Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

# Clustering and heuristics algorithm for the vehicle routing problem with time windows

**Andrés Felipe León Villalba[a]* and Elsa Cristina González La Rotta[a]**

[a]*Catholic University of Colombia, Engineering School, Industrial Engineering Program, Bogotá, Colombia*

C H R O N I C L E

*Article history*:
Received February 21 2021
Received in Revised Format April 22 2021
Accepted December 17 2021
Available online December, 27 2021

Keywords:
*Vehicle routing problem*
*Urban logistics*
*Cluster first-route second*
*Nearest neighbor*
*Local search 2-opt*

A B S T R A C T

This article presents a novel algorithm based on the cluster first-route second method, which executes a solution through K-means and Optics clustering techniques and Nearest Neighbor and Local Search 2-opt heuristics, for the solution of a vehicle routing problem with time windows (VRPTW). The objective of the problem focuses on reducing distances, supported by the variables of demand, delivery points, capacities, time windows and type of fleet in synergy with the model's taxonomy, based on data referring to deliveries made by a logistics operator in Colombia. As a result, good solutions are generated in minimum time periods after fulfilling the agreed constraints, providing high performance in route generation and solutions for large customer instances. Similarly, the algorithm demonstrates efficiency and competitiveness compared to other methods detailed in the literature, after being benchmarked with the Solomon instance data set, exporting even better results.

## 1. Introduction

The urban freight distribution is one of the most complex logistical challenges for the productive sector of any country, given the constant growth of cities and the high complexity in the distribution of supplies arising from the multiple variables of the environment; satisfy the needs of people and obtain an economic reward for such activities, must transform the logistics paradigm to avoid generating high costs in the supply chain (Anaya, 2015; Gutiérrez, Hincapié & León, 2019). This situation arises largely by the complex typology of cities, the high level of congestion and traffic, the high environmental impact of pollutant and greenhouse gas emissions caused by vehicle combustion, noise, accidents, deterioration of infrastructure, high logistics costs (in the United States and Europe logistic costs represent between 10% and 12% of gross domestic product (Bowersox & Calantone, 1998)), insecurity, lack of technological development, in addition to various regulatory restrictions by the government, among others factors; represents the externalities of urban freight distribution (Moen, 2016; Fernández, 2008; Best Urban Freight Solutions, 2006). Given these particularities, an alternative that helps to mitigate some externalities is to solve the vehicle routing problem (VRP). The VRP is originated from Traveling Salesman Problem (TSP) which is attributed to Hamilton and Kirkman from the 1800s, with the premise that a seller visits a set of customers sequentially and returns to the point of origin in the most economical way possible (Applegate, Bixby, Chvátal, & Cook, 2011; Rahman, 2012). Based on the TSP models, the VRP models arise, proposed by Dantzig & Ramser (1959), involving not a single seller, but multiple sellers, in addition to a depot. In this way, the VRP is defined as "the problem of designing least-cost delivery routes from a depot to a set of geographically scattered customers, subject to side constraint" (Laporte, 2009, p.408). The VRP is classified based on the taxonomic characteristics of the problem to be treated and the constraints involved (Braekers, Ramaekers & Nieuwenhuyse, 2016); this article only contemplates capacity and time windows constraints, better known as VRPTW or CVRPTW. One of the representative characteristics of VRP is its high computational complexity, given its ***n*** possibilities of solution to a greater number of customers (Lüer, Benavente, Bustos & Venegas, 2009; Yepes, 2002). Generally, these are categorized as polynomial problems, require a means where can be solved and an algorithm or solution method (Guerequeta & Vallecillo, 2000; Ladner, 1975); under this premise, the VRP is cataloged as NP-hard (Cordeau,

* Corresponding author Tel: +57 3132039620
E-mail: afleon47@ucatolica.edu.co (A. F. L. Villalba)

2022 Growing Science Ltd.
doi: 10.5267/j.ijiec.2021.12.002

Desaulniers, Desrosiers, Solomon & Soumis, 2000) and as solution algorithm are included techniques of exact methods, heuristic, metaheuristic, or hybrid (the combination of several methods) (Lüer et al., 2009; Toro, Escobar & Granada, 2016).

The literature details various algorithms and solution techniques for a VRP, where the previously mentioned methods are applied (exact, heuristic or metaheuristic). This article presents an algorithm under the cluster first-route second method, which links clustering techniques and solution methods (Bodin, 1975). This method starts with clustering, the unsupervised classification of patterns, by which objects are grouped based on their characteristics, resulting in the grouping of similar items (Hair, Black, Babin & Anderson, 2014; Galba, Balkić & Martinovi, 2013). There are several clustering methods as a function of their algorithm such as: the connectivity, the centroid, the distribution, the density, the subspace, the group, and the graphics-based models (Galba et al., 2013). The algorithm for routing phase could be exact, provides a feasible solution to the problem; a heuristic, which obtains quality solutions without ensuring optimality, in shorter execution times (Lüer et al., 2009); or a metaheuristic, which tends to explore more complex algorithms that sometimes emulate the behavior of nature or population, generating computational efficiency and optimality (Gendreau, Potvin, Bräumlaysy, Hasle & Løkketangen, 2008). Throughout the years, there have been several studies that apply the cluster first-route second methodology in VRP problems, including TSP; which integrate several types of clustering, followed by an exact, heuristic or metaheuristic method, as for example, the investigations of: Hiquebran, Alfa, Shapiro & Gittoes (1993); Kim, Kim & Sahoo (2006); Dondo & Cerdá (2007); Ghoseiri & Ghannadpour (2010); Shin & Han (2011); Qi, Lin, Li & Miao (2012); Kao & Chen (2013); Ayu, Septivani, Xu, Kwan & Ani (2015); López & de Jesús (2015); Cömert, Yazgan, Sertvuran & Şengül (2017); Abbatecola, Fanti, Pedroncelli & Ukovich (2018) and Fachini & Armentano (2020). From these documents, it is interesting to observe how the solutions are generated and the emphasis they put on ease of application, since the complexity of the problem is greatly reduced when any clustering method is performed and is integrated with a method solution of those previously mentioned (exact, heuristic or metaheuristic). Robust algorithms are designed that export good and efficient solutions which meet the objective function and the stipulated constraints. It should be clarified that these algorithms link mathematical formulation and are not limited to the application of only one method in either phase, whether in the clustering or routing phase.

As a literature review, it is important to highlight the works taken as a reference for the algorithm design, starting with the study by Cömert et al. (2017), in which they develop a cluster first-route second solution method for VRPTW, where they assign the constraints with the unsupervised classification algorithms K-means, K-medoids and Dbscan, followed by a routing phase that constructs the routes for a commercial chain composed of a depot, 78 customers and a homogeneous fleet, using an exact solution method of mixed integer linear programming (MILP). The present research applies a similar development to Cömert et al. (2017), in terms of validation and constraints fulfillment through two clustering techniques; but in turn, it presents differences in the techniques, such as in the routing method used for the VRPTW solution and in a larger association of customers. This application demonstrates that the clustering phase is one of the most important in the cases of constraints fulfillment, allowing to obtain good results in conjunction with the routing method.

Secondly, there is the algorithm designed by Solano, Montoya & Guerrero (2019), where a large-scale single depot (SD)VRPTW is solved as a Decision Support System for a Colombian public utility company. In their research, during the clustering phase Sweep Algorithm is used to assign a set of customers to each technician, subsequently, the assignment is improved with K-means; for the routing phase they apply Nearest Neighbor as initial route sequence generator and Or-opt heuristic as route optimizer. The authors demonstrate operational efficiency and compliance against the current routing carried out by the company, although there is no evidence of time window association for each customer and their solution involves a high computational cost.

Last but not least, the research of Groër, Golden & Wasil (2010) and Bräysy & Gendreau (2005) are taken as a reference, for employing a heuristic solution method for the design of the routes of a VRP. Groër et al. (2010) and Bräysy & Gendreau (2005) apply the Local Search heuristic for route generation (Local Search is detailed in section 4.2 of this article). The practicality of application, its simple pseudocode, the results and/or near-optimal solutions that it exports, make it an interesting algorithm to explore as long as it is provided with good input data or a good initial solution, such as those generated by Clark and Wright, Sweep algorithm, Nearest Neighbor, among others. Based on the algorithms of Solano et al. (2019), Groër et al. (2010) and Bräysy & Gendreau (2005), the routing phase of this research was designed focusing on the specific parts of the methods that allowed highlighting the results obtained in these works. On the other hand, it is important to mention other heuristic and metaheuristic techniques for the generation of routes or solution of the problem, which in turn are not linked to the cluster first-route second or route first-cluster second method, such as the well-known works of: Gendreau, Hertz & Laporte (1994) and Barbarosoglu & Ozgur (1999), for applying the Tabu Search (TS) heuristic, which iteratively explores a set of possible solutions to the problem posed, performing moves of a solution over the neighborhood or group of nodes generated by another solution, until reaching the fulfillment of the objective function with a solution that qualifies as optimal or near-optimal (Glover, Taillard & de Werra, 1993). In these studies, after evaluating the behavior of TS in different customer scenarios, reaching a good solution in several hardware configurations, compromises in some cases considerable execution times for solution selection in the neighborhood based on constraints compliance. Other well-known techniques from the wide range of metaheuristics are the Large Neighborhood Search (LNS), Simulated Annealing (SA) and Genetic Algorithm (GA) solution methods, by take as example, which in studies such as Gendreau et al. (2008) and Tan, Lee, Zhu & Ou (2001) are highlighted for the quality of their results in a VRPTW. LNS proposed by Shaw (1998) is a method that "destructs part of the current solution while a repair method rebuilds the destroyed solution" (Pisinger & Ropke, 2010, p.406). LNS, together with its extensions such as *Adaptive/Variable* LNS, in applied VRPTW scenarios such as those of Bent & Van Hentenryck (2004) and Bräysy (2003); return competitive results against the best-known solutions according to Solomon (1986) benchmark, while optimizing, linking, and solving problem variables independently. Particularly SA, is characterized by resulting in a global optimum (Gendreau et al., 2008) given that its solution is based on a random local search, which escapes from the bad optimality by accepting the solution that provides the most value with the lowest cost (Elshaer & Awad, 2020; Gendreau et al., 2008). Regarding GA, the best solution is obtained based on the process of natural selection, by creating a solution chain (*bit* or *integer chain*) on an initial or previous solution to the current one, through *selection*, *crossover,* and *mutation* (Gendreau et al., 2008). From the literature reviewed regarding metaheuristic algorithms, especially the methods that were previously detailed (LNS, SA and GA) and took place as a case study or application, such as those of: Berger, Salois & Begin (1998); Gupta, Singh & Pandey (2010); Moon, Lee & Seong (2012); Pemberthy (2012); Buhrkal, Larsen & Ropke (2012); Taner, Galić & Carić (2012); Wy, Kim & Kim (2013); Karagül & Gungor (2014); Khodabandeh *et al.* (2016); Chen & Yang (2017); Stehling & Souza (2017); Kang & Lee (2018); Son, Kim & Shin (2018); Prag, Woolway & Jacobs (2019) and Tirkolaee, Abbasian, Soltani & Ghaffarian (2019). It is remarkable to note the gradual level of difficulty as a function of its execution in complex real-life scenarios, and more so those that include stochastic components; despite generating optimal competitive solutions in most cases and/or better solutions compared to an exact or heuristic method; given its high computational algorithmic and application complexity, as well as hardware requirements, there are few algorithms that have solved the complexity of a VRP satisfactorily in its entirety, neither is it possible to validate that there is an infallible method that adapts and optimally solves all the particularities of a VRPTW, given that most algorithms are designed according to a specific need; likewise, it is important to validate the instances and scenarios given the sensitivity of the algorithms in their parameters, which could infer in the results.

