Politecnico di Torino

Corso di Laurea Magistrale: Engineering and Management

The application of two clustering methods in the vehicle routing problem - A numerical example in the city of Turin

Relatori:

Arianna Alfieri

Erica Pastore

Candidato:

Vittorio Guglielmo Glave

Dicembre 2023

Contents

List of Abbreviations

Abbreviations	Description
ACO	Ant Colony Optimization
B&B	Branch and Bound
B&C	Branch and Cut
CVRP	Capacitated Vehicle Routing Problem
DCVRP	Distance Constrained Vehicle Routing Problem
DP	Dynamic Programming
DVRP	Dynamic Vehicle Routing Problem
GA	Genetic Algorithm
GLS	Guided Local Search
HFVRP	Heterogeneous Fleet Vehicle Routing Problem
MDVRP	Multi-Depot Vehicle Routing Problem
PSO	Particle Swarm Optimization
PVRP	Periodic Vehicle Routing Problem
RVRP	Rich Vehicle Routing Problem
SA	Simulated Annealing
TRP	Traveling Repairman Problem
TS	Tabu Search
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPPD	Vehicle Routing Problem with Pick-up and Delivery
VRPTW	Vehicle Routing Problem with Time Windows
WCSS	Within-Cluster Sum of Squares

1 Introduction

Often, when talking about logistics and transportation, the concept of last mile delivery comes up, which is a specific logistics phase that can be defined as the final step of a delivery process. Contrary to what one might think, last mile delivery is not always a short-distance delivery but, rather, it refers to a specific service.

The logistics sector has grown significantly in recent years, thanks to e-commerce and online purchases. More and more people prefer to buy a product online rather than visit a physical store. Every day millions of deliveries take place to businesses and private customers. Each individual delivery is the result of a highly complex process that starts from the manufacturing company and enables the delivery of products of any kind to retail points or private customers. Basically, the manufacturing company physically produces the product, then the goods are moved from the factory to one of the logistics hubs, and finally, the last-mile delivery is carried out. The process may seem very simple, but in reality, there are many complexities, as well as potential issues that may arise.

Among the main benefits of an efficient last-mile delivery, there are the capability for a company to get and retain customers since a quick and reliable delivery service delights clients, and the reduction of time and resources required to carry out the delivery. All this is possible only if a careful and strategic planning of routes followed by vehicles is performed in order to cut transportation expenses and increase the satisfaction of customers.

The last mile delivery is almost always carried out using road transport, typically with the deployment of vehicles like trucks, vans, or similar means. These vehicles are well-suited for traveling through urban and suburban areas to reach the final destination, be it a retail store or a private customer's address. The flexibility and accessibility of road transport make it the preferred choice for efficiently completing the last leg of the delivery process.

It is indeed transportation that plays a crucial role for companies today in the last mile logistics. The goal of transportation management is to connect all the pick-up and delivery points in the supply network while complying with time constraints demanded

by customers, within the limits of the distribution infrastructure's capacity, and at the lowest possible cost [1]. Hence, it is evident that the geographical component is a fundamental aspect in this type of problems and, in most of the cases, the routes built for drivers are the result of an analysis of the customers locations in the attempt to travel the shortest distance possible and thus lower transportation costs. Once the positions of customers have been studied, the critical issue consists in selecting the optimal routes for vehicles, resulting in a routing problem referred to as Vehicle Routing Problem (VRP).

A VRP is said static when the positions of customers to be served is known by the dispatcher before the construction of the routes and, once vehicles have left the depot, routes are not recomputed [2]. However, in real-word situations requests arrive in different moments of time before the departure of vehicles from the depot but all solution methods available in literature for the static VRP do not make a distinction among customers based on their arrival time. Even if these new customers enter the problem in a second moment, they can be still labelled like known customers if they appear before the departure time of vehicles.
In this thesis, new solution methods that incorporate this concept of adjusting routes based on new customers entering the problem in different moments of time are introduced. Intermediate solutions are computed and continuously changed in order to find the best solution for vehicles at the moment of departure. The last solution before the departure time of vehicles represents the routes followed by drivers to visit all customers. The problem still falls in the category of static VRPs since, even if new customers enter the problem in different moments, all of them appear before the departure time of vehicles. Hence, they can be still labelled like known customers.
In particular, in this document four different algorithms are proposed. Two different clustering approaches for the management of customers locations (i.e., the k-means method and the affinity propagation method) and two different solution methods (i.e., the insert method and the rebuild method) for the planning of routes have been implemented, giving rise to four different Python-based algorithms to solve the vehicle routing problem (i.e., k-means and insert method (K-I), k-means and rebuild method

