87.228923	15.0
9	22.333733	87.238404	18.0

Above figure shows the random generated locations & demand of customers. Row number 0 is the depot, which has no demand. In this example, the location of IIT KGP campus is set as a depot.

And the following are printed results, which show that one of four vehicles was unnecessary, and the total moving distance of the other three vehicles is 9.6002 km. This is the optimal result calculated under the given conditions!

```
Vehicle Requirements: 3
Moving Distance: 9.600264377411943
```



This is the plot for the route representation.

6. *Summary*

I explained CVRP (Capacitated Vehicle Routing Problem) and introduced the python code which calculates optimal routing using pulp. Mapped results show that output of the python code makes sense.

However, the problem set in this post is extremely simplified (for example, the capacities of all vehicles are the same). In other words, it is extremely complicated to make a model that considers all possible constraints in the real field.

Therefore, in order to create a VRP solution as a system in the real field, it is necessary to consider how constraints can be reflected properly on the model.

CVRPTW (Capacitated Vehicle Routing Problem with Time Windows)

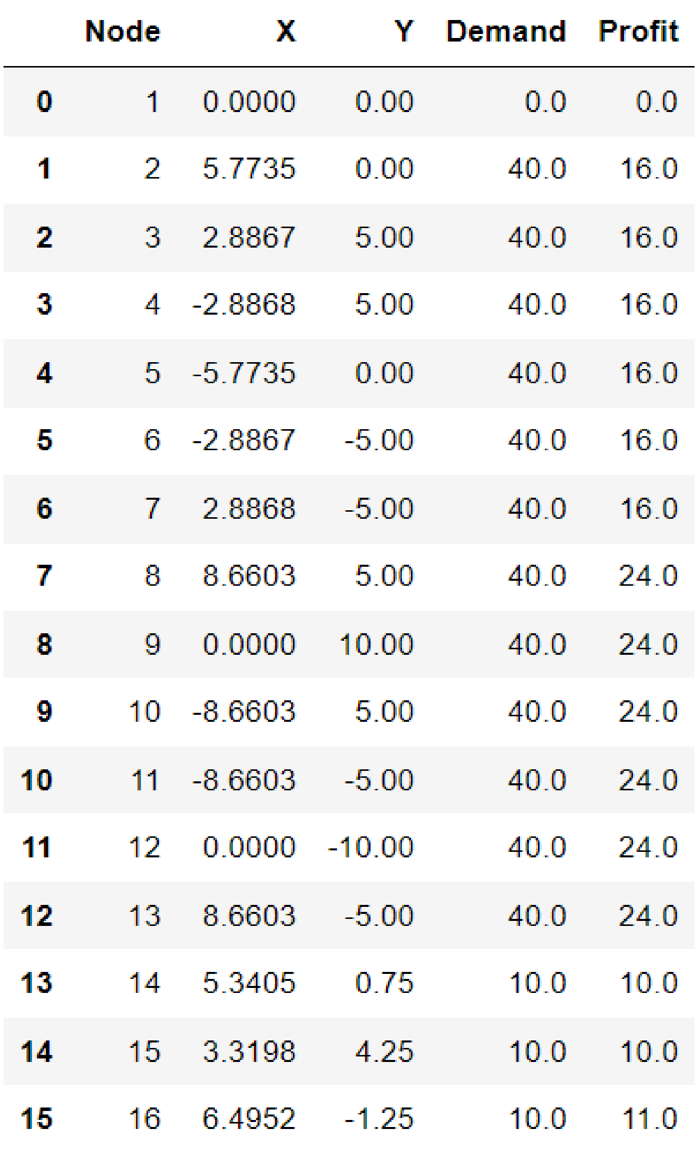
1. *Definition of the problem*

Basically, this is a vehicle routing problem with time windows. A typical VRP model consists of a depot and multiple customers in known distances from the depot. Challenge is to find the optimal route for the vehicle which starts the tour at the depot, to visit all the customers and come back to the depot with the minimum possible distance covered. We can make the problem more realistic by considering multiple vehicles available at depot and routes for each vehicle to visit all the customers. Hence, in this VRP, multiple vehicles available at depot will complete their respective tours starting and ending at depot so total distance travelled is minimum. This model is very helpful in saving time and cost of fuel of vehicles for any purpose or business by formulating minimum distance coverage tours. Some of the problems can be formulated considering multiple depots as well, but in this project, I am only focusing on the single depot scenario.