Similarly, the present article, by applying clustering techniques, intrinsically relates Machine Learning (ML). ML "consists of designing efficient and accurate prediction algorithms" (Mohri, Rostamizadeh & Talwalkar, 2018, p.1); for this reason, it is important to bring up a brief review on literature comprised of other ML techniques apart from clustering, on VRP models with ML or its extensions such as Reinforcement Learning (RL) that "involve learning what to do-how to map situations to actions-so as to maximize a numerical reward signal" (Sutton & Barto, 2018, p.2), Neural Network or Deep RL, among others.

Based on the above, Tamagawa, Taniguchi & Yamada (2010) design an algorithm for VRPTW-F (Forecasted) with Q-learning, an RL technique that focuses on states, actions and rewards, where they evaluate the impact of applying several logistic measures in a small network in the study city. On the other hand, Nazari, Oroojlooy, Takáč & Snyder (2018), detail better solutions and performance compared to other heuristics, by training a model using RL with a Neural Network for the solution of a VRP in small and medium scenarios, focused on the resulting calculation of the rewards and route feasibility, with the particularity of not incorporating a distance matrix. Bouhamed, Ghazzai, Besbes & Massoud (2019) apply Q-Learning in a randomized theoretical environment with Euclidean distance calculation, which considers the highest number of rewards for the agent according to the fulfillment of the time windows and the activity plan, without contemplating the amount of load to be transported/delivered. On the Kalakanti, Verma, Paul & Yoshida (2019) side, design and test an RL solver based on Q-learning, which generates competitive results compared to the small and medium instances of Solomon (1986). In the case of Yu, Yu, & Gu (2019) through Deep RL, they propose a model based on a Neural Network for an Online Green VRP with heterogeneous electric vehicles, which provides outstanding results in a short time with the support of a hardware robust, on several random scenarios supplied with real data from the study city. Zhao, Mao, Zhao & Zou (2020), program an adaptive Deep RL hybrid model in charge of generating the initial route sequence, which feeds the Local Search heuristic for the final solution of the model. Finally, other notable models of similar techniques that link ML can be seen in: Poullet (2018); Lin, Ghaddar & Nathwani (2020) and Furian, O'Sullivan, Walker & Çela (2021). From the reviewed works, the successful application of several ML algorithms in controlled environments of a VRP with a small-medium size of nodes is evidenced; however, some models despite being structured and of great adaptability, do not perform the linkage of many variables of the environment and in some cases compromise a modest computational load. The potential of this type of techniques, as well as its ability to solve large dynamic scenarios of greater complexity, has not been fully demonstrated, in addition to the fact that there are few works that have been dedicated to solving a VRPTW.

In research related to this branch of study (ML with VRP), the authors Bai *et al.* (2021) detail that the nature of the data and the taxonomic characteristics of the problem in some situations are not specified; it is difficult to access the information used; the mathematical programming and the constraints of the VRP problem according to its classification, are not fully explored or integrated with analytical methods; likewise, they highlight that more applications that contemplate the challenges of the real world are required. Similarly, few studies have been evidenced that contemplate the constraints of a VRPTW and generate near-optimal routes based on the cluster first-route second method, with high computational efficiency, without compromising high hardware requirements or memory usage for a real large-scale scenario. Therefore, this article contributes to the development of the technique with a novel solution to a mathematical model VRPTW guided by mixed integer programming for the variables, constraints, and objective function, where a database is used that gathers the deliveries made by a logistic operator in Colombia. The algorithm programmed in Python 3.7 for its solution links the K-means and Optics clustering techniques and the Nearest Neighbor and Local Search 2-opt heuristics, which obtain a set of solutions/routes, for each evaluated instance, with good results that determine high performance and efficiency, without compromising a high computational load. To confirm its performance, the algorithm is tested by means of a benchmark with the data of Solomon (1986), against various algorithms proposed in the literature.

To facilitate reading of this article, section **2** exposes the data used for the model, section **3** describes the mathematical formulation for the VRPTW, and sections **4** and **5** presents the proposed algorithm and its computational results. By last, section **6** presents the conclusion.

## 2. Data

The data used in the design of the algorithm were provided by a logistics operator in Colombia; the base includes the deliveries of products made during 65 weeks from 2016 and 2017 to more than 7,997 customers distributed in 196 cities. To determine the number of initial customers of the model, a statistical analysis was carried out, together with the extraction, transformation, and parametrization of data for the amount of cargo delivered, the number of orders, the time horizon, and the concentration of customers by city, as main variables. Of the 196 cities, Bogotá is established as the basis for the design of the algorithm, by integrating the largest number of customers, 35.6% of the total, that is, 2,766 customers and 23.2% to be highlighted, out of the total amount of 311,367 orders placed throughout the territory, with a daily maximum of 737 and an average of 211 orders delivered (see **Fig. 1**). This represents 32,309,395 units of product and 1,171.1 tons of cargo transported. **Fig. 2** illustrates the delivered load distribution on the heat map.

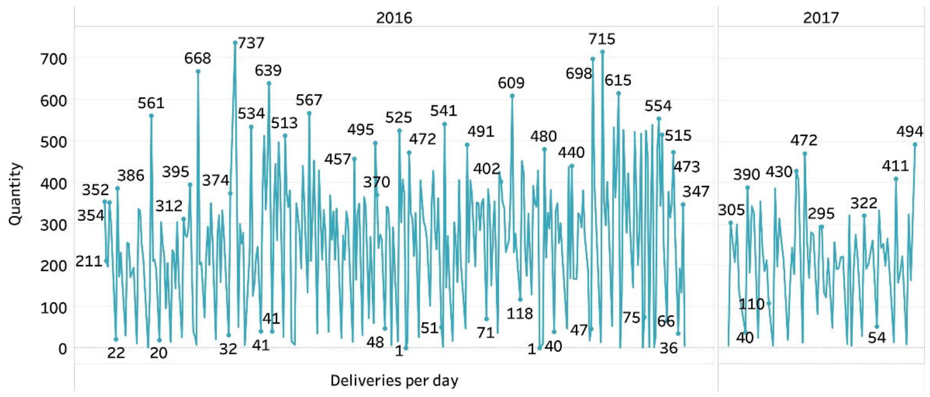

**Fig. 1.** Number of deliveries per day in Bogotá for 65 weeks

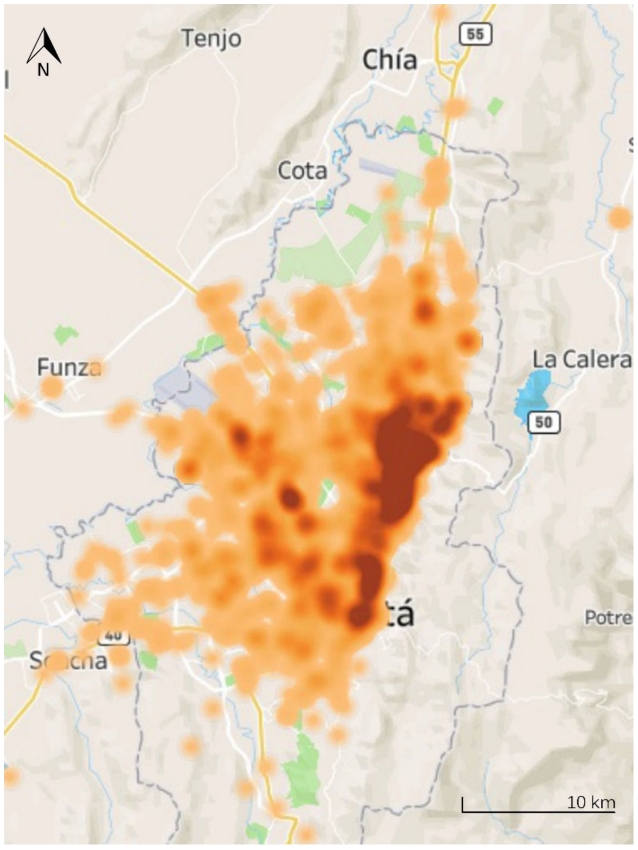

**Fig. 2.** Heat map of the amount of cargo delivered in Bogotá.

Additionally, to locate the initial nodes as input data for the model, a density based spatial clustering of applications with noise (**Dbscan**) was carried out to reduce the total number of customers and identify the most representative ones, thus reducing the exponential complexity of possible solutions according to the number of customers involved in a VRP type problem. The Dbscan algorithm is developed by Ester, Kriegel, Sander & Xu (1996), with the premise of generating clusters with the estimation of the density distribution of the data set, while identifying outliers not representative of any group. The Dbscan parameters configured in support of the Sklearn library (Pedregosa *et al.*, 2011) for Python were the ***number of samples*** or the number of points housed in a neighborhood, the ***neighborhood***, the area that conglomerates the samples, where three types of points are distinguished: the ***nucleus*** (central point), the ***edge*** and ***noise*** points. Neighborhoods are determined by the ***epsilon***, the maximum distance between two samples and the ***metric***, the distance function used (Ester et al., 1996; Tran, Drab & Daszykowski, 2013). For this clustering, the ***haversine*** function is used as a metric (Sinnott, 1984) to calculate the distance between the locations of a sphere based on latitude and longitude: $haversine(d/R) = haversine(\varphi_1 - \varphi_2) + \cos(\varphi_1)\cos(\varphi_2)\,haversine(\Delta\lambda)$ where: $\boldsymbol{d}$, is the distance between two points located on a circle; $\boldsymbol{R}$, is the radius of the sphere, in this case the earth 6,731.01 km (Lide, 2003); $\boldsymbol{haversine}$, the formula $sin^2(\theta/2)$; $\boldsymbol{\varphi_1}$, is the latitude of point **1**; $\boldsymbol{\varphi_2}$, is the latitude of point **2**; and $\boldsymbol{\Delta\lambda}$**,** is the difference in longitudes. Dbscan generated 52 clusters that bring together the 205 spatially most representative customers, after compressing by 92.6% the total of 2,766 customers (see **Fig. 3**); contrasting the heat map, are the ones customers to whom the greatest quantity of merchandise is delivered. This base of **205** customers is the input nodes for the algorithm.