(K-R), affinity propagation and insert method (A-I), affinity propagation and rebuild method (A-R)). All of them are characterized by an initial analysis of customers positions through the use of clustering methods and then, from it, the best routes for vehicles are planned. In literature the most common clustering method used to group customers in a VRP is the k-means method. In this work, the k-means method is also implemented in two of the four algorithms but the way routing construction is performed is different from literature since routes are adjusted during time until a final solution is found. Instead, in the other two solution techniques a second clustering approach called affinity propagation method is applied. The use of affinity propagation to group customers in clusters in the context of VRP is almost absent in literature and, for this reason, this document wants to focus on this technique.

Each pair of algorithms based on one of the two clustering methods distinguishes by the way in which the solution is calculated. In particular, in the insert solution method clusters are defined once at the beginning and, every time a new request arrives, it is included in these already established clusters and routes are recomputed. Hence, it is a method that finds the last solution to serve all customers by adjusting previous routes found by considering a smaller amount of customers. The final solution when all requests are known is affected by the clusters found at the beginning when only a fraction of requests is available. Instead, in the rebuild solution method clusters are defined every time new requests appear in the system and routes are recomputed according to new clusters. So, the last solution when all requests are known is not influenced by clusters built at the beginning considering a smaller amount of customers. This method simply recomputes again the clusters when all requests are available and routes are found based on it. Hence, the rebuild solution method can be seen as the classic clustering method used to solve VRPs in literature since, similarly, the last moment before vehicles leave the depot could be waited for and the clustering phase would be carried out only once at the last when all customers are known.

Hence, the first contribution of this thesis consists in introducing the concept of time also in static VRPs to reflect real-world situations in which customers call for service in different moments of time, differently from all solution methods available in literature for the static VRP. The impact of time is evident in the insert solution method since

the number of formed clusters and the way they are created depend only on customers that appear at time t_0 in the problem, regardless of whether a better grouping of customers and consequently better routes could be found by considering the total amount of requests at the end. However, the crucial constraint is that all customers enter the problem before the departure time of vehicles from the depot, otherwise the problem falls in the category of dynamic VRPs (this variant of VRP will be explained in detail later in the document). Then, the second contribution is the use of affinity propagation method in VRPs to group customers in clusters. In this case the aim is to assess how the affinity propagation technique performs compared to a widely established clustering method in the context of VRP like the k-means method.
Subsequently, a simulation of a numerical example placed in the city of Turin is conceived and the four algorithms (i.e., K-I, K-R, A-I and A-R) are applied to assess which is the best method in terms of shortest distance traveled, expressed in kilometers.

In particular, the sections of the work are organized as follow:

- Chapter 1: it is the current section where an initial introduction to the purpose of the thesis is provided. It also gives an overview of the work and how it is organized.

- Chapter 2: it is a literature review in which the definition of vehicle routing problem is outlined. Then, the most studied variants of the vehicle routing problem with their main features are explained.

- Chapter 3: it is entirely focused on the most common solution methods adopted in vehicle routing problems. It makes a distinction among exact, heuristic and metaheuristic techniques.

- Chapter 4: it illustrates how the four conceived algorithms work. First, a description of the two clustering methods used to divide customers in groups is shown. Then, the two methods adopted to plan routes is provided. The combination of the two clustering methods with the two solution techniques gives rise to the four different algorithms.

- Chapter 5: it describes the example placed in the city of Turin and how the algorithms have been applied. Finally, the obtained experimental results are analyzed.

