Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

Vehicle Routing Problem Solver

Sourabh Kulkarni
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ, USA

***Abstract*— Vehicle Routing problem is a problem which most of the postal service companies face every day. There are a number of addresses to which deliveries have to be made in a day and in a given amount of time. So, the main concern of the company is to save on time as well as on fuel. Hence, I developed an algorithm which can find the shortest path(s) out of the total number of addresses the delivery van has to travel and display it on google map using Google APIs in real-time. This algorithm first extracts the details out of google maps and computes the shortest path covering all the addresses and taking into account the maximum weight a vehicle can carry and plots it on the google map using libraries in python.**

Keywords*— Routing, Google APIs, Google Maps, Libraries, Python.

I. INTRODUCTION

Vehicle Routing problem is often classified as the classic VRP. Most of the postal service companies are generally hit by this problem and there is hardly a proper solution to fix this problem. The first algorithm invented to address this problem was by Clark et al. [1] in 1997. We build our algorithm keeping this as our base. For this, we need to have a good command over python.

The first approach was the basic savings algorithm. It was invented by Clark et. Al. [1]. The problem is to determine the allocation of the customers among routes, the sequence in which the customers shall be visited on a route, and which vehicle that shall cover a route. The objective is to find a solution which minimizes the total transportation costs.

The Second approach was the CVRP [2] approach. The concept of this algorithm is to hybridize the CW with tournament and roulette wheel selections to determine a new and efficient algorithm. The objective is to find the feasible solutions (or routes) to minimize travelling distances and number of routes.

Ribas et. al. [3] proposed the Vehicle Routing Problem with Time Windows is a particular case of the classical Vehicle Routing Problem in which the demands of each customer should be met within an established time window. Due to the combinatorial complexity of the problem its resolution by pure exact methods is, in many cases, computationally impractical. This fact motivates the development of heuristic algorithms, which are usually faster but do not guarantee the best solution for the problem. This work proposes a hybrid algorithm that combines the metaheuristic Iterated Local Search, the Variable Neighborhood Descent procedure and an exact Set Partitioning model. This is a high complexity algorithm.

We also need to look at the algorithm proposed by Gilbert Laporte [4]. This algorithm follows three steps i.e : (i) direct tree search methods; (ii) dynamic programming, and (iii) integer linear programming. But by this algorithm, the complexity increases.

Masum et. al. [5] proposed the complex combinatorial optimization problem that belongs to the NP-complete class. Due to the nature of the problem it is not possible to use exact methods for large instances of the VRP. Genetic algorithms provide a search technique used in computing to find true or approximate solution to optimization and search problems. However, we used some heuristic in addition during crossover or mutation for tuning the system to obtain better result.

In this paper, we build upon the savings algorithm invented by Clark and Wright. It is a simple approach and can be easily implemented. The complexity of our algorithm is low and hence we can compete this with the several existing algorithms and obtain at par or more better results. Moreover, we also provide this with a Google map interface and hence can be used for navigation in the delivery van.

II. ALGORITHM

Our algorithm is the simple algorithm based upon the Clark and wright's Savings algorithm. We extract the Details from google maps with the google API and use the extracted data to get our distance matrix. From this matrix we calculate the shortest path by applying the savings algorithm and then plot this on the google maps.

For simplicity purpose, we did it on Linux. We need to install all the necessary packages and libraries which will be in turn needed to extracted data. The code also serves the purpose of calculating the traffic on that path at a particular time of the day.

The postal service company has number of addresses to cover in a day. The first step is to put all these addresses in a file which we will use as our database. Our code in python extracts all the addresses from the file. Then by using the google API key we extract all the required information eg. Distances between all the addresses, time required by the vehicle between all the addresses, traffic in the respective area etc.

This algorithm is a heuristic algorithm. This method does, often yield a relatively good solution. That is, a solution which deviates little from the optimal solution. We need to understand the savings concept before proceeding. The basic savings concept expresses the cost savings obtained by joining two routes into one route as illustrated in figure 1, where point 0 represents the depot.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

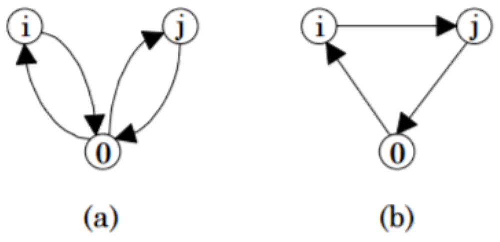


Fig. 1. Savings Algorithm [1]