2. Model and data to be used

Solution method used is Mixed Integer Programming (MIP) with the help of simplex solver of Gurobi software in Python-Gurobi interface. In Python programming, distance between each node is calculated with Pythagoras law. A model is started in the Gurobi platform, and decision variables are defined. Nature and dimension of variables are added to the model as mentioned in formulation, with Gurobi’s format of syntax. All the constraints are defined and added to the model and finally the objective is added. Finally, the model is optimised with Gurobi optimizer v9.5.1. Final solution is retributed and stored in a table array.

The data is stored in the text file, the format of the extracted data is as given below:



	Node	X	Y	Demand	Profit
0	1	0.0000	0.00	0.0	0.0
1	2	5.7735	0.00	40.0	16.0
2	3	2.8867	5.00	40.0	16.0
3	4	-2.8868	5.00	40.0	16.0
4	5	-5.7735	0.00	40.0	16.0
5	6	-2.8867	-5.00	40.0	16.0
6	7	2.8868	-5.00	40.0	16.0
7	8	8.6603	5.00	40.0	24.0
8	9	0.0000	10.00	40.0	24.0
9	10	-8.6603	5.00	40.0	24.0
10	11	-8.6603	-5.00	40.0	24.0
11	12	0.0000	-10.00	40.0	24.0
12	13	8.6603	-5.00	40.0	24.0
13	14	5.3405	0.75	10.0	10.0
14	15	3.3198	4.25	10.0	10.0
15	16	6.4952	-1.25	10.0	11.0

3. Formulate according to given constraints:

- No of depots: 1
- No of nodes: n
- No of vehicles: vehicle_count
- Vehicle capacity: vehicle_capacity
- Demand at nodes: D
- (i, j): Nodes indices
- dij = Distance between i & j
- dii = M.

1. **Variables:**
   Xij = {0, 1}, binary decision on going from node 'i' to 'j'
   Y(j) = Cumulative demand satisfied at node 'j' by a vehicle.
2. $z = min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij*x_{ij}}$

   s.t.

3. $\sum_{i=1}^{n} X_{0j} = 5$ $(i = 0;\ depot)$: start of 'm' tours at depot.
4. $\sum_{i=1}^{n} x_{i0} = 5$ $(j = 0;\ depot)$: end of 'm' tours at depot.
5. $\sum_{j=2}^{n} x_{ij} = 1$ $\forall i$ (i = 2, 3, …., n)
6. $\sum_{i=2}^{n} x_{ij} = 1$ $\forall j$ (j = 2, 3, …., n)
7. $x_{ij} + x_{ji} \leq 1$ $\forall i, j$ $(i, j = 1, 2, \ldots, n)$
8. Y[j] >= Y[i] + D[j]* X[i,j] – Q*(1-X[i,j]) (i, j = 2, 3, …., n)
9. D[j] <= Y[j] <= Q (j = 2, 3, ….,n)

4. *Coding*

Based on the given formulas, I wrote python code for solving the problem with the help of simplex solver of Gurobi software in Python-Gurobi interface. The code is attached in the repository.

