VEHICLE ROUTING PROBLEM

Cluster Innovation Centre

University of Delhi

AMIT | TUSHAR

April 2021

Month long project submitted for the paper of:

Advanced Algorithms Design

CERTIFICATE OF ORIGINALITY

The work embodied in this report entitled **“Vehicle Routing Problem”** has been carried out by **Amit and Tushar** for the paper of **“Advanced Algorithms Design”**. We declare that the work and language included in this project report is free from any kind of plagiarism.

The work submitted is original and has not been submitted earlier to any institute or university for the award of any degree or diploma.

Amit
(11807)

Tushar
(11840)

ACKNOWLEDGEMENT

With a deep sense of gratitude, we express our dearest indebtedness to **Prof. Anu Yadav** for her support throughout the duration of our project. We would like to thank her for giving us the opportunity to do this wonderful project. Her learned advice and constant encouragement has helped us complete this project. It is a privilege for us to be her students. Without her encouragement and guidance this project would not have materialized.

We are also thankful to our friends and family who helped us with their valuable suggestions and guidance that have been helpful in various phrases of the completion of the project.

ABSTRACT

The project aims at tackling the problem of inefficient vehicle sharing systems currently present in public transport domains and mobility on demand systems by suggesting changes in the way route planning and ride combining for sharing based systems is done while assuring optimization in several other domains.

In the *Vehicle Routing Problem (VRP)*, the goal is to find optimal routes for multiple vehicles visiting a set of locations. *(When there's only one vehicle, it reduces to the Traveling Salesman Problem* [1]*).*

But what do we mean by *"optimal routes"* for a VRP? One answer is the routes with the least total distance. However, if there are no other constraints, the optimal solution is to assign just one vehicle to visit all locations, and find the shortest route for that vehicle. This is essentially the same problem as the TSP.

A better way to define optimal routes is to minimize the length of the longest single route among all vehicles. This is the right definition if the goal is to complete all deliveries as soon as possible. Several constraints have also been implemented which gives us a perspective of the real world scenario.

TABLE OF CONTENTS

1. INTRODUCTION

One of the most important applications of optimization is *vehicle routing*, in which the goal is to find the best routes for a fleet of vehicles visiting a set of locations. Usually, "best" means routes with the least total distance or cost [2]. Here are a few examples of routing problems:

- A package delivery company wants to assign routes for drivers to make deliveries.
- A cable TV company wants to assign routes for technicians to make residential service calls.
- A ride-sharing company wants to assign routes for drivers to pick up and drop off passengers.

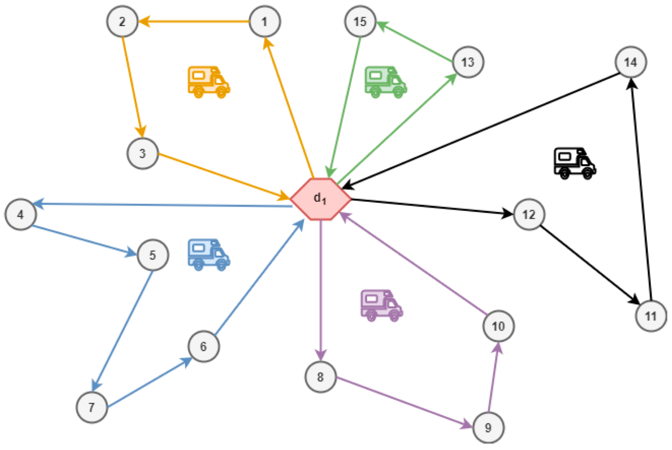


Fig.1: Representation of Vehicle Routing Problem

The most famous routing problem is the *Traveling Salesman Problem (TSP)*: find the shortest route for a salesman who needs to visit customers at different locations and return to the starting point. A TSP can be represented by a graph, in which the nodes correspond to the locations, and the edges (or arcs) denote direct travel between locations. [1]

The problem can be solved manually if there are less locations, like *4*. But if there are more than ten locations (not counting the starting point), the number of routes is *362880*. For *20* locations, the number jumps to *2432902008176640000*. An exhaustive search of all possible routes would be guaranteed to find the shortest, but this is computationally intractable for all but small sets of locations. For larger problems, optimization techniques are needed to intelligently search the solution space and find an optimal (or near-optimal) solution.

A more general version of the TSP is the vehicle routing problem (VRP), in which there are multiple vehicles.[2] In most cases, VRPs have constraints: for example, vehicles might have capacities for the maximum weight or volume of items they can carry, or drivers might be required to visit locations during specified time windows requested by customers. We have implemented many types of VRPs in this project, including the following:

- *Vehicle routing problem*, a generalisation of the TSP with multiple vehicles.
- *VRP with capacity constraints*, in which vehicles have maximum capacities for the items they can carry.
- *VRP with time windows*, where the vehicles must visit the locations in specified time intervals.
- *VRP with resource constraints*, such as space or personnel to load and unload vehicles at the depot (the starting point for the routes).