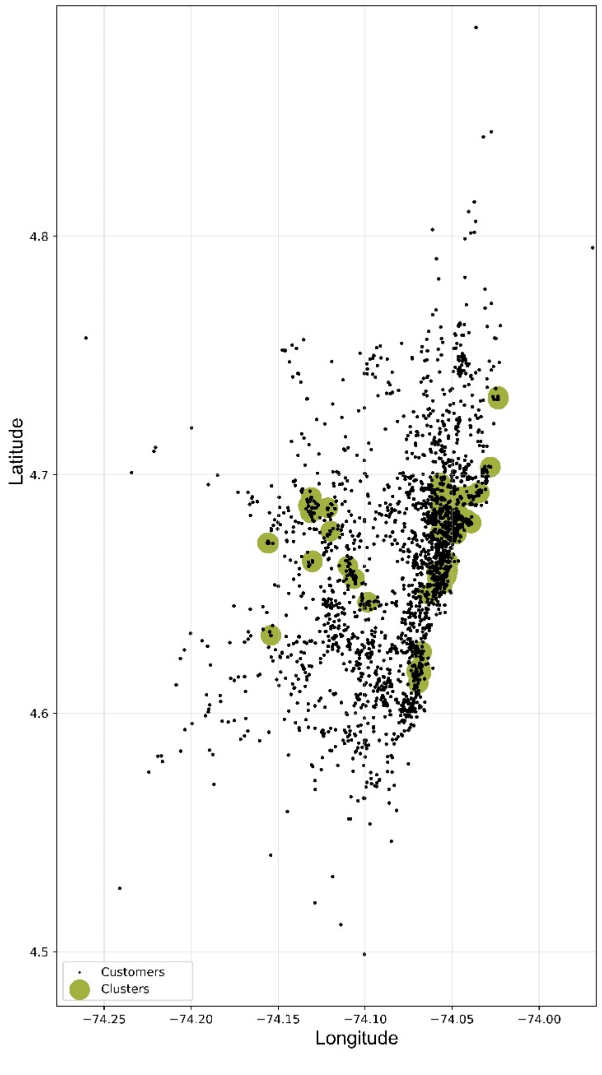

**Fig. 3.** Representative customers resulting from the Dbscan clustering

## 3. Model based on a vehicle routing problem with time windows

The model focuses its mathematical formulation on a VRPTW problem, in support of the taxonomy identified in Braekers et al. (2016) in order to identify the type of study, the scenario, physical characteristics, the information and data. To begin with, the type of study focuses the application of a method to a problem with the use of a mixed solution of two clustering methods and two heuristics. As characteristics of the scenario, a deterministic number of stops is established on the route, being the location of the nodes; load partitioning within vehicles is allowed and the quantity demanded is that required for the 205 customers resulting from the clustering for one business day without backhauls. In accordance with the waiting/service times on the site, deterministic values are established; likewise, the time windows are structured according to the regulatory restrictions regarding the movement of cargo vehicles within Bogotá for the year 2020, governed by decrees 840 of 2019 and 047 of 2020 issued by the Bogotá Mayor's Office (2019; 2020). At the physical level, the problem network is undirected, with the location of the customers in nodes (no deliveries are made during the arch route); only the delivery of merchandise is allowed; there is a single origin of the vehicles (the depot); presenting time window constraints on nodes and depot; with a limited number of homogeneous capacitated vehicles, to satisfy a distance-dependent objective. Finally, the information is static, centrally processed, of known quality and real provenance.

## *3.1. Mathematical formulation*

The problem is formulated based on existing mathematical models for VRPTW, taking as a guide Cordeau, Laporte, Savelsbergh & Vigo (2007). An undirected network is contemplated $\boldsymbol{G} = (\boldsymbol{I}, \boldsymbol{A})$, where $\boldsymbol{I}$ is the set of nodes represented by the letters $\boldsymbol{i}$ and $\boldsymbol{j}$ ($\boldsymbol{i} = \boldsymbol{j}$), and $\boldsymbol{A}$, is the set of arcs that join the nodes ($\boldsymbol{i}\ \boldsymbol{or}\ \boldsymbol{j}$). So that, $\boldsymbol{I} =\ i_n \cup \{i_0,\ dp\}$ where: $\boldsymbol{i_n}$, are the nodes to deliver the merchandise, ($\boldsymbol{i} = 1, \dots, I$); $\boldsymbol{i_0}$, the depot of origin and $\boldsymbol{dp}$, arrival depot; ($\boldsymbol{i_0} = \boldsymbol{dp}$, refers to the same depot). Likewise, $\boldsymbol{A} = \{(i,j) = (j,i): i,j\ \in I\}$ represents the set of arcs of the network and $\boldsymbol{K}$ the set of vehicles ($\boldsymbol{k} = 1, \dots, K$). Similarly, under these premises, the VRPTW can be formulated following a mixed integer programming model:

**Notation**

$E_i$: quantity of goods to be delivered in the node $\boldsymbol{i}$; $\{i = 1, \dots, I\}$ $[kg]$.

$E_i \geq 0$: the quantity of goods to be delivered in the node $\boldsymbol{i}$, it is not negative; $\{i = 1, \dots, I\}$.

$Q_k$: load capacity $\boldsymbol{Q}$ of vehicle $\boldsymbol{k}$; $(k \in K)$ $[kg]$.

$[a_i, b_i]$: time window; visit node $\boldsymbol{i}$ can only occur between time intervals of start $\boldsymbol{a_i}$ and end $\boldsymbol{b_i}$; $(i \in I)$ $[min]$.

$a_i$: start of time window at node $\boldsymbol{i}$; $(i \in I)$ $[min]$.

$b_i$: end of time window at node $\boldsymbol{i}$; $(i \in I)$ $[min]$.

$S_i$: service time at node $\boldsymbol{i}$ for unloading goods; $(i \in I)$ $[min]$.

$D_{ij}$: distance between the nodes of the arc $(\boldsymbol{i}, \boldsymbol{j})$; $\{(i,j) \in A\}$ $[km]$.

$D_{ij} \geq 0$: the distance between the nodes of the arc $(\boldsymbol{i}, \boldsymbol{j})$, it is not negative; $\{(i,j) \in A\}$.

$Tv_{ij}$: time to travel along the route of the arc $(\boldsymbol{i}, \boldsymbol{j})$ from node $\boldsymbol{i}$ to node $\boldsymbol{j}$; $\{(i,j) \in A\}$ $[min]$.

$Tv_{ij} \geq 0$: the time to travel along the route of the arc $(\boldsymbol{i}, \boldsymbol{j})$ from node $\boldsymbol{i}$ to node $\boldsymbol{j}$, it is not negative; $\{(i,j) \in A\}$.

$T_i$: time in which the vehicle $\boldsymbol{k}$ start service on node $\boldsymbol{i}$; $(i \in I)$ $[min]$.

**Decision variable**

$X_{ijk}$: binary variable; takes value $\mathbf{1}$ if vehicle $\boldsymbol{k}$ travels from node $\boldsymbol{i}$ to node $\boldsymbol{j}$ through the arc $(\boldsymbol{i}, \boldsymbol{j})$; $\{(i,j) \in A\}$ & $(k \in K)$ and $\mathbf{0}$ otherwise.

**Objective function**

$$\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} D_{ij} * X_{ijk} \tag{3.1}$$

**Subject to**

$$\sum_{j=1}^{J} \sum_{k=1}^{K} X_{ijk} = 1 \quad (i \in I) \tag{3.2}$$

$$\sum_{i=1}^{I} \sum_{k=1}^{K} X_{ijk} = 1 \quad (j \in I) \tag{3.3}$$

$$\sum_{j=1}^{J} X_{i_0 jk} = 1 \quad (k \in K) \tag{3.4}$$

$$\sum_{i=1}^{I} X_{i,dp,k} = 1 \quad (k \in K) \tag{3.5}$$

$$\sum_{i=1}^{I} X_{ijk} - \sum_{j=1}^{J} X_{jik} = 0 \quad (i \in I \,\&\, k \in K) \tag{3.6}$$

$$\sum_{i=1}^{I} E_i * \sum_{j=1}^{J} X_{ijk} \leq Q_k \quad (k \in K) \tag{3.7}$$

$$T_i \geq a_i \quad (i \in I) \tag{3.8}$$

$$T_i \leq b_i \quad (i \in I) \tag{3.9}$$

$$(T_i + S_i + Tv_{ij} - T_j) \leq \left( W * \left( 1 - \sum_{k=1}^{K} X_{ijk} \right) \right) \quad (i \in I) \tag{3.10}$$

$$X_{ijk} \in \{0,1\} \quad (i \in I \,\&\, k \in K) \tag{3.11}$$

$$E_i \geq 0 \tag{3.12}$$

$$D_{ij} \geq 0 \tag{3.13}$$

$$Tv_{ij} \geq 0 \tag{3.14}$$

In the model, the objective function (**3.1**) minimizes the distances traveled in the distribution of goods. Constraints (**3.2**) and (**3.3**) indicates that each vehicle $\boldsymbol{k}$ that reaches the node $\boldsymbol{i}$ has a node $\boldsymbol{i}$ where it should arrive, except the arrival depot $\boldsymbol{dp}$; and each vehicle $\boldsymbol{k}$ that leaves the node $\boldsymbol{i}$ has a node $\boldsymbol{i}$ from which it arrives, except the departure depot $\boldsymbol{i_0}$. In constraints (**3.4**) and (**3.5**) each vehicle $\boldsymbol{k}$ must start the route at departure depot $\boldsymbol{i_0}$ and finish the route at the arrival depot $\boldsymbol{dp}$. Constraint (**3.6**) preserves network flow, that is, each vehicle $\boldsymbol{k}$ that reaches the node $\boldsymbol{i}$ must be the same as that leaves the node $\boldsymbol{i}$. In constraint (**3.7**) the quantity $\boldsymbol{Q}$ of merchandise to be delivered to node $\boldsymbol{i}$ must not exceed the capacity of the vehicle $\boldsymbol{k}$. Constraints (**3.8**) and (**3.9**) indicate time windows; that is, the vehicle $\boldsymbol{k}$ must arrive at the same time or after the time window $\boldsymbol{a_i}$ starts in the node $\boldsymbol{i}$, and leaves at the same time or before the time window $\boldsymbol{b_i}$ ends in the node $\boldsymbol{i}$.

Constraint (**3.10**) is adapted from the model of Miller, Tucker & Zemlin (1960) to remove the subtours, where the sum of the operation times (the time of arrival, service, travel, and departure) of the node $\boldsymbol{i}$, must be less than a very large number $\boldsymbol{W}$ which takes value if the path of the arch $(\boldsymbol{i,j})$ has not yet been used to reach the node $\boldsymbol{i}$. In other words, if the node $\boldsymbol{i}$ has already been visited by the route of the arch $(\boldsymbol{i,j})$, the value of the binary variable is **1**, then the very large number $\boldsymbol{W}$ is **0**, expressing that the node has already been visited $\boldsymbol{i}$; for the opposite case, if the arch route $(\boldsymbol{i,j})$ has not yet been used to reach the node $\boldsymbol{i}$, the value of the binary variable will be **0** and the very large number $\boldsymbol{W}$ will take value, indicating that the node has not yet been visited $\boldsymbol{i}$.

Finally, the constraint (**3.11**) is presented which expresses the binary variable and constraints (**3.12**), (**3.13**) and (**3.14**) the non-negative variables.

## 4. The proposed algorithm

The proposed algorithm is performed in Python version 3.7, in Jupyter Notebook environment using the clustering modules of the library *Scikit Learn* (Pedregosa *et al.*, 2011); the management of vectors and matrices is done with the library *NumPy* (Oliphant, 2006); the analysis and management of databases with *Pandas* (McKinney, 2010); finally *Matplotlib* library (Hunter, 2007) is used for the generation of graphs; together with the *time* and *math* modules predefined by Python (Van Rossum, 1993) for time and mathematical functions, respectively. Model programming follows a solution flow (see **Fig. 4**) which generates routes after carrying out two clustering methods (K-means and Optics) and two heuristic algorithms, Nearest Neighbor to generate initial routes and Local Search 2-opt to optimize them, following the cluster first-route second method as a reference. Similarly, the steps of the algorithm and pseudocode are exposed in **Table 1** and **2,** respectively.