5. *Plots and results obtained*

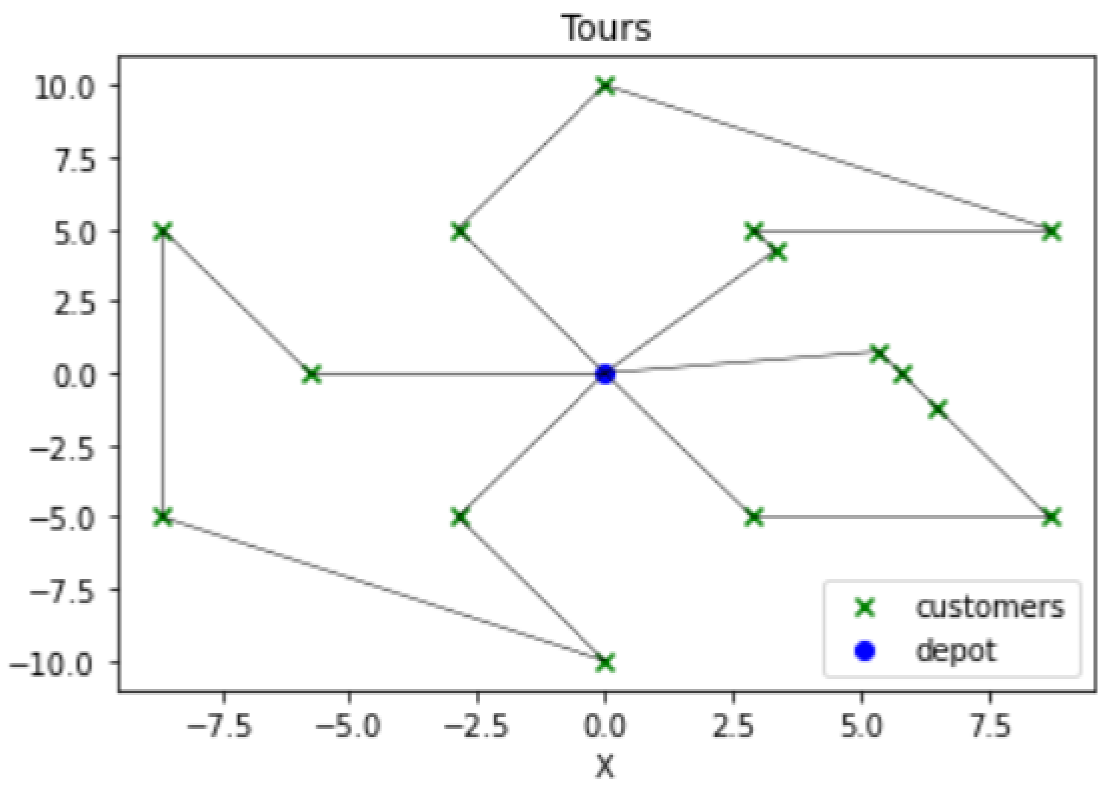
Case 1: vehicle_capacity = 200, vehicle_count = 3 and customer_count = 15

```
Objective is: 100.25318202062216

Decision variable X (binary decision of travelling from one node to another):
 [[0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]]

Decision variable z:(service start time of every customers in minutes)
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Decision variable y (cumulative demand collected at every customer node):
 [ 50  50 200  40 200 200  90 160  80 120 160 140  10  10 100]
```