- Chapter 6: it is the final chapter of the document in which final considerations are given. Moreover, possible future extensions of algorithms are outlined.

2 Vehicle Routing Problem and its variants

2.1 Historical perspective

One of the first combinatorial optimization problems to be widely studied is the Traveling Salesman Problem (TSP). The problem consists in determining the shortest distance that allows a traveling salesman to visit a set of cities once and come back to the departure city [3]. The TSP can be applied in different fields such as the logistics, the production and the vehicle routing problem. Indeed, the TSP and the VRP are two extremely correlated optimization problems which belong to the category of routing and scheduling problems.

"The Truck Dispatching Problem", written by Dantzig & Ramser in 1959, was the first study to introduce the concept of vehicle routing problem. A procedure based on a linear programming method was developed to find the minimum distance travelled by a fleet of homogeneous trucks to dispatch oil in a number of gasoline stations [4]. Subsequent years witnessed the rise of numerous works based on diverse principles, encompassing savings, geographical proximity, customer matchings, alongside steps to enhance both intra-route and inter-route aspects [5]. Among all researches performed, the most promising study was the one proposed in 1964 by Clarke & Wright who were able to extend the linear optimization problem to the logistics and transport sector. They analyzed a network made up by a central depot and a set of delivery points geographically dispersed around it, each of them served once and by a fleet of trucks with varying capacities [6].

However, modern VRP models diverge significantly from the initial formulations presented above. This is due to their growing focus on integrating real-world complexities. For example, they incorporate factors such as travel times influenced by time-dependent factors like traffic congestion, time constraints for both pickup and delivery, and dynamic changes in input data like demand information. Incorporating these aspects introduces considerable intricacy in the problem [7].

2.2 Definition of VRP

The Vehicle Routing Problem (VRP) is a typical operational challenge in distribution networks which relates to the determination of routes for a set of vehicles to serve a group of customers.

Starting from one or more central depots, a set of vehicles with specific requirements need to visit a set of customers, also with specific characteristics. Customers are typically dispersed in a geographical network and the problem is about coming up with an optimal route to serve all customers at the lowest possible cost.

Every customer order possesses a distinct location and the magnitude of order may vary from customer to customer. The objective is to devise the most efficient routes in which each customer is visited once and, at the same time, capacity constraints are met. The task entails determining the optimal sequence for visiting these locations, ensuring that each location is served by only one vehicle (see Figure 2.1).

The constraint is that the cumulative demand of all customers along a specific route does not surpass the vehicle capacity. In addition, another crucial assumption is that each vehicle starts and ends the route at the source (i.e., the depot).

Potential applications of the VRP are frequent in detailed logistics and distribution issues. For instance, one can envision scenarios where goods must be distributed (or collected). Real-world applications often introduce additional constraints that can complicate the structure of the problem. In this chapter, an overview of the different versions of VRP with their operational constraints is provided. Finally, some applications of VRPs in real life are illustrated.

Every variant of VRP is characterized by a similar solution approach, which entails satisfying customers within the following constraints [8]:

- each route starts and ends at the depot;
- each customer is served by a vehicle only once;
- the total demand of each route does not outperform the total capacity of the vehicle.

The network where depots and customers are located can be easily represented

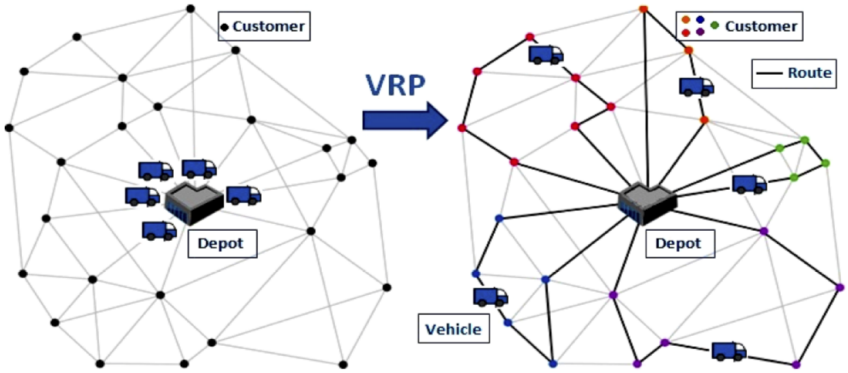

Figure 2.1: Classical VRP (source: Gupta et al. [9])

