ROUTE OPTIMIZATIONS USING CLUSTERING AND GENETIC ALGORITHMS

Alexandru – Ionut MUSTAȚĂ[1], Tudor – Robert TUDORUȚ[2], Anca MORAR[3]

Lately, an increase in traffic on public roads has been observed to be correlated with the increase in the number of vehicles. This directly affects the time periods needed to travel certain distances inside a city. In the present situation, there is a need to create a way of managing routes distributed for the courier companies that are forced to face the high degree of cars on the roads, the increase of fuel prices and the increasingly excessive customer demands related to the delivery in time of the ordered parcels. We propose a new artificial intelligence route planning algorithm that solves the "Last Mile Delivery" problem and compare it with existing solutions.

Keywords: algorithm, artificial intelligence, route planning, cluster, architecture

1. Introduction

In recent years, considerable progress has been made in the area of route calculation for cars. These advances were most noticeable in reducing fuel consumption, thus addressing a problem of the present generation, as well as the efficient use of resources by planning routes in advance. The extent of the development of map programs both of web nature and those incorporated in cars show an increase of interest in the field of route planning.

We propose a new algorithm for solving a problem that derives from those of route planning, namely the distributed route planning in order to solve the "Last Mile Delivery" [1] problem.

In the following section we will analyze the main existing tools for calculating and planning routes using different techniques. Section 3 describes the architecture of our route planning system, and in section 4 we present the proposed algorithm. In section 5 we evaluate the new route planning algorithm, in comparison with Google Maps, Bing Maps and HERE Maps. Section 6 presents the conclusions.

[1] Eng., MSc. Student, Dept. of Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: alexmustata19@gmail.com
[2] Eng., MSc. Student, Dept. of Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: almantud@gmail.com
[3] Associate Professor, Dept. of Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: anca.morar@cs.pub.ro

2. Related Work

This section details the main problems of route planning and several existing heuristics that solve them.

2.1. The Traveling Salesman Problem

The traveling salesman problem (TSP) [2] asks the following question: "Given a list of cities and distances between each city, which is the shortest possible route that visits each city and is returning to the starting point?". This is a NP-hard problem with a special importance in the field of mathematics and computers.

The "Held-Karp Lower Bound" [3] offers an approximation for the optimal tour, considering that for instances of the problem with a number of very large nodes the calculation using a dynamic programming method would take a long time (this would have at best a temporal complexity of $O(2^n \times n^2)$ and spatial complexity of $O(2^n)$). As a consequence, that the main approach is studying big instances of this problem, it has become a practice in the field to compare the heuristics with a standard in order to obtain a ranking of them. This is called the "lower limit of length due to Held - Karp" [4]. The limit represents the simplified solution of the TSP for linear programming.

In the following paragraphs we present several heuristics that solve the TSP problem.

Nearest neighbor [5] is the most popular and intuitive solution to this problem because choosing a new node at each step will look for the nearest neighbor and add it to the resulting tour. Thus, the complexity of this algorithm is $O(n^2)$. There is also a greedy version of the algorithm, with a complexity of $O(n^2 \times \log n)$, which connects the nodes such that they do not have a degree greater than two and no cycles are formed smaller than a proportional length with the maximum distance between cities. Normally, the optimum route length is between the minimum spanning tree (MST) [6] length and twice that length. The intuitive explanation is that if we double each edge of the tree we will obtain a cycle that visits all the nodes.

In *the Clark – Wright heuristic* [7], a node is chosen as a warehouse (initial starting point) at which the traveler will go before visiting any other city. Thus, all the addresses in the city will be connected with the address of the warehouse through two edges. For the addresses that do not have the status of deposit (that is, from the respective address you can only perform the operation of lifting the packages), edge cuts will be applied, in order to lower the cost of the tour by prohibiting the creation of cycles that contain only non-deposit cities. This heuristic has a temporal complexity of $O(n^2 \times \log n)$.

The *Christofides heuristic* [8] is an approximation algorithm designed in 1976 which in the worst case offers a solution 1.5 times slower than the optimal solution. The algorithm has four simple steps:

- Find the minimum spanning tree
- Create duplicates for each edge of the tree in order to create an Eulerian graph
- Find an Eulerian cycle through all the nodes of the graph
- Convert the problem to TSP. If a city is visited twice, a shortcut will be created from the city visited before to the one visited after it.

