VRP Algorithm Challenge

Ekin NURBAŞ
nurbasekin@gmail.com

November 18, 2021

1 Introduction

In this case study, the problem of the vehicle routing and capacitated vehicle routing were investigated in different aspects. In the first section, definition of the VRP is given in short and the both exact and heuristic approaches for the solution of the given problem were given. Moreover, in the next sections two selected algorithms (Brute-Force and Genetic Algorithm) and their implementation details were given. Also, several simulations were done and the results were detailed in Section 3.

2 Vehicle Routing Problem

Vehicle routing problem [VRP] asks what is the optimal set of routes for a fleet of vehicles to traverse. by generalising the well-known travelling salesman problem (TSP) and minimizing the total route cost.

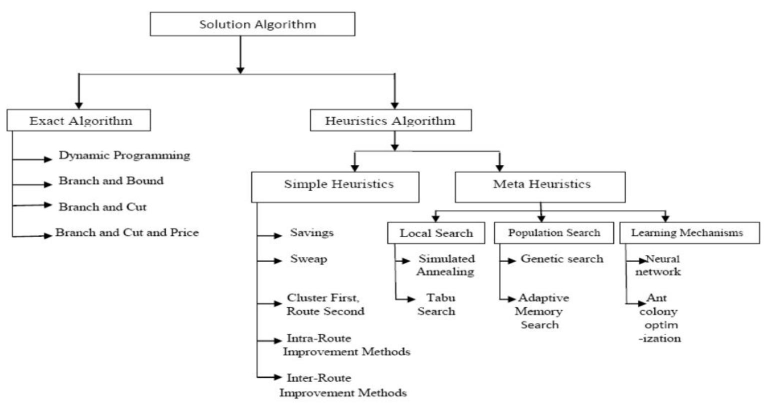


Figure 1: Exact and Heuristic Solutions for VRP [1]

In the figure above, list of the both exact and heuristics algorithms for VRP are given. As it can be seen in the figure, there exist multiple algorithms in the literature but in the scope of this challenge only Brute-Force and Genetic Algorithm were investigated.

2.1 Brute-Force Solution for VRP

In this approach, total route costs(duration) for each vehicle - jobs assignment combination and their route permutations were calculated in a loop to find the optimum assignment combination and as well as the optimum routes for each vehicle. Since the job locations stored in a vertex list as $V = 1, 2, 3, ..., m$ and there exist n available vehicles around the service area, algorithm steps given below can be applied to find the optimal solution for the given problem.

- Step 1: initialize $V = V_1, V_2, ..., V_m$ and $V_r = V$
- Step 2: Choose a set for V_i in V_r for vehicle i

- Step 3: Calculate the optimum route for V_i starting from the starting location of the vehicle i
- Step 4: Update the remaining vertices set $V_r = V\ V_n$
- Step 5: Repeat Step 2 to Step 4 for each vehicle and sum the costs for each vehicle's optimum route.
- Step 6: Repeat Step 2 to Step 5 until every jobs-vehicle combination is covered.
- Step 7: Find the optimum job assignment combination and optimal routes by looking for the minimum total route cost.

2.1.1 Implementation Details

In this section, details of the methods used in the Brute-Force Algorithm implementation were given.

- $bfsolver(V, n, C, S, D, L, capacitated)$: A recursive function which calculates every route cost for each n vehicle and V jobs, return the optimal route , route costs and total cost as the sum of the all route costs.

  V: Set of vertices(jobs)
  n: Number of vehicles
  C: Cost matrix
  S: Starting vertices of the vehicles
  D: Demand(Delivery) of the vertices(jobs)
  L: Limit(Capacity) of the vehicles
  capacitated: boolean which is True if the problem will be treated as CVRP and vice-versa

  Function chooses a subset V_i of V and calculates the optimal route for V_i. Updates the remaining vertex set $V_r = V\ V_i$, decreases the number of vehicles by one and recall itself with V_r and $n-1$ until number of vehicles reaches to the value of one. This procedure iterates for each subset V_i and at the end, optimal job-vehicle assignments and optimal routes for the vehicles are obtained. Moreover if the value of capacitated is True, algorithm considers the vehicle capacites and job demands so that, if any vehicle-job assignment combination violates the capacity condition of the vehicle, cost of the corresponding combination is assigned as maxint() value.

- $calcualte\_optimal\_route(V, C, k, K, V_p, capacitated)$: A recursive function which calculates the optimal route for the given set of vertices.

