Electric Cars: Logistics For the Environment

Juan David Echeverri Villada
Universidad Eafit
Colombia
jdecheverv@eafit.edu.co

Juan Sebastián Guerra Hernández
Universidad Eafit
Colombia
jsguerrah@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

ABSTRACT

Finding a way to combat climate change is one of the most important priorities in the last few years, that is why reducing the use of fossil fuels is a must for all the people. On other hand, the delivering companies have been growing exponentially and its services are more and more demanded with the time, occasioning high emissions of CO2 due to transportations matter. The electronic vehicles are clearly one of the best alternatives to solve this problem, but due to its characteristics, implementing them represents a complex logistic challenge. Looking to the precedents, the traveler salesman and its variant are problems that have a similar approach and the given solutions to them are a fundamental way to tackle the issue.

After modeling the data and deciding which was the most indicated solution, we decided to implement a random local search algorithm, a variation of the nearest client algorithm. With this solution we managed to achieve an incredible execution time, along with a low memory usage (asides of the used data) and highly optimized routes, thus creating an algorithm capable of provide R routes, given the coordinates of K clients, S stations and a depot.

Solving the logistic challenge that implementing electric vehicles represents in the delivery industry might reduce drastically the emissions of CO2 by this sector, and the algorithm created is definitively an appropriate solution for doing it.

ACM CLASSIFICATION Keywords.

•Applied computing → Operations research → Transportation.

•Applied computing → Operations research → Decision analysis → multi-criterion optimization and decision-making.

•Mathematics of computing → Discrete mathematics → Graph theory → Paths and connectivity problems.

•Theory of computation → Design and analysis of algorithms → Graph algorithms analysis → Shortest paths.

•Theory of computation → Design and analysis of algorithms → Streaming, sublinear and near linear time algorithms → Nearest neighbor algorithms.

1. INTRODUCTION

Due to the fact of the logistics needed to coordinate all the solicitudes and complete them with efficiency, delivering packages has always been a complex topic. Every time that a new technology can be implemented in this ambit, the logistics need to be recalculated, improving the efficiency but giving a hard time to the coordinators. That is the case of one of the most recent technologies developed, the electric cars, whose implementation could significantly improve the use of fossil fuels thus benefiting the environment. Nevertheless, implementing this technology has a certain price.

2. PROBLEM

To explain the problem, it is necessary to know what does implementing electric cars implies since they have certain limitations that must be considered by the time of calculating the logistics needed. Some of them are its limited conduction range and its long charging time, that represent a certain problem for the drivers, because they have a work schedule that cannot be exceeded. From here comes the necessity to adapt the planning and optimize the logistic process. developing a tool that evaluates the routes of each vehicle in a fleet and finds the most proper ones will allow the drivers to administrate efficiently the energy and the time, making easier the implementation of electric cars in this ambit and thus reducing the contamination of the environment.

.3. RELATED WORK

3.1 The Traveler Salesman Problem

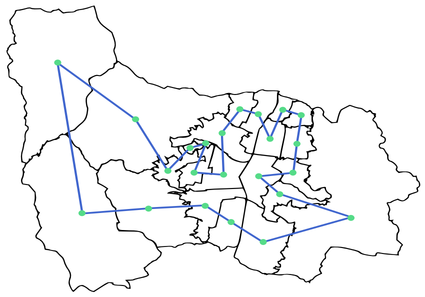

Figure 1: Illustration that represents a generalization of the TSP problem on Medellín.

Proposed by William Hamilton and Thomas Kirkman, consists in searching for the best path that can be carried out in a set of n nodes, passing only once through each node until returning to the first one. Various solutions to this optimization problem have been proposed throughout the years. One of them is the Christofides' algorithm, which consists in choosing and replacing edges to keep less distance. It is a heuristic algorithm since its operation looks for an approximate route with the weights that are raised in the data [1].

3.2 Vehicle Routing Problem (VRP)

The VRP is the distribution of the Traveler Salesman Problem to different vehicles. It "decides which vehicle handles which request in which sequence so that all vehicle routes can be feasibly executed" [2]. In this case, a viable solution is the Clark and Wright savings algorithm, which consists in combining two possible routes, taking the savings that are produced from each one. Thus, in an approximate way, it is possible to attribute the best path for each vehicle.

