The Shortest Route

CAPACITATED VEHICLE ROUTING PROBLEM

1st Jakub Michalewicz
Wroclaw University of Technology
Wroclaw, Poland
226741@student.pwr.edu.pl

2nd Hubert Grobelny
Wroclaw University of Technology
Wroclaw, Poland
235281@student.pwr.edu.pl

***Abstract*—The following documentation describes the project The Shortest Route. The goal of the project is to develop an application that solves the problem of routing. The program implements the appropriate optimization algorithms and presents the results using a simple graphical interface.**

***Index Terms*—Vehicle routing problem, Transport routes**

I. Introduction

The project can support small transport management companies. The Shortest Route applies to capacitated vehicle routing problem. The problem is to determine the optimal transport routes for a certain number of means of transport, whose task is to serve a set of customers at different points. An example is below.

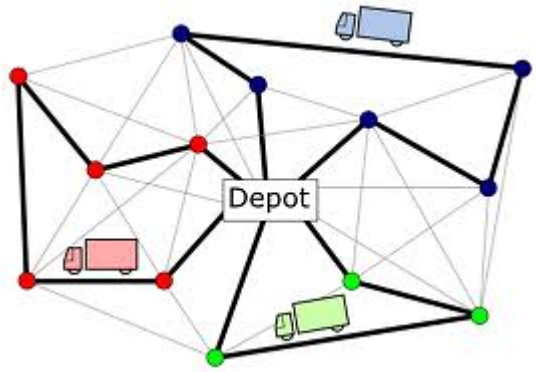

Fig. 1. Example of a routing problem.

In the middle is Depot from which the trucks take off and drive with the load to customers. The program will calculate the shortest route for trucks. Option CVRPLIB - Capacitated Vehicle Routing Problem Library was chosen. It is a variant of the routing problem with additional conditions regarding the load capacity of means of transport

II. Problem definition

The first step was to download data files and process them to the appropriate format. Program data is taken from this page: http://vrp.galgos.inf.puc-rio.br/index.php/en/. The variables in the sample file are:

- CAPACITY - int, capacity limit per vehicle,
- NORDE_CORD_SECTION - matrix of points to handle,

$$\text{distance\_matrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{n1} & a_{n2} \end{bmatrix}$$

- DEMAND_SECTION - demand vector,

  demand = $[\ 0 \quad b_1 \quad b_2 \quad \dots \quad b_n\ ]$ the starting point has a demand of 0.

- dist - matrix of distance between points (cost of a given segment) calculated on the basis of Euclidean distance,

$$\mathrm{d}_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

$$\text{dist} = \begin{bmatrix} 0 & d_{12} & \dots & d_{1n} \\ d_{21} & 0 & \dots & d_{2n} \\ d_{n1} & d_{n2} & \dots & 0 \end{bmatrix}$$

- num_of_vehicles - int, number of vehicles. Value taken from the data description with an example format: „*COMMENT: (Christophides and Eilon, Min no of trucks: 3, Optimal value: 569)*”
- time_limit -int, maximum algorithm running time

III. Algorithms description

Python and the OrTools library were used to solve the routing problem. They are presented in figures 2 and 3.

Fig. 2. Logo of Python.

Fig. 3. Logo of Python.

Algorithms used:

- Path cheapest - starting from the initial node, the algorithm looks for the shortest route (with the lowest cost) and continues the same operation for subsequent nodes,
- Christofides - an approximation algorithm, which assumption is to determine the minimum spanning tree for the graph in which we are looking for the optimal route and to attach to it edges containing the smallest distances between the vertices of the odd-numbered tree. Then the Euler cycle is determined for the graph, which in the process is converted into a Hamilton cycle by shortening, i.e. skipping visited vertices,

Metaheuristics used:

- Simple gradient - a numerical algorithm looking for a minimum of the local target function,
- Guided local search - a meta heuristic method that uses penalties to get out of local minima,
- Taboo search - the algorithm searches the space of all possible solutions using a sequence of motions, avoids oscillations around the local optimum because it stores information about already tested solutions.

