UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN.

“Facultad de Ingeniería Mecánica y Eléctrica”

Proyecto Integrador de Aprendizaje

Equipo 2

Jorge Alejandro Gónzalez Guerra 1889169

Juan Francisco Zermeño Puente 1679055

Bryan Alexis Carranza Reta 1668023

TEACHER: Iris Abril Martínez Salazar

HOUR: M4-M6 **FREQUENCY:** Thursday

CONTENT

Title: Solving an MTVRP problem.

Authors:

Gonzalez Guerra Jorge Alejandro

Zermeño Puente Juan Francisco

Carranza Reta Bryan Alexis

Abstract

The objective of this project was to design a heuristic algorithm and implement and code it in the Python programming language so that it solves the Multitrip Vehicle Routing Problem (MVRP) with the objective function of minimizing the sum of the times that customers wait to receive their product / service.

The steps of this algorithm were shown, as well as the examples used to test said algorithm and the interpretation of each of these results.

Keywords: Heuristic algorithm, MVRP, VRP, MLP, Python.

Introduction

The multi-trip vehicle routing problem (MTVRP) is a variant of the capacitated vehicle routing problem where each vehicle can perform a subset of routes, called a vehicle schedule, subject to maximum driving time constraints.

In this project you will see how a series of MTVRP problems were solved with the help of the heuristic algorithm designed and encoded in Python. This algorithm was designed so that it can be used with N number of clients and show the resulting routes and the clients visited in each one of these, as well as the result of the objective function. In this way the algorithm can be used in many cases as will be seen later this could help in many cases where vehicle routing is used. This problem can help solve routing problems for parcel companies like DHL, Fedex, Ups and Estafeta.

It was investigated in a series of articles related to the resolution of MVRP but only solutions have been found to problems that take multiple vehicles into account to carry out the defined routes.

The problems to be dealt with below have a certain number of clients, each of these clients has a certain demand and there is only one vehicle to visit each of the clients. At the end of this project, a time comparison can be made using one vehicle as in this problem or multiple vehicles as the articles previously investigated.

Literature review

A vehicle routing problem consists of three main elements: vehicles, customers, and warehouses. Broadly speaking, the problem is to determine a set of routes that begin and end in warehouses, so that vehicles visit customers. There is a cost measure (usually the distance traveled, the time or a combination of both) that is the one that is sought to minimize. The characteristics of the vehicles, customers and warehouses give rise to different restrictions that, combined together, generate different variants of the problem.

Vehicles

Vehicles often have limited capacity. Said capacity may have one or more dimensions (for example, weight and volume). Each vehicle can have both a fixed cost and a variable cost proportional to the distance it travels. Problems in which all vehicles have the same attributes (capacity, cost, etc.) are called homogeneous fleet, and, if there are differences, these are heterogeneous fleet problems.

Legal regulations could limit the time that a vehicle can be in circulation and even prohibit the passage of certain vehicles through certain areas of the road network. In some cases it may be required that the amount of work done by different vehicles (eg number of customers visited or travel time) is not very uneven.

The number of vehicles available could be an input or a decision variable. In general, it is assumed that each vehicle travels a single route in the planning period, but lately models in which the same vehicle can travel more than one route have been studied.

Clients

Each customer has a demand that must be satisfied by a vehicle. In many cases, demand is a good that takes place in vehicles. The requirement may be to distribute the merchandise among the clients, or collect the merchandise located in the clients and transport it to the warehouse. It could also happen that the merchandise must be transported to the clients but it is not initially in the warehouse, but distributed in certain supplier sites. In this case, suppliers must be visited before clients.

In other cases the object of the demand is not a good but a service: each client must simply be visited by the vehicle. The same vehicle could potentially visit all customers. In another variant of the problem, each customer has an origin location and wants to be transported to a destination site. Here the capacity of the vehicle imposes a limit on the number of clients that it can simultaneously host.

Usually, each client is required to be visited exactly once. However, in certain cases it is accepted that a customer's demand is satisfied at different times and by different vehicles.

Customers may have restrictions regarding their hours of service. These restrictions are usually expressed in the form of time intervals (called time windows) in which a vehicle can arrive at the customer.

In problems with several different vehicles there may be compatibility restrictions between these and customers. In these cases, each client can only be visited by some of the vehicles (for example, some very heavy vehicles cannot enter certain locations).

Deposits

Both the vehicles and the merchandise to be distributed (if any) are usually initially located in warehouses. It is common to require that each route start and end at the same depot, although this may not be the case in some applications (for example, it could happen that the trip must start or end at the driver's home).

In problems with multiple tanks, each could have different characteristics, such as its location and maximum production capacity. It may also happen that each depot has a fleet of vehicles assigned a priori that this allocation is part of what you want to determine. Each route is typically required to start and end at the same depot, although routes between different depots may also be allowed.

Deposits, like clients, may have associated time windows. In some cases, the time it takes to load or prepare a vehicle before your route begins, or the time spent cleaning it when you return, must be considered. Even, due to limitations of the tanks themselves, it may be necessary to avoid that too many vehicles are operating in the same tank at the same time.

