Comparing Heuristic Algorithm and Brute Force Algorithm on Real Life Data to solve Traveling Salesman Problem (TSP) at Retailo Technologies

Osama Rizwan
Karachi School of Business & Leadership
5/1/22

Contents

Abstract

The Traveling Salesman Problem (TSP) is a crucial problem that got associated with NP-hard problem by Karp in 1972 at the early stages of development of NP-completeness theory (Michael Jünger, 1995). The TSP is a well-known algorithmic problem that resides in the domain of operations research & computer science studies. It primarily focuses upon finding the shortest yet most optimum route for a salesman to take given a set of locations (Ma, 2020). This paper presents a comparative study between two core algorithms for instance Brute-Force Algorithm & Heuristic Approach (Genetic Algorithm) on the real-life data points made available by Retailo Technologies and highlight the implementation of the most efficient algorithm to find the shortest yet most efficient route to travel within a city. The results would be shared alongside with the final datapoints collection and methodology.

Introduction

Traveling Salesman Problem

The Traveling Salesman Problem has been identified as the most NP-hard problem and got its association to NP-completeness theory back in 1972. It is highlighted as an NP hard problem majorly because there seems to be no guarantee to obtain the most efficient route and no exact algorithm to solve it in polynomial time. Over the course of many decades, TSP has been linked to many domains outside salesman journey such as vehicle routing problems, logistics, scheduling and planning within multiple streams of profession.

Retailo Technologies

Retailo Technologies is a recently established and fast-growing online b2b marketplace in MENAP region that aims to empower 10 million retailers through easy stocking facility along with free delivery facility. It’s transforming the retail industry through digitalizing and supercharging the mainstream business flow and eliminating the middlemen who would exploit the retailers by procuring a large amount of inventory and dominating the prices by levering the supply. Currently Retailo operates in Pakistan and Saudi Arabia and provides it service within 6 cities.

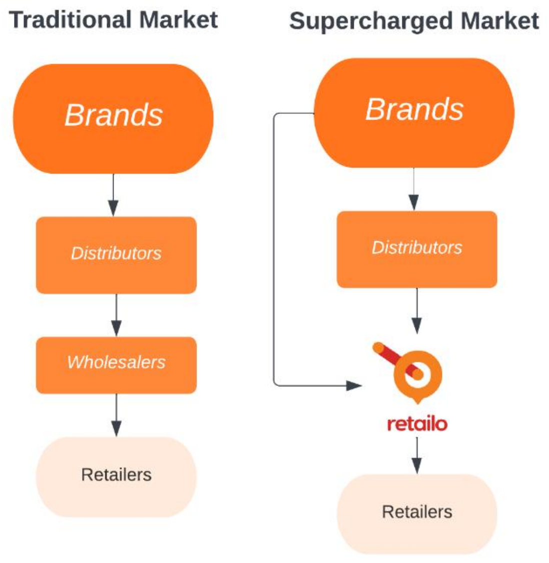


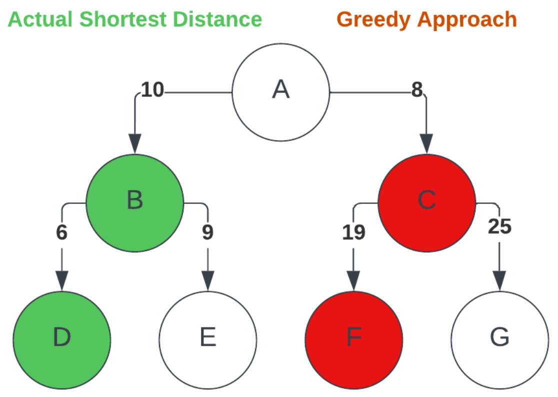
Figure on the right summarizes the business flow of Retailo quite comprehensively. This image is a simple representation of what Retailo does. In the traditional market, distributors would play a major role in managing the supply distribution of products in any specific market. The wholesalers would procure the stock from distributor at any specific price which the distributor would charge and then place a Trade Price onto the retailers which would result in less margin for the retailers to gain as for the most products the MRP remains the same.

However, in the supercharged market where Retailo exists. It is trying to abridging the stock flow in the market which would eliminate the wholesalers who cash in a lot by increasing the margins in Trade Price. Since Retailo focuses upon the inefficiencies of the wholesalers and try to benefit itself by working on those inefficiencies, it has developed itself a model that is cost effective and allows itself to pass on the margins to the retailers. One of the reasons for being cost effective in the market is due to the reasons that Retailo procures multiple different SKUs in the warehouse, whereas Wholesalers are generally allocated to work upon only one specific SKU. This way Retailo ensures that the space is highly optimized and the cost per unit decreases when an order is delivered. Also, the diagram illustrates that for some SKUs’ procurement, Retailo is in direct contact with the brands, this results in the cut off of some distributors and more margins for the retailers at the end of the day.

Problem Identification

This research paper focuses to solve one of the problems at Retailo which is highly related to Traveling Salesman Problem. Retailo has around 200 sales agents that are on field and they are primarily required to book orders from the shops that are assigned to them. On average a sales agent is typically required to visit 15 shops in a given day with an over of 90 unique shops in a whole month (the number varies depending upon the segment the agent works).

Once the shops are assigned the agent will commence his route to visit those 30 shops that would be visible on his mobile application. Once a customer has been visited, the customer would be marked as visited and the agent would continue his route by looking at the next possible close customer to his current location. This method is similar to what we call 'Greedy Approach'. The Greedy Approach Algorithm looks for the local optima whilst optimizing the local best solution to reach the global optima (Raju, 2020). To better understand, the Greedy Approach illustrated on the right seeks to find the route with shortest distance. It does this by selecting the shortest available number at each step. The greedy approach fails to find the shortest route, because it makes it decision on each step without taking into consideration the whole overall problem (Karleigh Moore, 2022).



In terms of Retailo, the overall journey optimization would allow Retailo to save cost & time for the agents. Currently as explained above, Greedy Approach fails to satisfy in this particular situation, hence the remaining study will work upon utilizing other approaches like Brute-Force Algorithm & Genetic Algorithm see whether the route decreases in terms of distance or not.

Brute Force Algorithm

When thinking to solve TSP, the most basic method that might come to mind is called Brute Force Algorithm. This algorithm primarily focuses upon generating all possible combinations or routes and compute their respective distances. The route with the least computed distance is considered to be an optimal route (Saiyed, 2012). The number of combinations is simply illustrated through using ***n! = 1*2*3*….*(n – 1)*n***

Given the dataset provided, we will try to compute the optimal route in methodology section.

Nevertheless, the Brute Force Algorithm is highly used with the domain of computer science where new approaches are targeted through this algorithm to get new insights and developments. Some of the core benefits are highlighted below for the usage of Brute Force to solve TSP:

- It will provide the shortest route without any doubt. When an algorithm selects all the possible outcomes into consideration, it is highly likely to give the shortest route available.
- It is easier to conceptualize the algorithm as it just looks at all possible outcomes and bring the best solution. (Case Study: Solving the Traveling Salesman Problem, 2012)

However, this algorithm is slow though it certainly always reaches the global minima. As the number of combination or location increases the algorithm starts to take more time to compute as the combinations amongst them increases exponentially.

The Genetic Algorithm

It is a search heuristic algorithm that is inspired by Charles Darwin's theory of evolution where the game is all about fittest individuals as they are selected for the reproduction in order to produce offspring. It is one of the techniques to find approximate solutions to combat optimization problems & to solve complex problems that would take a long time so solve (Genetic Algorithm in Machine Learning, n.d.).

In this paper, we propose and test a new crossover operator for traveling salesman problem to minimize the total distance.

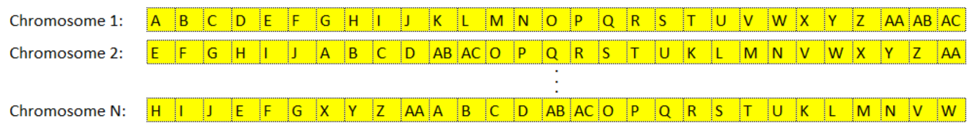
Standard practice for genetic algorithm would be followed to solve the TSP as listed below (Traveling Salesman Problem using Genetic Algorithm, 2021):

1. Creating or initializing the population
2. Calculating fitness
3. Selecting & identifying the best genes
4. Crossing over
5. Mutating to introduce the variations

Create & Initialize the Population (With Retailo's example)

The creation of population starts with randomly generated **k chromosomes**, which represents the population. Each chromosome would represent an individual solution in the population.

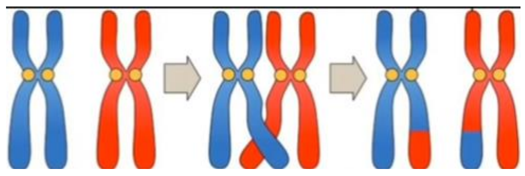
Let's assume that there are 30 possible retailers whom an agent has to visit in a particular day. The number of possible outcomes equals to 30! = 2.6525286e+32. Combined possible outcomes can be denoted as a total population where as each path can be identified as a single chromosome.



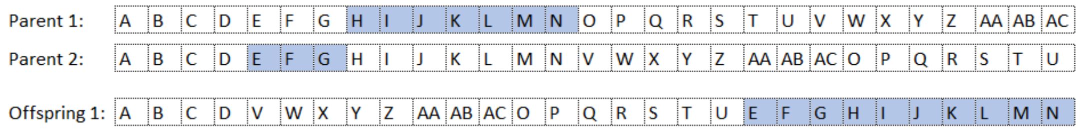
Chromosome 1:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
Chromosome 2:	E	F	G	H	I	J	A	B	C	D	AB	AC	O	P	Q	R	S	T	U	K	L	M	N	V	W	X	Y	Z	AA
															⋮														
Chromosome N:	H	I	J	E	F	G	X	Y	Z	AA	A	B	C	D	AB	AC	O	P	Q	R	S	T	U	K	L	M	N	V	W

Crossover

Crossovers are genetic reproduction of the chromosomes to create a new population from the previous generation. The idea is to swipe some part of the two parents (two chromosomes) and create an offspring. This process is iteratively repeated until the population is totally new and different from the previous one. On the right we can see how two chromosomes performed crossover and, in the end, blue chromosome consists of some part of red where as the red consists some part of blue.