3.3 Electric Vehicle Routing Problem (E-VRP)

“As corporations are becoming more conscious about the environment and the associated externality costs, electric commercial vehicles are gaining tractions in firms that deliver products" [3]. Under this premise, a variant of the vehicle routing problem was created, the E-VRP, which focuses on finding an optimal routing strategy with minimal travel time cost. energy cost and electric vehicles dispatched. Doing a heuristic approach could be the best way to tackle the EVRP, such as the algorithm of Held Karp or random search inspired algorithms.

3.4 Sequential Ordering Problem (SOP)

It is a variant of the TSP where there are precedence conditions. That means that there are certain nodes that must be visited before another ones, therefore, the most optimal route may not be correct. Carina Vega (2014) used the algorithm SOP 3 Exchange to solve this problem [4]. This algorithm has 2 components. The first one oversees checking the infraction of a precedence while the other ones change axis. When changing axis while preserving roads you get in a heuristic way a path that has all the requisites.

4. Greedy Pathfinding applied to VRP

4.1 Data Structure:

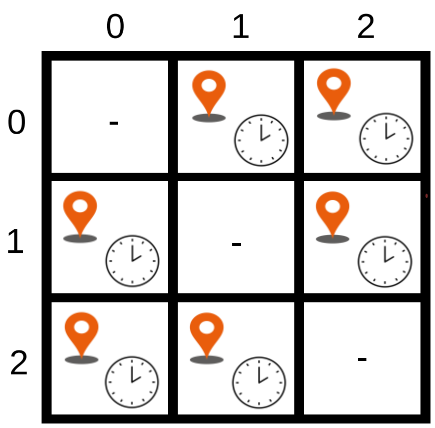

Figure 1: Adjacency Matrix with weights of pairs Distance - Time. Indexes represent the ID of the respective nodes. The diagonal is empty due both IDs refer to the same node.

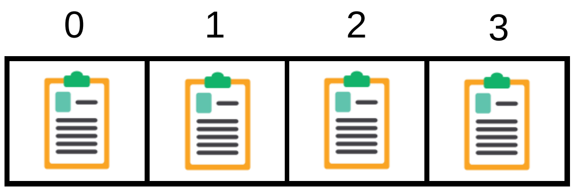

Figure 2: Array that contains the data of each node [Name of the ubication, x coordinate, y coordinate, type of node (either deposit, station, or client) and if is a station, the type of station (Fast, normal, or slow charging)]. Index represents the ID of the node.

4.2 Operations of the data structure

In what follows we explain the operations of the data structure what are going to be used by the algorithm to get or save the required data.

4.2.1 Getting edges:

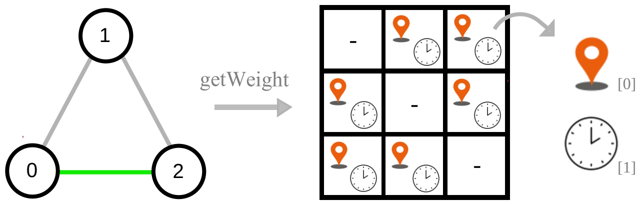

Figure 3: Given two IDs of references, we access to a specific position of the adjacency matrix and extract its data.

4.2.2 Find the successors:

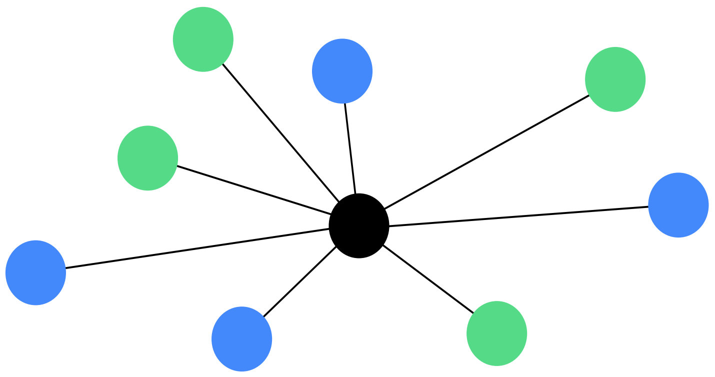

Figure 4: From a start node (black), find all the adjacent nodes (green), discarding those that have been visited (blue).