Definition of the problem

$$L = \sum_{i=1}^{n} l_{[i]} = \sum_{i=1}^{n} (n-i+1) c_{[i-1][i]}$$

When visualizing the problem we relate it to a gentleman of post office or any company that is dedicated to make or collet shipments online, or whether they are international shipments, ect.

We think together that it would be the most viable or would be the most practical for him to carry out your deliveries in the shortest possible time, thinking of possible solutions, we deduce that we could mention or assign that your fist stop would be that of the client closest to him.

Since it would take less time to reach the client who is in a more distant area, thus carrying out the requests of the clients to collect in their homes the material or the product to be transported, from a shorter distance to a greater distance, once the deposit is completed, or reaches its maximum capacity, return to the warehouse to download the products and start again from the next customer who continues as a short distance to the warehouse.

Databases

Some databases used to test the algorithm.

MT-DMP10s0-01.txt

nbClients: 10

nbTrips: 2

VehCapacity: 120

ClientDemands:

10 20 30 10 20 10 20 30 30 30

MT-DMP10s0-05.txt

nbClients: 10

nbTrips: 2

VehCapacity: 120

ClientDemands:

10 20 20 30 10 30 30 30 10 20

MT-DMP15s0-03.txt

nbClients: 15

nbTrips: 3

VehCapacity: 120

ClientDemands:

10 30 20 20 30 20 30 10 10 10 10 30 30 20 20

MT-DMP15s0-04.txt

nbClients: 15

nbTrips: 3

VehCapacity: 120

ClientDemands:

20 30 10 20 10 10 30 20 10 30 30 20 30 10 20

VRPNC1m.TXT

nbClients: 50

nbTrips: 5

VehCapacity 160

ClientDemands:

7 30 16 9 21 15 19 23 11 5 19 29 23 21 10 15 3 41 9 28 8
8 16 10 28 7 15 14 6 19 11 12 23 26 17 6 9 15 14 7 27 13
11 16 10 5 25 17 18 10

VRPNC3m.TXT

nbClients: 100

nbTrips: 8

VehCapacity 200

ClientDemands:

10 7 13 19 26 3 5 9 16 16 12 19 23 20 8 19 2 12
17 9 11 18 29 3 6 17 16 16 9 21 27 23 11 14 8 5 8
16 31 9 5 5 7 18 16 1 27 36 30 13 10 9 14 18 2 6
7 18 28 3 13 19 10 9 20 25 25 36 6 5 15 25 9 8 18
13 14 3 23 6 26 16 11 7 41 35 26 9 15 3 1 2 22 27
20 11 12 10 9 17

Proposed solution method

Our heuristic was defined as follows, and explained in the following steps:

1.- The Euclidean distances are calculated from the deposit to each of the clients and between each of the clients.

2.- The "deposito-cliente" array is created that contains the distances from the deposit to each of the clients, and the variable "Conteo_Rutas" is created with an initial value equal to 1.

3.- The "deposito-cliente" array is ordered from lowest to highest.

4.- Adds to the "rutas" array the client with the shortest distance from the deposit and removes it from the "deposito-cliente" array, also adds its value of "time" (distance) to the variable "Tiempo_total".

5.- Added client demand added to the "rutas"array to the "demanda_Total" variable.

6.- The next client to be added to the "rutas" array will be the client with the shortest distance from the client previously added to the "routes" array.

7.- Client demand added to the "routes" array is added to the "demanda_Total" variable and its value of "time" (distance) plus the previous times are added to the "Total_Time" variable.

8.- It is validated that the variable "demanda_Total" is less than or equal to the maximum capacity of the vehicle.

9.- If the "demanda_Total" variable is less than the vehicle's limit capacity, go back to step 6.

10.- If the "demanda_Total" variable is greater than or equal to the vehicle's capacity, a new route is created and the "demanda_Total" variable is initialized to 0 and 1 is added to the value of the "Conteo_Rutas" variable.

11.- Iterate again from step 4 if the size of the "deposito-client" array is greater than 0.

12.- If the size of the "deposito-cliente" array is equal to 0, the routes obtained and the value of the "Total_Time" variable are printed.

As we could see, our heuristic is based on starting from the smallest distances to iterate to carry out the complete tour of the clients to visit, this was the methodology that was implemented.

Computational experimentation

Instance name	OF Value	CPU Time	# Clients	# Routes
MT-DMP10s0-01.txt	1528	0.00200	10	2
MT-DMP10s0-05.txt	2052	0.00086	10	2
MT-DMP15s0-03.txt	5064	0.00138	15	3
MT-DMP15s0-04.txt	3921	0.001487	15	3
VRPNC1m.txt	2618.4162	0.0057	50	5
VRPNC2m.txt	6154.9680	0.0093	75	10
VRPNC3m.txt	7259.1719	0.0143	100	8
VRPNC4m.txt	17359.8108	0.0279	150	8
VRPNC5m.txt	27247.8203	0.0389	199	16
VRPNC11m.txt	9849.6052	0.0138	120	7
VRPNC12m.txt	7603.5705	0.0104	100	10