This algorithm has a temporal complexity of $O(n^3)$, but with the help of certain optimizations it can reach a temporal complexity of $O(n^2 \times \sqrt{n})$.

2.2. The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) [9] [10] [11] [12] [13] represents an extra step towards finding the perfect algorithm for the real case of the problem because it requires the division of the TSP instance into several "sub-routes", or, as the VRP specifies - "vehicle fleets".

Wilck and Cavalier introduced the concept of Split Delivery Vehicle Routing Problem (SDVRP) [14] which allows customers to be assigned to multiple routes. Their solution uses a hybrid genetic algorithm that uses different fitness functions such as shortest route to find the best tour given a set of locations.

2.3. The Capacity Vehicle Routing Problem with Time Windows

Another constraint in route planning is limiting the transport capacity per vehicle, through the Capacity Vehicle Routing Problem (CVRP) [15]. This way, one can simulate a situation encountered in real life by the delivery companies.

The last constraint, which also represents the purpose of the present paper, is related to time. In the Capacity Vehicle Routing Problem with Time Windows (CVRP – TW) [16], couriers are constrained to be present with the cars in certain locations at a certain time. Basically, it is the real problem of a fleet of vehicles with finite capacity respecting time windows.

Solving the CVRP – TW problem requires additional information, such as: statistics on traffic at the time of delivery, the current status of the roads (under construction, blocked, usable, etc.) and the degree of loading per courier (one working day is equivalent to eight hours). This approach aims, for a car, to maximize the number of deliveries (of addresses visited) achieved, as well as to minimize fuel consumption and human resources.

In order to obtain such an algorithm, an incremental approach is required starting from the types of problems mentioned above, due to the fact that a direct approach is too difficult to ensure the compliance with all the constraints of the real life problem.

Gao, Sun and Yuan [17] are currently researching the same topic but taking into account a different nature of the problem. They are using genetic algorithms combined with knapsack problem logic in order to create a method that can split deliveries of large orders to a fleet of vehicles. The purpose of their work is to find an optimum combination in which vehicles will be loaded with parcels as close as possible to their maximum capacity.

3. The Route Planning System

In order to test various route planning algorithms, we designed a system that consists of a virtual private network, a distance metrics calculator, a geocoding server, a map server, a database and a web server, as illustrated in Fig. 1.

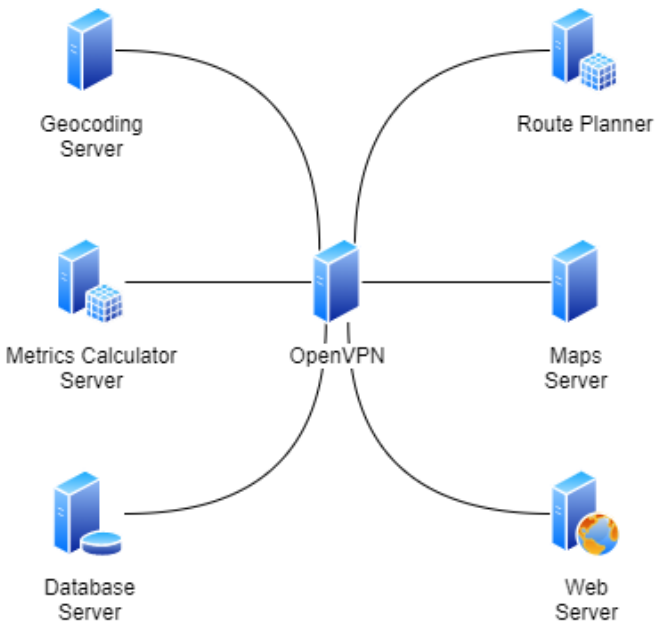

Fig. 1. The architecture of the proposed route planning system

OpenVPN [18] is a tool that creates a Virtual Private Network (VPN) that secures the internal network. This allows us to expose to the public only the part of the infrastructure that is required for interfacing with the system, respectively

the website and the map server. This measure was taken to prevent future network attacks.

To be able to have data on the road between two locations (both in time and distance) we used a server that computes the distance between each two locations. For the *distance metrics calculator* we chose the Open Street Routing Machine project (OSRM) [19] which has an API that calculates the matrix of distances between *N* points.