  V: Set of vertices(jobs)
  C: Cost matrix
  k: Number of remaining vertices
  K: Number of total vertices
  Vp: Previous vertex
  capacitated: boolean which is True if the problem will be treated as CVRP and vice-versa

  Function selects a vertex V_i in V and calculates the distance between V_p and selected vertex and recall itself with updated vertex set $V_r = V - V_i$ and $V_p = V_i$ until the number of remaining vertices reaches to one. This process iterates for each combination and the optimal route determined by looking for the minimum cost.
  Moreover if the value of capacitated is True, algorithm considers the returning back to the depot case by adding the cost for the last visited point to depot.

2.2 Genetic Algorithm for VRP

To implement the genetic algorithm approach for vehicle routing problem details and definitions in [2] were investigated and the summarized below.
GAs operate on well-established principles. A population of solutions is maintained, and a reproductive process enables the population's parent solutions to be selected. Offspring solutions are generated that exhibit characteristics of both parents. Each solution's fitness can be related to the value of the objective function, in this case the total distance traveled, and the degree to which any constraint

is violated. A population is more likely to survive and reproduce if its offspring have relatively high fitness levels, with the expectation that fitness levels throughout the population will continue to rise. The genetic coding for the VRP problem is defined as a vector which consist of m (number of jobs) integers which represents the vehicles. In this expression if the i^{th} element of the vector filled with integer j it represents that i^{th} job will be delivered by the vehicle j.

Jobs	1	2	3	4
Chromosome	1	1	2	3

(a) Caption

Parent 1	1	1	2	3
Parent 2	2	2	3	1

(b) Caption

Offspring 1	1	2	2	1
Offspring 2	2	1	3	3

(c) Caption

In the given example above, given chromosome vector represents that Job 1 and Job 2 will be delivered by Vehicle 1, Job 3 will be delivered by Vehicle 2 and Job 4 will be delivered by Vehicle 3. Moreover, the fitness value of each chromosome is considered inversely proportional to its optimum route cost.
In the initialization phase, multiple parents are generated to create a population. After initialization process two parents are randomly selected from the population and generates offsprings by crossover operation defined above.
After offsprings are generated, a mutation operation can be applied by randomly changing the value of the random index in the offsprings. After mutation process,optimum offspring routes as well as their total cost are calculated and compared with the parent list. If there exist a parent whose cost is larger than the offspring's cost, offspring and parent replaced. This procedure continues until the total iteration reaches to the predefined maximum iteration number.
With the given explanation above, algorithm can be summarized in seven steps given below.

- Step 1: Initialize population by randomly generated parents and evaluate their costs
- Step 2: Randomly select two parents in the population.
- Step 3: Crossover the selected parents and generate offsprings
- Step 4: Evaluate the cost of offsprings
- Step 5: If the cost of the offspring is less than the maximum cost of the parents, replace offspring with parent
- Step 6: Repeat Step 2 to Step 5 for predefined number of iterations
- Step 7: Find the optimum job assignment combination and optimal routes by looking for the minimum total route cost

2.2.1 Implementation Details

In this section, details of the methods used in the Genetic Algorithm implemtation were given.

- $gasolver(V, C, S, D, L, n, K, T, prob, capacitated)$: A function which generates a population with a given size K and iterates to generate offsprings and updates the population by given number of iterations T V: list of vertices
  C: cost matrix
  S: starting vertives of the vehicles
  D: Demand(Delivery) of the jobs
  L: Limit(Capacity) of the vehicles
  n: number of total vehicles
  K: population size
  T: number of iterations
  prob: probability of mutation
  capacitated: boolean which is True if the problem will be treated as CVRP and vice-versa

- $generate(V, n, K)$: A function which randomly generates a population with K members from a vertex set V according to the number of vehicles n.
  V: Set of vertices(jobs)
  n: Number of vehicles
  K: Population size

- $crossover(parents)$: A function which selects two parents from given population and apply the crossover operation to generate offsprings.
  parents: list of parent chromosomes

- $mutation(offsprings, n, prob)$: A function which changes random index of the given offsprings with random value with given probability. offsprings: list of offsprings
  n: total number of vehicles
  prob: probability of mutation

- $replace(parents, route\_parent, cost\_parent, cost\_total_parent,$
  $offspring, route\_offspring, cost\_offspring, cost_total_offspring, index)$: A function whic replaces the given offspring with the selected parent.
  parents: list of parents
  route_parent: list of parent routes
  patent_cost: list of parent costs
  cost_total_parent: total route costs of parents
  offspring: offspring to be replaced
  route_offspring: route of the offspring to be replaced
  cost_offspring: route costs of the offspring to be replaced
  cost_total_offspring: total route cost of the offspring to be replaced

- $evaluate(parents, V, C, S, D, L, n, capacitated)$: A function which evaluates the costs of the given parents.
  parents: list of parents to be evaluated
  V: list of vertices
  C: cost matrix
  S: start vertices of the vehicles
  D: Demand(Delivery) of the jobs L: Limit(Capacity) of the vehicles n: total number of vehicles
  capacitated: boolean which is True if the problem will be treated as CVRP and vice-versa

3 Simulation Results

In this section, performance evaluation of the algorithms were done over several experiments. In the experiments, number of vehicles, number of jobs and locations were changed and optimum routes as well as the optimum costs were investigated.