MT-DMP10s0-01.tx

```
Elapsed Time: 0.0020035810000000043 seconds
Expected Trips: 2
Actual Trips: 2
Total Distance: 1528

Deposit         ->      Client 3        |       Capacity: 120 - 30 -> 90 ✔
Client 3        ->      Client 1        |       Capacity: 90 - 10 -> 80 ✔
Client 1        ->      Client 2        |       Capacity: 80 - 20 -> 60 ✔
Client 2        ->      Client 8        |       Capacity: 60 - 30 -> 30 ✔
Client 8        ->      Client 9        |       Capacity: 30 - 30 -> 0 ✔
Client 9        ->      Deposit

Deposit         ->      Client 10       |       Capacity: 120 - 30 -> 90 ✔
Client 10       ->      Client 4        |       Capacity: 90 - 10 -> 80 ✔
Client 4        ->      Client 5        |       Capacity: 80 - 20 -> 60 ✔
Client 5        ->      Client 6        |       Capacity: 60 - 10 -> 50 ✔
Client 6        ->      Client 7        |       Capacity: 50 - 20 -> 30 ✔
```

VRPNC3m.TXT

```
Elapsed Time: 0.013784182000000006 seconds
Expected Trips: 8
Actual Trips: 8
Total Distance: 7259.171966581872

Deposit         ->      Client 7        |       Capacity: 200 - 5 -> 195 ✔
Client 7        ->      Client 52       |       Capacity: 195 - 9 -> 186 ✔
Client 52       ->      Client 39       |       Capacity: 186 - 31 -> 155 ✔
Client 39       ->      Client 67       |       Capacity: 155 - 25 -> 130 ✔
Client 67       ->      Client 49       |       Capacity: 130 - 30 -> 100 ✔
Client 49       ->      Client 4        |       Capacity: 100 - 19 -> 81 ✔
Client 4        ->      Client 19       |       Capacity: 81 - 17 -> 64 ✔
Client 19       ->      Client 26       |       Capacity: 64 - 17 -> 47 ✔
Client 26       ->      Client 27       |       Capacity: 47 - 16 -> 31 ✔
Client 27       ->      Client 28       |       Capacity: 31 - 16 -> 15 ✔
Client 28       ->      Client 88       |       Capacity: 15 - 9 -> 6 ✔
Client 88       ->      Client 62       |       Capacity: 6 - 19 -> -13 ✘
Client 88       ->      Client 69       |       Capacity: 6 - 6 -> 0 ✔
Client 69       ->      Deposit

Deposit         ->      Client 53       |       Capacity: 200 - 14 -> 186 ✔
Client 53       ->      Client 56       |       Capacity: 186 - 6 -> 180 ✔
Client 56       ->      Client 40       |       Capacity: 180 - 9 -> 171 ✔
Client 40       ->      Client 21       |       Capacity: 171 - 11 -> 160 ✔
Client 21       ->      Client 48       |       Capacity: 160 - 36 -> 124 ✔
Client 48       ->      Client 47       |       Capacity: 124 - 27 -> 97 ✔
Client 47       ->      Client 72       |       Capacity: 97 - 25 -> 72 ✔
Client 72       ->      Client 36       |       Capacity: 72 - 5 -> 67 ✔
Client 36       ->      Client 82       |       Capacity: 67 - 16 -> 51 ✔
Client 82       ->      Client 58       |       Capacity: 51 - 18 -> 33 ✔
```

Conclusions

After analyzing the two problems proposed to carry out a heurysitic together, we came to the conclusion of how important they are, since they could be said to be the simple way to solve a problem of that category, in addition to the divisions in the team that we can do to carry out the work together, such as two people carrying out the creation of the algorithm, applying the logic and taking into account both the restrictions and variables that it would have, and a person carrying out its implementation in Python code in this case.

We also learned the degrees of difficulty that these problems can have for a machine or rather to carry out their implementation for a machine, for example for a human being it is very easy to take a distance matrix as a base and start to iterate until you reach a given result, but for a machine it is more complex to know what number to take and in what order, we also found it interesting to change costs or change prices at times, since in these cases in this type of routing problems, the cost is added cumulatively in each of the stops that the vehicle made.

Once the heuristic algorithm has been implemented in Python, we reflect a little about the problems that could be solved with such a program, we are clear that in real life much greater distances are handled than you handle in this project, however, we believe that it is a good basis if you want to implement on a larger scale, for example, determine the delivery routes to follow for parcel vans such as DHL, Fedex, Amazon, etc.

These problems are undoubtedly very interesting and even more so how their implementation is carried out in programming language.

Bibliographic references

Lee, C.-G., Epelman, M., White III, C., & Bozer, Y. (2006). A shortest path approach to the multiple-vehicle routing problem with split pick-ups. *Transportation Research*, 265-284.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research*, 377-386.

Oliva, A. (Agosto de 2005). Memorias adaptativas para el problema de ruteo de vehículos con múltiples viajes. Uruguay: Colibri.

Tognarelli, P. N. (Junio de 2016). Enfoque híbridos para ruteo de vehículos con múltiples viajes,costos dependientes del tiempo y cola en la bodega. Santiago de Chile, Chile.