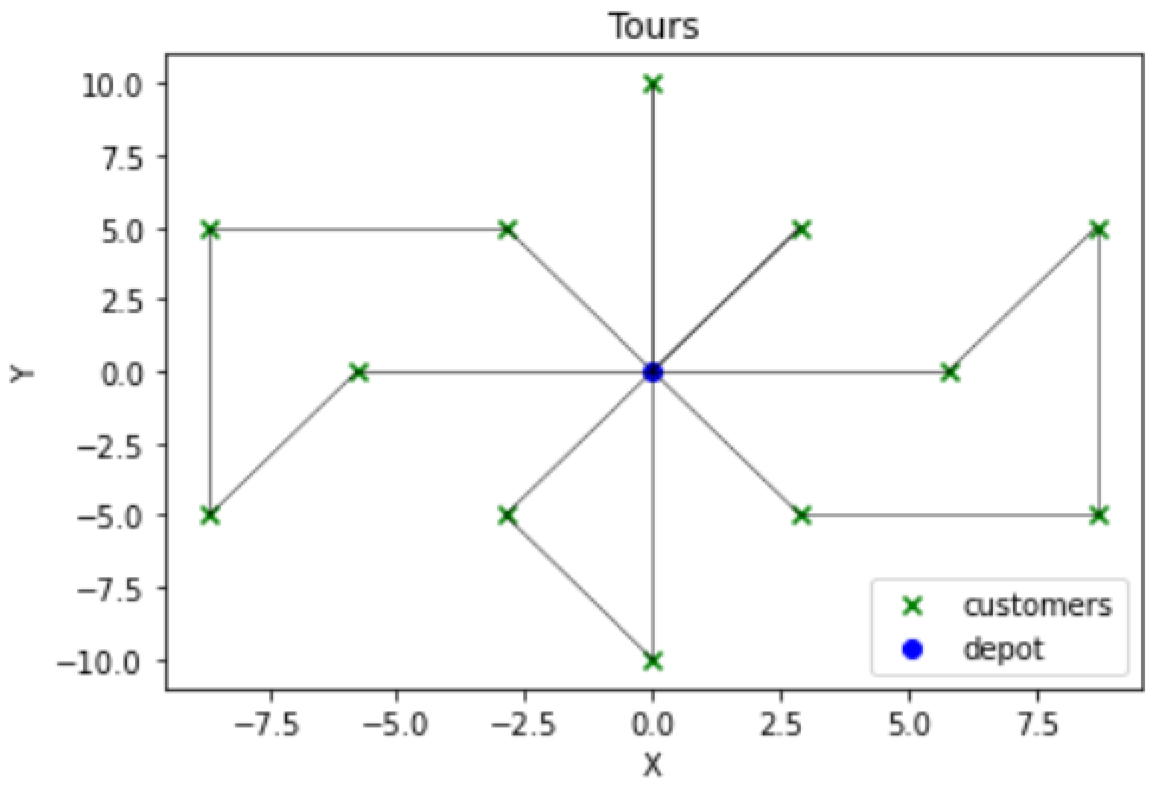
Case 2: vehicle_capacity = 200, vehicle_count = 5 and customer_count = 12

```
Objective is: 119.28201615267753

Decision variable X (binary decision of travelling from one node to another):
 [[0 1 1 0 1 1 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0]]

Decision variable z:(service start time of every customers in minutes)
 [0 0 0 0 0 0 0 0 0 0 0 0]

Decision variable y (cumulative demand collected at every customer node):
 [ 80  40 200  80  40 200 120  40 160 120  80 160]
```



Case 3: vehicle_capacity = 300, vehicle_count = 5 and customer_count = 25

Objective is: 121.71983980518628

Decision variable X (binary decision of travelling from one node to another):