The input of the API of the above-mentioned OSRM system consists of points codified as a tuple of (longitude, latitude). Since our proposed algorithm receives addresses in textual format, such as "Bucharest, Calea Victoriei 18", a *geocoding server* is required to perform transformation of the addresses in the mentioned tuple form.

Since the OSM project [20] is an open source project updated very often, we used it as a *map server*. Currently we limit the system to the map of Romania, but thanks to the OSM construction we can easily add maps of other countries.

We have created a server that will store the maps. In addition, we have made changes to the map tile rendering service to be able to accept API keys for the map rendering service as well. In this way we will have control over the use of our map system by authenticating each user with an API key. In the future, this service will be modified to provide an API management system for the available keys.

To solve the requirement of having a map system that can be updated very quickly, we opted for storing the maps in a PostgreSQL [21] *database*. PostgreSQL uses the postgis [22] extension to save the vectors that subsequently make up a tile of the map.

The user interface of the system consists of a *website* created with a combination of Nginx [23] and PHP 7.0 [24]. The website, which hosts the graphical interface which facilitates the communication with our routing algorithm, was created with the Bonfire [25] framework, an extension of the CodeIgniter [26] framework.

4. The Route Planning Algorithm

The pipeline of the proposed algorithm is presented in Fig. 2. Following a request to calculate a route, this request is transmitted to the routing server which will subsequently communicate with the address translation and route generation server to obtain the final route.

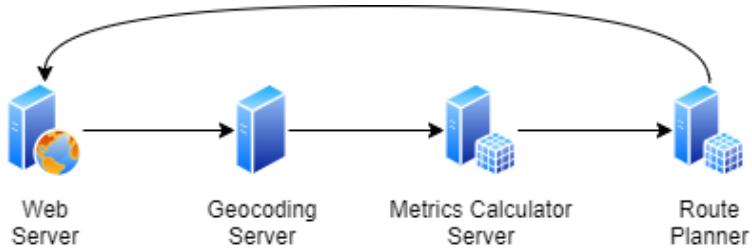

Fig. 2. The pipeline of the proposed route planning algorithm

4.1. Translating Addresses into Geographical Coordinates

The first step of the route generation process is to determine the geographical information of the addresses that are part of the initial request. Within the algorithm, we geocode the addresses through the Nominatim system [27] in tuples like (longitude, latitude). Addresses that will not be geocoded will be returned to the web interface so that the user will be notified if geocoding errors occur.

4.2. Concentrated Areas of Interest

The problem we are trying to solve is twofold. The first issue is the creation of concentrated areas of interest, visiting clusters, for each vehicle, while the second problem is represented by the calculation of the optimal route in each resulting area. A concentrated area of interest is an area that should be visited with a single vehicle.

Grouping the locations into concentrated areas of interest is done with the K-Means algorithm [28], which generates groups of addresses according to their geographical location.

4.3. Optimal Route Calculation

After obtaining the areas of interest for each vehicle, it is necessary to compute the optimum route for each area.

The optimal route calculation starts from a basic model of a genetic algorithm [29] [30]. The most difficult step was determining the component functions of the genetic algorithm, namely the fitness function, the selection function, the cross-over function and the mutation function.

The fitness function measures the quality of the individuals, estimating the time required to complete the route as well as the length of the journey. Subsequently the individuals are ranked in decreasing order according to the scores they obtained for these two criteria.

For the *selection function*, we considered that the best 20% individuals in a generation are enough to create a new generation. This percentage was chosen due to the fact that it offers us a fairly large variety (80%) in the newly created generation and at the same time ensures the preservation of the elitist genes of the last generation thus improving the chances of finding a near-optimal route.

For the *cross-over function* we used the "Longest Common Substring" (LCS), the longest continuous substring in a string, to obtain the best gene from both parents.

The *mutation function* interchanges two locations in the journey by applying a 2-OPT type move [31].

5. Evaluation

In this section we provide comparisons with other de facto systems from the routing domain that offer the possibility to calculate routes between multiple points. The de facto systems that we decided to use for comparison are: Google Maps, Bing Maps and HERE Maps, because of the routing options you are allowed to use.

5.1. Comparison for One Vehicle

In Figs. 3, 4, 5 and 6 we present the routes generated by our solution, Google Maps, Bing Maps and HERE Maps respectively, for a single vehicle that must reach three destinations in Bucharest (Obor Square, Romana Square, Victoriei Square), starting from Crangasi Square.