2. FORMULATION OF PROBLEM

2.1. Theoretical Concepts:

2.1.1. *Traveling Salesman Problem:* The travelling salesman problem is a graph theory optimization problem in which the nodes (cities) of a graph are bound by directed edges (routes), with the weight of an edge indicating the distance between two cities. The challenge is to find a route that reaches each city only once, returns to the starting point, and minimises the distance travelled. [1] The only known general solution algorithm that provides the shortest path has a solution time that grows exponentially with the size of the problem (i.e., the number of cities). It is an example of an NP-complete (from nonpolynomial) problem for which no efficient (i.e., polynomial time) algorithm is known.

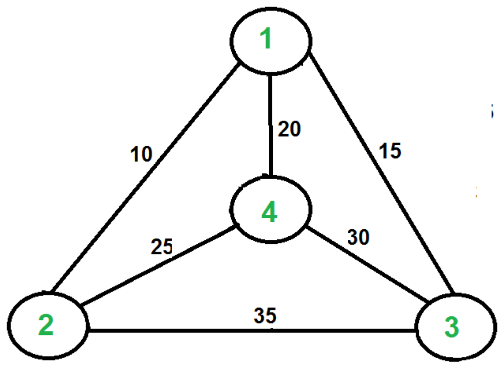


Fig.2: Travelling Salesman Problem example

This problem was first proposed in 1930 and has been one of the most intensively researched topics in optimization. Numerous optimization approaches use it as a benchmark. Despite the fact that the problem is computationally difficult several heuristics and identical algorithms are known, just so that the instances with large numbers of cities can be solved fully and problems involving millions of cities can be approximated to be within a small fraction of 1%.

Even in its basic form, the TSP has many potential applications, which includes organizing, supply chain, and microchip manufacturing.

2.1.2. *Capacity Constraints:* 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. [3]

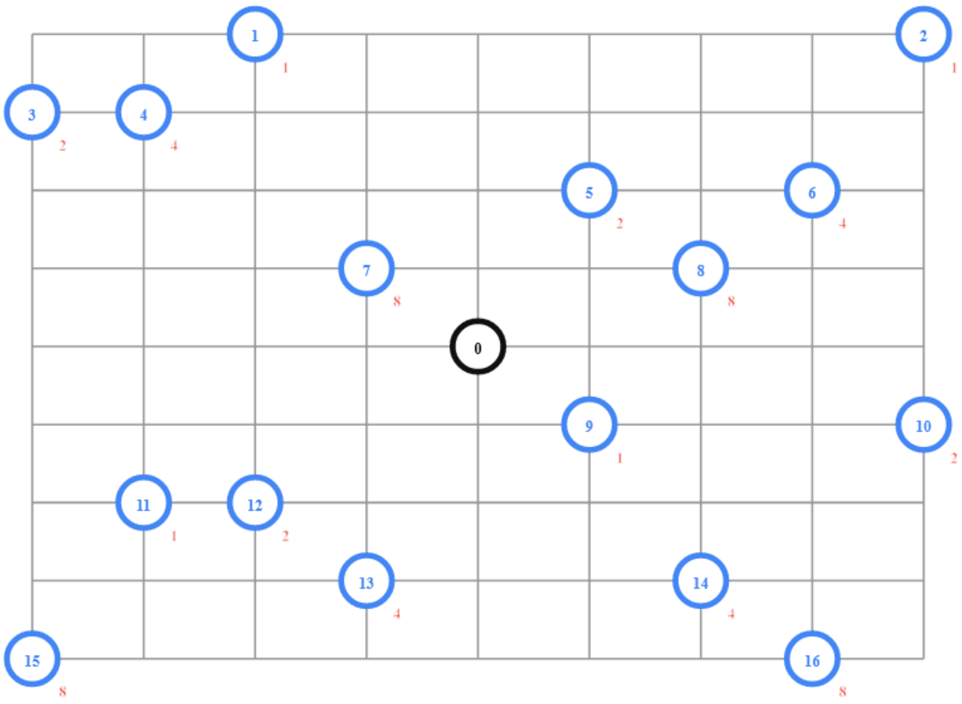


Fig.3: Example showing VRP with Capacity constraint

The example extends the previous VRP example and adds the following requirements. At each location there is a demand corresponding to the quantity of the item to be picked up. Also, each vehicle has a maximum capacity of 15. (We aren't specifying units for the demands or capacity). The grid above shows the locations to visit in blue and the company location in black. The demands are shown at the lower right of each location.

The problem is to find an assignment of routes to vehicles that has the shortest total distance, and such that the total amount a vehicle is carrying never exceeds its capacity.

2.1.3. *Pickups and Delivery Constraints:* This is a VRP in which each vehicle picks up items at various locations and drops them off at others. The problem is to assign routes for the vehicles to pick up and deliver all the items, while minimizing the length of the longest route.