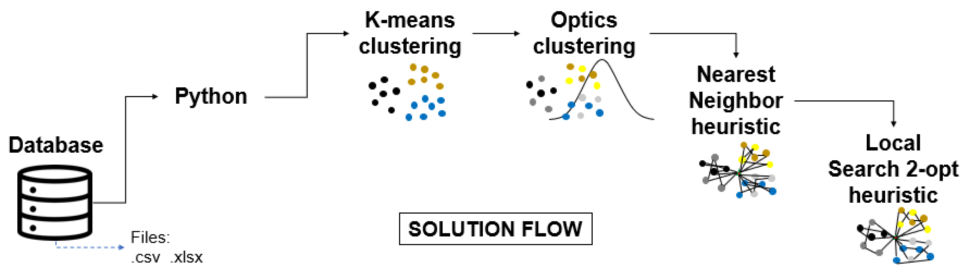

**Fig. 4.** Solution flow

**Table 1**
Proposed algorithm

| Step | Activities |
|---|---|
| 1 | Load the databases. |
| 2 | Identify the number of vehicles required for the model according to its capacity and the total amount of cargo to be transported. |
| 3 | Apply K-means clustering. |
| 4 | Generate Optics clustering for each cluster resulting from K-means after normalizing variables to follow a Gaussian distribution. |
| 5 | If the amount of cargo to be transported per cluster exceeds the capacity of the vehicles and the time windows are not fulfilled, go to step 6. Otherwise, go to step 7. |
| 6 | Modify Optics clustering parameters for constraints compliance. |
| 7 | Generate the distance matrix with the haversine function and get your distance scalar for the heuristics, if applicable. |
| 8 | Generate the initial route with the Nearest Neighbor heuristic for each resulting Optics cluster. |
| 9 | Implement the Local Search 2-opt heuristic for each route generated by Nearest Neighbor. |
| 10 | Repeat the Nearest Neighbor and Local Search 2-opt heuristics for each resulting Optics cluster. |
| 11 | If all routes for all clusters have been generated, continue to step 12. Otherwise, return to step 10. |
| 12 | Plot all routes. |
| 13 | If subtours or crossovers per route are generated, return to Step 9. Otherwise, continue to Step 14. |
| 14 | Evaluate compliance with loading constraints and time windows per route. |
| 15 | If all constraints are not met, go back to step 6. |

**Table 2**
Pseudocode

**Algorithm for VRPTW (K-means, Optics, Nearest Neighbor and Local Search 2-opt)**

```
01: Start procedure
02: Load databases
03: Read $\{i, Lat, Lon, E_i, S_i, a_i, b_i, K, Q_k\}$
04: $K'$ clusters $ceil\left[\left(\frac{\sum E_i}{Q_k}\right)/2\right]$
05: Assign axes K-means $\{Lat, Lon\}$
06: Initialize $K'$ cluster with their centroids $\theta$
07: while not converge:
08:     for $i$ in range $(I)$: // $I = dataset \leftarrow \{Lat, Lon\}$
09:         $C_{k'} := argmin \, \|x_i - \theta_{k'}\|^2$
10:     end for
11:     for $j$ in range $(K')$:
12:         $\theta_j := \frac{1}{N}\sum_{i=1}^{N} X_i$
13:     end for
14: end while
15: Assign $i$ to its respective $k'$ cluster
16: for $k'$ cluster in range $(K')$:
17:     Scaling variables to bring to a comparable level
18:     $\beta = E_i . S_i . a_i . b_i$
19:     Standard Scaler $(\beta)$
20:     Normalizing $(\beta)$ to approximately follows a Gaussian distribution
21:     // Apply Optics clustering $(O')$ to $\beta$ normalized
22:     Optics $(min\_samples, \varepsilon, xi, cluster\_size)$ parameters
23:     // See pseudocode of Optics clustering in Ankerst, Breunig, Kriegel & Sander (1999)
```

```
24:         **for** $o'$ cluster in range $(O')$:
25:             **if** $\sum_{i=1}^{I} E_{io'} \leq Q_k$:
26:                 and $a_{i0} + S_{i0} + [random\ (a_{io'_1}) - a_{i0}] + \sum_{i=1}^{I} S_{io'} < b_{idp}$
27:                     Assign $i$ to its respective $O'$ cluster
28:                 else
29:                     Modify $O'$ parameters
30:             **end if**
31:         **end for**
32: **end for**
33: $i = (radians(i[0]).\ radians(i[1]))$
34: $j = (radians(j[0]).\ radians(j[1]))$
35: $dist = (\text{acos}(\sin(i[0]) * \sin(j[0]) + \cos(i[0]) * \cos(j[0]) * \cos(i[1] - j[1]))) * R$
36: **for** $node1, v$ in $D.items()$:
37:         **for** $node2, d$ in $v.items()$:
38:             $(node1, node2, d)$
39:             $(node1, node2, d)$
40:         **end for**
41: **end for**
42: **for** $o'$ cluster in range $(k')$:
43:         nodes = [$i$ for $i$ in range $(I_{o'})$]
44:         arcs = [$(i, j)$ for $i$ in range $(I_{o'})$ for $j$ in range $(I_{o'})$ if $i\ != j$]
45:         Select starting node $i_0$
46:         Visited nodes = $I'$
47:         **while** $i_0 + I' <$ nodes:
48:             Last node visited $(w) = I'[-1]$
49:             Calculate dist $(\sigma_w) = \{(w, j)$: $dist[(w, j)]$ for $j$ in range $I_{o'}$ if $w\ != j$ and $j$ not in $I'\}$
50:             Locate the min $(\sigma)$
51:             Add $w$ in $I'$
52:         **end while**
53:         Add $i_0$
54:         Select $I'_{o'}$
55:         min $change = 0$
56:         **while** (min $change < 0$):
57:             **for** $i$ in range $I'_{o'}$ until $i = i_0 - 2$:
58:                 **for** $j$ in range $j + 2$ until nodes:
59:                     $change = \big(dist(i.j) + dist(i + 1.j + 1)\big) - (dist(i.i + 1) - dist(j.j + 1)$
60:                     **if** (min $change$) $> change$:
61:                         (min $change$) $= change$
62:                         $route[i{:}j] = route[(j - 1){:}(i - 1){:}-1]$
63:                     **end if**
64:                 **end for**
65:             **end for**
66:         **end while**
67: **end for**
68: **if** $subtour$ or $crossing \geq 1$:
69:         Review 2-opt implementation
70: **end if**
71: **if** all constraints are not met:
72:         Modify $O'$ parameters
73: **end if**
74: **End procedure**.
```

### *4.1. K-means and Optics clustering*

Based on **Table 1** and **2**, the proposed algorithm begins with the loading of the databases that contain information regarding the clients or nodes, their respective location in terms of latitude and longitude, the amount of cargo to be delivered, the vehicles available and their capacity, the service time, the start and end of the time windows per node (in the algorithm the minute "0" is 00:00 hours and the minute "1,439" is 23:59 hours). From the databases, the algorithm applies **K-means** clustering, a centroid-based method developed by MacQueen (1967), under the premise of partitioning a data set into subsets (clusters) with the sum of Euclidean distances of the data and the centroid or center of the cluster (Likas, Vlassis & Verbeek, 2003); as a required parameter of K-means, the number of clusters is defined by the division of the total amount of cargo and the capacity of the vehicles, its result is divided by two and is approximate to the highest integer. In this step (three), it does not infer the Euclidian distance calculation since this is not required to carry out any route; on the contrary, it is carried out to generate conglomerates by proximity. The K-means clustering indicates the cluster to which the client is assigned; then, to each cluster resulting from K-means a second clustering is applied, in this case the **Optics** (ordering points to identify the clustering structure) clustering developed by Ankerst et al. (1999) which finds clusters based on their density, with the premise of "creates an ordering of a database, additionally storing the core-distance and a suitable reachability-distance for each object" (Ankerst et al., 1999, p.52). Optics clustering generates groups depending on the ordering of the points, the accessibility values and the cores; while constructing an accessibility graph that assigns a range distance to each point and together with its parameters determines the point's membership of the cluster (Ankerst et al., 1999). Likewise, one of the peculiarities of Optics is that it does not require the desired number of clusters to be indicated, since it identifies the ideal amount for the data series.

For the algorithm, the variables referring to constraints are normalized to bring them to a comparable level following a Gaussian distribution, with the objective that they are the input data of the Optics clustering and are mathematically evaluated in terms of compliance. In the Optics design, the values in some of its parameters were modified (*number of samples*, *epsilon*, *xi value* and *cluster size*), because, if a value is indicated in all its 12 parameters, only the execution time is reduced, which reduces optimality in the generation of results, in addition to causing non-compliance with constraints of the routing model. In step five of the algorithm, it is necessary to validate whether the amount of load to be transported assigned to each Optics cluster according to the nodes that comprise it, does not exceed the capacity of the vehicle, as well as the fulfillment of the time variables. Once the validations have been carried out, it is understood that each cluster resulting from Optics is composed of a select group of nodes, which are the input data for the initialization of the routing phase. Before carrying out routing with the Nearest Neighbor and Local search 2-opt heuristics, each georeferenced node is converted into radians to generate the distance matrix with the *haversine* function, to contemplate the real distance between all the points of the network; it should be noted that this algorithm does not include the road network or the topography of the city. Generally, heuristics are flexible according to the distance function used; therefore, if a function other than *haversine* is used, a scalar must be generated to obtain distances close to the real ones.

The Nearest Neighbor is applied as a heuristic to generate an initial route and Local Search 2-opt to optimize it, together they are made to each cluster that generated Optics (the detail of the heuristics is presented in section 4.2). Finally, if subtours, crossovers are generated or not all the model constraints are satisfied, it is necessary to review the heuristics and/or modify the Optics clustering parameters to reassign nodes in other clusters.

## *4.2. Nearest Neighbor and Local Search 2-opt*

The heuristic Nearest Neighbor "is a constructive method for generating initial feasible solutions for CVRP with the simple idea of inserting the nearest neighbor of the last inserted customer in the route" (Caric & Gold, 2008, p.18). The first customer selected for all the clusters is the depot, from this point onwards customers begin to be inserted according to their criteria of proximity. The steps used were the following, taking as reference Kizilateş & Nuriyeva (2013): **1**, selection of depot. **2**, find an unvisited customer near and go there. **3**, Are there any customers left without visiting? If correct, repeat step **2**. Otherwise, **4**, return to the depot.

After the Nearest Neighbor algorithm is executed, an initial feasible route is obtained, which in most cases presents crossings, therefore Local Search 2-opt is used. Local Search is a heuristic generally used to solve difficult optimization problems, which is assigned a set of possible solutions with a cost function, which assigns numerical values to each solution (Aarts & Lenstra, 2003) in order to "iteratively move from a solution to a neighboring solution by applying local perturbations until an optimum has been reached" (Smet & Thomas, 2016, p.15). The so-called perturbations are small movements that consist in changing a small part of the current solution to obtain an alternative solution similar to the previous one; all these solutions with perturbations are stored in a set of solutions called 'neighborhood', from which the best solution is obtained after exploring all the possible stored solutions after applying the intensification that selects the next improvement movement to reach a local optimum (Smet & Thomas, 2016). Generally, an initial solution generated by another algorithm is used as a neighborhood, because, in neighborhoods with a considerable size of nodes, it will take longer to explore and therefore to solve (Aarts & Lenstra, 2003; Smet & Thomas, 2016). 2-opt algorithm belongs to the Local Search family, proposed by Croes (1958) with the objective of relating 2 arcs and 4 nodes to eliminate the present crossing, reordering these two arcs to each other and calculating a new distance (Englert, Röglin & Vöcking, 2006). That is, 2-opt consists of finding pairs $(\boldsymbol{i},\boldsymbol{i}+\mathbf{1})$ and $(\boldsymbol{j}+\mathbf{1},\boldsymbol{j})$ which when changed by $(\boldsymbol{i},\boldsymbol{j})$ and $(\boldsymbol{i}+\mathbf{1},\boldsymbol{j}+\mathbf{1})$ minimizes the distance. During 2-opt two functions are calculated that evaluate which pair is going to be changed: **1**, $\boldsymbol{Current\,cost} = \boldsymbol{distance}(\boldsymbol{i},\boldsymbol{i}+\mathbf{1}) + \boldsymbol{distance}(\boldsymbol{j},\boldsymbol{j}+\mathbf{1})$ and **2**, $\boldsymbol{New\,cost} = \boldsymbol{distance}(\boldsymbol{i},\boldsymbol{j}) + \boldsymbol{distance}(\boldsymbol{i}+\mathbf{1},\boldsymbol{j}+\mathbf{1})$. If the change: $\boldsymbol{Change} = \boldsymbol{New\,cost} - \boldsymbol{Current\,cost}$ result is negative, that is, the new distance is less than the current one, such change will be applied, usually "called a 2-exchange" (Verhoeven, Aarts & Swinkels, 1995, p.177); the steps used are the following taking by reference to Croes (1958).