through a graph $G = (N, A)$ in which $N = \{0, \ldots, n\}$ are the points that vehicles must reach. Usually, in case of a single depot, the point $\{i = 0\}$ corresponds to the depot location while all the other nodes $i = \{1, \ldots, n\}$ describe the customers dispersed in the network. Arcs $A = \{(i, j) : i \neq j \wedge (i, j) \in (0, \ldots, n)\}$ represent the roads that connect each pair of points. Each of them has a cost c_{ij} that, in many cases, corresponds to its length. Arcs can be directed or undirected depending on if the road can be traveled in only one or both directions. The amount of goods demanded by each customer i is described by the variable q_i. Finally, the last actor in the problem is the fleet of vehicles $V = \{1, \ldots, v\}$. Each of them has a fixed capacity that usually reflects the number of goods carried out. The route travelled by each vehicle starts and ends in the depot location.

2.3 Variants of the VRP

Starting from the classical VRP, a number of variants have been studied to increase the complexity of the problem and bring research closer to the reality. Each of the VRP versions that are described below includes additional features that lead the problem to slightly deviate from the simple VRP proposed in the above section:

- *The Capacitated VRP (CVRP)*: it is the classical VRP in which each customer has a specific demand for the good and vehicles have finite capacity. All vehicles have the same characteristics and the network presents only one central depot [10]. In this case, the constraint must be updated to take into account the

different demands of each customer and not exceed the finite capacity of vehicles.

- *The Distance Constrained VRP (DCVRP)*: it is assumed that vehicles have a limited autonomy in terms of kilometres travelled [11]. This entails that each specific distance between two nodes must be considered in order to not exceed the total capacity of vehicles in terms of kilometres. If a vehicle can not serve the following node, it must come back to the depot and let another vehicle visit it. This restriction can either substitute the capacity constraint of the CVRP or integrate it.

- *The VRP with Time Windows (VRPTW)*: in this variant of the VRP, the concept of time becomes fundamental. Routes of each vehicle follow a time-table since deliveries to a given customer must occur in a specific time frame, which is different for each customer in the network. There are two types of time windows. When the delay in the delivery to a given point does not involve a penalty cost, time windows are called soft. Instead, when scheduled time must be strictly respected, time windows are referred to as hard time windows. If a vehicle visits the customer too early, it must wait until the time window opens and it is not permitted to arrive when the time frame is expired [12]. In this case, a time window is assigned to each customer with a starting time when the client can be visited and a maximum time when he can be served. Concerning the vehicle, it is added a travel time that describes the needed time to serve the customer. Finally, the starting time when a vehicle starts to serve a specific customer is considered in this variant.

- *The Heterogeneous Fleet VRP (HFVRP)*: it is another popular extension of VRP, also called Mixed Fleet VRP, that considers vehicles whose capacities vary [13]. The constraint about the equal maximum quantity that all vehicles can carry out is replaced by the new condition that reflects different capacities of vehicles.

- *The VRP with Pick-up and Delivery (VRPPD)*: in this type of VRP, in addition to the delivery of goods, vehicles must also to pick up amounts of the good from customers. Each customer in the network might have either a delivery request or

a collection request or both. Vehicles collect goods from the central depot that are delivered to customers but, simultaneously, collect goods from some clients which are supplied to other customers or returned to the depot [14]. Hence, in this variant a distinction between demand and supply of customers is necessary. Moreover, additional inclusions about the load of vehicles after leaving nodes, the remaining demand after leaving nodes and the remaining supply after leaving nodes are made to guarantee the load of each vehicle and the requirement of not exceeding the maximum capacity of the vehicle. It needs that the remaining demand can be handled by the load and the remaining space in each vehicle.