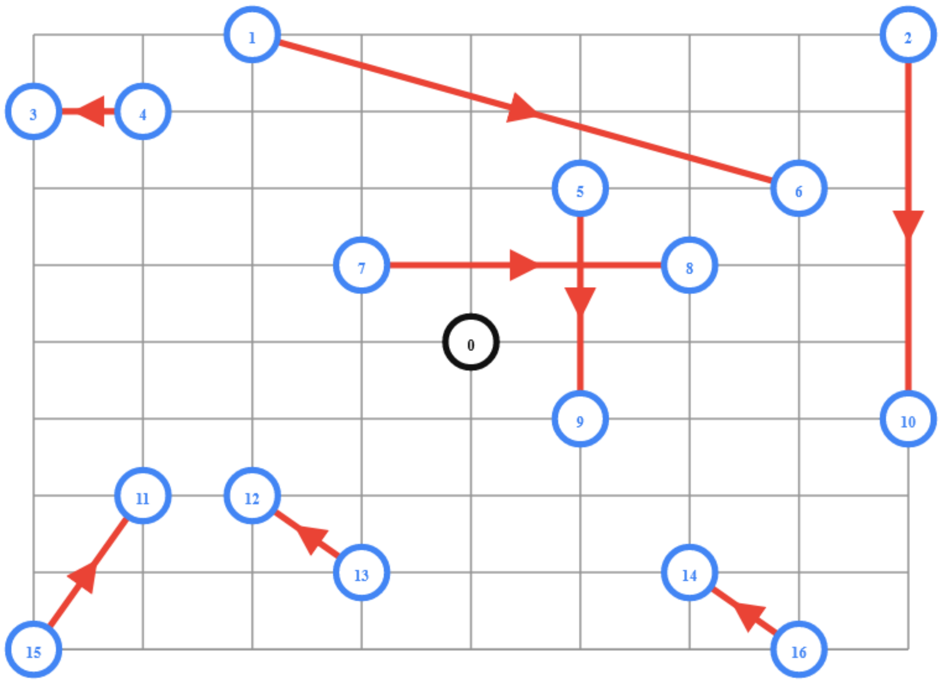


Fig.6: VRP with Pickups and Delivery Constraints

The diagram above shows the pickup and delivery locations on a grid similar to the one in the previous VRP example. For each item, there is a directed edge from the pickup location to the delivery location.

2.1.4. *Time Window Constraints:* Many vehicle routing problems involve scheduling visits to customers who are only available during specific time windows. These problems are known as vehicle routing problems with time windows (VRPTWs). [4]

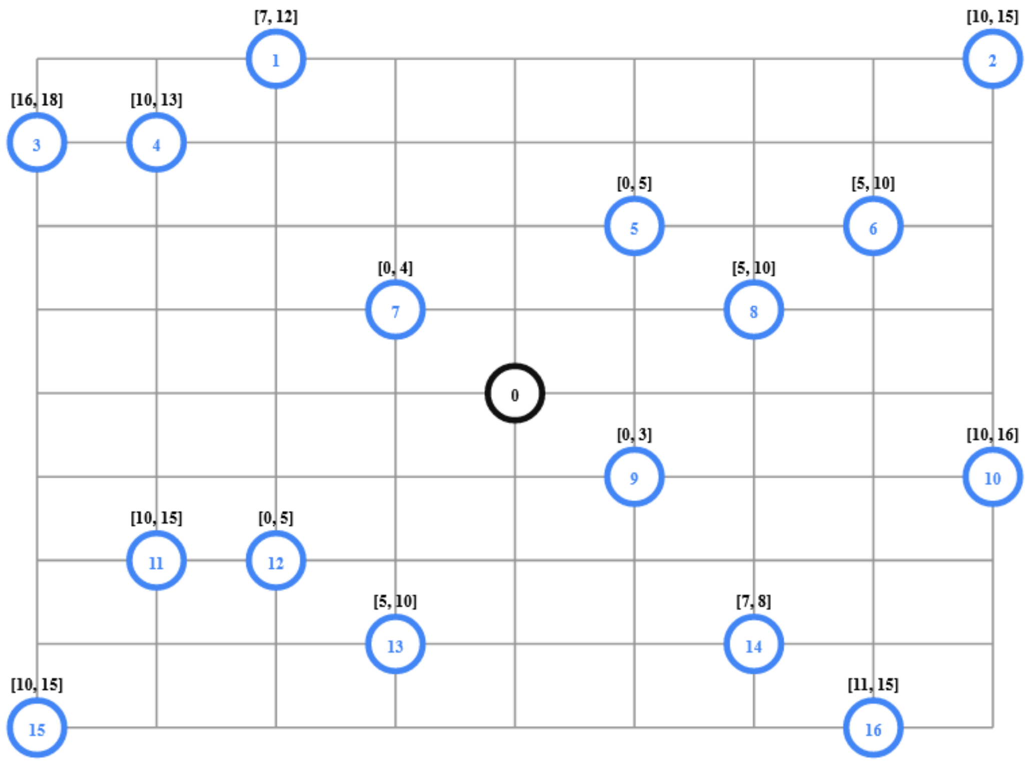


Fig.4: Example showing VRP with Time-Window constraint

Since the problem involves time windows, the data include a time matrix, which contains the travel times between locations (rather than a distance matrix as in previous examples). The diagram above shows the locations to visit in blue and the depot in black. The time windows are shown above each location.