On the right, we can see that offspring has been created using crossover of two parents. The blue areas aka the customers were selected from the parents and have been shifted to the end of tail on the offspring. This would create a new path and would allow a new generation to flow in.

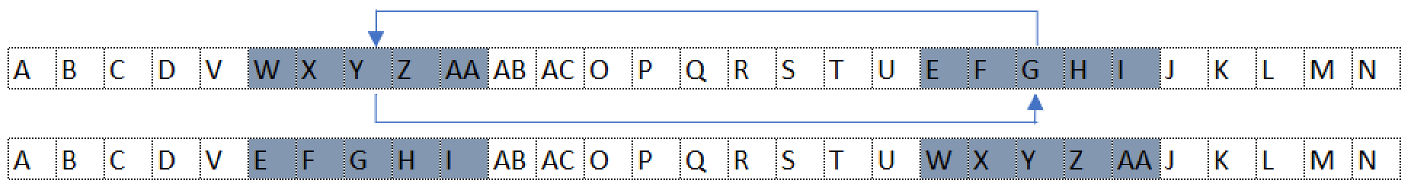


Parent 1:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
Parent 2:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	V	W	X	Y	Z	AA	AB	AC	O	P	Q	R	S	T	U
Offspring 1:	A	B	C	D	V	W	X	Y	Z	AA	AB	AC	O	P	Q	R	S	T	U	E	F	G	H	I	J	K	L	M	N

Mutation

Mutation involves selection on a random bases where certain alteration and changes are made in the genes of the chromosomes. This would provide a random new chromosome to generate within the Genetic Algorithm search process so that more of the search space is considered (Murray-Smith, 2012). Also, mutation is performed after crossover to prevent all solutions to fall into local optimum (Avni Rexhepi, 2013).

In the case of Retailo, swap mutation has been identified to generate the new population. Following image shows, how the genes are swapped within the chromosomes:


A B C D V W X Y Z AA AB AC O P Q R S T U E F G H I J K L M N
A B C D V E F G H I AB AC O P Q R S T U W X Y Z AA J K L M N

In the image above, we can see a new chromosome has been generated by simply swapping the customers within the chromosome. This will create a new path for the agent to travel with its own path cost which is the traveling distance. The idea is to not get trapped in local optimum and keep on creating new generation with the shortest distance available.

Literature Review

This paper presents the comparative analysis between two most widely known algorithm to solve Traveling salesman Problem, Brute Force Algorithm and Heuristic Algorithm (Genetic Algorithm) on a real-life data which has been presented by Retailo Technologies which is a startup based out of Saudi Arabia.

Everyday Retailo incurs a cost of operation that is relevant and direct for Retailo to maintain the business and sustain in the business. Retailo's agents are the key to success and contribute the highest in the overall month Sales (GMV). With the rising inflation and investors moving their investments in more secured assets, it's inevitable to reduce cost. The TSP problem has been identified as one of the problems where inefficiencies exist and achieving them would allow businesses to reduce cost and time significantly.

There have been many studies associated with Traveling Salesman Problem (TSP). Some of the relevant work that has been completed on the same issue related to minimizing the traveling time for a salesman have been identified below along with the gaps and themes that can however be challenged upon.

In 2014, Sonam K. & Dr. Goswami concluded a research paper which constitutes of solving Traveling Salesman Problem (TSP) using a metaheuristic approach called Genetic Algorithm. Genetic Algorithm is an excellent approach to find out a good solution which also depends on how the problem has been encoded and what crossover and mutations are used. They concluded that the Genetic Algorithm will not provide the most minimum value but the solution could be the optimum global value.

In 2014, Antima Sahalot & Sapna Shrimali presented a research paper that stated that Brute Force method always work if given enough time & care. They also stated that since Brute Force method looks at all the possibilities to compare, it is convenient for small number of instances.

In 2009, Zicheng Wang along with two other authors proposed in their comparative study for TSP that a two-stage simulated annealing algorithm has presented a better result than four recent algorithms such as ant colony optimization algorithm, self-organization maps algorithm, particular swarm algorithm & constructive optimizer neural network algorithm. The SA algorithm is inspired by evolutionary theory and is another form of heuristic approach. In SA algorithm, the temperature is equivalent to the randomness amongst the instances, the higher the temperature, larger random chances are created, hence avoiding the risk of becoming trapped in local minima.

One of the most interesting research papers that highlighted about dynamic traveling salesman problem written by Zizhen Zhang and three other authors, expresses how Traveling salesman can encounter an issue of traffic and customers' requested time (Zizhen Zhang, 2021). While the general research papers solely work upon standard variables like fixed distances and other parameters like assuming in Retailo's case that the customer is available, this research paper by Zizhen has created a massive impact that could be counted as one of the best working forward for Retailo. The agents visiting shops should be dynamic as the traffic plays a significant part in the distance calculation. A short route might have more traffic than a longer route, which could impact the fuel consumption.

Data Fetching & Modification

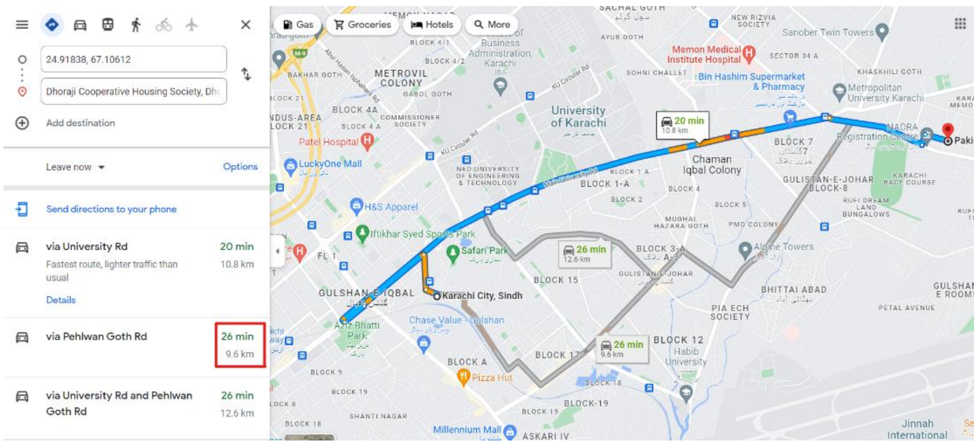
Data points gathered for the research includes customer id along with their respective latitude & longitude. The data reflects a journey route for an agent who belongs from *Wholesaler* segment, which means that the overall route is shorter than the rest of the agents' route who do not belong from this particular segment. This is primarily because of two core reasons mentioned below:

1. There are fewer wholesalers as compared to retailers
2. Since wholesalers' contribution is significant in the overall GMV (Revenue), Retailo has a set number of agents who are specialized and trained. Moreover, for relationship building purpose, a smaller number of shops are assigned to have better focus resulting in better correspondence.

On the right, the customer IDs along with their respective latitudes and longitudes are available. However, one of the cost indicators is missing from this dataset. That is the distance between each shop with another shop. This work had to be done manually through google maps as shown below. Where each shop's lat/long is compared with another shop's lat/long to find the minimum distance available between them and so on.

Once all of the customers' distances were calculated, we were able to formula customer distance matrix as shown below:

Customer Id	longitude	latitude
0	67.106122	24.918383
1	67.1720299	24.9365715
2	67.1411283	24.8809269
3	67.100613	24.9089314
4	67.1388983	24.8805672
5	67.14077778	24.88103722
6	67.142662	24.8808336
7	67.10035756	24.90048165
8	67.1070123	24.9169072
9	67.100516	24.900486
10	67.09533881	24.904709
11	67.1040059	24.9014026
12	67.1390761	24.8842676
13	67.1403859	24.8795902


24.91838, 67.10612
Dhoraji Cooperative Housing Society, Dhc
Add destination
Leave now
Options
Send directions to your phone
via University Rd — 20 min — 10.8 km
Fastest route, lighter traffic than usual
Details
via Pehlwan Goth Rd — 26 min — 9.6 km
via University Rd and Pehlwan Goth Rd — 26 min — 12.6 km

Customer Matrix														
#	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	9.6	7.3	1.9	7.4	7.3	7.5	4.5	1.9	4.5	2.8	3	6.8	7.4
1	9.6	0	11.8	9.4	12.1	11.9	11.7	11.7	9.5	11.7	11	10.6	11.8	11.85
2	7.3	11.8	0	6.7	0.29	0.057	0.16	7.7	7.1	7.8	7.81	7.82	0.6	0.22
3	1.9	9.4	6.7	0	6.5	6.4	6.6	1.3	2.5	1.283	0.95	1.1	5.6	6.5
4	7.4	12.1	0.29	6.5	0	0.26	0.45	7.7	7.1	7.7	7.8	7.82	0.55	0.24
5	7.3	11.9	0.057	6.4	0.26	0	0.21	7.72	7.1	7.76	7.77	7.82	0.5	0.19
6	7.5	11.7	0.16	6.6	0.45	0.21	0	7.9	7.3	7.93	8	8.2	0.75	0.35
7	4.5	11.7	7.7	1.3	7.7	7.72	7.9	0	3.2	0.021	0.75	0.55	5.5	6.1
8	1.9	9.5	7.1	2.5	7.1	7.1	7.3	3.2	0	6.3	2.9	3	6	6.6
9	4.5	11.7	7.8	1.283	7.7	7.76	7.93	0.021	6.3	0	0.8	0.55	5.5	6.1
10	2.8	11	7.81	0.95	7.8	7.77	8	0.75	2.9	0.8	0	1.2	6.2	6.8
11	3	10.6	7.82	1.1	7.82	7.82	8.2	0.55	3	0.55	1.2	0	5.1	5.7
12	6.8	11.8	0.6	5.6	0.55	0.5	0.75	5.5	6	5.5	6.2	5.1	0	0.65
13	7.4	11.85	0.22	6.5	0.24	0.19	0.35	6.1	6.6	6.1	6.8	5.7	0.65	0

Assumptions:

1. Customer ID 1 represents the starting position of the agent which means that the ending note has to be Customer ID 1 in order to end the journey.
2. The distances provided above in the Customer Distance Matrix are in ***Kilometers.***
3. The distance selected between each customer from the recommended distances is the most minimum one or the route where vehicles are allowed to travel.
    a. This was done primarily because Google Map would recommend distances that are longer because there was less traffic, so had to normalize the distances for all the data points.
    b. It was recommending routes which were only oriented to walk, hence no vehicles allowed.