4.2.3 Find the nearest client:

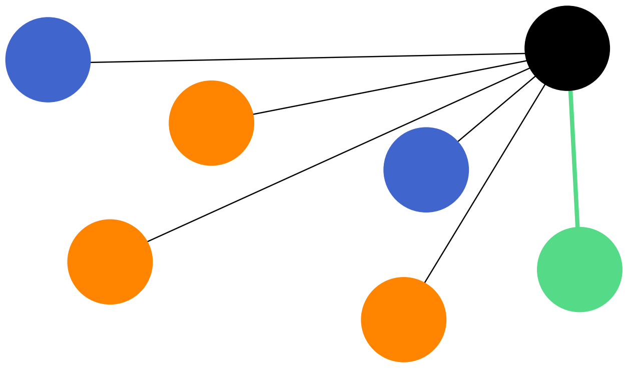

Figure 5: From a start node (black) we search in our adjacency matrix the closest node (green), discarding those that have been visited (blue) and temporarily the other ones (orange).

4.2.4 Show the graph:

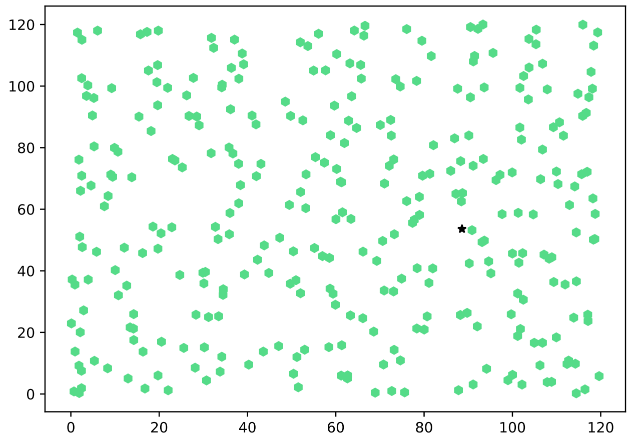

Figure 6: Plot of the graph using *matplotlib.pyplot* library and information saved in the array that has all the information of the nodes. The black star represents the depot, the other ones are clients.

4.3 Design criteria of the data structure

Analyzing the problem, is notorious that every node has a connection with all the other ones, making a map that can be interpreted as a complete graph. Now, knowing the bases of the different types of graphs implementations, it is seen that adjacency list implementation is not optimum at all in complete graph (since it will consume the same memory that adjacency matrix, it only makes worst the access) and for that reason, we choose adjacency matrix. This decision is clearer by the time of searching data in the graph, where adjacency matrix allow us to get it in O(1) unlike adjacency list, who takes O(n) to search.

4.4 Complexity analysis

Firstly, it is necessary the variables that are going to be used to define complexity in each section.

Variable	***Description***
N	Number of points of interest in the map. *Equals to W*R*
S	Number of clients that have not been visited.
W	The maximum number of customers that a vehicle can visit.
R	Number of routes.

Table 1: Dictionary of variables used to express asymptotic complexity.

Operation	***Asymptotic complexity***
Create the graph	$O(N^2)$
Access to an edge	O(1)
Find successors	O(N)
Find the nearest client	O(S)
Show the graph	O(N)

Table 2: Table to report complexity analysis.

4.5 Execution time

Data set	Best case (sec)	Average case (sec)	Worst case (sec)
1	1.03	1.13	2.19
2	1.03	1.12	1.34
3	1.04	1.13	1.27
4	1.04	1.14	1.29
5	1.08	1.15	1.27
6	1.06	1.13	1.29
7	1.14	1.23	1.34
8	1.14	1.24	1.34
9	1.18	1.24	1.37
10	1.13	1.24	1.35
11	1.13	1.21	1.32
12	1.14	1.21	1.3

Table 3: Execution time during the creation of the data structure for each dataset.

4.6 Memory consumption

Data set	Memory consumption (MB)
1	8.5039
2	8.4960
3	6.6992
4	6.9570
5	6.2382
6	6.4414
7	6.4414
8	6.6992
9	5.9257
10	6.4414
11	5.4101
12	5.9257

Table 4: Memory used during the creation of the data structure for each dataset.

4.7 Algorithm: Nearest client

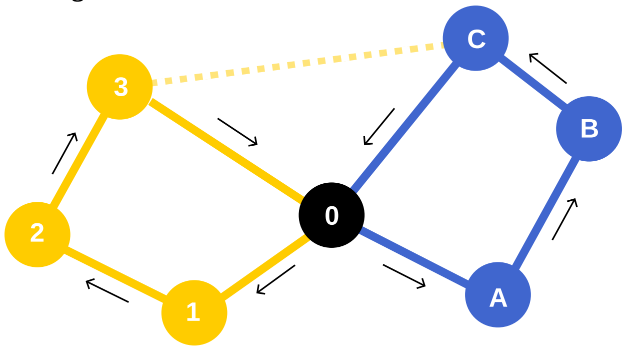

Figure 7:

Step 1: The nearest client is found, and the algorithm travel there.