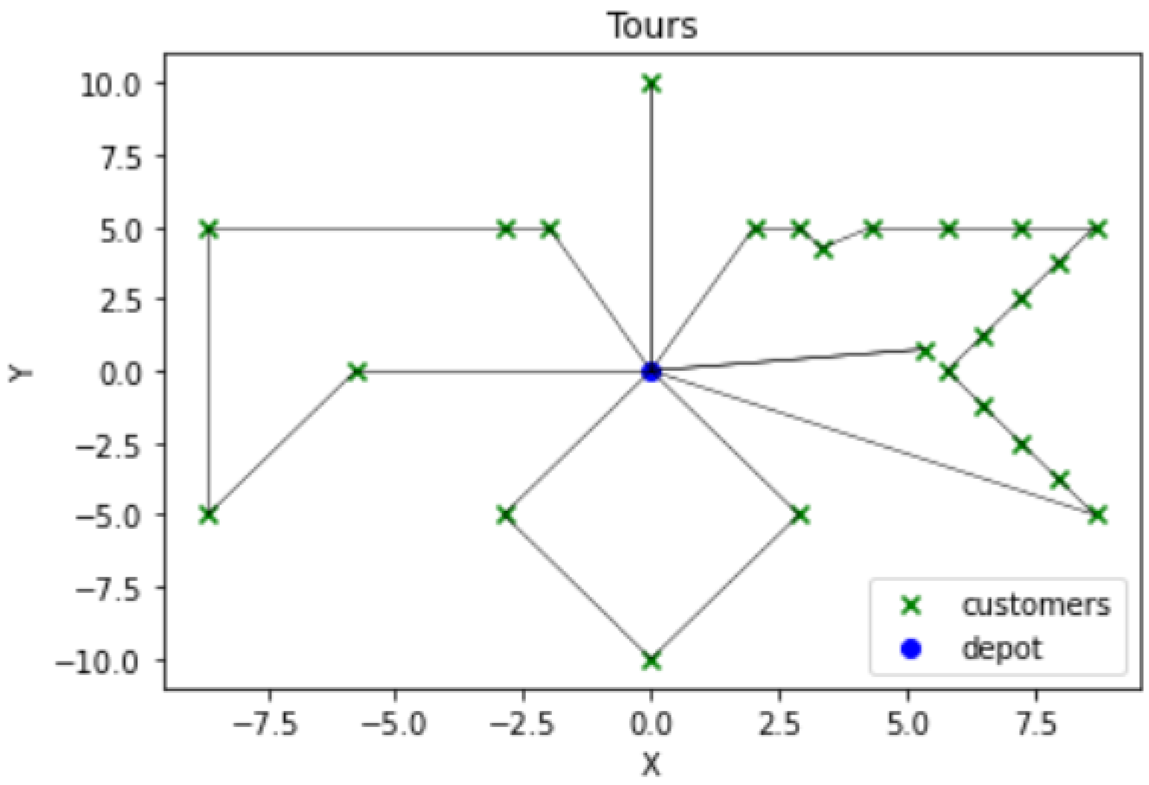
```
[[0 0 0 0 1 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]]
```

Decision variable z:(service start time of every customers in minutes)

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Decision variable y (cumulative demand collected at every customer node):

```
[120 290 290  40 300  40 200  40 250  80  80  40  10 250  80  70  50 130
 150 160 300 300 240 230 210]
```



6. *Summary*

This model is very useful to determine routes for vehicles to reduce cost of travel. I formulated and solved a multiple VRP with time windows model with single depot with

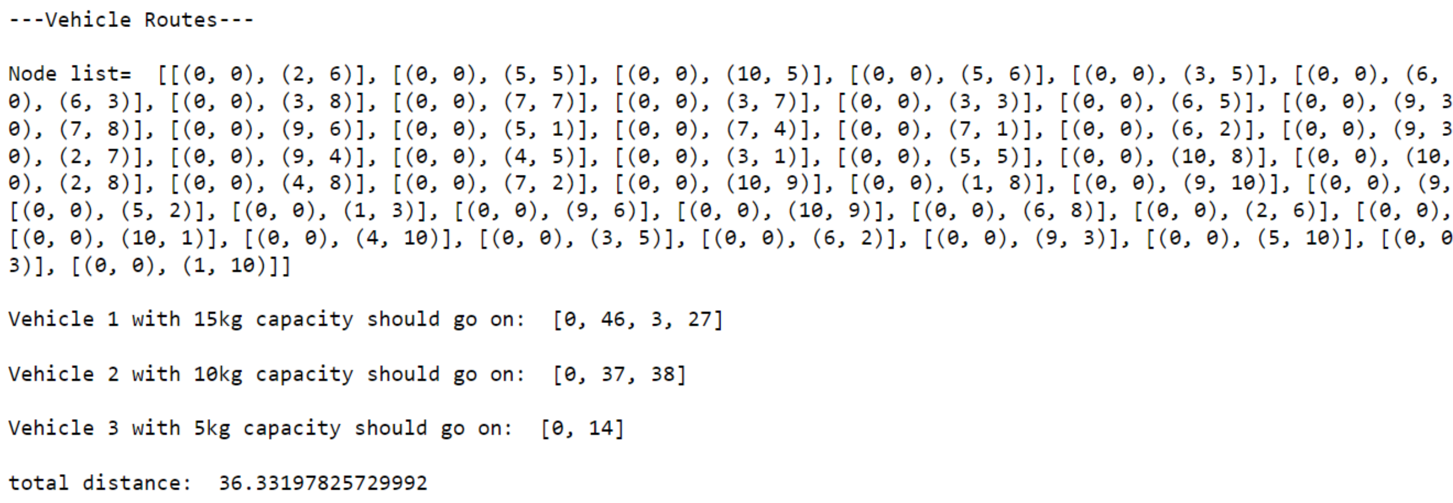
Gurobi solver in Gurobi-Python interface. This is a NP hard problem and computational time rises with the increasing size of the problem. This model is capable of solving MVRP(multiple vehicle routing problem) for bigger scenarios as well as TSP(travelling salesman problem) when the number of vehicles is just one. For real life businesses and other routing purposes, Optimal route plans with this model will minimise the fuel cost and travel duration of vehicles on day to day operation which is a huge economic and time saving advantage.