In this figure, customers i and j are visited on separate routes. An alternative solution to this is to visit the two customers on the same route, for example in the sequence i-j as illustrated in figure 1(b). Because the transportation costs are given, the savings that result from driving the route in figure 1(b) instead of the two routes in figure 1(a) can be calculated. Denoting the transportation cost between two given points i and j by cij , the total transportation cost Da in figure 1(a) is:

$$D_a = c_{0i} + c_{i0} + c_{0j} + c_{j0} \qquad \text{….(1)}$$

The transportation cost Db in figure 1(b) is:

$$D_b = c_{0i} + c_{ij} + c_{j0} \qquad \text{….(2)}$$

By combining the two routes we can find the savings:

$$S_{ij} = D_a - D_b = c_{i0} + c_{0j} - c_{ij} \qquad \text{….(3)}$$

It is attractive with respect to cost if we have a relatively larger value of Sij.

In our approach, we consider 5 customers. We will be using cities instead of the exact address as the demonstration would be easier. The following diagram represents the cities which have to be covered.

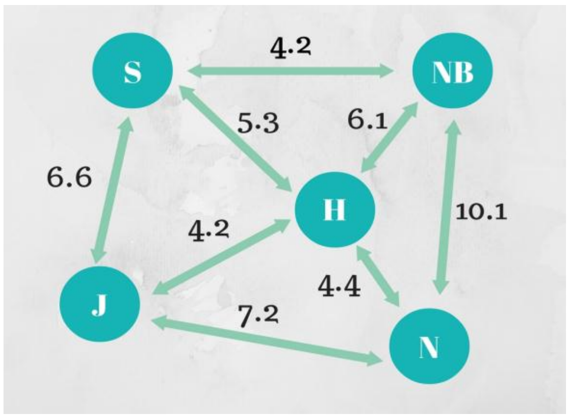
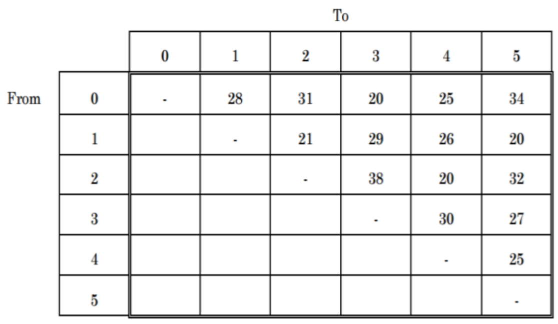


Fig. 2. Algorithm Schema

The diagram represents the cities and the distance between each on them. H is Hoboken, N is New York, NB is North Bergen, J is Jersey City, S is Secaucus. We consider our depot to be at Hoboken. We intend to first find the transportation cost between all the 5 destinations. The transportation costs between all pairs of points are shown in the following table, where 0 represents the depot (the costs are symmetric, and for that reason only the upper half of the table is filled in)



		To					
		0	1	2	3	4	5
From	0	-	28	31	20	25	34
	1		-	21	29	26	20
	2			-	38	20	32
	3				-	30	27
	4					-	25
	5						-

Table I: Costs between all pair of points [1]

In this algorithm, we also assume that the vehicle has certain maximum capacity. Assuming the capacity is 100 units. Now we take into consideration, how much units of load each customer wants to send.

Customer	Quantity
1	37
2	35
3	30
4	25
5	32

Table II: Quantity per Customer [1]

The savings value are calculated based on these values. We only consider the upper triangular values of the matrix as the matrix is symmetric.

Once we are done with calculating this, we have the pairs of points. We then sort them in descending order. And from that we find the shortest path(s). The type of algorithm used is the sequential savings algorithm.

In the example customers 1 of 5 are considered first. They can be assigned to the same route since their joint demand for 69 units does not exceed the vehicle capacity. Now we establish the connection 1-5, and thereby points 1 and 5 will be neighbors on a route in the final solution. Next, we consider customers 1 and 2. If customers 1 and 2 should be neighbors on a route, this would require the customer sequence 2-1-5 or 5-1-2 on a route, because we have established already that 1 and 5 must be visited in immediate succession on the same route. The total demand (104) on this route would exceed the vehicle capacity (100).

Therefore, customers 1 and 2 are not connected. If points 2 and 4, which is the next pair in the list, were connected at this stage, we would be building more than one route (1-5 and 2-4). Since this algorithm is limited to making only one route at a time, we disregard the point pair 2 and 4.