The first algorithm may not find the optimal solution or fall into the local minimum, which is why the latter are also looking for a better solution. Their duration of action is determined in the last step of application.

IV. Simulation Environment and Testing Scenarios Description

The results were analyzed for the selected data set due to the range of the analysis, but in the program it is possible to obtain a solution for other data sets. There are 8 files in total.

Instancja E-n22-k4, solution 1

Data: capacity - 6000,
num_of_vehicles - 4, optimal value - 375
Algorithm: Path cheapest
Meta heuristics: Simple gradient
Time limit: 10 s, 30 s, 60 s

Result

Vehicle No. 1:
Vehicle path: 0 → 10 → 9 → 7 → 3 → 4 → 11 → 0
Route length: 119
Loading the vehicle: 5300

Vehicle No. 2:
Vehicle path: 0 → 5 → 2 → 1 → 6 → 8 → 13 → 0
Route length: 132
Loading the vehicle: 5700

Vehicle No. 3:
Vehicle path: 0 → 12 → 15 → 18 → 19 → 0
Route length: 104
Loading the vehicle: 5600

Vehicle No. 4:
Vehicle path: 0 → 14 → 17 → 20 → 21 → 16 → 0
Route length: 83
Loading the vehicle: 5900

Total route length: 438
Total load: 22500

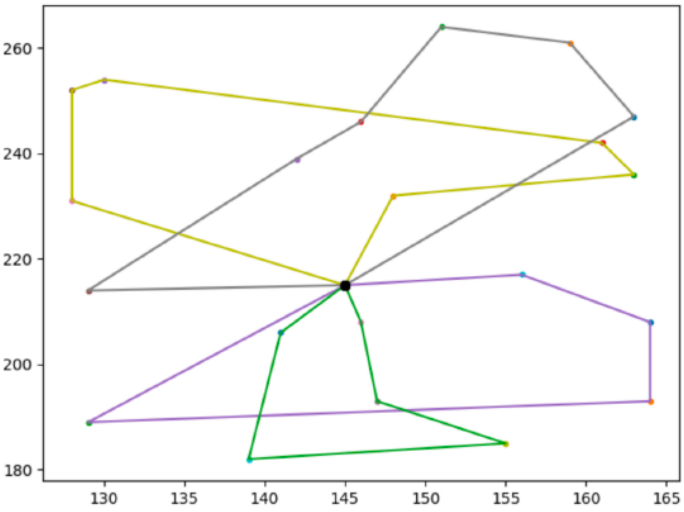

Fig. 4. E-n22-k4, solution 1

The algorithm has not found the optimal solution for the above mentioned parameters. Vehicle routes are chaotic, but the search time is short. Changing the time limit does not change the duration of the program.

Instance E-n22-k4, solution 2

Data: capacity - 6000,
num_of_vehicles - 4, optimal value - 375
Algorithm: Christofides
Meta heuristics: Simple gradient
Time limit: 10 s, 30 s, 60 s

Result

Vehicle No. 1:
Vehicle path: 0 → 16 → 14 → 7 → 2 → 1 → 6 → 0
Route length: 130
Loading the vehicle: 6000

Vehicle No. 2:
Vehicle path: 0 → 19 → 21 → 9 → 5 → 0
Route length: 150
Loading the vehicle: 5800

Vehicle No. 3:
Vehicle path: 0 → 8 → 3 → 4 → 11 → 13 → 0
Route length: 100
Loading the vehicle: 4800

Vehicle No. 4:

Vehicle path: 0 → 17 → 20 → 18 → 15 → 12 → 0
Route length: 83
Loading the vehicle: 5900