**1**, starts with an initial solution generated by another algorithm. **2**, two paths are executed in all network nodes; the first path starts at the $\boldsymbol{i_0}$ and ends in a node before the end $\boldsymbol{i_{-2}}$ and the second, starts on a next node $\boldsymbol{j_{+2}}$ to the start node $\boldsymbol{j}$ ($\boldsymbol{i}$=$\boldsymbol{j}$). **3**, the distances are calculated. And **4**, the criteria of the change are evaluated; if it meets the criteria, it reverses the sequence of the pairs, otherwise it returns to the step **2**.

In addition, the advantage of the second heuristic is that it finds near optimal, and in some cases, optimal solutions in relatively short spaces of time, with low memory consumption, because it only stores the small movements, instead of several complete solutions of the whole network (Smet & Thomas, 2016; Aarts & Lenstra, 2003; Croes, 1958). The proposed algorithm for route generation starts with the solution generated by Nearest Neighbor, its sequence and distance obtained for each route. It then applies Local Search 2-opt and exports the generated optimized sequence, the distance, the solution time, and the total number of iterations.

## 5. Results

To begin with, the K-means clustering generated three clusters (see **Fig. 5**), to which, for each resulting cluster, Optics clustering was applied, resulting in nine clusters, which translates into nine routes (see **Fig. 6-7**). In this way, each node was assigned to the cluster that was deferred by -1, 0 and 1. The load assigned to transport does not exceed the vehicle capacity of 5 tons in any of the cases, providing attention to 100% of the nodes, delivering the entire amount of cargo demanded without subtours or crossings on the route. In terms of distance, Nearest Neighbor returns a total of 221.09 km; However, with the Local Search 2-opt optimization this is reduced by 5.06%, resulting in a total distance of 209.91 km for the nine routes, for which 88.9% were optimized by eliminating the intersections present. According to the solution/execution time to generate the nine routes, the algorithm took 0.019016 seconds, with a total of 41 iterations. **Table 3** presents in detail the results by route.

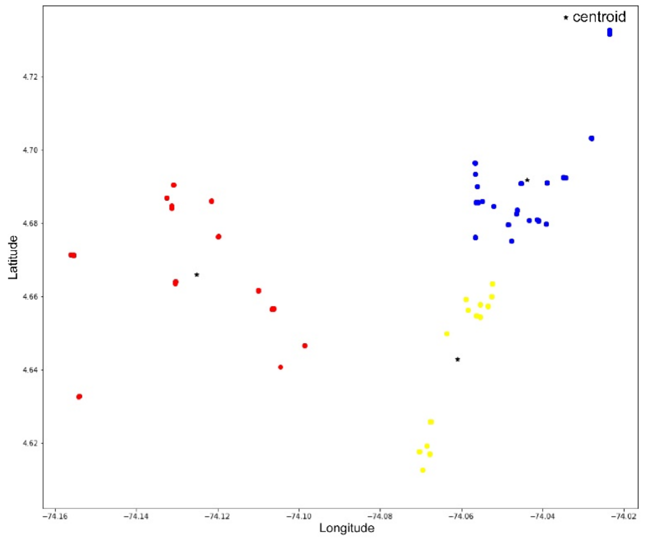

**Fig. 5**. K-means clusters

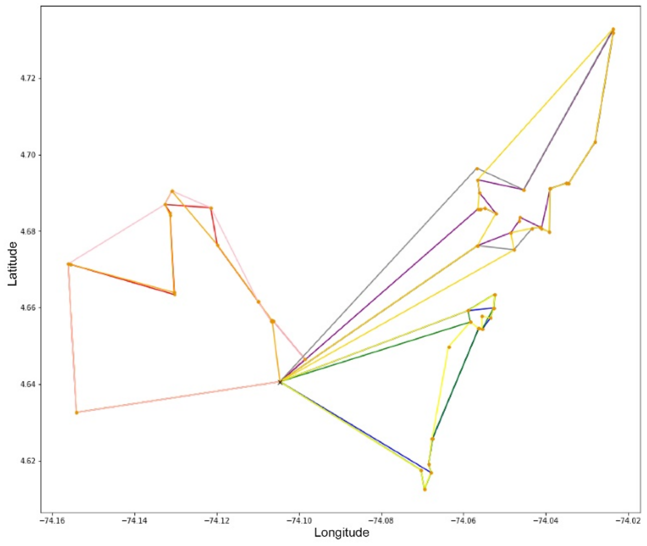

**Fig. 6**. Routes generated for 205 customers

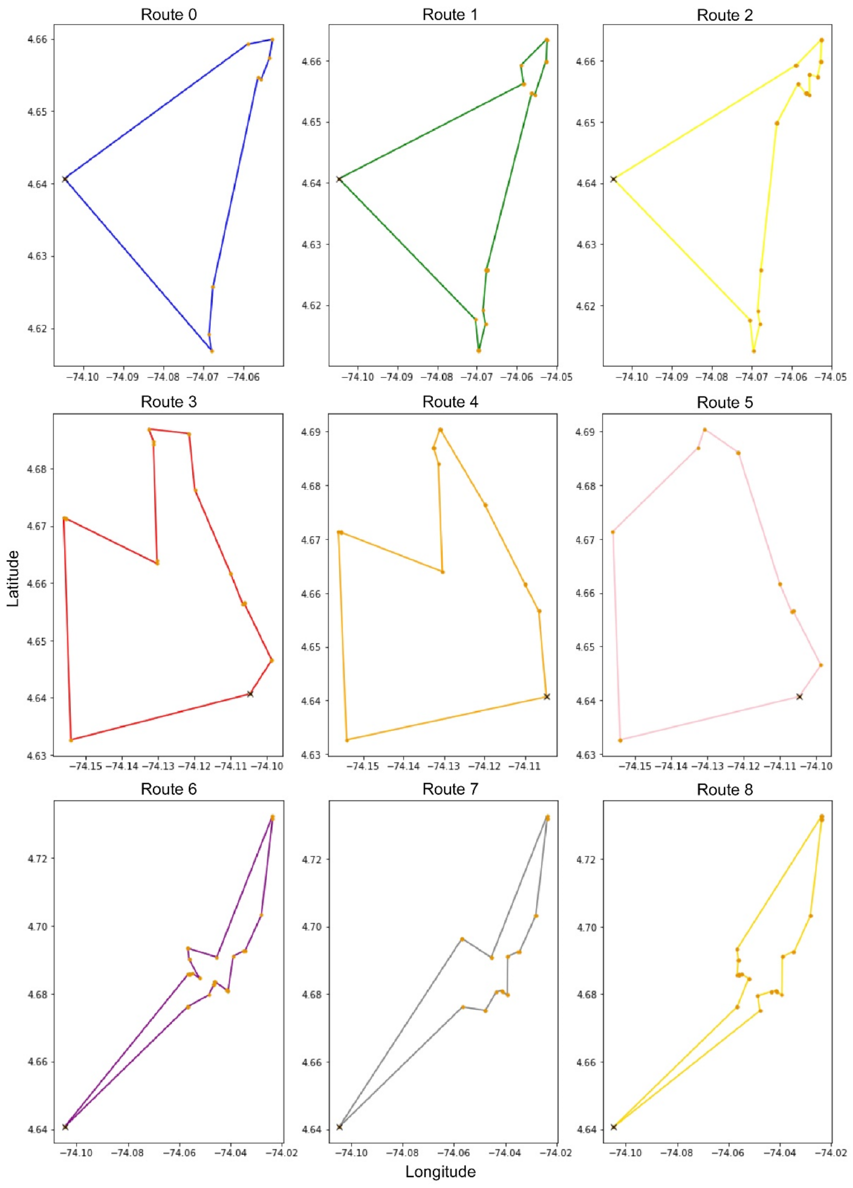

**Fig. 7**. Routes details

**Table 3**
Results per route for 205 nodes (customers)

| Clustering | | Route | Number of nodes | Cargo to transport [Kg] | Distance travelled | | | Execution time [Seg] | Number of iterations |
|---|---|---|---|---|---|---|---|---|---|
| K-means | Optics | | | | NN* [km] | LS* 2-opt [km] | Diff. | | |
| 0 | -1 | 0 | 9 | 1,311.30 | 16.80 | 16.28 | -3.10% | 0.001 | 2 |
| | 0 | 1 | 27 | 3,846.36 | 19.49 | 17.98 | -7.75% | 0.004 | 10 |
| | 1 | 2 | 30 | 1,325.13 | 19.44 | 18.34 | -5.66% | 0.002 | 4 |
| 1 | -1 | 3 | 19 | 4,863.40 | 23.17 | 22.93 | -1.04% | 0.001 | 4 |
| | 0 | 4 | 16** | 263.85 | 22.59 | 22.33 | -1.15% | 0.001 | 3 |
| | 1 | 5 | 14 | 151.27 | 20.71 | 20.71 | 0.00% | 0.000 | 1 |
| 2 | -1 | 6 | 33 | 3,156.78 | 34.44 | 30.72 | -10.80% | 0.003 | 5 |
| | 0 | 7 | 23 | 4,722.57 | 33.61 | 30.28 | -9.91% | 0.002 | 5 |
| | 1 | 8 | 35 | 2,756.80 | 30.84 | 30.34 | -1.62% | 0.005 | 7 |
| **Total** | **9** | **9** | **206** | **22,397.46** | **221.09** | **209.91** | **-5.06%** | **0.019** | **41** |

*NN (Nearest Neighbor), LS (Local Search). **Includes depot.

The service time constraints and time windows are fulfilled in 100% of the routes for all nodes from which a delivery time is obtained for the nine routes of: {617.23; 792.91; 828.64; 916.80; 1,000.24; 878.37; 894.38; 776.34; 846.61 [min] (includes the 360 minutes that have elapsed since 00:00 hours)} respectively, complying with the time window of the depot that starts from minute 360 to 1,140 of the day, in accordance with the mobility restrictions and depot constraints. In short, the clustering algorithms carried out made it possible to reduce the total number of possible solutions, while assigning the constraints based on their compliance; Nearest Neighbor is defined as not generating optimal solutions; however, the calculated routes are of good quality unless they have crossings, which is why Local Search 2-opt was indexed as a solution optimization, from which routes without crossings and very close to the optimal ones diverge. Thus, the solution flow allows the integration of new techniques to solve the complexity of the problem, complying with all the proposed constraints.