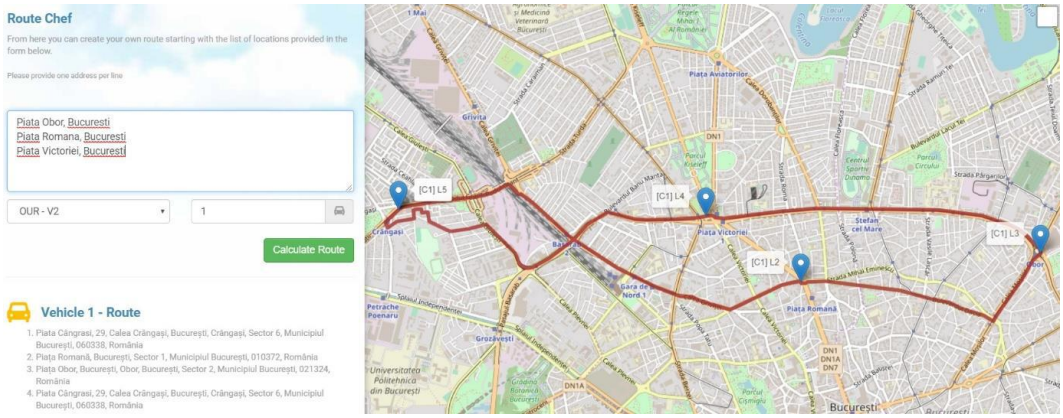

Fig. 3. Route generated with proposed algorithm for one vehicle

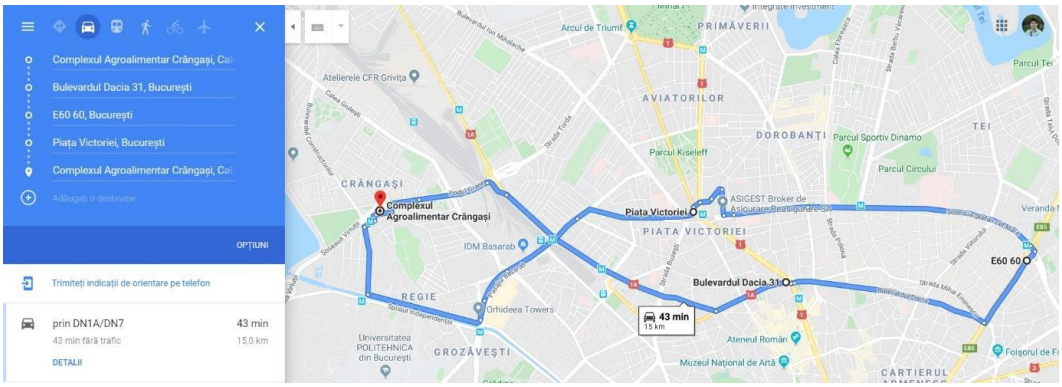

Fig. 4. Route generated with Google Maps for one vehicle

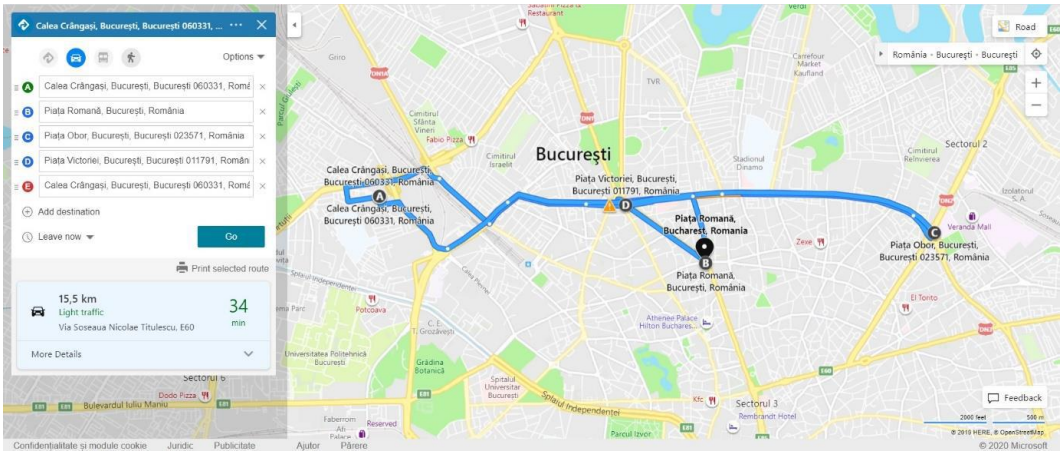

Fig. 5. Route generated with Bing Maps for one vehicle

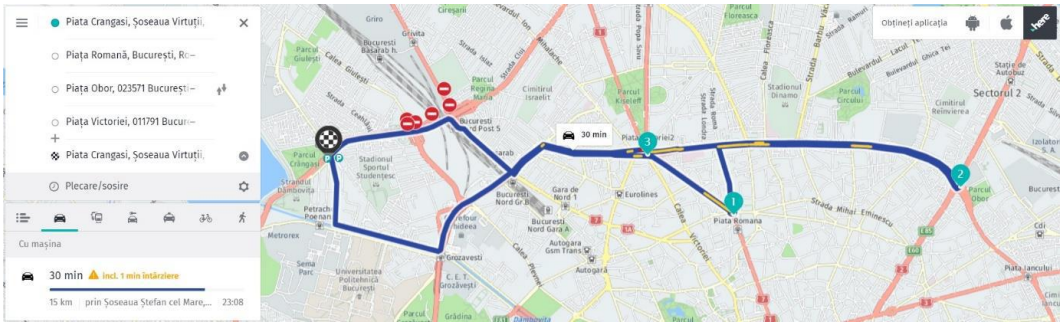

Fig. 6. Route generated with HERE Maps for one vehicle

Table 1 presents the total distance and estimated travel time that for each solution. It can be observed that the proposed algorithm offers a good result for the tour problem. Another aspect worth mentioning is that our solution

approximates a noticeably shorter time for the tour because we have not yet implemented a real-time traffic data processing service. The same behavior can be observed for HERE Maps, where either the traffic is not updated in real time, or the departure date is not being taken into account. Even if the simulations for Google Maps and Bing Maps were run at nighttime and with the traffic off option, it seems that these applications still consider some constraints due to traffic.