The combination of the next pair of points, 4 and 5, results in the route 1-5-4 with a total demand of 94. This combination is feasible, and we establish the connection between 4 and 5 as a part of the solution. We can add no more points due to capacity restriction. Hence the route formed is 0-1-5-4-0. The max load on the vehicle is 98 and the other route which covers the remaining addresses will be 0-2-3-0. The load will be 89.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

III. EXPERIMENTAL RESULTS

I wrote a code in python and implemented the above written algorithm. The code first extracts the details of the addresses and the distances between them. Then it calculates the shortest path by taking the savings and the cost into consideration.

j

i	1	2	3	4	5
1	-	38	19	27	42
2		-	13	36	33
3			-	15	27
4				-	34
5					-

Table III: Calculated Savings

The paths are displayed on the google maps. To do this we need to first install gmplot library in python. This helps us to open google maps in a new tab on the web browser and exports your results to the maps. Hence the results are plotted and this makes the driver of the vehicle, easy to navigate.

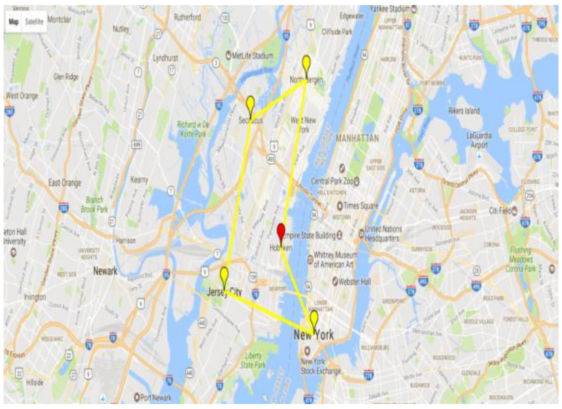
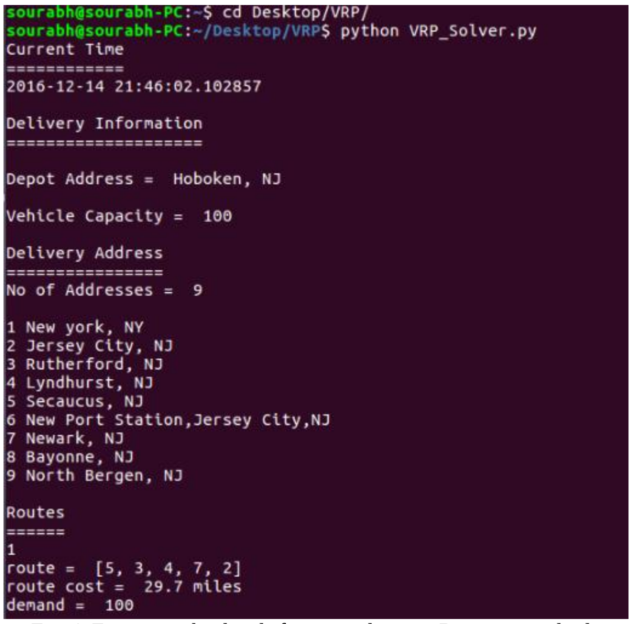


Fig. 3. Result for the above example

The yellow line represents the path for the above-mentioned cities. We may also increase the number of addresses. It is easy to find shortest route when the number of addresses are small but when the number increases, there are a huge number of permutations and combinations.

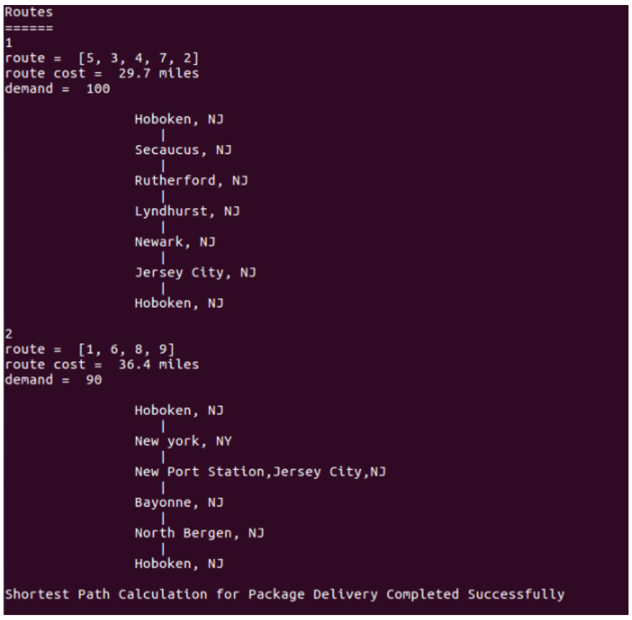
I tried it with more number of addresses.



```
sourabh@sourabh-PC:~$ cd Desktop/VRP/
sourabh@sourabh-PC:~/Desktop/VRP$ python VRP_Solver.py
Current Time
============
2016-12-14 21:46:02.102857

Delivery Information
====================

Depot Address =  Hoboken, NJ

Vehicle Capacity =  100

Delivery Address
================
No of Addresses =  9

1 New york, NY
2 Jersey City, NJ
3 Rutherford, NJ
4 Lyndhurst, NJ
5 Secaucus, NJ
6 New Port Station,Jersey City,NJ
7 Newark, NJ
8 Bayonne, NJ
9 North Bergen, NJ

Routes
======
1
route =  [5, 3, 4, 7, 2]
route cost =  29.7 miles
demand =  100
```