## *5.1. Performance*

This section evaluates the algorithm in terms of execution, compliance with constraints and distance for a larger number of nodes in the network; for this a Dbscan clustering with 2 samples and an epsilon of 0.00345 was designed from the original database, resulting in 274 clusters, that is, 758 nodes. This number of nodes translates to 2.85% more customers than the logistics operator has served in one day in Bogotá. From K-means, five clusters are generated, to which the Optics clustering is subsequently applied with its respective modification in the value of parameters, resulting in 19 clusters, indicating 19 routes for the attention of all nodes, as can be seen in **Fig. 8**. To all routes, the distance and the execution time was 545.54 km for Local Search 2-opt, in contrast to Nearest Neighbor of 619.76 km in total, for the 19 routes, generating a reduction 11.98% of the distance traveled; through 176 iterations in 0.720317 seconds (see **Table 4**). The attention of all customers is fully met, without subtours or crossings; all cargo is delivered, and each vehicle transports the quantity demanded, without exceeding its 7.5 tons capacity selected for this scenario within the time windows.

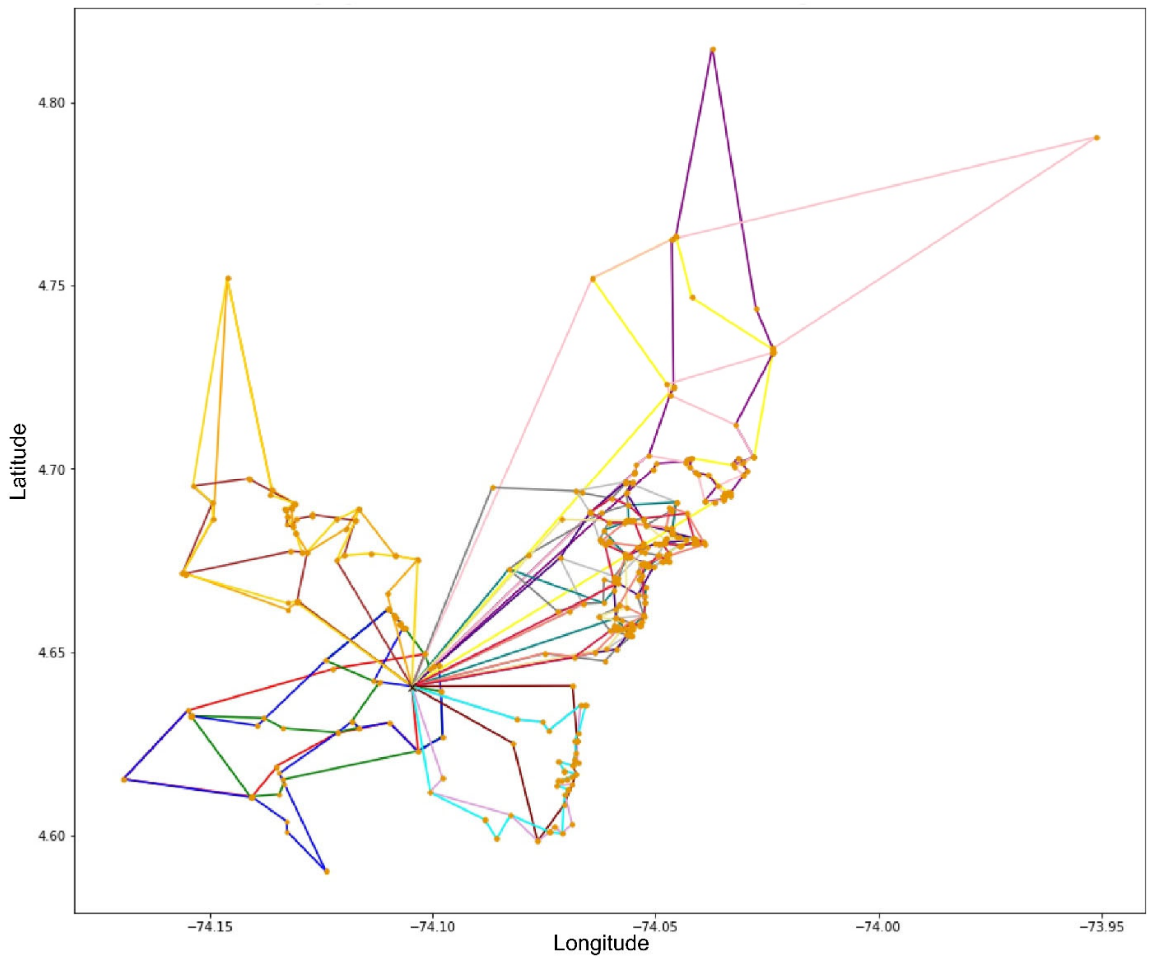

**Fig. 8**. Routes generated for 758 customers.

**Table 4**
Results per route for 758 nodes (customers)

| Clustering | | Route | Number of nodes | Cargo to transport [Kg] | Distance travelled | | | Execution time [Seg] | Number of iterations |
|---|---|---|---|---|---|---|---|---|---|
| **K-means** | **Optics** | | | | **NN* [km]** | **LS* 2-opt [km]** | **Diff.** | | |
| 0 | -1 | 0 | 12** | 383.59 | 23.34 | 20.41 | -12.55% | 0.000 | 4 |
| | 0 | 1 | 43 | 4,077.68 | 26.15 | 22.89 | -12.47% | 0.010 | 7 |
| | 1 | 2 | 44 | 3,805.12 | 39.35 | 30.71 | -21.96% | 0.017 | 11 |
| 1 | -1 | 3 | 20 | 6,142.88 | 42.22 | 38.56 | -8.67% | 0.000 | 5 |
| | 0 | 4 | 60 | 8371.2 | 53.05 | 49.29 | -7.09% | 0.050 | 16 |
| | 1 | 5 | 37 | 4284.51 | 64.63 | 57.43 | -11.14% | 0.010 | 7 |
| 2 | -1 | 6 | 24 | 6,347.98 | 38.51 | 33.30 | -13.53% | 0.000 | 7 |
| | 0 | 7 | 31 | 267.65 | 25.60 | 22.69 | -11.37% | 0.007 | 7 |
| | 1 | 8 | 38 | 6,535.28 | 44.15 | 34.93 | -20.88% | 0.008 | 10 |
| 3 | -1 | 9 | 177 | 5,838.93 | 44.03 | 40.94 | -7.02% | 0.558 | 26 |
| | 0 | 10 | 40 | 1,858.46 | 29.96 | 28.11 | -6.17% | 0.010 | 7 |
| | 1 | 11 | 24 | 1,429.53 | 26.57 | 23.41 | -11.89% | 0.000 | 7 |
| | 2 | 12 | 22 | 920.27 | 26.07 | 23.03 | -11.66% | 0.006 | 6 |
| | 3 | 13 | 22 | 140.00 | 23.54 | 21.54 | -8.50% | 0.005 | 13 |
| | 4 | 14 | 22 | 988.92 | 25.69 | 22.49 | -12.46% | 0.000 | 7 |
| | 5 | 15 | 45 | 376.91 | 25.27 | 24.86 | -1.62% | 0.010 | 6 |
| 4 | -1 | 16 | 10 | 2,650.01 | 16.70 | 15.03 | -10.00% | 0.000 | 2 |
| | 0 | 17 | 50 | 3,977.76 | 21.49 | 18.00 | -16.24% | 0.015 | 12 |
| | 1 | 18 | 38 | 3,622.82 | 23.41 | 17.93 | -23.41% | 0.017 | 16 |
| **Total** | **19** | **19** | **759** | **62,019. 5** | **619.7** | **545.55** | **-11.97%** | **0.720** | **176** |

*NN (Nearest Neighbor). LS (Local Search). **Includes depot.

To contemplate a solution for more customers, considering the performance of the algorithm, it is necessary to modify the parameters of the clusters, since that is where the compliance with the constraints lies. In a **2,000 nodes** scenario, a solution is generated in **2.71 seconds** by performing 497 iterations. This demonstrates efficiency of performance by the algorithm developed in terms of execution and memory use. It should be noted that the experiments carried out were run on a 1.80 GHz Intel Core i5 8th Gen processor with 8.00 GB of memory and a 64-bit operating system.

## *5.2. Computational studies*

In this section, computational studies are carried out to compare the performance of the designed algorithm against models from the existing literature, taking the instances of Solomon (1986) as a reference in which six types of problems that affect the behavior of the algorithms for VRPTW are raised. These are categorized into Random (R), Clustered (C) and Random & Clustered (RC) which in turn are classified into two types; type **1**, short scheduling horizon (R1, C1, RC1); and, type **2**, long scheduling horizon (R2, C2, RC2). The data integrates the variables of number of customers, geographic location, amount of demand, time window (ready and due time) and service time, which are taken from Solomon (2005). In this benchmark, the designed algorithm is compared against the best-known solutions identified by heuristics before March 24, 2005. To measure its performance, the formula $\%GAP_{best} = [(PA - best\ known\ solution)/best\ known\ solution] * 100\%$ posed by (Küçükoğlu & Öztürk, 2015) is adapted. Where PA (Proposed Algorithm), indicates the solution of the algorithm proposed in this article. Thus, 12 problems with Euclidean distance are considered for 100 customers: R101, R102, C101, C102, RC101 and RC102, with 200 capacity in the vehicle; R201, R202, RC201 and RC202 with 1,000 capacity in the vehicle; and finally, C201 and C202 with 700 capacity in the vehicle. It should be noted that the algorithm is adjusted to a Euclidean distance to obtain comparable distances. The results are shown in **Table 5**.

Based on **Table 5**, of the 12 problems considered, 66.7% generated better results than those identified as the best solution in the literature. A decrease of 10.87% in the total number of vehicles (NV) used is noteworthy, as well as a decrease in the total distance of 2.09% (14,140.8) compared to 14,442.67, despite the failure to comply with the four type C problems evaluated. In fact, those problems for which no better results were obtained are close to the solutions identified in the literature, without exceeding 3.67%, 3.00% and 1.49% of the best solution for the four cases, respectively.

In this benchmark, the number of iterations executed by each case and the generation of solutions in minimum execution times (does not exceed 0.05 seconds for all problems) stands out; as an example, the minimum average times of 11.90 seconds, of the best runs of RC problems for 100 customers executed by Taillard et al. (1997) was exceeded by 99.94%. However, the value of CPU is not comparable given the difference in hardware used to solve the problems.