- *The VRP with Backhauls (VRPB)*: it is a VRP with pick up and delivery with the difference that on each route all deliveries take precedence over any pickups. Customers can be grouped in linehaul and backhaul customers. Linehaul nodes are customers that require deliveries by vehicles while backhaul nodes are points where vehicles pick up goods from customers. The crucial constraint of this variant is that in any route all linehaul customers must visited before than all backhaul customers [15]. Sometimes, in this model it is assumed that routes containing only linehaul nodes are not allowed.

- *The Multi-Depot VRP (MDVRP)*: it is a VRP extension in which the network is characterized by more depots spread among customers [16]. This VRP may require either that all vehicles leave and return to the same depot or not.

- *The Periodic VRP (PVRP)*: in this version of VRP, a planning horizon of multiple periods is considered and the objective is to minimize the total travelling cost provided that each node in the network can be visited by vehicles only in a specific set of possible visit schedules [17]. Vehicle routes must be organized over multiple days and a schedule is defined as a set of days of the planning period in which customers can be served. Allocating a node to a schedule implies that the node will receive goods in every day of that schedule [18]. Hence, by taking into account the number of visits required by each customer during the planning period, a specific schedule is defined for each of them.

- *The Dynamic VRP (DVRP)*: it is one of the most complex variants of VRP. Basically, the distinctive feature of DVRP is that not all relevant inputs and information for the resolution of the problem are available at the beginning of the routing process [2]. Routes are recomputed and changed dynamically as soon as new data is known. This means that the network is described by both constantly changing data and data known in advance. Consequently, the model is solved frequently as new information enters it and huge computational resources are needed [19]. A graphical representation of a simple dynamic routing problem and the required real-time communication scheme are shown. These examples are depicted, respectively, in Figure 2.2 and Figure 2.3. In the former, there is a single vehicle starting its route in the depot at time t_0 and it must serve all known customers (A, B, C, D, E). During the travelling of the route, at time t_1 the vehicle receives two new requests (X, Y) and the previously defined route must be adjusted to visit new customers. Finally, at time t_f a new different route is executed (A, B, C, D, Y, E, X).

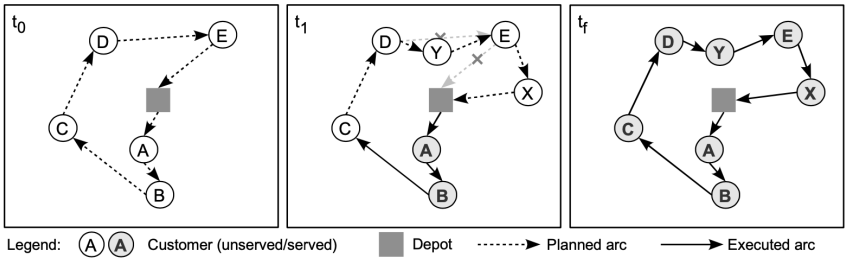

Figure 2.2: A simple dynamic vehicle routing problem (source: Villac et al. [20])

In the second figure, the needed real-time communication that allows the dynamic adjustment of routes is explained. The environment represents the real world where the vehicles move while the dispatcher is the centralized power that gives guidelines to the drivers. As soon as the vehicle is ready to leave the depot, the dispatcher instructs the driver with the next customer to be visited (A). Once the vehicle starts and finishes to serve the customer A, it notifies the dispatcher that, in turn, makes a new decision about the next customer to add in the route.

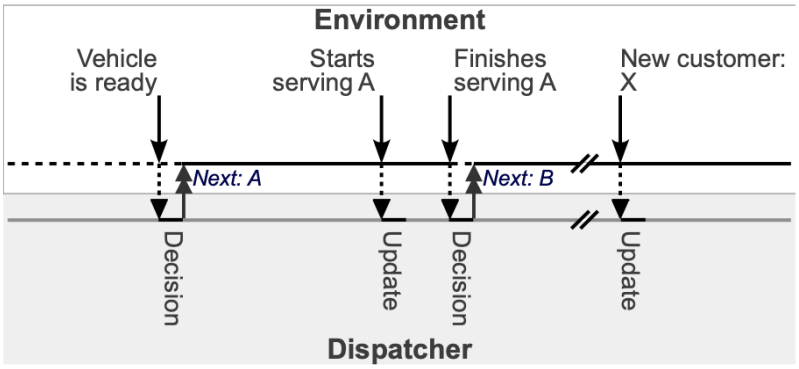

Figure 2.3: The real-time communication between dispatcher and vehicle (source: Villac et al. [20])