Fig. 4. Extracting the details from google maps By giving multiple addresses as input

Now we need to find out the path(s) keeping in mind, the cost savings and demand.

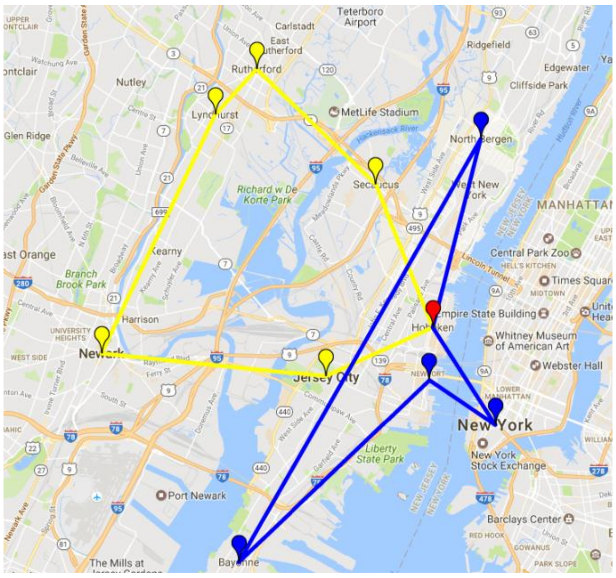


```
Routes
======
1
route =  [5, 3, 4, 7, 2]
route cost =  29.7 miles
demand =  100

                    Hoboken, NJ
                        |
                    Secaucus, NJ
                        |
                    Rutherford, NJ
                        |
                    Lyndhurst, NJ
                        |
                    Newark, NJ
                        |
                    Jersey City, NJ
                        |
                    Hoboken, NJ

2
route =  [1, 6, 8, 9]
route cost =  36.4 miles
demand =  90

                    Hoboken, NJ
                        |
                    New york, NY
                        |
                    New Port Station,Jersey City,NJ
                        |
                    Bayonne, NJ
                        |
                    North Bergen, NJ
                        |
                    Hoboken, NJ

Shortest Path Calculation for Package Delivery Completed Successfully
```

Fig. 5. Possible shortest paths

I tried to display the paths on the Ubuntu terminal for a better understanding. Though they can be directly displayed on Google maps.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)



This is the final result which takes into account all the possible paths which we have calculated and displays them on the map for navigation.

IV. CONCLUSION

From this we can conclude that we have been successful in computing the shortest path for the given number of addresses. This can reduce the complexity to a great extent. The future scope is to implement this in form of an application on well-known platforms like android and iOS so that it can be used in real-time by Postal service companies. More over the complexity of this algorithm is low and hence is easy to implement.

V. ACKNOWLEDGEMENT

I would like to thank Professor Mukundan Iyenger from Department of Electrical and Computer engineering at Stevens Institute of Technology for his guidance and support.

VI. REFERENCES

[1] Clarke & Wright's Savings Algorithm Jens Lysgaard (translated by Michael M. Sørensen) Department of Management Science and Logistics The Aarhus School of Business Fuglesangs Allé 4 DK-8210 Aarhus V.

[2] Tantikorn Pichpibu and Ruengsak Kawtummachai, "An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem"

[3] Sabir RIBAS a , Anand SUBRAMANIAN a , Igor Machado COELHO , Luiz Satoru OCHI a , Marcone Jamilson Freitas SOUZA "A hybrid algorithm for the Vehicle Routing Problem with Time Windows"

[4] Gilbert Laporte, "The Vehicle Routing Problem: An overview of exact and approximate algorithms", Centre de Recherche sur les Transports, Canada H3C 3J7 Universit~ de Montreal, C.P. 6128 Station A, Montreal,

[5] Abdul Kadar Muhammad Masum, Mohammad Shahjalal, Md. Faisal Faruque, and Md. Iqbal Hasan Sarker4, "Solving the Vehicle Routing Problem using Genetic Algorithm (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 7, 2011

(This work is licensed under a Creative Commons Attribution 4.0 International License.)