Step 2: As seen in the figure, step one is repeated until it is not possible to go to another node due to the lack of time (In the figure, the algorithm tries to go from '3' to 'c' but since it is not possible, 'c' is removed from the path and we return to '3', were the algorithm come back to '0' and saves the path).

Step 3: Steps 1 and 2 are repeated until all the clients are visited.

4.8 Complexity analysis of the algorithm

Steps	**Asymptotic complexity**
Find the best client	O(S)
Found a route	O(S*W)
Found all the routes	O(S*W*R)
Final algorithm	**O(S*N)**

Table 5: Complexity of each step in the algorithm. Dictionary of variables available in Table 1.

4.9 Design criteria of the algorithm

After researching of all the different approaches that can be done, we decided to focus in a greedy one, trying constantly to find the nearest client by the time of calculate a route. This has several optimization benefits since it groups the clients by zones, maximizing the time of the routes and ending with a less use of vehicles. Besides its good complexity, which is O(S*W) to find each of the different routes, the algorithm has shown a great yet improvable performance in the used dataset.

4.10 Execution times

Dataset	Best case (sec)	Average case (sec)	Worst case (sec)
1	0.16804	0.1973	0.26572
2	0.14065	0.15778	0.19604
3	0.14917	0.17525	0.23807
4	0.1563	0.18258	0.25008
5	0.14067	0.16158	0.2274
6	0.15626	0.16626	0.18904
7	0.16359	0.18758	0.21882
8	0.15628	0.16878	0.21387
9	0.09352	0.15085	0.24463
10	0.1094	0.1191	0.1563
11	0.09378	0.10596	0.13384
12	0.09377	0.10833	0.12629

Table 6: Execution time of the algorithm for different datasets.

4.11 Memory consumption

Data set	Memory consumption (MB)
1	1.0567754
2	1.2037474
3	1.1925678
4	1.0484738
5	1.2387344
6	1.1849829
7	1.0487832
8	1.2284026
9	1.1799384
10	1.0418389
11	1.2579394
12	1.2038384

Table 7: Memory consumption of the algorithm for different datasets.

4.12 Analysis of the results

Data sets	Executions' time	Memory used	Total vehicles	Routes' time	Total clients
1	1.3273 s	9.5606 MB	34	322 h	320
2	1.2777 s	9.6997 MB	28	265 h	320
3	1.3052 s	7.8917 MB	30	283 h	320
4	1.3225 s	8.0054 MB	34	322 h	320
5	1.3115 s	7.4769 MB	28	265 h	320
6	1.2962 s	7.6263 MB	30	283 h	320
7	1.4175 s	7.4901 MB	34	322 h	320
8	1.4087 s	7.9276 MB	28	265 h	320
9	1.3908 s	7.1056 MB	30	283 h	320
10	1.3591 s	7.4832 MB	34	322 h	320
11	1.3159 s	6.6680 MB	28	265 h	320
12	1.3183 s	7.1295 MB	30	283 h	320

Table 8: Analysis of the results obtained from the creation of the data structure and algorithm execution.

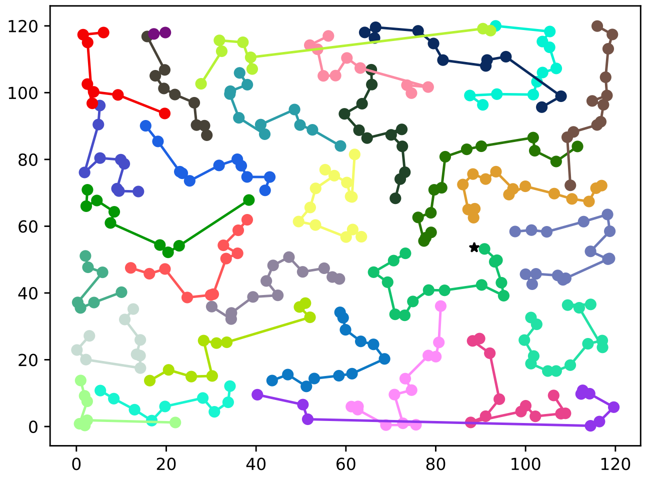

Figure 8: Illustration designed with *matplotlib*. It represents each of the routes calculated for dataset 9. Keep in mind that each color represents a route, and every route starts and ends in the black star, which represents the deposit.

