Report for Project 2

Group 4:

Chaerim Lim

Charlotte Millot

Faheem Abdeen Kulam Magdoon

For project 2, we used Python and did our code on Google Colab to be able to all participate in the project. We have a set of vehicles with a known capacity, set of customers and depot and an optimal solution. We first imported the dataset to the colab and then list the global variables.

We represent the solution as a list of nodes separated by the number “-1” to separate the routes. Each route begins with a “-1” and we assume the “-1” at the beginning of the solution is there and we don’t put it explicitly in the route. Example :

[1, 2, -1, 3, 4] would give → Route 1 : [1,2] and Route 2 : [3,4]

To select the list of proper routes we use multiple functions :

**GeneratePopulation** where we create a list of chromosomes and for each we create nine routes and add the nodes randomly beginning with the ones with greater capacity until all routes are full. If it is not possible to respect a capacity of 100 for each route or if one route is empty, we just do another chromosome. We stop when all chromosomes are created.

Then to optimize the routes better to reduce the travelling time, the **function** **DoCrossover** selects two parents with a rank selection based on fitness. Our crossover is an order crossover but because of our choice for representing routes, we have to be careful when deleting the -1 from parent 2. We check that no route is empty and the capacity is respected with the function **check\_capacity** andthe function **next\_to** verify no -1 is next to each other**.** Fitness is evaluated with **EvaluateFitness** which returns for each chromosome 1/total\_distance with a geometric distance based on the coordinates of the nodes.

Then the function **DoMutation** uses the mutation rate to mutate the chromosomes with this probability. It uses three functions :

* **swapped** : swaps two nodes randomly and redo it until capacity is respected and no route is empty
* **shifted** : takes one element randomly and try to insert it at a random position until capacity is respected and no route is empty
* **inverted** : inverts two nodes until capacity is respected and no route is empty

To calculate distance, the **function** **calDis** where the nodes 1,2 are represented by four axis (x1,y1,x2,y2) and by the given formula in the question the distance is calculated and then the **function** **totalDistance** uses it to calculate the total distance of the solution and return it.

Then we have a **function GenerateFitList** which creates a list and fills it with the best evaluated elements.

The **function GAsolve is the main function**. It creates a new child list and a population list using the previous functions. A child of parent 1 and parent 2 is created using doCrossover and added to the population. **childList,newPop** are updated and the best solution is printed as the **BestChromosome** of the genetic algorithm with fitlist.

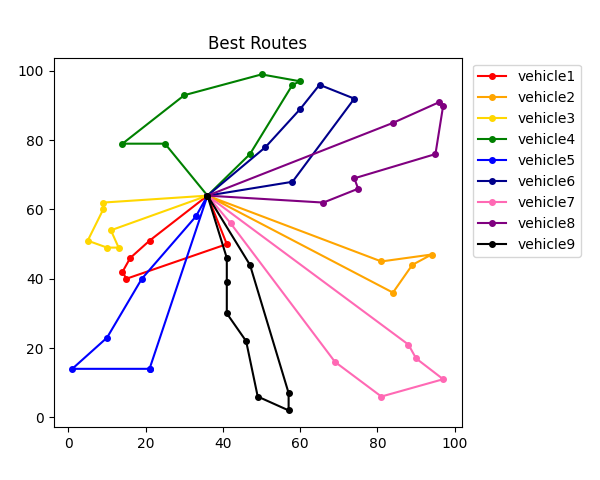
Now a **function routes\_from\_list** extracts the routes from the solution and prints them.

After running the program 30 times, the results are as follows:

| **Iteration** | **Total**  **Distance** | **CPU**  **time** | **Iteration** | **Total Distance** | **CPU**  **time** | **Iteration** | **Total**  **Distance** | **CPU**  **time** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 1114 | 193 | **11** | 1104 | 199 | **21** | 1139 | 188 |
| **2** | 1134 | 195 | **12** | 1141 | 192 | **22** | 1141 | 188 |
| **3** | 1109 | 200 | **13** | 1148 | 189 | **23** | 1156 | 200 |
| **4** | 1129 | 201 | **14** | 1170 | 189 | **24** | 1189 | 181 |
| **5** | 1165 | 200 | **15** | 1173 | 199 | **25** | 1174 | 198 |
| **6** | 1145 | 202 | **16** | 1103 | 196 | **26** | 1180 | 188 |
| **7** | 1167 | 195 | **17** | 1137 | 200 | **27** | 1138 | 194 |
| **8** | 1107 | 200 | **18** | 1145 | 200 | **28** | 1136 | 188 |
| **9** | 1186 | 189 | **19** | 1148 | 192 | **29** | 1142 | 190 |
| **10** | 1182 | 200 | **20** | 1166 | 182 | **30** | 1202 | 187 |

The average of Total Traveling Distance and CPU[[1]](#footnote-0) time is 1149, 193.8 seconds.

Our best solution is #11: total traveling distance is 1104.



| **Route #1:** 21 34 17 3 37  **Route #2:** 6 1 45 43  **Route #3:** 26 14 46 20 31 42  **Route #4:** 25 19 27 22 30 7 4  **Route #5:** 36 33 2 51 11 15  **Route #6:** 29 54 13 28 41  **Route #7:** 8 47 39 49 9 35  **Route #8:** 23 52 24 44 50 48 18  **Route #9:** 12 10 5 53 40 16 38 32 |
| --- |

1. 12th Gen Intel(R) Core(TM) i7-1260P 2.10 GHz [↑](#footnote-ref-0)