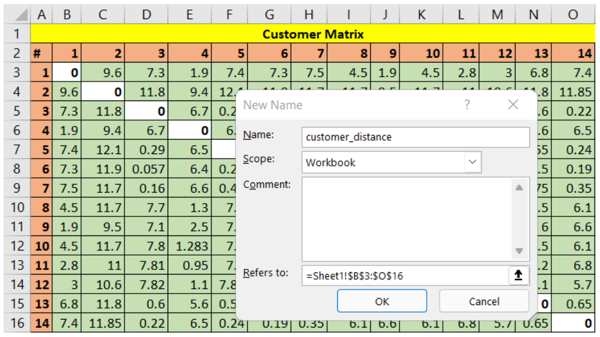
Data Rambling

In this section, we will test out different methods and algorithms to our given dataset primarily because we want to calculate the shortest distance and see which method provided us the fastest solution as well. Remember the TSP problem is a NP hard problem which means that there is no one best answer when taken time into consideration.

Excel Approach

In order to find the minimum distance for the sales agent, we would first test out an excel method using Solver for our TSP (Jiang, 2010). Following are the steps for excel approach:

Step 1:



New Name

Name: customer_distance

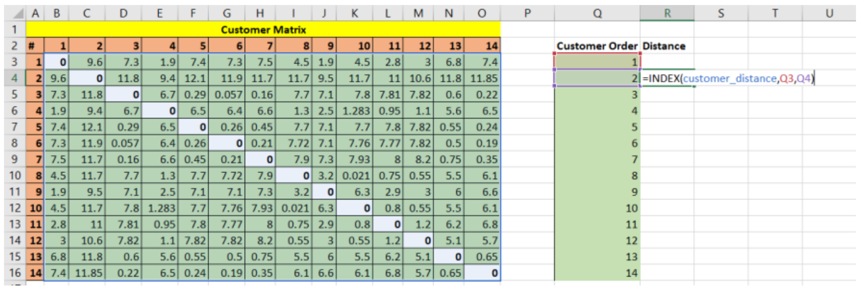
Scope: Workbook

Comment:

Refers to: =Sheet1!$B$3:$O$16

Initializing the name of the dataset to 'customer_distance' from cell B3 to O16.

Step 2:

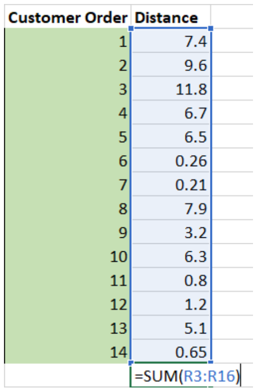


Customer Matrix														
#	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	9.6	7.3	1.9	7.4	7.3	7.5	4.5	1.9	4.5	2.8	3	6.8	7.4
2	9.6	0	11.8	9.4	12.1	11.9	11.7	11.7	9.5	11.7	11	10.6	11.8	11.85
3	7.3	11.8	0	6.7	0.29	0.057	0.16	7.7	7.1	7.8	7.81	7.82	0.6	0.22
4	1.9	9.4	6.7	0	6.5	6.4	6.6	1.3	2.5	1.283	0.95	1.1	5.6	6.5
5	7.4	12.1	0.29	6.5	0	0.26	0.45	7.7	7.1	7.7	7.8	7.82	0.55	0.24
6	7.3	11.9	0.057	6.4	0.26	0	0.21	7.72	7.1	7.76	7.77	7.82	0.5	0.19
7	7.5	11.7	0.16	6.6	0.45	0.21	0	7.9	7.3	7.93	8	8.2	0.75	0.35
8	4.5	11.7	7.7	1.3	7.7	7.72	7.9	0	3.2	0.021	0.75	0.55	5.5	6.1
9	1.9	9.5	7.1	2.5	7.1	7.1	7.3	3.2	0	6.3	2.9	3	6	6.6
10	4.5	11.7	7.8	1.283	7.7	7.76	7.93	0.021	6.3	0	0.8	0.55	5.5	6.1
11	2.8	11	7.81	0.95	7.8	7.77	8	0.75	2.9	0.8	0	1.2	6.2	6.8
12	3	10.6	7.82	1.1	7.82	7.82	8.2	0.55	3	0.55	1.2	0	5.1	5.7
13	6.8	11.8	0.6	5.6	0.55	0.5	0.75	5.5	6	5.5	6.2	5.1	0	0.65
14	7.4	11.85	0.22	6.5	0.24	0.19	0.35	6.1	6.6	6.1	6.8	5.7	0.65	0

Customer Order	Distance
1	
2	=INDEX(customer_distance,Q3,Q4)
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

Calculate the distance between each consecutive datapoints by using index function. This is done showing the following image on the right. The idea is to get distance of current city from the previous city. Please note, that for customer order with value 1 will be calculated through comparing the distance journey from last customer point, this is because we want the journey to get completed when the agent gets back to the home.

Step 3:



Customer Order	Distance
1	7.4
2	9.6
3	11.8
4	6.7
5	6.5
6	0.26
7	0.21
8	7.9
9	3.2
10	6.3
11	0.8
12	1.2
13	5.1
14	0.65
	=SUM(R3:R16)

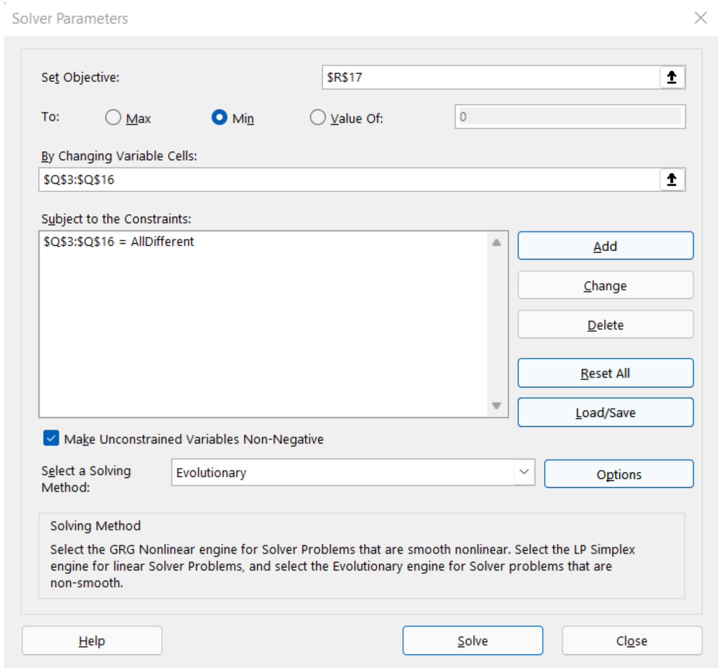
The diagram on the left illustrates one of the possibilities of customer pattern with a distance of each customer from the previous customer visited.

Also, note that after the solver is run on this particular table, the column name 'Customer Order' will change providing the shortest route.

For the current route as shown, the total distance covered is 67.62 Kms. The total distance cell is also going to be our Set Objective from Solver.

Step 4:

In this particular step, we are going to use solver from *Data > Analyze > Solver.*


Solver Parameters

Set Objective: $R$17

To: Max [✓ Min] Value Of: 0

By Changing Variable Cells: $Q$3:$Q$16

Subject to the Constraints:

$Q$3:$Q$16 = AllDifferent

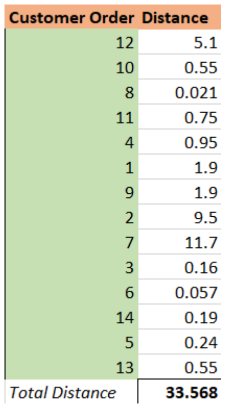
[✓] Make Unconstrained Variables Non-Negative

Select a Solving Method: Evolutionary

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

1. Setting the Objective cell is required because it is the cell where our answer would be visible.
2. Since we are working to find the shortest distance, it is required to select Min.
3. As mentioned above, the 'Customer Order' column would be selected in the field 'By Changing Variable Cells'.
4. A single constraint has been added which is stating that Customer Order column should be all different. If this is not activated then it will only show 1 in all cells as it will create the smallest distance.
5. The solving method has been selected as evolutionary. This is primarily because evolutionary method is based on Natural Selection Theory which is inspired by the survival of fittest. In this the model is plugged in with the input values and a target value is defined. The set of input values that are closest to the target value (min. distance) will be selected and a second population of offspring are selected (EXCEL SOLVER: WHICH SOLVING METHOD SHOULD I CHOOSE?, 2022)
6. Click 'Solve'.



Customer Order	Distance
12	5.1
10	0.55
8	0.021
11	0.75
4	0.95
1	1.9
9	1.9
2	9.5
7	11.7
3	0.16
6	0.057
14	0.19
5	0.24
13	0.55
Total Distance	**33.568**

Step 5:

Compare the results. Overall, the iteration process of solver took about 25 seconds to provide the minimum distance. We can see a severe reduction in the total distance for the overall route journey. This would reduce the overall cost and save time for the agent to an extensive degree.