Total route length: 463
Total load: 22500

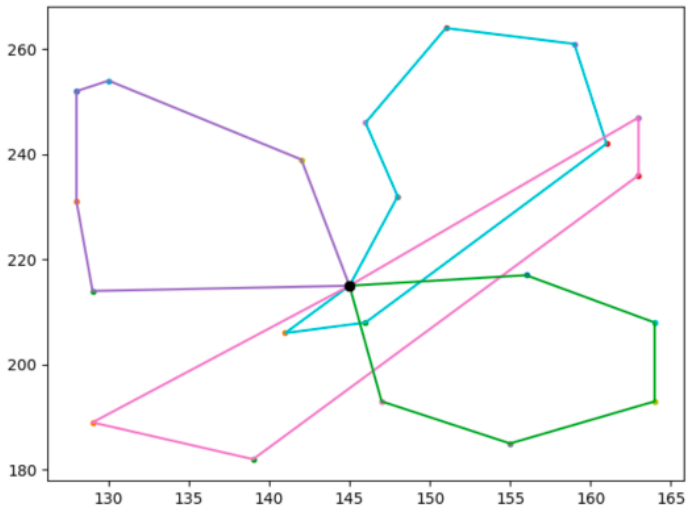

Fig. 5. E-n22-k4, solution 2

The algorithm has not found the optimal solution for the above mentioned parameters and it is worse than previously obtained with the shortest path algorithm. The duration of searching for a solution is short and unchanging for the time limit.

Instance E-n22-k4, solution 3

Data: capacity - 6000,
num_of_vehicles - 4, optimal value - 375
Algorithm: Path cheapest
Meta heuristics: Guided local search
Time limit: 10 s

Result

Vehicle No. 1:
Vehicle path: 0 → 13 → 11 → 4 → 3 → 8 → 10 → 0
Route length: 102
Loading the vehicle: 5400

Vehicle No. 2:
Vehicle path: 0 → 6 → 1 → 2 → 5 → 7 → 9 → 0
Route length: 113
Loading the vehicle: 5600

Vehicle No. 3:
Vehicle path: 0 → 17 → 20 → 18 → 15 → 12 → 0
Route length: 83
Loading the vehicle: 5900

Vehicle No. 4:
Vehicle path: 0 → 14 → 21 → 19 → 16 → 0
Route length: 77
Loading the vehicle: 5600

Total route length: 375
Total load: 22500

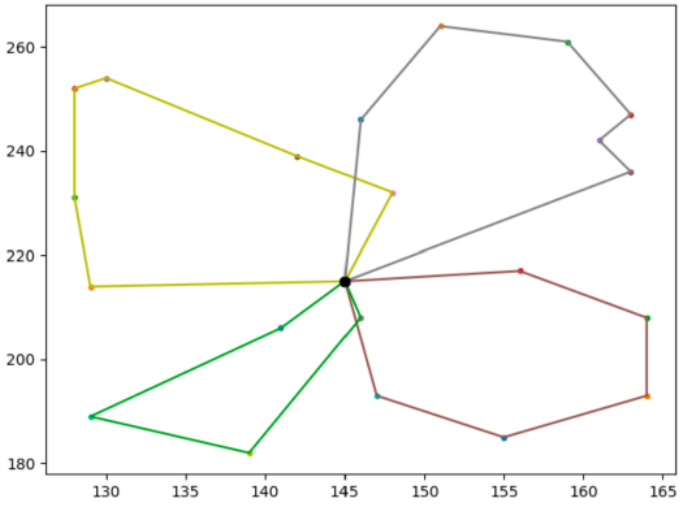

Fig. 6. E-n22-k4, solution 3

An optimal solution was obtained for the above mentioned parameters. The program was launched only once with a time limit of 10 seconds, because the optimal solution was found the first time.