The goal is to minimize the total travel time of the vehicles.

2.1.5. <u>*Resource Constraints:*</u> This is the type of VRP with time windows that also has constraints at the depot: all vehicles need to be loaded before departing the depot and unloaded upon return. Since there are only two available loading docks, at most two vehicles can be loaded or unloaded at the same time. As a result, some vehicles must wait for others to be loaded, delaying their departure from the depot. The problem is to find optimal vehicle routes for the VRPTW that also meet the loading and unloading constraints at the depot.

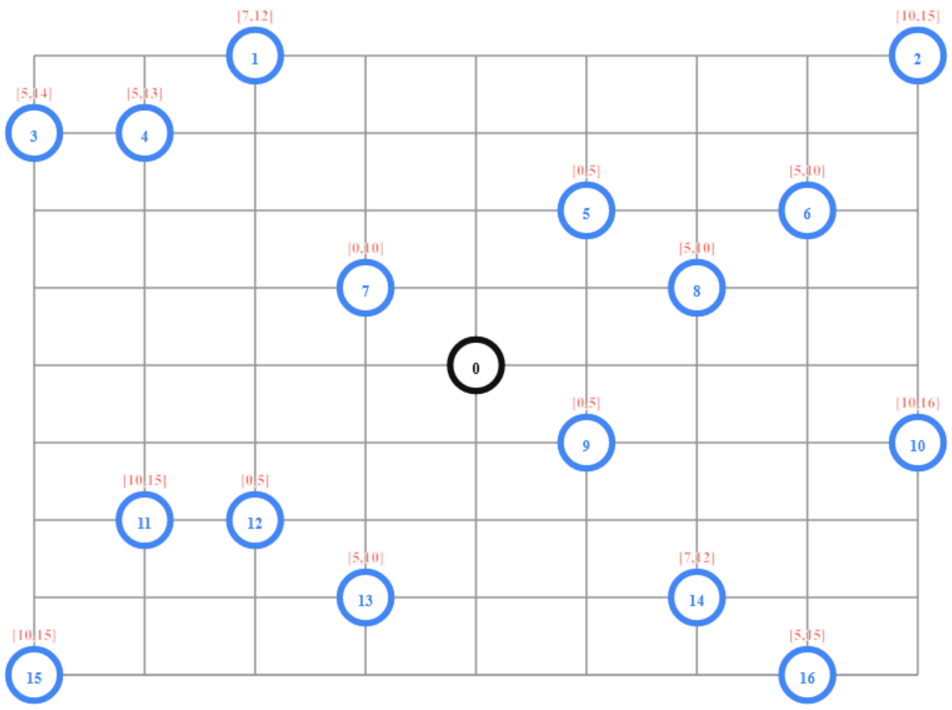


Fig.5: The diagram shows a VRPTW with resource constraints.

2.2. Problem statement & Challenges:

Constraint Programming normally uses depth-first-branch & bound methodology to solve problems of optimization. While this procedure can be effective in finding the optimal solution, the time taken to find can be prohibitive for large problems. This project presents an approach for using local search methods within a restricted programming framework and this approach extends to routing issues with vehicles. We have integrated a vehicle routing programming model and a framework for combining local search and constraint programming. The task can then be optimized with quick local inspections by controlling key constraints, while the constrained propagation system is left with more complicated constraints. [5]

The problem can be given by a set of customers $N = \{1,2,3,...n\}$, residing at n different locations. Every pair of locations (i, j), where $i, j \in N$ and $i \neq j$, is associated with a travel time t_{ij} and a distance travelled d_{ij} that are symmetrical ($t_{ij} = t_{ji}$

and $d_{ij} = d_{ji}$). Denote by q_i , $= 1, 2, \ldots\ldots n,$ the demand at point i . The central depot is denoted by 0.

The customers are served from one depot with a homogeneous and limited fleet of vehicles. The vehicle leaves and returns to the depot. There is a set V of vehicles, $V = \{1, 2, \ldots m\}$, with identical capacities. The capacity of each vehicle $k \in V$is represented by a_k .

We let $R_i = \{r_i(1), \ldots r_i(n_i)\}$ denote the route for vehicle i, where r_i (j) is the index of the jth customer visited and n_i is the number of customers in the route. It is assumed that every route finishes at the depot, i.e., $r_i (n_i + 1) = 0$.

- *<u>Time Windows:</u>* Each customer $i \in N$ has a time windows, i.e. an interval $[e_i , l_i]$, where $e_i \leq l_i$, which corresponds, respectively, to the earliest and latest time to start to service customer i . Let s_i be the service time at customer i.

- *<u>Split Deliveries:</u>* The demand of a customer may be fulfilled by more than one vehicle. This occurs in all cases where some demand exceeds the vehicle capacity, but can also turn out to be cost effective in other cases.

The decision variables of the model are :

$x_{ij}^k = 1,$ if j is supplied after i by vehicle k ;

$0,$ otherwise.

b_i^k = moment at which service begins at customer i by vehicle k, $i = 1, \ldots n$ and $k = 1, \ldots m$

y_i^k = fraction of customer's demand i delivered by vehicle k.