BestRoute: 1-9-2-7-3-6-14-5-13-12-10-8-11-4-1

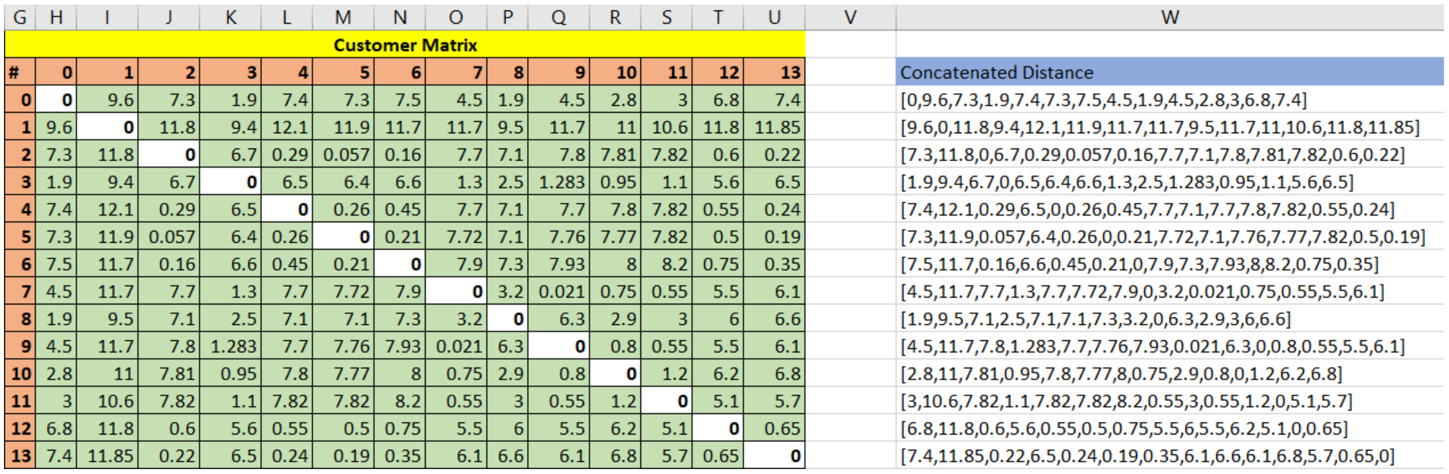
Distance: ***33.568 kms***

Brute-Force Algorithm Approach

As discussed earlier regarding Brute Force Approach. It is a popular approach to solve TSP as it provides accurate answer but has the limitation of time as it would go through all the possibilities in order to come up with a minimum distance.

Modification in the data

As per the algorithm, the routes are required to be placed within squared bracket with comma separation, as shown in figure below.



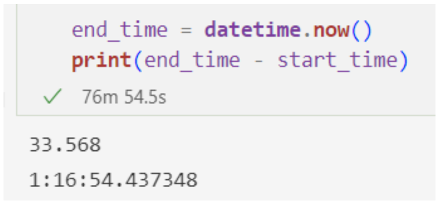
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Customer Matrix																
#	0	1	2	3	4	5	6	7	8	9	10	11	12	13		Concatenated Distance
0	0	9.6	7.3	1.9	7.4	7.3	7.5	4.5	1.9	4.5	2.8	3	6.8	7.4		[0,9.6,7.3,1.9,7.4,7.3,7.5,4.5,1.9,4.5,2.8,3,6.8,7.4]
1	9.6	0	11.8	9.4	12.1	11.9	11.7	11.7	9.5	11.7	11	10.6	11.8	11.85		[9.6,0,11.8,9.4,12.1,11.9,11.7,11.7,9.5,11.7,11,10.6,11.8,11.85]
2	7.3	11.8	0	6.7	0.29	0.057	0.16	7.7	7.1	7.8	7.81	7.82	0.6	0.22		[7.3,11.8,0,6.7,0.29,0.057,0.16,7.7,7.1,7.8,7.81,7.82,0.6,0.22]
3	1.9	9.4	6.7	0	6.5	6.4	6.6	1.3	2.5	1.283	0.95	1.1	5.6	6.5		[1.9,9.4,6.7,0,6.5,6.4,6.6,1.3,2.5,1.283,0.95,1.1,5.6,6.5]
4	7.4	12.1	0.29	6.5	0	0.26	0.45	7.7	7.1	7.7	7.8	7.82	0.55	0.24		[7.4,12.1,0.29,6.5,0,0.26,0.45,7.7,7.1,7.7,7.8,7.82,0.55,0.24]
5	7.3	11.9	0.057	6.4	0.26	0	0.21	7.72	7.1	7.76	7.77	7.82	0.5	0.19		[7.3,11.9,0.057,6.4,0.26,0,0.21,7.72,7.1,7.76,7.77,7.82,0.5,0.19]
6	7.5	11.7	0.16	6.6	0.45	0.21	0	7.9	7.3	7.93	8	8.2	0.75	0.35		[7.5,11.7,0.16,6.6,0.45,0.21,0,7.9,7.3,7.93,8,8.2,0.75,0.35]
7	4.5	11.7	7.7	1.3	7.7	7.72	7.9	0	3.2	0.021	0.75	0.55	5.5	6.1		[4.5,11.7,7.7,1.3,7.7,7.72,7.9,0,3.2,0.021,0.75,0.55,5.5,6.1]
8	1.9	9.5	7.1	2.5	7.1	7.1	7.3	3.2	0	6.3	2.9	3	6	6.6		[1.9,9.5,7.1,2.5,7.1,7.1,7.3,3.2,0,6.3,2.9,3,6,6.6]
9	4.5	11.7	7.8	1.283	7.7	7.76	7.93	0.021	6.3	0	0.8	0.55	5.5	6.1		[4.5,11.7,7.8,1.283,7.7,7.76,7.93,0.021,6.3,0,0.8,0.55,5.5,6.1]
10	2.8	11	7.81	0.95	7.8	7.77	8	0.75	2.9	0.8	0	1.2	6.2	6.8		[2.8,11,7.81,0.95,7.8,7.77,8,0.75,2.9,0.8,0,1.2,6.2,6.8]
11	3	10.6	7.82	1.1	7.82	7.82	8.2	0.55	3	0.55	1.2	0	5.1	5.7		[3,10.6,7.82,1.1,7.82,7.82,8.2,0.55,3,0.55,1.2,0,5.1,5.7]
12	6.8	11.8	0.6	5.6	0.55	0.5	0.75	5.5	6	5.5	6.2	5.1	0	0.65		[6.8,11.8,0.6,5.6,0.55,0.5,0.75,5.5,6,5.5,6.2,5.1,0,0.65]
13	7.4	11.85	0.22	6.5	0.24	0.19	0.35	6.1	6.6	6.1	6.8	5.7	0.65	0		[7.4,11.85,0.22,6.5,0.24,0.19,0.35,6.1,6.6,6.1,6.8,5.7,0.65,0]

Brute Force Algorithm:

```
from sys import maxsize
from itertools import permutations
V = 14

# implementation of traveling Salesman Problem
def travellingSalesmanProblem(graph, s):

    # store all vertex apart from source vertex
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)

    # store minimum weight Hamiltonian Cycle
    min_path = maxsize
    next_permutation=permutations(vertex)
    for i in next_permutation:

        # store current Path weight(cost)
        current_pathweight = 0

        # compute current path weight
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]

        # update minimum
        min_path = min(min_path, current_pathweight)

    return min_path

# Driver Code
if __name__ == "__main__":

    # matrix representation of graph
    graph = [[0,0.8,5.8,1.9,0.65,6.6,1.4,1.7,2.2,6.5,1.8,1.6,1.3,1.3,10,],
[0.8,0,5.7,1.3,0.9,7,0.75,0.95,1.4,6.9,3.2,2,1.9,1.7,10.5,],
[5.8,5.7,0,7,4.6,0.85,6.7,7,6.9,0.7,3.5,6.4,6.5,4.3,10.8,],
[1.9,1.3,7,0,2.7,6,0.75,0.75,0.25,5.9,2.1,4.7,3.6,2.2,10.5,],
[0.65,0.9,4.6,2.7,0,6.5,1.6,2,2.4,6.4,1.3,1.6,2.1,0.85,10,],
[6.6,7,0.85,6,6.5,0,7.5,7.8,7.7,0.45,4.3,7.2,7.4,7,10.8,],
[1.4,0.75,6.7,0.75,1.6,7.5,0,0.18,1,6.5,3.7,3.2,2.6,2.4,10.6,],
[1.7,0.95,7,0.75,2,7.8,0.18,0,1,6.7,3.8,2.9,2.9,3.4,10.6,],
[2.2,1.4,6.9,0.25,2.4,7.7,1,1,0,5.6,1.8,3.3,3.4,2,10.2,],
[6.5,6.9,0.7,5.9,6.4,0.45,6.5,6.7,5.6,0,5.2,7,7.2,6.9,11.2,],
[1.8,3.2,3.5,2.1,1.3,4.3,3.7,3.8,1.8,5.2,0,3,3.7,0.95,9,],
[1.6,2,6.4,4.7,1.6,7.2,3.2,2.9,3.3,7,3,0,1.9,1.8,9.4,],
[1.3,1.9,6.5,3.6,2.1,7.4,2.6,2.9,3.4,7.2,3.7,1.9,0,2.6,9.6,],
[1.3,1.7,4.3,2.2,0.85,7,2.4,3.4,2,6.9,0.95,1.8,2.6,0,8.5,],
[10,10.5,10.8,10.5,10,10.8,10.6,10.6,10.2,11.2,9,9.4,9.6,8.5,0,]
]
    s = 0
    print(travellingSalesmanProblem(graph, s))
```

The results:

Brute Force algorithm had a success factor associated to it as it was able to find the shortest distance possible. However, it took 1 hour 16 minutes and 54 seconds to compute the answer; whereas, the evolutionary solver method above only took 20 seconds to find the same answer: 33.568 Kms.



```
end_time = datetime.now()
print(end_time - start_time)
✓ 76m 54.5s

33.568
1:16:54.437348
```

The Genetic Algorithm

Genetic algorithm as explained earlier would be utilized on the segments that are mostly related to retailers. Retailers are more in quantity and have more to offer as compare to other segments as they are more in number.