5. Random search applied in e-VRP

5.1 Data Structure

No major changes were done in the data structure, since all the requirements to do this adaptation were fulfilled previously.

(Data structure operations, complexity, execution times and memory consumption keep the same values as mentioned in the section 4)

5.8 Algorithm: Random Local Search

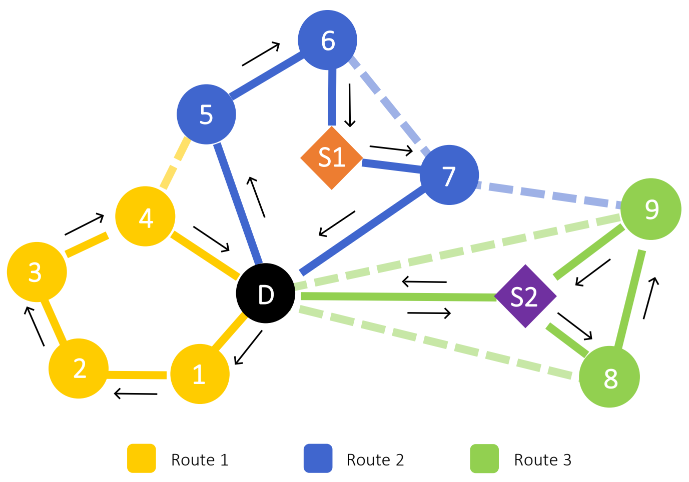

Figure 6: The algorithm is designed to work in several complex situations, the most commons are represented in the figure as an individual route:

Route 1: The algorithm finds all the nearest client, runs out of time after trying to go to 5 and then goes back to depot.

Route 2: The algorithm tries to go from 6 to seven and ran out of energy, it goes to charge, then goes to 7, runs out of time and goes back to depot.

Route 3: The algorithm is not able to start traveling without charging first, so it charges, visit all the clients, charge again, and goes back to the depot.

5.9 Complexity analysis of the algorithm

Steps	**Asymptotic complexity**
Find a better random POI	O(N)
Found a route	O(N*W)
Found all the routes	O(N*W*R)
Recreate all with a different seed	O(N*W*R)
Final algorithm	**O(N^2)**

Table 8. Complexity of each subproblem that is part of the algorithm. Let N be number of clients, W the maximum number of interest points that can be visited in one route, and R the maximum number of routes that can be visited in the same map.

5.10 Design criteria of the algorithm

Unlike the nearest client algorithm, random local search algorithms do not get stuck in local minimums, they go all the way through the cost function finding different minimums thus allowing the creation of more optimized routes without sacrificing much execution time. Based on that, it is certainly one of the best approaches for e-VRP.

5.11 Execution times

Dataset	Best case (sec)	Average case (sec)	Worst case (sec)
1	3.81128	3.87553	3.92024
2	3.09814	3.10244	3.12267
3	2.87452	2.92764	2.993
4	3.68958	3.79476	4.066
5	2.25342	2.3123	2.41406
6	2.67286	2.74125	2.77312
7	3.4899	3.65148	4.17748
8	2.15364	2.23134	2.35968
9	2.59232	2.7165	2.7948
10	3.44148	3.52738	3.69084
11	2.29312	2.37526	2.44696
12	2.50348	2.55964	2.63098

Table 9: Execution time of the algorithm for different datasets.

5.12 Memory consumption

Data set	Memory consumption (MB)
1	79.63
2	82.10
3	76.23
4	78.09
5	73.91
6	75.68
7	80.32
8	72.82
9	74.98
10	77.75
11	74.21
12	75.96

Table 10: Memory consumption of the algorithm for different datasets.