So, the objective of the model is to minimize the total distance travelled respecting the time window constraints.

The objective function can be written as follows :

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} d_{ij} x_{ij}^{k}$$

The model constraints are :

$$\sum_{j=1}^{n} x_{0j}^{k} = 1 \quad ; k = 1, ..., m \qquad \rightarrow \textbf{(1)}$$

Constraint (1) guarantees that each vehicle will leave the depot and arrive at a determined customer.

$$\sum_{i=0}^{n} x_{ip}^{k} - \sum_{j=0}^{n} x_{pj}^{k} = 0 \, ; \, p = 0,..., n; \; k = 1,..., m \qquad \rightarrow \textbf{(2)}$$

Constraint (2) is about entrance and exit flows, guarantees that each vehicle will leave a determined customer and arrive back to the depot.

$$\sum_{k=1}^{m} y_{i}^{k} = 1 \, ; \, i = 1,..., n \qquad \rightarrow \textbf{(3)}$$

Constraint (3) guarantees that the total demand of each customer will be fulfilled.

$$\sum_{i=1}^{n} q_{i} y_{i}^{k} \leq a_{k} ; \, k = 1,..., m \qquad \rightarrow \textbf{(4)}$$

Constraint (4) guarantees that the vehicle capacity will not be exceeded.

$$y_{i}^{k} \leq \sum_{j=0}^{n} x_{ji}^{k} \, ; \, i = 1,..., n \, ; \, k = 1,..., m \qquad \rightarrow \textbf{(5)}$$

Constraint (5) guarantees that the demand of each customer will only be fulfilled if a determined vehicle goes by that place. It can be noticed that, adding to constraint (5) the sum of all vehicles and combining to equation (3) we have the constraint $\sum_{k=1}^{m}\sum_{i=0}^{n} x_{ij}^{k} \geq 1 \; ; \; j = 0,\dots,n$, which guarantees that each vertex will be visited at least once by at least one vehicle.

$$b_i^k + s_i + t_{ij} - M_{ij}(1 - x_{ij}^k) \leq b_j^k \; ; \; i = 1,\dots,n; \; j = 1,\dots,n; \; k = 1,\dots,m \quad \rightarrow \textbf{(6)}$$

Equation (6) sets a minimum time for beginning the service of customer *j* in a determined route and also guarantees that there will be no sub tours. The constant M_{ij} is a large enough number, for instance, $M_{ij} = l_i + t_{ij} - e_j$.

$$e_i \leq b_i^k \leq l_i \; ; \; i = 1,\dots,n \qquad \rightarrow \textbf{(7)}$$

Constraint (7) guarantees that all customers will be served within their time windows.

$$y_i^k \geq 0 \; ; \; i = 1,\dots,n; \; k = 1,\dots,m$$

$$\rightarrow \qquad \textbf{(8)}$$

$$b_i^k \geq 0 \; ; \; i = 1,\dots,n; \; k = 1,\dots,m$$

Equation (8) guarantees that both the decision variables, y_i^k and b_i^k , are positive.

$$x_{ij}^k \in \{0, 1\} \; ; \; i = 0,\dots,n; \; j = 0,\dots,n; \; k = 1,\dots,m \qquad \rightarrow \textbf{(9)}$$

Finally, equation (9) guarantees the decision variables, x_{ij}^k , to be binary.

3. METHODOLOGY

3.1. Softwares used:

- *System Softwares:*
  - *Windows 10 Pro (v1809)*

- *Application Softwares:*
  - *Google Colab*

3.2. Working of code:

➢ Define the dictionary with the given elements, which need to be passed to the later functions.

  - *distance_matrix*: Array of distances between locations calculated as Manhattan Distance between two coordinates.
  - *num_locations*: The number of vehicles in the fleet.
  - *depot:* The index of the depot, the location from where all the vehicles start and end the route.
  - *demands:* In case of capacity constraint, each location has a demand corresponding to the quantity of the item to be picked up.
  - *vehicle_capacities:* In case of capacity constraint, each vehicle has a capacity which can never be exceeded while loading the items to it.
  - *pickups_deliveries:* In case of pickup-delivery constraint, it is a list of pairs of pickup and delivery locations corresponding to the directed edges. For each pair, the first entry is the index of the pickup location, and the second is the index of the delivery location.

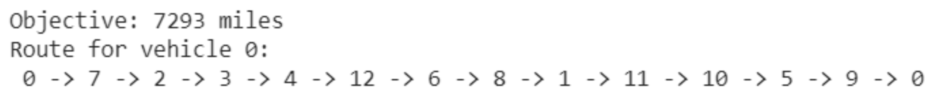
  - *time_matrix:* In case of time-windows constraint as well as resource constraint, *distance_matrix* is replaced with *time_matrix*. It is an array of travel times between the locations.
  - *time_windows:* In case of time-windows constraint as well as resource constraint, it is a list of pairs denoting the range of time or the time window for the requested visits to the locations.
  - *vehicle_load_time:* In case of resource constraint, it is the time required to load the vehicle at the depot.
  - *vehicle_unload_time:* In case of resource constraint, it is the time required to unload a vehicle at the depot.
  - *depot_capacity :* In case of resource constraint, it is the maximum number of vehicles allowed to load or unload at the same time.