Overall, on average an agent visits around 29 to 30 shops per day and hence we have selected an agent named 'Danish Mustafa' and his route-based customers for Monday.

Cust_name	Cust_index	customer_phone	latitude	longitude	Coordinates
A	1	923072904443	24.84967	67.05083	24.8496651,67.0608296
B	2	923102844184	24.85436	67.04863	24.854359,67.0286335
C	3	923142978989	24.84687	67.05349	24.8488653,67.0764864
D	4	923003628676	24.85169	67.05013	24.860689,67.0401254
E	5	923012556575	24.84411	67.04949	24.84111,67.02949
F	6	923438075563	24.84876	67.05325	24.8487596,67.09325

The dataset above represents the 6 customers out of 29 others and their respective details. These are the customers Danish would visit on Monday and we can get the location as well from the dataset.

Data Automation for distance

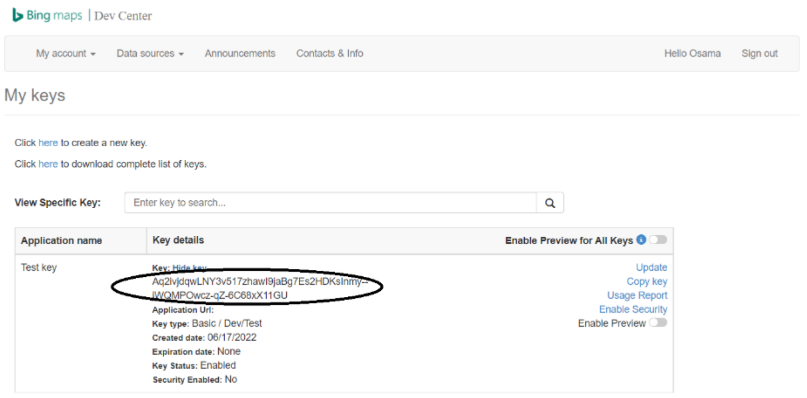
As seen above, we used Google maps to create the customer matrix which shows the distances between each customer's shop. The process was done manually and was limited to time constraint. It could take an hour of manual work only for wholesale agent that visits only 14-15 shops per day. Hence, if we had to do this for retailer-based agents, it would take us a couple of hours just to solve the matrix.

For this purpose, we have automated the dashboard by creating macros that can help extracting the distances amongst the customers within few minutes.

Step 1*: *Create API

To generate the distance from the lat-long of two customers, we need an API assistance from the Maps, either Google or Bing. The API called from Google does not come with free package hence Bing was utilized for this purpose.

The key for the API can be extracted from bingmapsportal.com and can be seen on the imagine on the right.



Bing maps | Dev Center

My account · Data sources · Announcements · Contacts & Info · Hello Osama · Sign out

My keys

Click here to create a new key.

Click here to download complete list of keys.

View Specific Key: Enter key to search...

Application name	Key details	Enable Preview for All Keys
Test key	Key: Hide key Aq2ivjdqwLNY3v517zhawl9jaBg7Es2HDKsInmy--jWQMPOwcz-qZ-6C68xX11GU Application Url: Key type: Basic / Dev/Test Created date: 06/17/2022 Expiration date: None Key Status: Enabled Security Enabled: No	Update Copy key Usage Report Enable Security Enable Preview

Once the key has been generated, we can take this key to our dashboard on Excel.

Source Location	**AC**
Destination Location	**AC**
Source Coordinate	24.839586,67.054656
Destination Coordinate	24.839586,67.054656
key	Aq2ivjdqwLNY3v517zhawl9jaBg7Es2HDKsInmy--jWQMPOwcz-qZ-6C68xX11GU
Distance (km)	0.00

Step 2: Maintain Dashboard

The dataset shown above would be stored in another tab with the name 'real_database'. The working tab would be named 'matrix' and is shown on the left.

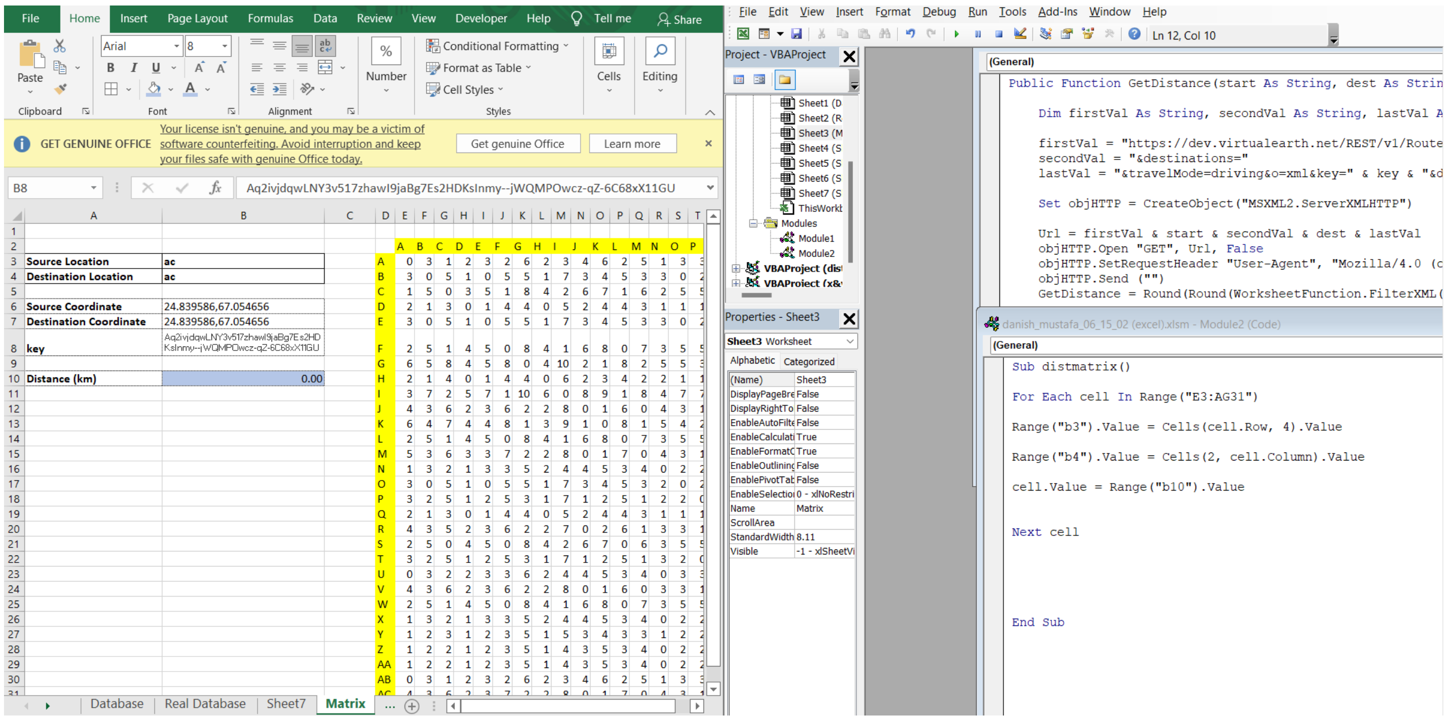
Cust_name would be inserted in the cells defining Source Location and Destination Location, whereas the Source Coordiante cell and Destination Coordinate are dependent on the source location and destination location. So for instance, if source location is 'A' and Destination location is 'C', it would plot the Coordinates in the respective cells below.

The Key cell shows the Key that was generated by the Bing system. For distance in KM, we have used the following formula:

=getDistance("Key", "Source Coordinate", "Destination Coordinate")

The API would utilize the coordinates and then plot the distance between the customers.

Step 3: Creating and Running Macros for Matrix



B8 Aq2ivjdqwLNY3v517zhawI9jaBg7Es2HDKsInmy--jWQMPOwcz-qZ-6C68xX11GU

	A	B
3	Source Location	ac
4	Destination Location	ac
6	Source Coordinate	24.839586,67.054656
7	Destination Coordinate	24.839586,67.054656
8	key	Aq2ivjdqwLNY3v517zhawl9jaBg7Es2HDKsInmy--jWQMPOwcz-qZ-6C68xX11GU
10	Distance (km)	0.00

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	0	3	1	2	3	2	6	2	3	4	6	2	5	1	3
B	3	0	5	1	0	5	5	1	7	3	4	5	3	3	0
C	1	5	0	3	5	1	8	4	2	6	7	1	6	2	5
D	2	1	3	0	1	4	4	0	5	2	4	4	3	1	1
E	3	0	5	1	0	5	5	1	7	3	4	5	3	3	0
F	2	5	1	4	5	0	8	4	1	6	8	0	7	3	5
G	6	5	8	4	5	8	0	4	10	2	1	8	2	5	5
H	2	1	4	0	1	4	4	0	6	2	3	4	2	2	1
I	3	7	2	5	7	1	10	6	0	8	9	1	8	4	7
J	4	3	6	2	3	6	2	2	8	0	1	6	0	4	3
K	6	4	7	4	4	8	1	3	9	1	0	8	1	5	4
L	2	5	1	4	5	0	8	4	1	6	8	0	7	3	5
M	5	3	6	3	3	7	2	2	8	0	1	7	0	4	3
N	1	3	2	1	3	3	5	2	4	4	5	3	4	0	2
O	3	0	5	1	0	5	5	1	7	3	4	5	3	2	0
P	3	2	5	1	2	5	3	1	7	1	2	5	1	2	2
Q	2	1	3	0	1	4	4	0	5	2	4	4	3	1	1
R	4	3	5	2	3	6	2	2	7	0	2	6	1	3	3
S	2	5	0	4	5	0	8	4	2	6	7	0	6	3	5
T	3	2	5	1	2	5	3	1	7	1	2	5	1	3	2
U	0	3	2	2	3	3	6	2	4	4	5	3	4	0	3
V	4	3	6	2	3	6	2	2	8	0	1	6	0	3	3
W	2	5	1	4	5	0	8	4	1	6	8	0	7	3	5
X	1	3	2	1	3	3	5	2	4	4	5	3	4	0	2
Y	1	2	3	1	2	3	5	1	5	3	4	3	3	1	2
Z	1	2	2	1	2	3	5	1	4	3	5	3	4	0	2
AA	1	2	2	1	2	3	5	1	4	3	5	3	4	0	2
AB	0	3	1	2	3	2	6	2	3	4	6	2	5	1	3