3.1 Experiment 1

In this experiment, six jobs at different locations will be delivered by three vehicles. Jobs locations were selected different than the vehicles starting points and the details about this selection, number of carboys to be delivered and vehicles capacities can be seen in the table below.

(a) Jobs Table

Jobs	0	1	2	3	4	5
Location	1	2	3	4	5	6
Delivery	1	1	1	1	1	2

(b) Vehicle Table

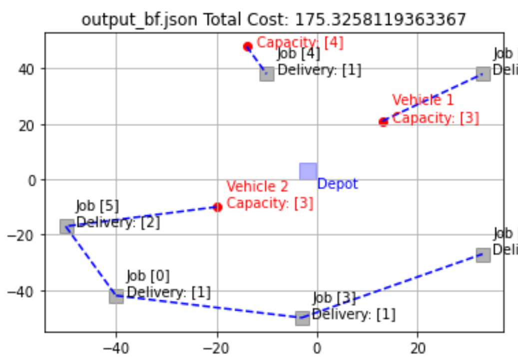
Vehicle	1	2	3
Start Location	7	8	9
Capacity	3	3	4

In addition, the cost matrix which consist of the duration between each two vertices (i,j) are given below So that the i^{th} row or column represents the duration from i^{th} vertex to other vertices.

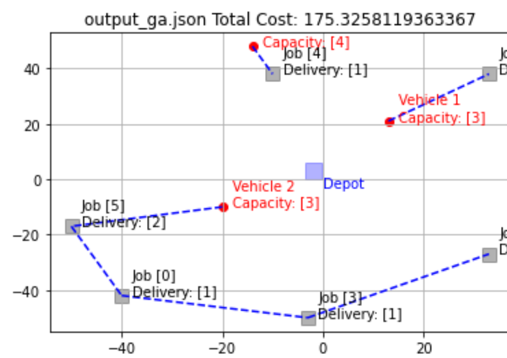
Table 3: Cost Matrix

Cost	0	1	2	3	4	5	6	7	8	9
0	0	58.8982	49.4975	46.0977	53.0094	35.9026	52	23.4307	22.2036	46.5725
1	58.8982	0	108.301	74.5252	37.855	85.44	26.9258	82.3286	37.7359	93.6803
2	49.4975	108.301	0	65	95.0789	43	99.5691	26.2488	71.5052	48.0521
3	46.0977	74.5252	65	0	42.72	77.9359	83.6002	52	55.6597	88.5099
4	53.0094	37.855	95.0789	42.72	0	88.278	57.4282	72.7805	43.4626	98.6154
5	35.9026	85.44	43	77.9359	88.278	0	68.0074	28.6007	49.0306	10.7703
6	52	26.9258	99.5691	83.6002	57.4282	68.0074	0	73.5731	30.8058	74.3034
7	23.4307	82.3286	26.2488	52	72.7805	28.6007	73.5731	0	45.2769	38.1838
8	22.2036	37.7359	71.5052	55.6597	43.4626	49.0306	30.8058	45.2769	0	58.3095
9	46.5725	93.6803	48.0521	88.5099	98.6154	10.7703	74.3034	38.1838	58.3095	0

For the scenario given above, both Brute-Force and Genetic Algorithm based solutions were run with capacitated and not capacitated options. In the figures given below, resulting routes for not capacitated problem is given. In this case, it is assumed that vehicles can carry infinite carboys so that they do not need to go back to the depot. Moreover additional parameters population size and number of iterations are selected as 2000 and 20 respectively.