5.13 Analysis of the results

Data sets	Executions' time	Memory used	Total vehicles	Routes' time	Total clients
1	3.8753 s	79.63 MB	42	397 h	320
2	3.1024 s	82.10 MB	31	292 h	320
3	2.9276 s	76.23 MB	38	361 h	320
4	3.7946 s	78.09 MB	45	428 h	320
5	2.3123 s	73.91 MB	32	303 h	320
6	2.7415 s	75.68 MB	37	352 h	320
7	3.6514 s	80.32 MB	47	448 h	320
8	2.2314 s	72.82 MB	30	283 h	320
9	2.7165 s	74.98 MB	35	332 h	320
10	3.5278 s	77.75 MB	49	435 h	320
11	2.3756 s	74.21 MB	31	291 h	320
12	2.5596 s	75.96 MB	34	325 h	320

Table 11. Analysis of the results

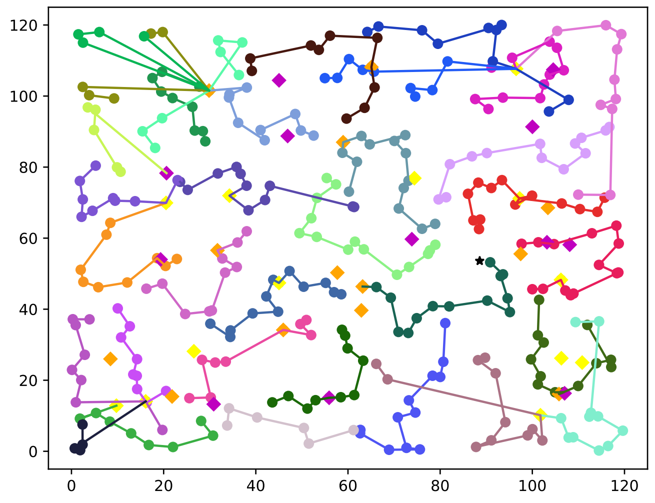

Figure 9: Illustration designed with *matplotlib*. It represents each of the routes calculated for dataset 9. Keep in mind that each color represents a route, and every route starts and ends in the black star, which represents the deposit. Additionally, the purple, orange and yellow diamonds represent respectively the slow, regular, and fast charging stations.

6. CONCLUSIONS

In summary, there are several approaches that can be done to solve VRP and e-VRP, and each one of them can generate very different solutions with different particularities. Both approaches realized in this journey got extremely high performance, and even though the base where they work is quite similar (greedy-based algorithms) they have a huge difference on its results.

The data structure designed for both problems were the same since they are quite similar conceptually, so the main difference between each approach is relayed in the elaboration of the algorithm.

Once the nearest client algorithm is executed, it is not capable of optimize more since it already find a local minimum, but that is not the case with the random local search algorithm, because most of the times it is able to find more optimized solutions through each iteration, and its only limitations is the quantity of times that it is executed (but since it is quite optimized, doing so does not take too much time)

In conclusion, random search algorithm showed to be a great alternative in this problem, and only by sacrificing a few seconds we can create highly optimized routes without much effort.

6.1 Future work

Since the algorithm is already developed, we are planning to develop a more user-friendly interface that will allow everyone to use it and plan the logistics at their own business, thus getting the impact that we were looking for. We are also planning to search for other proposals of solutions that might have different results with personalized datasets so it can allow the people to find the best algorithm for their needs.

ACKNOWLEDGEMENTS
We thank for assistance in the elaboration of the main concept to Miguel Angel Correa Manrique, student at EAFIT university.

We thank for assistance in the debugging of the algorithm to Kevin Alejandro Sossa Chavarria, student at EAFIT university.

We thank for contextualizing the e-VRP and solving a couple of questions that we had over the ride to Alejandro Montoya Echeverri, professor at EAFIT university.

REFERENCES

[1] Bernal, J., Hontoria, E. and Alekvsovski, D., 2015. El problema del viajante de comercio: Búsqueda de soluciones y herramientas asequibles. ASEPUMA, 16(2), pp.117-133.

[2] Irnich, S., Toth, P., and Vigo, D., 2014. Vehicle Routing Problems, Methods, and Applications. Philadelphia: Society for Industrial and Applied Mathematics.

[3] Jane Lin, Wei Zhou, Ouri Wolfson, Electric Vehicle Routing Problem, Transportation Research Procedia, Volume 12, 2016, 508-521, https://doi.org/10.1016/j.trpro.2016.02.007.

[4] Vega, C., 2014. *BRKGA para el Problema de Ordenamiento Secuencial*. [online] Buenos Aires. Available at: <http://dc.sigedep.exactas.uba.ar/media/academic/grade/thesis/vega.pdf> [Accessed 21 February 2021].