Table 1

Comparison of distance to travel and estimated travel time with Google Maps, Bing Maps HERE Maps for one vehicle

System	Results	
	Distance to travel (km)	Estimated travel time (minutes)
Proposed algorithm	14.153	23
Google Maps	15	43
Bing Maps	15.5	34
HERE Maps	15	30

5.2. Comparison for Multiple Vehicles

We also compare our solution with Google Maps on each zone generated for a multiple vehicle instance. For this type of comparison, we chose a set of 16 addresses that must be visited using two vehicles. Our algorithm produces a grouping of these addresses by regions, dividing the city into two areas: west and east, as seen in the Fig. 7.

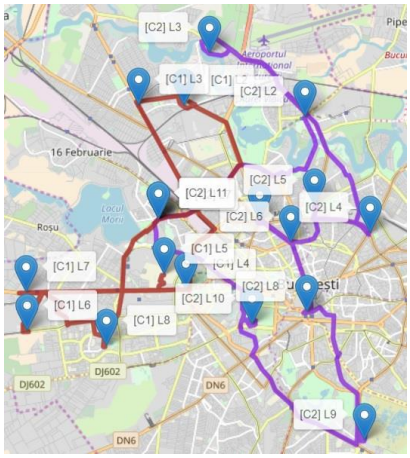

Fig. 7. Routes generated by proposed algorithm for two vehicles

In order to make a comparison, we manually entered the route data of each cluster in the Google Maps web interface to access the result from the perspective

of the Google Maps routing engine. The following figure show the two clusters introduced in Google Maps.