CVRP with greedy approach

1. *Definition of the problem*
In this approach, we will try to find the optimal routes for all the vehicles used and try to minimise the distance travelled in multiple vehicle routing problems(MVRP). This problem is similar to the other problems solved. In this approach, all the vehicles will have different capacities. And the difference will be in the approach of finding the routes to all the customers from the node.

2. *Model and data to be used*
The model used is not defined as I will be completely relying on the greedy approach to get the routes for all the vehicles simultaneously taking into account the minimum distance criterion.
The data is completely generated using the random module.

3. Results obtained



```
---Vehicle Routes---

Node list=  [[(0, 0), (2, 6)], [(0, 0), (5, 5)], [(0, 0), (10, 5)], [(0, 0), (5, 6)], [(0, 0), (3, 5)], [(0, 0), (6,
0), (6, 3)], [(0, 0), (3, 8)], [(0, 0), (7, 7)], [(0, 0), (3, 7)], [(0, 0), (3, 3)], [(0, 0), (6, 5)], [(0, 0), (9, 3
0), (7, 8)], [(0, 0), (9, 6)], [(0, 0), (5, 1)], [(0, 0), (7, 4)], [(0, 0), (7, 1)], [(0, 0), (6, 2)], [(0, 0), (9, 3
0), (2, 7)], [(0, 0), (9, 4)], [(0, 0), (4, 5)], [(0, 0), (3, 1)], [(0, 0), (5, 5)], [(0, 0), (10, 8)], [(0, 0), (10,
0), (2, 8)], [(0, 0), (4, 8)], [(0, 0), (7, 2)], [(0, 0), (10, 9)], [(0, 0), (1, 8)], [(0, 0), (9, 10)], [(0, 0), (9,
[(0, 0), (5, 2)], [(0, 0), (1, 3)], [(0, 0), (9, 6)], [(0, 0), (10, 9)], [(0, 0), (6, 8)], [(0, 0), (2, 6)], [(0, 0),
[(0, 0), (10, 1)], [(0, 0), (4, 10)], [(0, 0), (3, 5)], [(0, 0), (6, 2)], [(0, 0), (9, 3)], [(0, 0), (5, 10)], [(0, 0
3)], [(0, 0), (1, 10)]]

Vehicle 1 with 15kg capacity should go on:  [0, 46, 3, 27]

Vehicle 2 with 10kg capacity should go on:  [0, 37, 38]

Vehicle 3 with 5kg capacity should go on:  [0, 14]

total distance:  36.33197825729992
```

Above diagram represents the result obtained through the code, it comprises the minimum total distance, paths to be taken by all vehicles and the vehicle route.

4. *Summary*

The above approach solved the problem mainly by creating new routes, combining two routes having similar edges, adding nodes on edges of the route and finally to get the route with maximum weight utilised for the current vehicle.

This naive approach developed great intuition for the capacitated vehicle routing problem with multiple vehicles of different weighing capacities.