**Table 5**
Benchmark results

| Problem type | Best known solution | | | PA | | | | %GAP |
|---|---|---|---|---|---|---|---|---|
| | NV | Distance | Author(s) | NV | Distance | CPU [seg] | Iterations | |
| R101 | 19 | 1,645.79 | Homberger (2000) | 16 | 1,567.59 | 0.0194 | 41 | -4.75 |
| R102 | 17 | 1,486.12 | Rochat & Taillard (1995) | 15 | 1,404.98 | 0.0111 | 36 | -5.46 |
| C101 | 10 | 828.94 | Rochat & Taillard (1995) | 10 | 859.34 | 0.0075 | 24 | 3.67 |
| C102 | 10 | 828.94 | Rochat & Taillard (1995) | 10 | 859.34 | 0.0075 | 24 | 3.67 |
| RC101 | 14 | 1,696.94 | Taillard, Badeau, Gendreau, Geurtin & Potvin (1997) | 12 | 1,623.72 | 0.0066 | 26 | -4.31 |
| RC102 | 12 | 1,554.75 | Taillard, Badeau, Gendreau, Geurtin & Potvin (1997) | 11 | 1,505.87 | 0.0030 | 30 | -3.14 |
| R201 | 4 | 1,252.37 | Homberger & Gehring (1999) | 3 | 1,248.71 | 0.0309 | 34 | -0.29 |
| R202 | 3 | 1,191.70 | Rousseau, Gendreau & Pesant (2002) | 2 | 1,130.84 | 0.0170 | 16 | -5.11 |
| C201 | 3 | 591.56 | Rochat & Taillard (1995) | 3 | 609.33 | 0.0499 | 23 | 3.00 |
| C202 | 3 | 591.56 | Rochat & Taillard (1995) | 3 | 600.38 | 0.0294 | 24 | 1.49 |
| RC201 | 4 | 1,406.91 | Mester (2002) | 4 | 1,395.30 | 0.0218 | 25 | -0.83 |
| RC202 | 3 | 1,367.09 | Czech & Czarnas (2002) | 3 | 1,335.35 | 0.0325 | 20 | -2.32 |
| **Total** | **102** | **14,442.67** | | **92** | **14,140.8** | **0.2367** | **323** | **-2.09** |

## 6. Conclusion

This study considered a vehicle routing problem with time windows, usually known as VRPTW. Given the complexity of the problem, based on its constraints and objective function, a robust solution algorithm programmed in Python 3.7 is proposed, which takes as a reference the database of deliveries made by a logistics operator in Colombia. By integrating the K-means and Optics clustering techniques, as well as the Nearest Neighbor heuristics for the generation of routes and Local Search 2-opt for their optimization, good solutions for the proposed scenarios are generated. To evaluate the designed algorithm, several scenarios were made varying the number of nodes and vehicles, the associated load, service times and time windows. Starting from an initial base of 205 customers, a solution is found in 0.019 seconds by executing 41 iterations; then, a scenario of 758 customers is solved in 0.720 seconds by means of 176 iterations. Finally, a scenario of 2,000 customers is tested, giving results in 2.71 seconds after 497 iterations. This demonstrates high performance and efficiency for large instances in terms of execution, compliance with constraints and memory usage. Additionally, a benchmark is made based on the instances of Solomon (1986) against the results obtained from 12 problems looked at by several authors of the existing literature. Of this section, the short execution times, the reduction of 2.09% in the total distance and 10.87% in the number of vehicles required stand out, as well as the obtaining of better solutions in 66.7% of the problems evaluated. From this comparison, the performance of the algorithm and its adaptability to any type of problem is confirmed. Finally, the evaluation and the benchmark allow scaling the capacity of the designed algorithm, where it shows high performance in terms of route generation and solution while at the same time providing routes close to the optimal ones for large customer instances, demonstrating competitiveness against other solution methods.

**Acknowledgements**

We give special thanks to the Catholic University of Colombia for funding this research as a result of the 2020 young researcher program, as well as to all the reviewers who with their contributions improved the quality of this writing.

## References

Aarts, E. & Lenstra, J. K. (Ed). (2003). *Local Search in Combinatorial Optimization*. Princeton. Oxford: Princeton University Press. ISBN 9780691115221

Abbatecola, L., Fanti, M. P., Pedroncelli, G., & Ukovich, W. (2018). A New Cluster-Based Approach for the Vehicle Routing Problem with Time Windows. *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE),* *Automation Science and Engineering (CASE), 2018 IEEE 14th International Conference On*, 744–749. doi: 10.1109/COASE.2018.8560419

Anaya, J. (2015). *El transporte de mercancías: Enfoque logístico de la distribución*. España: ESIC.

Ankerst, M., Breunig, M. M., Kriegel, H.-P. & Sander, J. (1999). OPTICS: Ordering Points To Identify the Clustering Structure. *ACM SIGMOD Record, 28*(2), 49–60. doi: 10.1145/304181.304187

Applegate, D., Bixby, R., Chvátal, V. & Cook, W. (2007). *The Traveling Salesman Problem: A Computational Study*. New Jersey: Princeton University Press.

Ayu, K. G., Septivani, N., Xu, S., Kwan, S., & Ani. (2015). Optimum clustering and routing model using CVRP cluster-first, route-second in a 3PL provider. *2015 International Conference on Industrial Engineering and Operations Management (IEOM)*. doi: 10.1109/ieom.2015.7093800

Bai, R., Chen, X., Chen, Z. L., Cui, T., Gong, S., He, W., Jiang, X., Jin, H., Jin, J., Kendall, G., Li, J., Lu, Z., Ren, J., Weng, P., Xue, N. & Zhang, H. (2021). Analytics and Machine Learning in Vehicle Routing Research. *arXiv:2102.10012*.

Barbarosoglu, G., & Ozgur, D. (1999). A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research, 26*(3), 255–270. doi: 10.1016/s0305-0548(98)00047-1

Bent, R. & Van Hentenryck, P. (2004). A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. *Transportation Science, 38*(4), 515–530. doi: 10.1287/trsc.1030.0049

Berger, J., Salois, M., & Begin, R. (1998). A hybrid genetic algorithm for the vehicle routing problem with time windows. In *Conference of the Canadian society for computational studies of intelligence. Lecture Notes in Computer Science,* 114–127. Springer Verlag. doi: 10.1007/3-540-64575-6_44

Best Urban Freight Solutions. (2006). *Quantification of Urban Freight Transport Effects I.*

Bodin, L. D. (1975). A taxonomic structure for vehicle routing and scheduling problems. *Computers & Urban Society, 1*(1), 11–29.

Bogotá Mayor's Office (2019). *Decreto 840 de 2019 por medio del cual se establecen las condiciones y restricciones para el tránsito de los vehículos de transporte de carga en el Distrito Capital y se dictan otras disposiciones.* Bogotá. Retrieved from http://www.andi.com.co/Uploads/Decreto%20840%202019.pdf

Bogotá Mayor's Office (2020). *Decreto 047 de 2020 Por medio del cual se toman medidas transitorias y preventivas en materia de tránsito en las vías púbicas en el Distrito Capital y se dictan otras disposiciones.* Bogotá. Retrieved from http://www.andi.com.co/Uploads/Decreto%20047%20de%202020.pdf

Bouhamed, O., Ghazzai, H., Besbes, H. & Massoud, Y. (2019). Q-learning based routing scheduling for a multi-task autonomous agent. *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 634-637. doi: 10.1109/MWSCAS.2019.8885080

Bowersox, D. J., & Calantone, R. J. (1998). Executive Insights: Global Logistics. *Journal of International Marketing, 6*(4), 83–93. doi: 10.1177/1069031x9800600410

Braekers, K., Ramaekers, K. & Nieuwenhuyse, I. V. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering, 99*, 300-313. doi: 10.1016/j.cie.2015.12.007

Bräysy, O. (2003). A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows. *INFORMS Journal on Computing, 15*(4), 347-368.

Bräysy, O., & Gendreau, M. (2005). Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science, 39*(1), 104–118. doi: 10.1287/trsc.1030.0056

Buhrkal, K., Larsen, A. & Ropke, S. (2012). The waste collection vehicle routing problem with time windows in a city logistics context. *Procedia - Social and Behavioral Sciences, 39,* 241-254. doi: 10.1016/j.sbspro.2012.03.105

Caric, T. & Gold, H. (Ed.). (2008). *Vehicle Routing Problem*. Austria. Vienna: In-Teh is Croatian branch of I-Tech Education and Publishing KG. ISBN 978-953-7619-09-1

Chen, D. & Yang, Z. (2017). Multiple depots vehicle routing problem in the context of total urban traffic equilibrium. *Journal of Advanced Transportation*, *2017*, 1–14. doi: 10.1155/2017/8524960

Cömert, S., Yazgan, H., Sertvuran, İ., & Şengül, H. (2017). A new approach for solution of vehicle routing problem with hard time window: an application in a supermarket chain. *Sadhana*, *42*(12), 2067–2080. doi: 10.1007/s12046-017-0754-1

Cordeau, J. F.; Desaulniers, G.; Desrosiers, J.; Solomon, M. M. & Soumis, F. (2000). The VRP with Time Windows. *Les Cahiers du GERAD.*

Cordeau, J. F., Laporte, G., Savelsbergh, M. W. P. & Vigo, D. (2007). Chapter 6: vehicle routing. In: Barnhart. C. y Laporte. G. *Handbooks in Operations Research and Management Science: Transportation, 14*, 367-428. Amsterdam: North-Holland. Elsevier.

Croes, G. (1958). A Method for Solving Traveling-Salesman Problems. *Operations Research, 6*(6), 791-812. doi: 10.2307/167074

Czech, Z. J., & Czarnas, P. (2002). Parallel simulated annealing for the vehicle routing problem with time windows. *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing*, Canary Islands, Spain. doi: 10.1109/empdp.2002.994313

Dantzig, G. & Ramser, J. (1959). The truck dispatching problem. *Management Science, 6*(1), 80-91. doi: 10.1287/mnsc.6.1.80

Dondo, R., & Cerdá, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research, 176*(3), 1478–1507. doi: 10.1016/j.ejor.2004.07.077

Elshaer, R. & Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering 106242, 140.* doi: 1016/j.cie.2019.106242

Englert, M., Röglin, H. & Vöcking, B. (2006). Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP. *Electronic Colloquium on Computational Complexity. Report No. 92*, 190–264. ISSN 1433-8092.

Ester, M., Kriegel, H. P., Sander, J. & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96), AAAI Press*, 226–231.

Fachini, R. F., & Armentano, V. A. (2020). Logic-based Benders decomposition for the heterogeneous fixed fleet vehicle routing problem with time windows. *Computers & Industrial Engineering*, *148*. doi: 10.1016/j.cie.2020.106641

Fernández, I. (2008). *Modelización de la distribución urbana de mercancías* (undergraduate thesis). Universitat Politécnica de Catalunya, Cataluña, España.

Furian, N., O'Sullivan, M., Walker, C. & Çela, E. (2021). A machine learning-based branch and price algorithm for a sampled vehicle routing problem. *OR Spectrum*. doi: 10.1007/s00291-020-00615-8

Galba, T., Balkić, Z. & Martinović, G. (2013). Public Transportation BigData Clustering. *International journal of electrical and computer engineering systems, 4*(1), 21-26.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science, 40*(10), 1276–1290. doi: 10.1287/mnsc.40.10.1276

Gendreau, M., Potvin, J. Y., Bräumlaysy, O., Hasle, G. & Løkketangen, A. (2008). Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography. In B. Golden; S. Raghavan y E. Wasil (eds), *The Vehicle Routing Problem: Latest Advances and New Challenges. Operations Research/Computer Science Interfaces*, 43. Boston: Springer. doi: 10.1007/978-0-387-77778-8_7

Ghoseiri, K., & Ghannadpour, S. F. (2010). A hybrid genetic algorithm for multi-depot homogenous locomotive assignment with time windows. *Applied Soft Computing Journal*, *10*(1), 53–65. doi: 10.1016/j.asoc.2009.06.004

Glover, F., Taillard, E., & de Werra, D. (1993). A user's guide to tabu search. *Annals of Operations Research, 41*(1), 1–28. doi: 10.1007/bf02078647

Groër, C., Golden, B., & Wasil, E. (2010). A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation, 2*(2), 79–101. doi: 10.1007/s12532-010-0013-5

Guerequeta, R. & Vallecillo, A. (2000). *Técnicas de diseño de algoritmos*. Málaga, España: Servicio de publicaciones de la Universidad de Málaga. ISBN 84-7496-666-3

Gupta, R., Singh, B. & Pandey, D. (2010). Multi-Objective Fuzzy Vehicle Routing Problem: A Case Study. *Int. J. Contemp. Math. Sciences, 5*(29), 1439–1454.