(Name)	Sheet3
DisplayPageBre	False
DisplayRightTo	False
EnableAutoFilte	False
EnableCalculati	True
EnableFormatC	True
EnableOutlining	False
EnablePivotTab	False
EnableSelectio	0 - xlNoRestri
Name	Matrix
ScrollArea	
StandardWidth	8.11
Visible	-1 - xlSheetVi

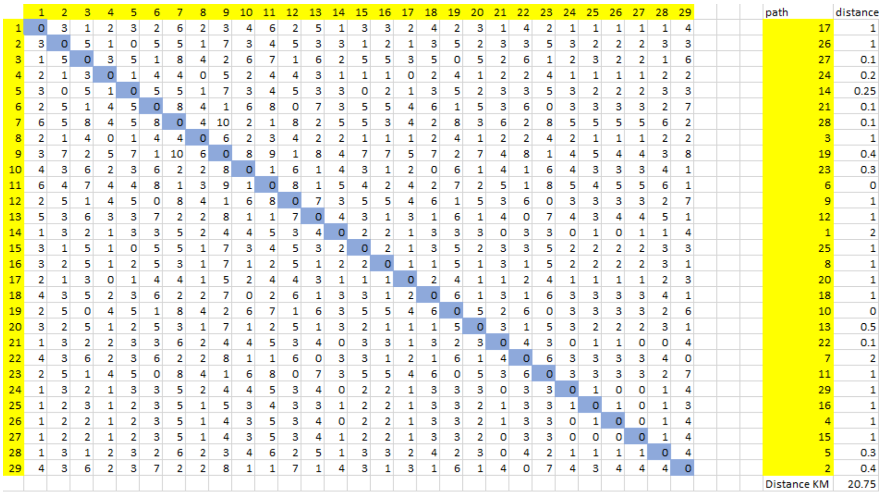
```
Public Function GetDistance(start As String, dest As Strin

    Dim firstVal As String, secondVal As String, lastVal A

    firstVal = "https://dev.virtualearth.net/REST/v1/Route
    secondVal = "&destinations="
    lastVal = "&travelMode=driving&o=xml&key=" & key & "&d

    Set objHTTP = CreateObject("MSXML2.ServerXMLHTTP")

    Url = firstVal & start & secondVal & dest & lastVal
    objHTTP.Open "GET", Url, False
    objHTTP.SetRequestHeader "User-Agent", "Mozilla/4.0 (c
    objHTTP.Send ("")
    GetDistance = Round(Round(WorksheetFunction.FilterXML(
```

```
Sub distmatrix()

For Each cell In Range("E3:AG31")

Range("b3").Value = Cells(cell.Row, 4).Value

Range("b4").Value = Cells(2, cell.Column).Value

cell.Value = Range("b10").Value


Next cell



End Sub
```

The above image illustrates the macro running and computing the distances between each customer. The macro allowed us to significantly reduce the time taken for the overall matrix to get completed. It only took 13 mins to get it completed via macros.

The final matrix can be seen on the right.

The **Evolutionary** Algorithm calculated the shortest route at ***20.75 kms*** for the agent.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	path	distance
1	0	3	1	2	3	2	6	2	3	4	6	2	5	1	3	3	2	4	2	3	1	4	2	1	1	1	1	1	4	17	1
2	3	0	5	1	0	5	5	1	7	3	4	5	3	3	1	2	1	3	5	2	3	3	5	3	2	2	2	3	3	26	1
3	1	5	0	3	5	1	8	4	2	6	7	1	6	2	5	5	3	5	0	5	2	6	1	2	3	2	2	1	6	27	0.1
4	2	1	3	0	1	4	4	0	5	2	4	4	3	1	1	1	0	2	4	1	2	2	4	1	1	1	1	2	2	24	0.2
5	3	0	5	1	0	5	5	1	7	3	4	5	3	3	0	2	1	3	5	2	3	3	5	3	2	2	2	3	3	14	0.25
6	2	5	1	4	5	0	8	4	1	6	8	0	7	3	5	5	4	6	1	5	3	6	0	3	3	3	3	2	7	21	0.1
7	6	5	8	4	5	8	0	4	10	2	1	8	2	5	5	3	4	2	8	3	6	2	8	5	5	5	5	6	2	28	0.1
8	2	1	4	0	1	4	4	0	6	2	3	4	2	2	1	1	1	2	4	1	2	2	4	2	1	1	1	2	2	3	1
9	3	7	2	5	7	1	10	6	0	8	9	1	8	4	7	7	5	7	2	7	4	8	1	4	5	4	4	3	8	19	0.4
10	4	3	6	2	3	6	2	2	8	0	1	6	1	4	3	1	2	0	6	1	4	1	6	4	3	3	3	4	1	23	0.3
11	6	4	7	4	4	8	1	3	9	1	0	8	1	5	4	2	4	2	7	2	5	1	8	5	4	5	5	6	1	6	0
12	2	5	1	4	5	0	8	4	1	6	8	0	7	3	5	5	4	6	1	5	3	6	0	3	3	3	3	2	7	9	1
13	5	3	6	3	3	7	2	2	8	1	1	7	0	4	3	1	3	1	6	1	4	0	7	4	3	4	4	5	1	12	1
14	1	3	2	1	3	3	5	2	4	4	5	3	4	0	2	2	1	3	3	3	0	3	3	0	1	0	1	1	4	1	2
15	3	1	5	1	0	5	5	1	7	3	4	5	3	2	0	2	1	3	5	2	3	3	5	2	2	2	2	3	3	25	1
16	3	2	5	1	2	5	3	1	7	1	2	5	1	2	2	0	1	1	5	1	3	1	5	2	2	2	2	3	1	8	1
17	2	1	3	0	1	4	4	1	5	2	4	4	3	1	1	1	0	2	4	1	1	2	4	1	1	1	1	2	3	20	1
18	4	3	5	2	3	6	2	2	7	0	2	6	1	3	3	1	2	0	6	1	3	1	6	3	3	3	3	4	1	18	1
19	2	5	0	4	5	1	8	4	2	6	7	1	6	3	5	5	4	6	0	5	2	6	0	3	3	3	3	2	6	10	0
20	3	2	5	1	2	5	3	1	7	1	2	5	1	3	2	1	1	1	5	0	3	1	5	3	2	2	2	3	1	13	0.5
21	1	3	2	2	3	3	6	2	4	4	5	3	4	0	3	3	1	3	2	3	0	4	3	0	1	1	0	0	4	22	0.1
22	4	3	6	2	3	6	2	2	8	1	1	6	0	3	3	1	2	1	6	1	4	0	6	3	3	3	3	4	0	7	2
23	2	5	1	4	5	0	8	4	1	6	8	0	7	3	5	5	4	6	0	5	3	6	0	3	3	3	3	2	7	11	1
24	1	3	2	1	3	3	5	2	4	4	5	3	4	0	2	2	1	3	3	3	0	3	3	0	1	0	0	1	4	29	1
25	1	2	3	1	2	3	5	1	5	3	4	3	3	1	2	2	1	3	3	2	1	3	3	1	0	1	0	1	3	16	1
26	1	2	2	1	2	3	5	1	4	3	5	3	4	0	2	2	1	3	3	2	1	3	3	0	1	0	0	1	4	4	1
27	1	2	2	1	2	3	5	1	4	3	5	3	4	1	2	2	1	3	3	2	0	3	3	0	0	0	0	1	4	15	1
28	1	3	1	2	3	2	6	2	3	4	6	2	5	1	3	3	2	4	2	3	0	4	2	1	1	1	1	0	4	5	0.3
29	4	3	6	2	3	7	2	2	8	1	1	7	1	4	3	1	3	1	6	1	4	0	7	4	3	4	4	4	0	2	0.4
																														Distance KM	20.75

Data Coding for Genetic Algorithm

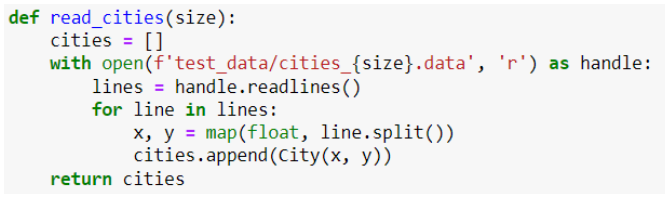
Defining the distance:

On the right, the distance is being calculated using Haversine formula (Program for distance between two points on earth, 2021).

The idea is to get the distance between two coordinates that lie on a sphere so we would be using a simple distance function. The idea is prevent any hypothesis assumption that can be later considered as overall drawback for the analysis.

```
class City:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance(self, city):
#           return math.hypot(self.x - city.x, self.y - city.y)
        dlon = self.y - city.y
        dlat = self.x - city.x
        a = sin(dlat / 2)**2 + cos(city.x) * cos(self.x) * sin(dlon / 2)**2
        dist = 2 * asin(sqrt(a))
        if dist < 1:
            return 2 * asin(sqrt(a))
        else:
            return round(2 * asin(sqrt(a)),2)

    def __repr__(self):
        return f"({self.x}, {self.y})"
```

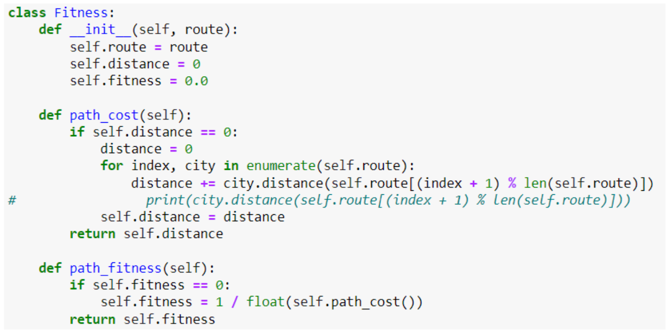
Reading the source file:



```python
def read_cities(size):
    cities = []
    with open(f'test_data/cities_{size}.data', 'r') as handle:
        lines = handle.readlines()
        for line in lines:
            x, y = map(float, line.split())
            cities.append(City(x, y))
    return cities
```

The code on the left is defining a function to extract the latitude and longitude from the source file that is available in test_data folder.