- ➢ Define *distance_callback* function which returns the distance between the location nodes. Cost of travel can also be passed into this function.
- ➢ Define *distance_dimension* object to compute the cumulative distance travelled by each vehicle. This cumulative distance is calculated to keep track of quantities to be accumulated over a vehicle's route.
- ➢ Define *print_solution* function to display the routes for the vehicles and the total distances of the routes.

4. RESULTS

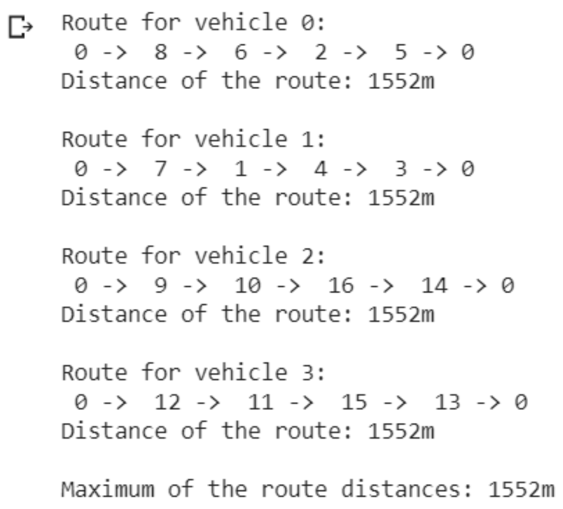
The output shows the set of the index of the location in an order that forms the route for the particular vehicle.

- *<u>Travelling Salesman Problem:</u>*



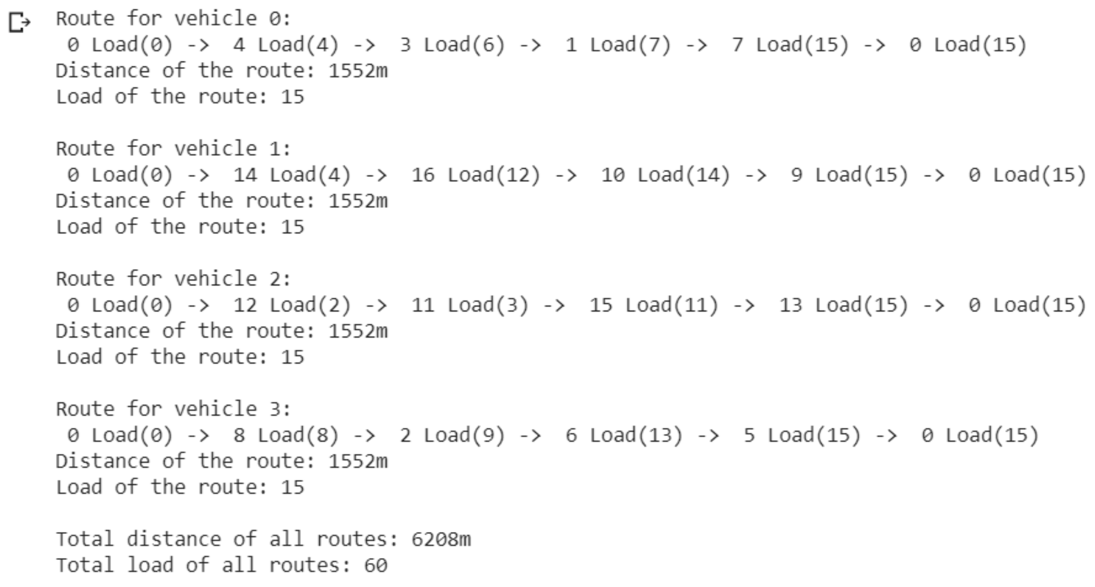
```
Objective: 7293 miles
Route for vehicle 0:
 0 -> 7 -> 2 -> 3 -> 4 -> 12 -> 6 -> 8 -> 1 -> 11 -> 10 -> 5 -> 9 -> 0
```

- *<u>Vehicle Routing Problem (No constraints included):</u>*



```
Route for vehicle 0:
 0 ->  8 ->  6 ->  2 ->  5 -> 0
Distance of the route: 1552m

Route for vehicle 1:
 0 ->  7 ->  1 ->  4 ->  3 -> 0
Distance of the route: 1552m

Route for vehicle 2:
 0 ->  9 ->  10 ->  16 ->  14 -> 0
Distance of the route: 1552m

Route for vehicle 3:
 0 ->  12 ->  11 ->  15 ->  13 -> 0
Distance of the route: 1552m

Maximum of the route distances: 1552m
```