Nowadays, most of these variants are merged in a unique instance of VRP to form a set of problems known as Rich VRP (RVRP), where multiple constraints are included to study a complex problem that can embody aspects of real-life [21].

2.4 Examples of VRP in real life

The applications of vehicle routing range from commercial to non-commercial scenarios. Basically, three main types of routing problems can be identified:

- *Transport of goods*: The most common example of transport of goods is the work of a courier delivering packages in urban areas. These packages can be either document or goods. Once the customer position is known and the request has being accepted, the courier is sent to pick-up the good that, in turn, will be delivered to another location [22]. Other typical example of this category are food delivery services or e-commerce.

- *Transport of people*: The transport of people is similar to transport of goods but it is characterized by additional constraints such as waiting times and travel times for passengers. A classical scenario of transport of people is represented by taxi services. Clients can be known in advance but, most of the cases, a high percentage of them is revealed dynamically during the working day. A taxi must be allocated in the shortest time and typically the closest free driver is the

one that takes the ride [23]. Another evidence of this category is the Dial-a-ride problem which focuses on the online planning of routes for passengers, instead of goods [24]. All on-demand transports such as the transportation of elderly and handicapped people, the transportation of children, for instance from home to school, or patients transport are included in this type of routing problem.

- *Services*: The dispatching of emergency services (e.g., ambulances, police or fire services) implies a careful planning of routes. The response time is a crucial aspect and the time between the call and the arrival time must be the shortest possible. In addition, an adequate coverage of the service area and the avoidance of traffic congestion must be granted [25]. Other real-life situations can be the maintenance operations (e.g., for heating systems, elevators or other machines). Companies have contracts with clients and send technicians for periodical and planned maintenance visits. This problems is called Traveling Repairman Problem (TRP) and it is one of the most studied routing problems [26].

Nowadays, as shown in Figure 2.4, 32% of papers study the transportation of goods, 14% are about services, 14% about the transportation of people and last 20% of articles study generic applications [27].

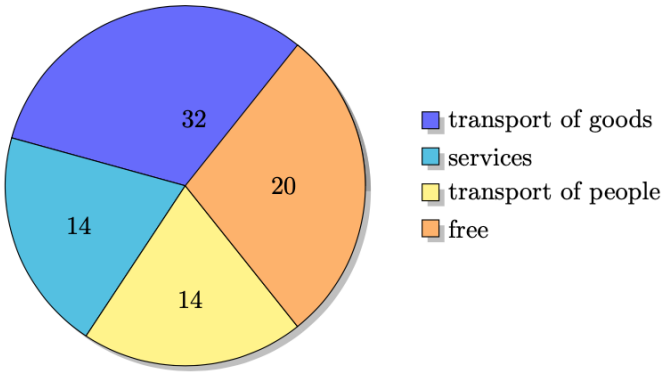

Figure 2.4: Division of articles according to the application analyzed (source: Rios et al. [27]

3 Solution methods for VRPs

After an introduction about the vehicle routing problem and its variants by outlining constraints involved in the construction of routes for a fleet of vehicles that must serve a group of customers, in this chapter the focus shifts to the existing solutions method adopted to solve the vehicle routing problems. Before starting with any kind of consideration, it is important to emphasize that the complexity of the VRP rapidly grows with the number of customers, the diversity of vehicles and the specific constraints of the problem.

An equilibrium between the search for an optimal solution and the limitation of computational resources should be considered for the selection of a proper methodology to solve the VRP in an efficient way. Basically, the solution methods can be grouped in three main categories: exact methods, heuristic methods and metaheuristic methods.