The notepad consists of latitudes and longitudes which have been converted into radian by simply dividing the latitude and longitude by 180/Pi.



```python
class Fitness:
    def __init__(self, route):
        self.route = route
        self.distance = 0
        self.fitness = 0.0

    def path_cost(self):
        if self.distance == 0:
            distance = 0
            for index, city in enumerate(self.route):
                distance += city.distance(self.route[(index + 1) % len(self.route)])
#                print(city.distance(self.route[(index + 1) % len(self.route)]))
            self.distance = distance
        return self.distance

    def path_fitness(self):
        if self.fitness == 0:
            self.fitness = 1 / float(self.path_cost())
        return self.fitness
```

Defining the fitness function:

A fitness function tells us how good each particular route is (for us how short the distance is). So, the shorter the distance, the superior and fitter the chromosome.

Function path_cost is defining the distance of each city with each city and storing it.

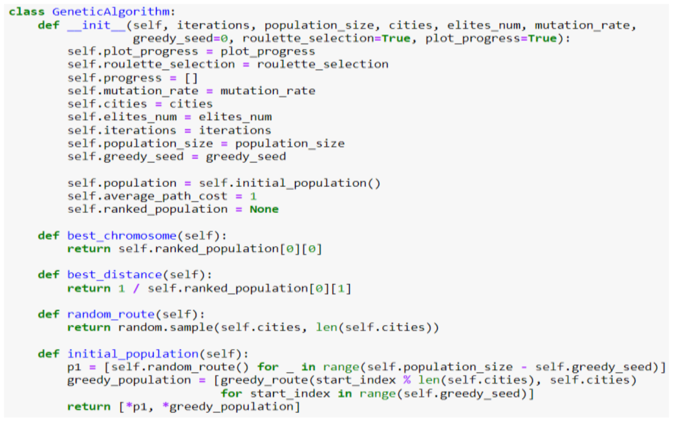
The path fitness would get path cost associated with it, if the path fitness is 0 then we have to divide the cost that is associated with path.

Defining the Genetic Algorithm:

This is initializing process that is taking place. We are identifying the functions for their respective work.

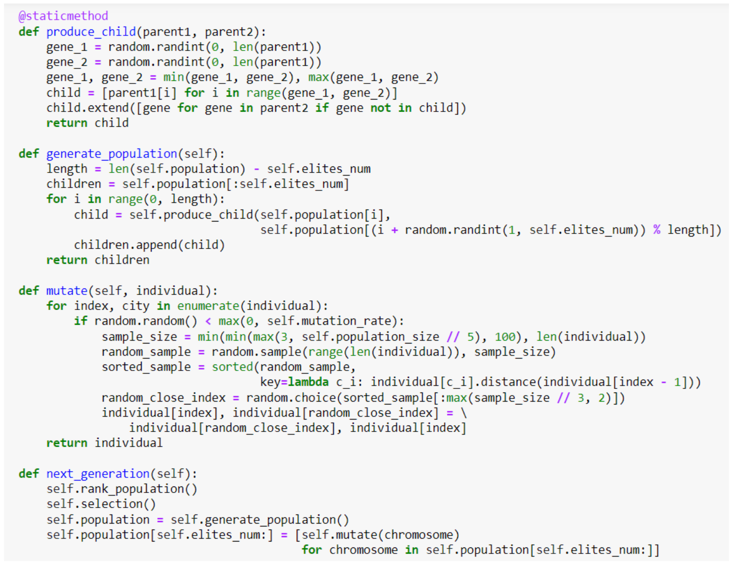
Some of the best functions for this algorithm are defined here as we get the best chromosomes and distances out of the variables.

Function initial population is setting up the starting point to get the population from the file and utilize some other approach like Greedy approach to kick start the basic function.



```python
class GeneticAlgorithm:
    def __init__(self, iterations, population_size, cities, elites_num, mutation_rate,
                 greedy_seed=0, roulette_selection=True, plot_progress=True):
        self.plot_progress = plot_progress
        self.roulette_selection = roulette_selection
        self.progress = []
        self.mutation_rate = mutation_rate
        self.cities = cities
        self.elites_num = elites_num
        self.iterations = iterations
        self.population_size = population_size
        self.greedy_seed = greedy_seed

        self.population = self.initial_population()
        self.average_path_cost = 1
        self.ranked_population = None

    def best_chromosome(self):
        return self.ranked_population[0][0]

    def best_distance(self):
        return 1 / self.ranked_population[0][1]

    def random_route(self):
        return random.sample(self.cities, len(self.cities))

    def initial_population(self):
        p1 = [self.random_route() for _ in range(self.population_size - self.greedy_seed)]
        greedy_population = [greedy_route(start_index % len(self.cities), self.cities)
                             for start_index in range(self.greedy_seed)]
        return [*p1, *greedy_population]
```

Main body of Genetic Algorithm:



```python
@staticmethod
def produce_child(parent1, parent2):
    gene_1 = random.randint(0, len(parent1))
    gene_2 = random.randint(0, len(parent1))
    gene_1, gene_2 = min(gene_1, gene_2), max(gene_1, gene_2)
    child = [parent1[i] for i in range(gene_1, gene_2)]
    child.extend([gene for gene in parent2 if gene not in child])
    return child

def generate_population(self):
    length = len(self.population) - self.elites_num
    children = self.population[:self.elites_num]
    for i in range(0, length):
        child = self.produce_child(self.population[i],
                                   self.population[(i + random.randint(1, self.elites_num)) % length])
        children.append(child)
    return children

def mutate(self, individual):
    for index, city in enumerate(individual):
        if random.random() < max(0, self.mutation_rate):
            sample_size = min(min(max(3, self.population_size // 5), 100), len(individual))
            random_sample = random.sample(range(len(individual)), sample_size)
            sorted_sample = sorted(random_sample,
                                   key=lambda c_i: individual[c_i].distance(individual[index - 1]))
            random_close_index = random.choice(sorted_sample[:max(sample_size // 3, 2)])
            individual[index], individual[random_close_index] = \
                individual[random_close_index], individual[index]
    return individual

def next_generation(self):
    self.rank_population()
    self.selection()
    self.population = self.generate_population()
    self.population[self.elites_num:] = [self.mutate(chromosome)
                                         for chromosome in self.population[self.elites_num:]]
```

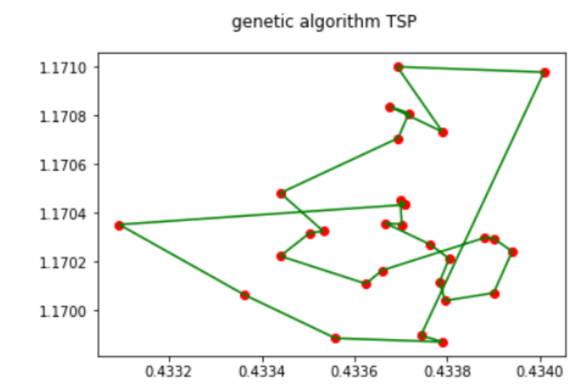
These functions would allow us to get information regarding the overall genetic algorithm. This is where crossover and mutation is being taken place.

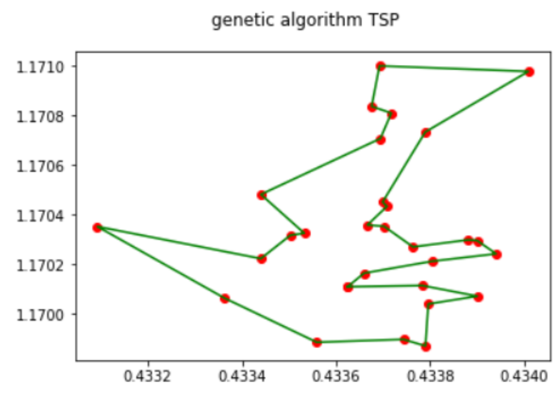
Function **produce_child** is considering two parents to create an offspring that would be used to in the next generation to produce more offsprings.

Mutation is taking place in the code to swipe the customers within the chromosomes to further generate new generation that would be more fit for our analysis.

Once the mutation process is completed, the next generation process is initiated, which repeats the whole process all over again.

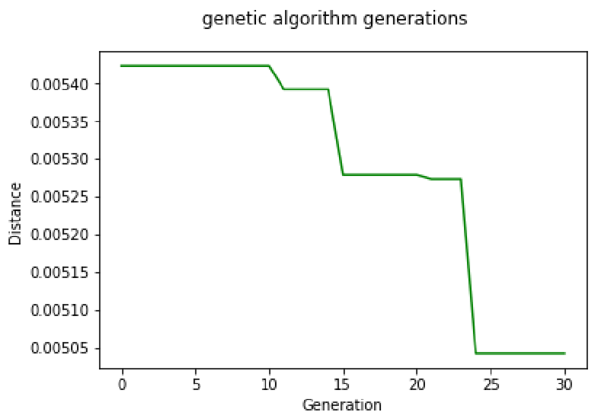
Final Results:

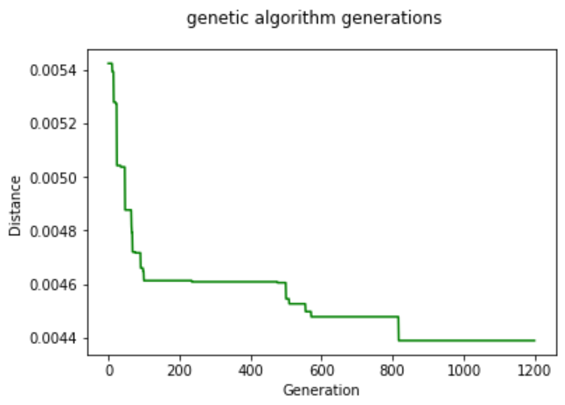




```
30.243141382938983
```

```
27.963048708396148
```





Once the algorithm runs, the latitude and longitude are shown in the visuals above. The latitude and longitude are in radian form along with the edges. The edges (lines) are the path that the algorithm creates. On the left we can see that initial paths are created for the agent through greedy approach, but then the algorithm starts to solve the algorithm using Genetic approach. The initial total distance gets reduced from 30.24 kms to 27.96 kms.