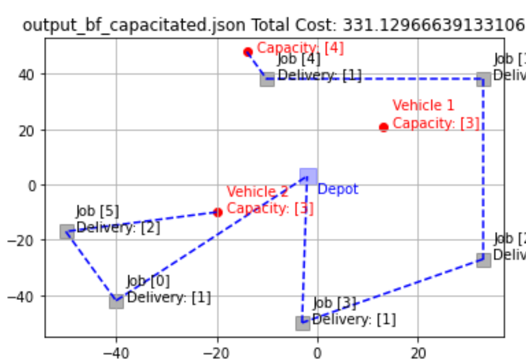


(a) Brute-Force Solution for Not Capacitated Vehicle Routing Problem

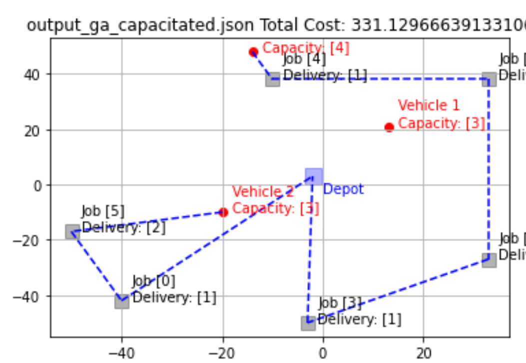


(b) Genetic Algorithm Solution for Not Capacitated Vehicle Routing Problem

As it can be seen from the figure given above, Genetic Algorithm reaches to the optimal solution obtained with the Brute-Force Algorithm. But it is not always guaranteed, especially if the initial population size (which was chosen as 2000 at first) and number of iterations (20) decreases, GA suffers in the sense of convergence to the optimal solution.



(a) Brute-Force Solution for Capacitated Vehicle Routing Problem



(b) Genetic Algorithm Solution for Capacitated Vehicle Routing Problem

In the figures above, resulting routes for the capacitated vehicle routing problem is given. In this case,

each vehicles has its own capacity which is given in the table above. So that, the number of carboys carried by a vehicle must satisfy its capacity condition.
As it can be seen from the figures, vehicle routes were changed since each vehicle can carry limited amount of carboys. For example in the previous not capaitated case, optimum solution was consist of a route where Vehicle 2 visits all job locations except Job 4 and Job 1. However it not possible in the capacitated scenario because Vehicle 2 can only carry 3 carboys. Moreover, since the vehicles has limited capacities, they must return to the depot after they complete their routes, therefore additional returning back to depot cost were also considered while searching for the optimal solution.

3.2 Experiment 2

In this scenario, similar to the previous experiment six jobs at different locations will be delivered by three vehicles. In the table below job locations, vehicle starting points, number of carboys to be delivered and vehicles capacities were given.

(a) Jobs Table

Jobs	0	1	2	3	4	5
Location	1	2	3	4	5	6
Delivery	2	1	1	1	2	2

(b) Vehicle Table

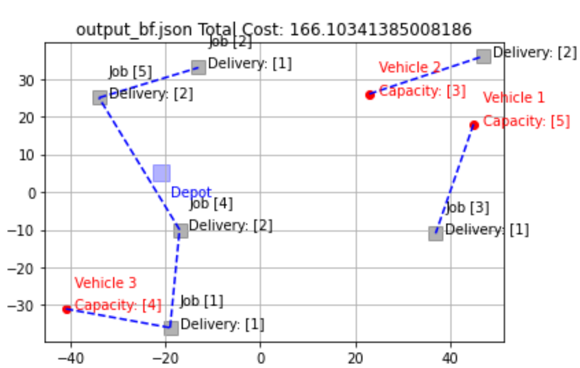
Vehicle	1	2	3
Start Location	7	8	9
Capacity	5	3	4

Moreover, similar cost matrix is given below to show the distances bewtween each two vertices.

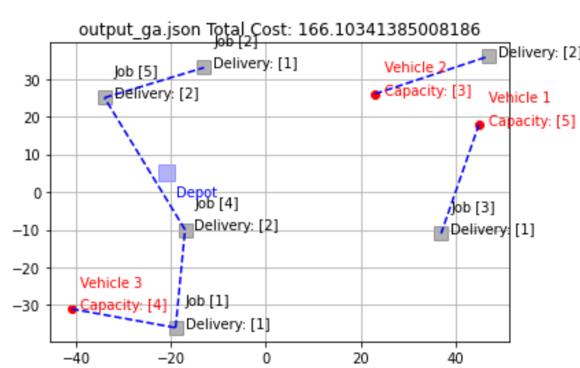
Cost	0	1	2	3	4	5	6	7	8	9
0	0	74.7329	41.0488	29.1204	60.1664	15.5242	23.8537	67.2681	48.7545	41.1825
1	74.7329	0	97.6729	60.075	48.0521	78.8162	81.7435	18.1108	26	110.603
2	41.0488	97.6729	0	69.2604	61.327	26.0768	62.8172	83.7377	74.8866	22.561
3	29.1204	60.075	69.2604	0	66.6033	43.1856	22.4722	59.9083	36.6742	69.857
4	60.1664	48.0521	61.327	66.6033	0	54.0093	79.6053	30.0832	39.5601	80.5233
5	15.5242	78.8162	26.0768	43.1856	54.0093	0	38.9102	68.0294	53.8145	31.8904
6	23.8537	81.7435	62.8172	22.4722	79.6053	38.9102	0	79.3095	57.0088	56.4358
7	67.2681	18.1108	83.7377	59.9083	30.0832	68.0294	79.3095	0	23.4094	98.9798
8	48.7545	26	74.8866	36.6742	39.5601	53.8145	57.0088	23.4094	0	85.703
9	41.1825	110.603	22.561	69.857	80.5233	31.8904	56.4358	98.9798	85.703	0