3.1 Exact methods

Exact methods are adopted in VRPs when the accuracy of the solution is required since the key characteristic of algorithms belonging to this category is their capability to reach the optimal solution.

3.1.1 Branch and Bound

The Branch and Bound (B&B) method is one of the most common exact methods used to solve small instances of the VRP. The main idea behind this technique is the division of the original problem in several sub-problems and the method focuses on the resolution of each of these sub-problems [28].

For each node all feasible solutions are evaluated and an upper bound and a lower bound are computed. The upper bound represents the most promising solution found till now while the lower bound is a lower estimation of the optimal value. At each iteration, if the lower bound of the node is larger or equal to the upper bound of the best solution found till now, the node is pruned because the integration of the node will lead to a feasible solution that is worse than the one found till now. When no unexplored node has a smaller bound than the length of the current best solution, this

best solution is optimal [29].
Hence, if the upper and the lower bounds are correctly defined, the branch and bound method leads to the optimal solution of the problem through the elimination of not promising nodes.

3.1.2 Branch and Cut

The Branch and Cut (B&C) algorithm is an extension of the branch and bound method in which, like in the previous technique, the original problem is described by several sub-problems but the set of feasible solutions is further restricted through the use of cutting strategies [30]. Similarly to the branch and bound method, it provides an optimal solution for small instances of the VRP. However, through the use of these cutting techniques which are specific to the problem under examination, this method reduces the amount of solutions in which the optimal one is chosen and achieves it more quickly.

3.1.3 Column generation

The column generation technique tries to solve the VRP by dynamically analyzing the feasible routes for the problem.
Instead of finding a solution in which all variables are considered since the beginning, it solves a reduced instance of the problem that involves a limited set of variables. Subsequently, the method starts to progressively include new variables to check if the solution found for the reduced problem can improve or not [31]. Hence, an estimate of the optimal solution is found by solving an initial basic problem and then the method identifies routes that can improve the solution by adding new columns, each representing a feasible route.

3.1.4 Dynamic Programming

The Dynamic Programming (DP) method is another technique that finds a solution for the original problem by dividing it in a group of small problems that can be easily addressed.

First the best routes for these small instances of the problem are built and then the final optimal solution is computed by combining all results achieved [32]. Basically, this method follows a bottom-up approach (i.e., from 'small' problems to the 'big' problem) and build the optimal solution by exploiting routes determined in smaller problems.

However, the pursuit of an optimal solution in problem-solving typically involves mathematical approaches. Constructing a representative mathematical model in problems where input data vary with time or stochastic factors is not always feasible. Occasionally, due to data complexity and problem scale, a mathematical model might yield the optimal solution but over an extended period of time [33].
VRP is an NP-hard problem and exact algorithms are only suitable when the data size of the problem is small since the required solution time increases exponentially with the number of nodes [34]. These methods explore the whole space of solutions to find the optimal one and, when the problem scale is large, they are impracticable. Moreover, the insertion of more constraints in the problem, such as time windows or limits in the capacity of vehicles, makes the mathematical formulation of the problem and its resolution very hard.
As a result, when the dimension of the problem is large, the computational resources are limited and a good enough solution is more than sufficient, other approaches to VRPs are preferred. These methods are referred to as heuristic and metaheuristic methods.

3.2 Heuristic methods

When the computation of the optimal solution is not necessary or the complexity of the problem is such that the time required for the optimal resolution is not reasonable, it is preferred the use of heuristic methods.
In practice, these algorithms solve VRPs by guaranteeing a near-optimal solution in acceptable computational time but they are not able to reach optimality [35]. If the search for a solution is computationally expensive, a feasible solution that gets closer to the optimal one is adequate.

The choice of the algorithm depends on the complexity of the problem, the available computational resources and the type of the solution pursued. In addition, in most of the cases, the heuristic approach is specific to the VRP under consideration and it may be inadequate to solve a different one.