Instance E-n22-k4, solution 4

Data: capacity - 6000,
num_of_vehicles - 4, optimal value - 375
Algorithm: Path cheapest
Meta heuristics: Tabu search
Time limit: 10 s

Result

Vehicle No. 1:
Vehicle path: 0 → 10 → 8 → 3 → 4 → 11 → 13 → 0
Route length: 102
Loading the vehicle: 5400

Vehicle No. 2:
Vehicle path: 0 → 9 → 7 → 5 → 2 → 1 → 6 → 0
Route length: 113
Loading the vehicle: 5600

Vehicle No. 3:
Vehicle path: 0 → 12 → 15 → 18 → 20 → 17 → 0
Route length: 83
Loading the vehicle: 5900

Vehicle No. 4:
Vehicle path: 0 → 16 → 19 → 21 → 14 → 0
Route length: 77

Loading the vehicle: 5600

Total route length: 375
Total load: 22500

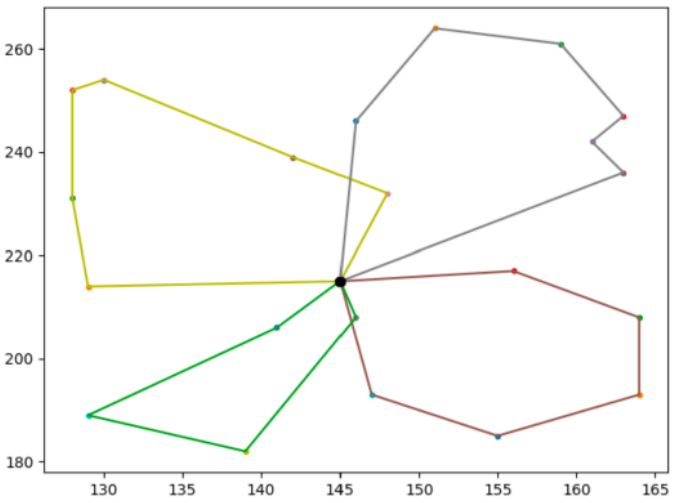

Fig. 7. E-n22-k4, solution 4

The algorithm has found the optimal solution for the above mentioned parameters. Using the search tab gave the same result as using guided local search, but the vehicle path (route) is reversed.

The Christofides algorithm using the search tab also obtained the optimal solution, but in this case the vehicle route looks as follows:

Vehicle No. 1:
Vehicle path: $0 \rightarrow 13 \rightarrow 11 \rightarrow 4 \rightarrow 3 \rightarrow 8 \rightarrow 10 \rightarrow 0$

Vehicle No. 2:
Vehicle path: $0 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 0$

Vehicle No. 3:
Vehicle path: $0 \rightarrow 17 \rightarrow 20 \rightarrow 18 \rightarrow 15 \rightarrow 12 \rightarrow 0$

Vehicle No. 4:
Vehicle path: $0 \rightarrow 14 \rightarrow 21 \rightarrow 19 \rightarrow 16 \rightarrow 0$

V. Results analysis

Of the two algorithms used, Path Cheapest has repeatedly presented better results than Christofides' using the same meta heuristics (not in every case). For the E-n22-k4 and taboo search data set, both algorithms returned the optimal solution. Definitely the worst results were obtained using a simple gradient, which returned the solution in a very short time, but far from optimal. Increasing the time limit did not affect its operation.

VI. Summary

The main goal of the project was to solve the capacitated vehicle routing problem so that the prepared application could help a small company that manages trucks. At this point, the application is able to complete the set goal. The Shortest Route can determine the optimal transport routes for a certain number of modes of transport. The final solution is displayed in the form of a diagram.

At this point, anyone who has Python with the OrTools library and Matplotlib can run the application.

There were no complications when creating the project. However, organization in the group was problematic because one member abandoned the project.

In the future, a graphical interface is planned to be added to the application because it currently uses the CMD console. In addition, it is planned to create a database that will store data files, because they are currently stored locally in text files. These files will then also be added from the graphical interface. With this change, the application can be used transport companies.