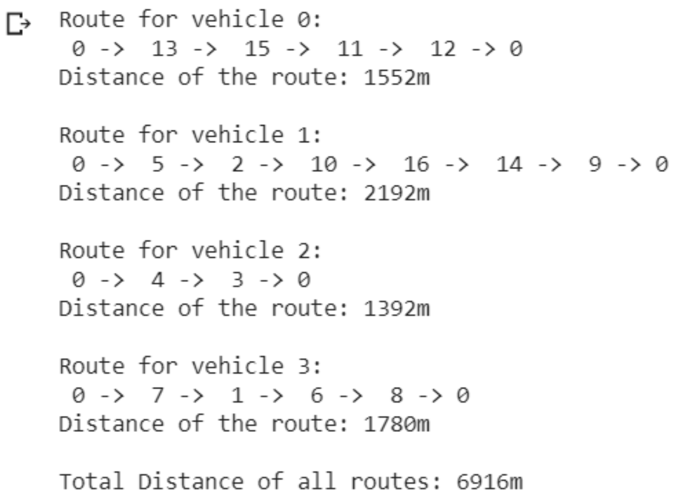
- *VRP with Capacity Constraint:*



```
Route for vehicle 0:
 0 Load(0) ->  4 Load(4) ->  3 Load(6) ->  1 Load(7) ->  7 Load(15) ->  0 Load(15)
Distance of the route: 1552m
Load of the route: 15

Route for vehicle 1:
 0 Load(0) ->  14 Load(4) ->  16 Load(12) ->  10 Load(14) ->  9 Load(15) ->  0 Load(15)
Distance of the route: 1552m
Load of the route: 15

Route for vehicle 2:
 0 Load(0) ->  12 Load(2) ->  11 Load(3) ->  15 Load(11) ->  13 Load(15) ->  0 Load(15)
Distance of the route: 1552m
Load of the route: 15

Route for vehicle 3:
 0 Load(0) ->  8 Load(8) ->  2 Load(9) ->  6 Load(13) ->  5 Load(15) ->  0 Load(15)
Distance of the route: 1552m
Load of the route: 15

Total distance of all routes: 6208m
Total load of all routes: 60
```

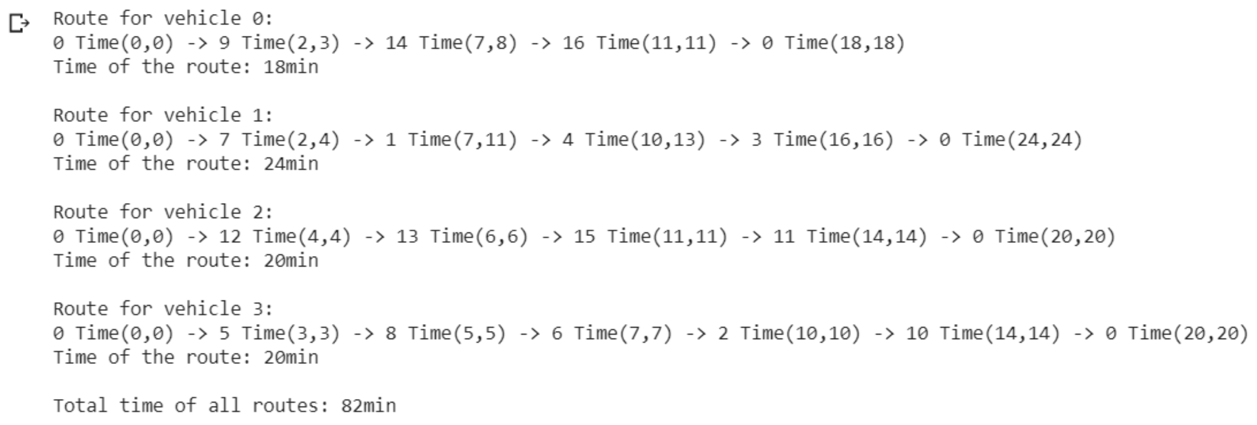
Along with the index of the location, the output also depicts *Load(a)*, in which '*a*' the total load carried by the vehicle when it departs that location.

- *VRP with Pickup-Delivery Constraint:*



```
Route for vehicle 0:
 0 ->  13 ->  15 ->  11 ->  12 -> 0
Distance of the route: 1552m

Route for vehicle 1:
 0 ->  5 ->  2 ->  10 ->  16 ->  14 ->  9 -> 0
Distance of the route: 2192m

Route for vehicle 2:
 0 ->  4 ->  3 -> 0
Distance of the route: 1392m

Route for vehicle 3:
 0 ->  7 ->  1 ->  6 ->  8 -> 0
Distance of the route: 1780m

Total Distance of all routes: 6916m
```