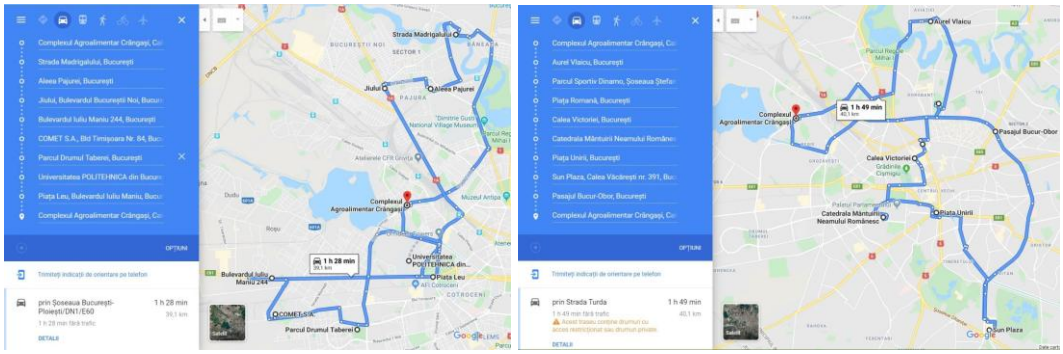

Fig. 8. Route generated by Google Maps for cluster 1 (left) and cluster 2 (right)

In Table 2 we have highlighted the results of these tests, presenting the distance to travel and the estimated travel time for each cluster and for the whole instance. It is noted that the Google Maps routing engine behaves similarly, but the results regarding the travel time differ due to the lack of real-time analysis of traffic conditions.

Table 2

Comparison for distance to travel and estimated travel time with Google Maps for two vehicles

System	Results for first cluster		Results for second cluster		Results for the whole instance	
	Distance to travel (km)	Estimated travel time (minutes)	Distance to travel (km)	Estimated travel time (minutes)	Distance to travel (km)	Estimated travel time (minutes)
Proposed system	32.357 km	54’	44.357 km	72’	76.714 km	126’
Google Maps	39.1 km	88’	40.1 km	109’	79.2 km	197’

6. Conclusions

Following the development and analysis carried out in this study, we managed to define the architecture and combine the infrastructure elements to obtain a visual environment to represent the results produced by various route planning algorithms. We also designed a new solution to the VRP problem that divides destinations for multiple vehicles into clusters with the K-means algorithm and computes for each of these clusters an optimal route with a genetic algorithm.

A major contribution of our algorithm is the possibility of splitting the addresses to be visited in multiple clusters that can be each associated to a vehicle.

This is useful for creating a route planning that uses a large data set which can be evenly distributed to the number of fleet vehicles a company uses.

Starting from the visual results we were able to draw several conclusions.

Our algorithm has achieved better results in matters of travelling distance compared to other de facto solutions such as HERE Maps and Google Maps. Even though the proposed algorithm has a better estimated travel time, we can assume this is because of the lack of real-time traffic data in the areas that the algorithm operates in, but this is subject to change as we are currently adding live traffic data to our system.

Due to the nature of the genetic algorithm that our system uses the identification of areas of interest for each vehicle is heavily influenced by the initialization stage because it relies on randomness to choose the clusters center points. Upon further investigation we decided to use, for the next system version, an initialization phase that derives from the K-Means++ [32] algorithm. In this way clusters center points will be uniformly chosen, in consequence reducing the variations that occur between different executions on the same instance of the problem.

We assumed that using the scheme of genetic algorithms for the routing algorithm will provide consistently good results, but we observed shifting results in the outputs of multiple executions on the same dataset. We believe that the randomness present in the clustering algorithm propagates through the route planner algorithm, yet this is a matter for further investigations.

REFERENCES

[1] *Macioszek, E.* (2017, September). First and last mile delivery–problems and issues. In Scientific And Technical Conference Transport Systems Theory And Practice (pp. 147-154). Springer, Cham.

[2] *G. Laporte*, „The Traveling Salesman Problem: An overview of exact and approximate algorithms”, European Journal of Operational Research, Vol. 59 (2), pp. 231 -247, 1992.

[3] *Valenzuela, C. L., & Jones, A. J.* (1997). Estimating the Held-Karp lower bound for the geometric TSP. European journal of operational research, 102(1), 157-175.

[4] *Johnson, D. S., McGeoch, L. A., & Rothberg, E. E.* (1996, January). Asymptotic experimental analysis for the Held-Karp traveling salesman bound. In Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (Vol. 341, p. 350). San Francisco: ACM Press.

[5] *Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U.* (1999, January). When is “nearest neighbor” meaningful?. In International conference on database theory (pp. 217-235). Springer, Berlin, Heidelberg.

[6] *N. Christofides*, „Technical Note - Bounds for the Travelling – Salesman Problem”, Operations Research, Vol. 20 (5), pp. 1044 – 1056, 1972.