Gutiérrez-Rubiano, D. F., Hincapié-Montes J. A. & León-Villalba, A. F. (2019). Collaborative distribution: strategies to generate efficiencies in urban distribution - Results of two pilot tests in the city of Bogotá. *DYNA, 86*(210), 42-51. doi: 10.15446/dyna.v86n210.78931

Hair, J. F., Black, W., Babin, B. & Anderson, R. (2014). *Multivariate Data Analysis*. Pearson.

Hiquebran, D. T., Alfa, A. S., Shapiro, J. A., & Gittoes, D. H. (1993). A revised simulated annealing and cluster-first route-second algorithm applied to the vehicle routing problem. *Engineering Optimization, 22*(2), 77–107. doi: 10.1080/03052159308941327

Homberger, J. (2000). *Verteilt-parallele Metaheuristiken zur Tourenplanung*. Gaber, Wiesbaden

Homberger, J., & Gehring, H. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research, 37*(3), 297–318. doi: 10.1080/03155986.1999.11732386

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi: 10.1109/mcse.2007.55

Kalakanti, A. K., Verma, S., Paul, T., & Yoshida, T. (2019). RL SolVeR Pro: Reinforcement Learning for Solving Vehicle Routing Problem. *2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, 94–99. doi: 10.1109/aidas47888.2019.8970890

Kang, H. Y. & Lee, A. (2018). An Enhanced Approach for the Multiple Vehicle Routing Problem with Heterogeneous Vehicles and a Soft Time Window. *Symmetry, 10*(11), 650. doi:10.3390/sym10110650

Kao, Y., & Chen, M. (2013). Solving the CVRP Problem Using a Hybrid PSO Approach. *Computational Intelligence,* 59–67. doi: 10.1007/978-3-642-35638-4_5

Karagül, K. & Gungor, I. (2014). A case study of heterogeneous fleet vehicle routing problem: Touristic distribution application in Alanya. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA), 4*(2), 67–76. doi: 10.11121/ijocta.01.2014.00185

Khodabandeh, E., Bai, L., Heragu, S. S., Evans, G. W., Elrod, T. & Shirkness, M. (2016). Modelling and solution of a large-scale vehicle routing problem at GE appliances & lighting. *International Journal of Production Research, 55*(4), 1100–1116. doi: 10.1080/00207543.2016.1220685.

Kim, B.-I., Kim, S., & Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers & Operations Research, 33*(12), 3624–3642. doi: 10.1016/j.cor.2005.02.045

Kizilateş, G., & Nuriyeva, F. (2013). On the Nearest Neighbor Algorithms for the Traveling Salesman Problem. *Advances in Computational Science. Engineering and Information Technology*, 111–118. doi: 10.1007/978-3-319-00951-3_11

Küçükoğlu, İ., & Öztürk, N. (2015). An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows. *Computers & Industrial Engineering, 86,* 60–68.

Ladner, R. (1975). On the structure of polynomial time reducibility. *Journal of the Association for Computing Machinery,* *22*(1), 155-171. doi: 10.1145/321864.321877

Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science, 43*(4), 408-416. doi: 10.1287/trsc.1090.0301

Lide, D. R. (Ed.). (2003). *Handbook of Chemistry and Physics*. (2003). United States: CRC Press. ISBN 0-8493-0481-4

Likas, A., Vlassis, N. & Verbeek, J. (2003). The global k-means clustering algorithm. *Pattern Recognition, 36*(2), 451–461. doi: 10.1016/s0031-3203(02)00060-2

Lin, B., Ghaddar, B. & Nathwani, J. (2020). Deep reinforcement learning for the electricvehicle routing problem with time windows. *arXiv:2010.02068v2*

López, E. R., & de Jesús, J. (2015). A hybrid column generation and clustering approach to the school bus routing problem with time windows. *Ingeniería (0121-750X)*, *20*(1), 111–127. doi: 10.14483/udistrital.jour.reving.2015.1.a07

Lüer, A., Benavente, M., Bustos, J., & Venegas, B. (2009). El problema de rutas de vehículos: Extensiones y métodos de resolución estado del arte. *Workshop Internacional EIG 2009 - Actas 3er Encuentro Informatica y Gestion,* 558.

MacQueen, J. B. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, 1*, 281-297.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9thPython in Science Conference, 445*, 56 –61. doi: 10.25080/Majora-92bf1922-00a

Mester, D. (1999). A parallel dichotomy algorithm for vehicle routing problem with time windows, Working paper, Minerva Optimization Center, Technion, Israel.

Miller, C., Tucker, A. & Zemlin, R. (1960). Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM (JACM), 7*(4), 326–329. doi: 10.1145/321043.321046

Moen, O. (2016). The Five-step Model – Procurement to Increase Transport Efficiency for an Urban Distribution of Goods. *Transportation Research Procedia, 12,* 861–873. doi: 10.1016/j.trpro.2016.02.039

Mohri, M., Rostamizadeh, A. & Talwalkar, A. (2018). *Foundations of Machine Learning*. Cambridge, MA: MIT Press. Retrieved from https://cs.nyu.edu/~mohri/mlbook/

Moon, I., Lee, J.-H. & Seong, J. (2012). Vehicle routing problem with time windows considering overtime and outsourcing vehicles. *Expert Systems with Applications, 39*(18), 13202-13213. doi: 10.1016/j.eswa.2012.05.081

Nazari, M., Oroojlooy, A., Snyder, L. V., Takáč, M. (2018). Reinforcement Learning for Solving the Vehicle Routing Problem. *arXiv:1802.04240v2*.

Oliphant, T. E. (2006). *Guide to NumPy*. MDTDR.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Pemberthy, J. (2012). *Implementación de un algoritmo metaheurístico para la solución de un problema de programación de transporte terrestre internacional* (master thesis). National University of Colombia, Medellín, Colombia.

Pisinger, D., & Ropke, S. (2010). Large Neighborhood Search. *International Series in Operations Research & Management Science*, 399–419. doi: 10.1007/978-1-4419-1665-5_13

Poullet, J. (2020). *Leveraging machine learning to solve the vehicle routing problem with time windows* (master thesis). Massachusetts Institute of Technology, Cambridge, United States.

Prag, K., Woolway, M. & Jacobs, B. (2019). Optimising the Vehicle Routing Problem with Time Windows under Standardised Metrics. *2019 6th International Conference on Soft Computing & Machine Intelligence (ISCMI), Soft Computing & Machine Intelligence (ISCMI), 2019 6th International Conference On*, 111–115. doi: 10.1109/ISCMI47871.2019.9004294

Qi, M., Lin, W.-H., Li, N., & Miao, L. (2012). A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows. *Transportation Research Part E*, *48*(1), 248–257. doi: 10.1016/j.tre.2011.07.001

Rahman, A. (2012). The traveling Salesman problem. 1-15. Retrieved from http://cs.indstate.edu/~zeeshan/aman.pdf

Rochat, Y., & Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics, 1*(1), 147–167. doi: 10.1007/bf02430370

Rousseau, L.M.; Gendreau, M. & Peasant. G. (2002). Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics, 8,* 43–58. doi: 10.1023/A:1013661617536

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. *Lecture Notes in Computer Science,* 417–431. doi: 10.1007/3-540-49481-2_30

Shin, K. & Han, S. (2011). A centroid-based heuristic algorithm for the capacitated vehicle routing problem. *Computing and Informatics, 30*, 721–732.

Sinnott, R. W. (1984). Virtues of the Haversine. *Sky and Telescope, 68*(2).

Smet, L. & Thomas, C. (2016). *Local Search for the Vehicle Routing Problem* (master thesis). Université Catholique de Louvain. Bélgica.

Solano, E., Montoya-Torres, J. & Guerrero-Rueda, W. (2019). A decision support system for technician routing with time windows. A case study of a Colombian public utility company. *Academia Revista Latinoamericana de Administración, 23*(2), 138-158. doi: 10.1108/ARLA-04-2017-0101

Solomon, M. M. (1986). On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks, 16,* 161–174.

Solomon, M. M. (2005). VRPTW benchmark problems. CBA. Retrieved from http://web.cba.neu.edu/~msolomon/problems.htm

Son, H., Kim, G. & Shin, H. (2018). Case study: vehicle routing problem with time windows for a shop in fisheries wholesale market. *International Journal of Management and Applied Science, 4*(4), 44-47. ISSN 2394-7926.

Stehling, T. M. & de Souza, S. R. (2017). A Comparison of Crossover Operators Applied to the Vehicle Routing Problem with Time Window. *2017 Brazilian Conference on Intelligent Systems (BRACIS),* 300–305. doi: 10.1109/BRACIS.2017.47

Sutton, R. S. & Barto, A. G. (2018). *Reinforcement learning: an introduction*. Cambridge, MA: MIT Press.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J.-Y. (1997). A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science, 31*(2), 170–186. doi: 10.1287/trsc.31.2.170

Tamagawa, D., Taniguchi, E., & Yamada, T. (2010). Evaluating city logistics measures using a multi-agent model. *Procedia - Social and Behavioral Sciences, 2*(3), 6002–6012. doi: 10.1016/j.sbspro.2010.04.014

Tan, K. C., Lee, L. H., Zhu, Q. L., & Ou, K. (2001). Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering, 15*(3), 281–295. doi:10.1016/s0954-1810(01)00005-x

Taner, F., Galić, A. & Carić, T. (2012). Solving Practical Vehicle Routing Problem with Time Windows Using Metaheuristic Algorithms. *Promet – Traffic&Transportation, 24*(4), 343-351.

Tirkolaee, E., Abbasian, P., Soltani, M. & Ghaffarian, S. (2019). Developing an applied algorithm for multi-trip vehicle routing problem with time windows in urban waste collection: A case study. *Waste Management & Research, 37*(1), 4–13. doi: 10.1177/0734242X18807001

Toro, E., Escobar A. & Granada, M. (2016). Literature review of vehicle routing problem in the green transportation context. *Revista Luna Azul, 42*, 362-387. doi: 10.17151/luaz.2016.42.21

Tran, T. N., Drab, K., & Daszykowski, M. (2013). Revised DBSCAN algorithm to cluster data with dense adjacent clusters. *Chemometrics and Intelligent Laboratory Systems*, *120*, 92–96. doi: 10.1016/j.chemolab.2012.11.006

Van Rossum, G. (1993). An Introduction to Python for UNIX/C Programmers. *Proceedings of the NLUUG najaarsconferentie. Dutch UNIX users group*, 1-8.

Verhoeven, M. G. A., Aarts, E. H. L., & Swinkels, P. C. J. (1995). A parallel 2-opt algorithm for the Traveling Salesman Problem. *Future Generation Computer Systems, 11*(2), 175–182.

Wy, J., Kim, B. I. & Kim, S. (2013). The rollon–rolloff waste collection vehicle routing problem with time windows. *European Journal of Operational Research, 224*(3), 466–476. doi: 10.1016/j.ejor.2012.09.001

Yepes, V. (2002). *Optimización heurística económica aplicada a las redes de transporte del tipo VRPTW* (doctoral thesis). Universidad Politécnica de Valencia, Valencia, España.

Yu, J. J. Q., Yu, W., & Gu, J. (2019). Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems,* 1–12. doi: 10.1109/tits.2019.2909109

Zhao, J., Mao, M., Zhao, X., & Zou, J. (2020). A hybrid of deep reinforcement learning and local search for the vehicle routing problems. *IEEE Transactions on Intelligent Transportation Systems,* 1–11. doi:10.1109/tits.2020.3003163