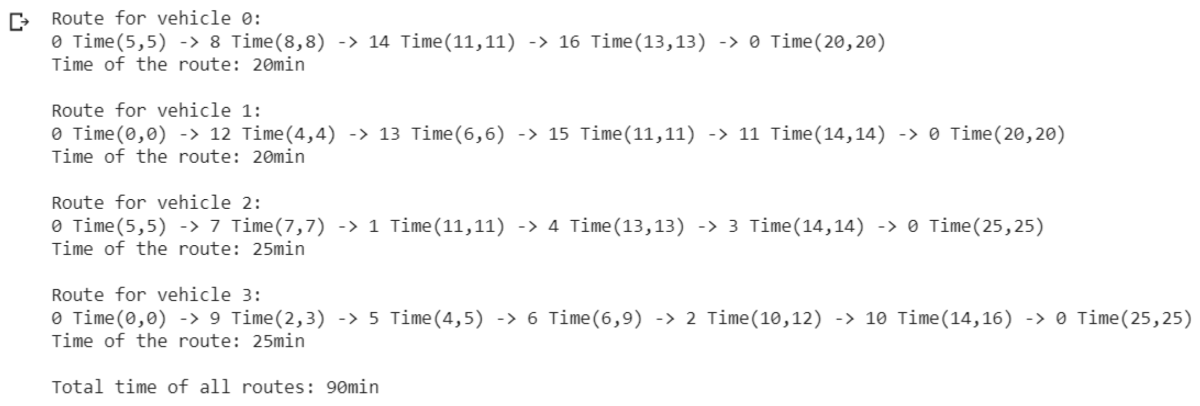
- *VRP with Time Windows Constraint:*



```
Route for vehicle 0:
0 Time(0,0) -> 9 Time(2,3) -> 14 Time(7,8) -> 16 Time(11,11) -> 0 Time(18,18)
Time of the route: 18min

Route for vehicle 1:
0 Time(0,0) -> 7 Time(2,4) -> 1 Time(7,11) -> 4 Time(10,13) -> 3 Time(16,16) -> 0 Time(24,24)
Time of the route: 24min

Route for vehicle 2:
0 Time(0,0) -> 12 Time(4,4) -> 13 Time(6,6) -> 15 Time(11,11) -> 11 Time(14,14) -> 0 Time(20,20)
Time of the route: 20min

Route for vehicle 3:
0 Time(0,0) -> 5 Time(3,3) -> 8 Time(5,5) -> 6 Time(7,7) -> 2 Time(10,10) -> 10 Time(14,14) -> 0 Time(20,20)
Time of the route: 20min

Total time of all routes: 82min
```

Along with the index of the location, the output also depicts *Time(a,b)*, in which '*a*' and '*b*' is the time interval in which the vehicle must visit the location to stay on its schedule.

- *VRP with Resource Constraint:*



```
Route for vehicle 0:
0 Time(5,5) -> 8 Time(8,8) -> 14 Time(11,11) -> 16 Time(13,13) -> 0 Time(20,20)
Time of the route: 20min

Route for vehicle 1:
0 Time(0,0) -> 12 Time(4,4) -> 13 Time(6,6) -> 15 Time(11,11) -> 11 Time(14,14) -> 0 Time(20,20)
Time of the route: 20min

Route for vehicle 2:
0 Time(5,5) -> 7 Time(7,7) -> 1 Time(11,11) -> 4 Time(13,13) -> 3 Time(14,14) -> 0 Time(25,25)
Time of the route: 25min

Route for vehicle 3:
0 Time(0,0) -> 9 Time(2,3) -> 5 Time(4,5) -> 6 Time(6,9) -> 2 Time(10,12) -> 10 Time(14,16) -> 0 Time(25,25)
Time of the route: 25min

Total time of all routes: 90min
```

Along with the index of the location, the output also depicts *Time(a,b)*, in which '*a*' and '*b*' is the time interval in which the vehicle must visit the location to stay on its schedule.

5. CONCLUSION

In essence, it is difficult to solve combinatorial optimisation problems. The VRP, CVRP and CVRP-TW are perhaps the most often used VRP problems in a number of logistical sectors. Owing to years of study, TSP solvers are extremely effective and are capable of solving a wide range of problems with very low computing times. We are aiming to continue exploring how we can broaden our concept to other constrained TSP variants like the TSP with penalties and drop-off trips along with some other dimensions, an essential and much more real life scenario which is more complicated. Further, UI design can also be implemented, which will make the results much easier to understand and will be more commercial.

Also dynamic time constraints can be implemented which can be helpful in a lot of sectors. For that, it is also important to increase dynamic data acquisition, as the state of traffic is not always stable. For instance, we cannot obtain potential traffic information in the afternoon while we schedule this tour in the morning.[6] This complex and scalable algorithm can then be used in the next version of this project.

6. REFERENCES

[1] Mosheiov, Gur. "The travelling salesman problem with pick-up and delivery." *European Journal of Operational Research* 79.2 (1994): 299-310.

[2] Toth, Paolo, and Daniele Vigo, eds. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.

[3] Ibrahim, Abdullahi Adinoyi, et al. "Capacitated vehicle routing problem." *International Journal of Research-Granthaalayah* 7.3 (2019): 310-327.

[4] Lau, Hoong Chuin, Melvyn Sim, and Kwong Meng Teo. "Vehicle routing problem with time windows and a limited number of vehicles." *European journal of operational research* 148.3 (2003): 559-569.

[5] Deudon, Michel, et al. "Learning heuristics for the tsp by policy gradient." *International conference on the integration of constraint programming, artificial intelligence, and operations research*. Springer, Cham, 2018.

[6] Sanguansat, Parinya. "Developing Applications for Vehicle Routing Problems with Real Time Data Acquisition." *origins* 13.100.532713&destinations (2019): 13-847881.