[7] *Jeřábek, K., Majercak, P., Kliestik, T., & Valaskova, K.* (2016). Application of Clark and Wright´ s Savings Algorithm Model to Solve Routing Problem in Supply Logistics. NAŠE MORE: znanstveno-stručni časopis za more i pomorstvo, 63(3 Special Issue), 115-119.

[8] *Hoogeveen, J. A.* (1991). Analysis of Christofides' heuristic: Some paths are more difficult than cycles. Operations Research Letters, 10(5), 291-295.
[9] *G. Laporte, H. Mercure, Y. Nobert*, „Generalized travelling salesman problem through „Sets of nodes: The asymmetrical case", Discrete Applied Mathematics, Vol. 18 (2), pp. 185 – 197, 1987.
[10] *M. Gendreau, G. Laporte, C. Musaraganyi, E. D. Taillard*, „A tabu search heuristic for the heterogeneous fleet vehicle problem", Computers & Operations Research, Vol. 26 (12), pp. 1153 – 1173, 1999.
[11] *A. Larsen*, "The dynamic vehicle routing problem", Kgs. Lyngby, Denmark: Technical University of Denmark. IMM-PHD, No. 2000-73, 2000.
[12] *N. Christofides*, „The vehicle routing problem", R.A.I.R.O. Recherche opérationnelle, Vol. 10 (V1), pp. 55 – 70, 1976.
[13] *Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L.* (2013). A review of dynamic vehicle routing problems. European Journal of Operational Research, 225(1), 1-11.
[14] *Wilck IV, J. H., & Cavalier, T. M.* (2012). A genetic algorithm for the split delivery vehicle routing problem.
[15] *Longo, H., De Aragao, M. P., & Uchoa, E.* (2006). Solving capacitated arc routing problems using a transformation to the CVRP. Computers & Operations Research, 33(6), 1823-1837.
[16] *Khachay, M., & Ogorodnikov, Y.* (2018, July). Efficient PTAS for the Euclidean CVRP with time windows. In International Conference on Analysis of Images, Social Networks and Texts (pp. 318-328). Springer, Cham.
[17] *Gao, Z., Sun, G., & Yuan, Z.* (2019, November). Genetic Algorithm to the Split Delivery Vehicle Routing Problem. In 2019 6th International Conference on Systems and Informatics (ICSAI) (pp. 626-630). IEEE.
[18] *OpenVPN*, available at https://openvpn.net/faq/what-is-openvpn/, accessed on 7.04.2020.
[19] OSRM project. http://project-osrm.org/, accessed on 7.04.2020.
[20] OSM project https://www.openstreetmap.org/about, accessed on 7.04.2020.
[21] *PostgreSQL*, available at https://www.postgresql.org/, accessed on 7.04.2020.
[22] *Postgis*, available at https://postgis.net/, accessed on 7.04.2020.
[23] *Nginx*, available at https://www.nginx.com/, accessed on 7.04.2020.
[24] *PHP*, available at https://www.php.net/, accessed on 7.04.2020.
[25] *Bonfire*, available at https://cibonfire.com/, accessed on 7.04.2020.
[26] *CodeIgniter*, available at https://www.codeigniter.com/ , accessed on 7.04.2020.
[27] *Nominatim*, available at http://nominatim.org/, accessed on 7.04.2020.
[28] *T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu*, "An efficient k-means clustering algorithm: Analysis and implementation", IEEE Transactions on Pattern Analysis & Machine Intelligence, (7), pp. 881-892, 2002.
[29] *F. Altiparmak, M. Gen, L. Lin, T. Paksoy*, "A genetic algorithm approach for multi-objective optimization of supply chain networks". Computers & industrial engineering, Vol. 51(1), pp. 196-215, 2006.
[30] *Berger, J., Salois, M., & Begin, R.* (1998, June). A hybrid genetic algorithm for the vehicle routing problem with time windows. In Conference of the canadian society for computational studies of intelligence (pp. 114-127). Springer, Berlin, Heidelberg.
[31] *M. Hasegawa, T. Ikeguchi, K. Aihara,* "Combination of chaotic neurodynamics with the 2-opt algorithm to solve traveling salesman problems". Physical Review Letters, Vol. 79(12), pp. 2344 - 2347, 1997.
[32] *Arthur, D., & Vassilvitskii, S.* (2006). k-means++: The advantages of careful seeding. Stanford.