Table 5: Caption

For the scenario given above, both Brute-Force and Genetic Algorithm based solutions were run with capacitated and not capacitated options. In the figures given below, resulting routes for not capacitated problem is given.

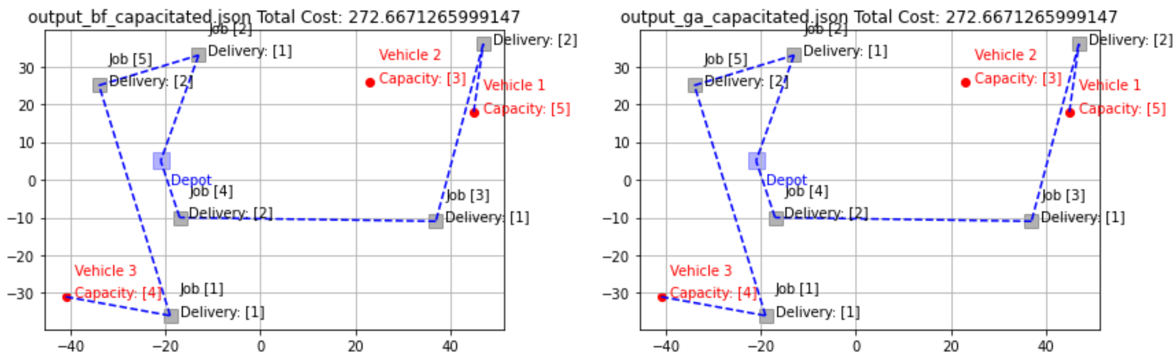


(a) Brute-Force Solution for Not Capacitated Vehicle Routing Problem



(b) Genetic Algorithm Solution for Not Capacitated Vehicle Routing Problem

In the figures above, resulting routes for the capacitated vehicle routing problem is given. As it can be seen from the figure given above, again Genetic Algorithm reaches to the optimal solution obtanied with the Brute-Force solution and the vehicle routes were not changed in this scenario.



(a) Brute-Force Solution for Capacitated Vehicle Routing Problem

(b) Genetic Algorithm Solution for Capacitated Vehicle Routing Problem

4 References

[1] W, Sapti S, Darmawan. (2015). The Characteristics Study Of Solving Variants Of Vehicle Routing Problem And Its Application On Distribution Problem.
[2] Barrie M. Baker, M.A. Ayechew,A genetic algorithm for the vehicle routing problem,Computers Operations Research,Volume 30, Issue 5,2003

5 Appendix

5.1 Solution Files

- *main.py*: Main file for calling the necessary functions to solve VRP and CVRP problem.
- *generate_test_case.py* : File that generates a test case for both VRP and CVRP problems and write the necesaary parameters to a .json file.
- *visualize_test_case* : File that visualizes the graph of the test scenario. To run this file, input .json file must contain position information of the vertices to be scattered in a plot. Therefore it can not be used for the given input json file for this challenge, however it can be used for the input files which are generated by the *generate_test_case.py* :
- *main.py*: main file to run the algorithms with the given input file.
- *bf.py*: File that contains the methods for Brute-Force Algorithm.
- *ga.py*: File that contains the methods for Genetic Algorithm.
- *RouteOptimizer.py*: File contains the base RouteOptimizer class as well as the BruteForce and Genetic Algorithmn classes
- $input_{t}est_{c}ase_{1}.json$: JSON file that constains the necessary parameters for the Experiment 1
- $input_{t}est_{c}ase_{2}.json$: JSON file that constains the necessary parameters for the Experiment 2
- $output_{b}f.json$: JSON file that constains Brute-Force Algorithm results for the given challenge input JSON file for not capacitated case.
- $output_{b}f_{c}ap.json$: JSON file that constains Brute-Force Algorithm results for the given challenge input JSON file for capacitated case

- $output_ga.json$: JSON file that constains Genetic Algorithm results for the given challenge input JSON file for not capacitated case.
- $output_bf_cap.json$: JSON file that constains Genetic Algorithm results for the given challenge input JSON file for capacitated case