Moreover, the iteration processed some result around 800 iterates after which it reached the best possible answer. Moreover, in the bottom left image above, we can see that the results were concluded within the first 10 iterations. This is because the creation of new generation takes some time and once, this was achieved, the distance started to reduce.

Final Verdict

Overall, we can see that the distances have been minimized by both the functions, Evolutionary and Genetic Algorithm. However, Evolutionary being more advanced and optimized has shown the shortest path of **20.75 k.ms** whereas Genetic has shown the least distance of **27.96 k.ms.**

Key Analysis

Retailo has implemented a new strategy over a course of last 3 weeks as the recession is hitting the markets which has produced high prices of oil ultimately resulting in high operational cost and low profitability.

Operational cost or last mile costs takes a major portion of the P&L statement. Hence, it is ultimately required to take steps to reduce the cost in more efficient manner that benefit the company.

Initially, it is required to identify the variables that impact the overall cost of the sales agents visiting the market.

Input Variables	Output Variable
Number of Agents	Total Volume Consumed in Rupees
Price of Fuel	
Average Distance Travelled	

Number of Agents:

Retailo Karachi has around 50 retailer-based agents that are working to facilitate only the retailers. As the number of agents increases, the overall cost of traveling also increases, hence it has been identified as an important variable.

Price of Fuel:

With the recession taking place and the prices of fuel going upward, it is an important factor that would impact the total volume consumed in Rupees. The current price as of 19th June 2022 is Rupees 233/litre.

Average Distance Traveled:

The distance traveled is given through the algorithms. We are assuming that all the agents on average traveled that particular distance. The distance given by the model is further divided by the average distance traveled by a 70cc bike in 1 litre. This shows us % consumption of petrol in kms. Moreover, the increase in distance traveled would impact the % consumption eventually.

Total Agents	50
Price of Fuel (Rupees)	$233
Avg. Distance Traveled	20
Avg. Distance in 1 litre	32
% Consumption (litre)	0.625

Volume (Agents x % consumption)	33.33
Total Consumption in Cost (Volume x Price)	$7,767

Saving

On average, if the distance travelled in 20 kms (Evolutionary Algorithm) on average for all the agents, then it would cost **7,767** Rupees per day. Simultaneously, if the distance travelled in increased to 27 kms (Genetic Algorithm) on average for all the agents, then it would cost us **10,485** Rupees per day.

	Cost per Day	**Cost per Week**	**Cost Per Month**
Genetic Algorithm	10,485	73,395	293,580
Evolutionary Algorithm	7,767	54,369	217,476
Saving in Rupees	*2,718*	*19,026*	*76,104*
Saving in Litres	*12*	*84*	*336*

**Please note this is based on only retailer-based agents. The saving in terms of Rupees and litres would be more than estimated above.*

**The volume increases from 33 litres to 45 litres when avg. kilometers traveled increased from 20 kms (Evolutionary) to 27 kms (Genetic).*

Sensitivity Analysis

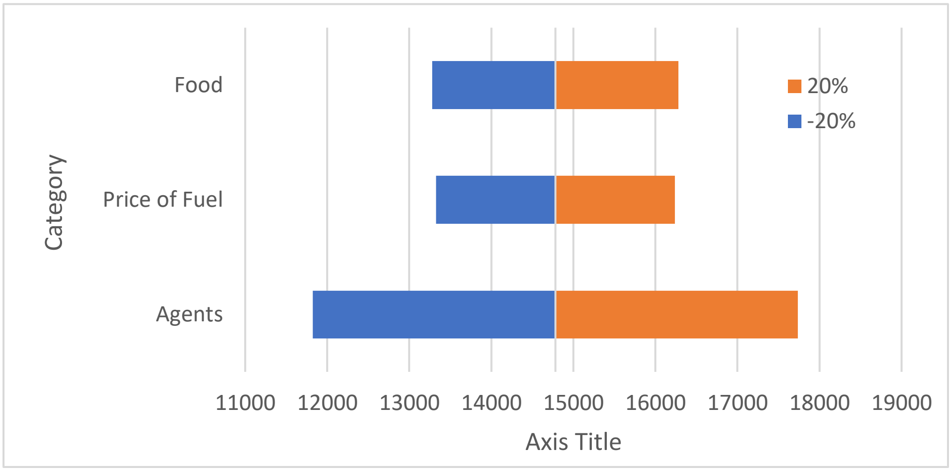
		Price						
	$ 7,767	$ 213	$ 223	$ 233	$ 243	$ 253	$ 263	$ 273
Litres	**30**	$ 6,390	$ 6,690	$ 6,990	$ 7,290	$ 7,590	$ 7,890	$ 8,190
	31	$ 6,627	$ 6,938	$ 7,249	$ 7,560	$ 7,871	$ 8,182	$ 8,493
	32	$ 6,863	$ 7,186	$ 7,508	$ 7,830	$ 8,152	$ 8,474	$ 8,797
	33	$ 7,100	$ 7,433	$ 7,767	$ 8,100	$ 8,433	$ 8,767	$ 9,100
	34	$ 7,337	$ 7,681	$ 8,025	$ 8,370	$ 8,714	$ 9,059	$ 9,403
	36	$ 7,573	$ 7,929	$ 8,284	$ 8,640	$ 8,995	$ 9,351	$ 9,707
	37	$ 7,810	$ 8,177	$ 8,543	$ 8,910	$ 9,276	$ 9,643	$ 10,010
	38	$ 8,047	$ 8,424	$ 8,802	$ 9,180	$ 9,558	$ 9,935	$ 10,313

Above, the sensitivity analysis of litres and price when the distance traveled in 20kms on average has been calculated. Since the total consumption in monetary values is around 7,767 rupees, cost incurring above it could be considered in to account as an increase in expenses.

The increase in price could only be catered by reduction in litres consumed. Litre consumption is dependent on number of agents on field and % distance consumed. Since, Evolutionary Algorithm has shown the least distance traveled, has the % distance consumed is stagnant, so any increase in price has to be catered through reduction in the number of agents on field.

Tornado Analysis

	Agents	Price of fuel	Food
80%	11,825	13,325	13,281
100%	14,781	14,781	14,781
120%	17,737	16,237	16,281



References

Berninger, I. (2014). *Vehicle Routing Problem on Android/iOS.* Austria: Institute of Computer Science Research Group DPS, University Innsbruck.

Case Study: Solving the Traveling Salesman Problem. (2012, Summer). Retrieved from cs: https://www2.cs.sfu.ca/CourseCentral/125/tjd/tsp_example.html#:~:text=Solving%20the%20TSP%20by%20brute,and%20finds%20the%20shortest%20one.

EXCEL SOLVER: WHICH SOLVING METHOD SHOULD I CHOOSE? (2022). Retrieved from EngineerExcel: https://engineerexcel.com/excel-solver-solving-method-choose/#:~:text=The%20Evolutionary%20method%20is%20based,What%20is%20this%3F&text=In%20simple%20terms%2C%20the%20solver,of%20sets%20of%20input%20values.

Genetic Algorithm in Machine Learning. (n.d.). Retrieved from Javatpoint: https://www.javatpoint.com/genetic-algorithm-in-machine-learning

Gosawmi, S. K. (2014). *A Solution of Genetic Algorithm for Solving Traveling Salesman Problem.* International Journal for Scientific Research and Development (IJSRD).

Jiang, C. (2010). *A Reliable Solver of Euclidean Traveling Salesman Problems with Microsoft Excel Add-in Tools for Small-size Systems .* Hangzhou: College of Information Management.

Karleigh Moore, J. K. (2022, May 13). *Greedy Algorithms* . Retrieved from Brilliant : https://brilliant.org/wiki/greedy-algorithm/

Ma, S. (2020, January 02). Retrieved from Routific : https://blog.routific.com/travelling-salesman-problem#:~:text=The%20Travelling%20Salesman%20Problem%20(TSP,computer%20science%20and%20operations%20research.

Michael Jünger, G. R. (1995). *Chapter 4 The traveling salesman problem.*

Program for distance between two points on earth. (2021, April 09). Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/program-distance-two-points-earth/

Raju, D. (2020, July 13). *Travelling Salesman Problem via the Greedy Algorithm*. Retrieved from Medium: https://medium.com/ivymobility-developers/algorithm-a168afcd3611

Saiyed, A. R. (2012). *The Traveling Salesman problem.* Indiana: Indiana State University.

SHRIMALI, A. S. (2014). *A COMPARATIVE STUDY OF BRUTE FORCE METHOD, NEAREST NEIGHBOUR AND GREEDY ALGORITHMS TO SOLVE THE TRAVELLING SALESMAN PROBLEM.* Udaipur: Department of Mathematics, Pacific Academy of Higher Education & Research University.

Teguh Narwadi, S. (2017). *An application of traveling salesman problem using the improved genetic algorithm on android google maps.*

Traveling Salesman Problem using Genetic Algorithm. (2021, Dec 23). Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/traveling-salesman-problem-using-genetic-algorithm/