3.2.1 The saving method of Clarke and Wright

The saving method of Clarke and Wright is one of the most known heuristic approaches developed. The main concept behind is the computation of cost savings obtained by combining two nodes on the same route, rather than serving them separately. This cost saving is defined as the difference between the cost of serving the two customers separately and the cost of serving them together. First, let i and j be two customers and (x_i, y_i) and (x_j, y_j) respectively their geographical locations, the cost distance matrix consists of Euclidean distances $(c_{i,j})$ expressed by the following equation [36]:

$$c_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1}$$

Then, the savings value between the two customers i and j is calculated as:

$$s_{i,j} = c_{1,j} - c_{i,j} \tag{2}$$

where $c_{1,j}$ represents the distance between the depot and the customer j while $c_{i,j}$ is the distance between customers i and j.
Subsequently, the equation (2) is modified and a list with all savings values sorted in a decreasing order is built in the following way:

$$s_{i,j} = c_{1,i} + c_{j,1} - c_{i,j} \tag{3}$$

At this point, the construction of routes takes place and it starts from the top of saving list which corresponds to the largest $s_{i,j}$. If all constraints involved in the VRP under consideration are respected, the two customers i and j are included in the same route. Usually, the solution offered by this method is an initial reasonable solution that needs further improvements through the application of other heuristic approaches to become more accurate.

3.2.2 The sweep method

The sweep method is another heuristic approach used to solve the VRP. The key feature of this technique is the sweeping around the depot of all nodes in the network to build the routes. In other words, customers are assigned to a route based on the angle that is formed by considering their position in relation to the depot [37].
In the first part of the algorithm, nodes are divided in clusters and clustering is carried out by joining in the same group the closest node to any node that is selected as the first one in the cluster. The closest node is the one that is found with the smallest angle. This procedure goes on by including the second closest, the third and so on until the constrains are violated. Indeed, if adding the next closest node to the cluster leads to a total demand higher than the capacity constraint of the vehicle, this node is rejected and becomes the first node of the second cluster. When all nodes belong to a cluster, routes for each cluster formed are computed.
Figure 3.1 shows how the sweep method works to build clusters.

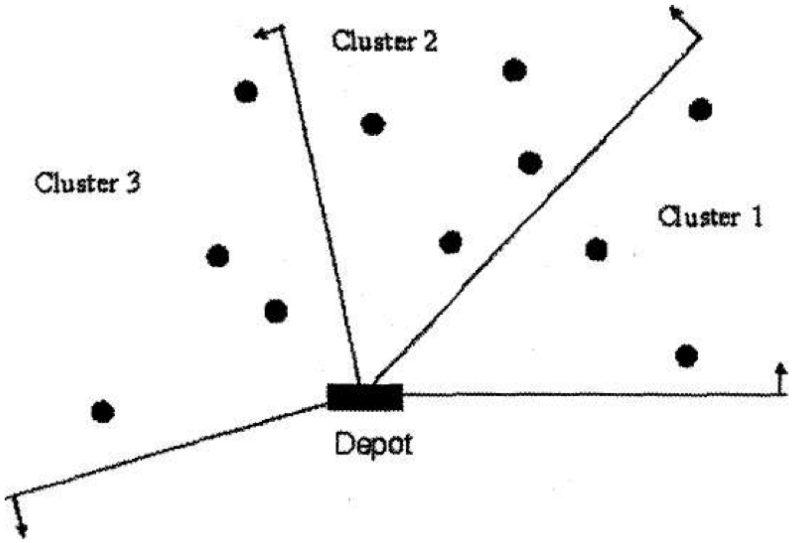

Figure 3.1: Clustering process in the sweep method (source: Nurcahyo et al. [37])

Like the saving method of Clarke and Wright, this procedure may not lead to highly qualitative solutions. However, the sweeping strategy can be useful to build routes for customers in a simple and fast way.

3.2.3 The neighborhood methods

The last described heuristic methods are the algorithms that exploits the concept of neighborhood. Through the local search of neighbors, these procedures are capable to modify the previously established routes for vehicles and the achievement of a good solution is guaranteed.
The composition of neighbors can be changed by carrying out one of the following elementary actions or through a combination of them [38]:

- *Insert action*: it randomly adds a node of the network to the route of a vehicle if the incremental cost is the lowest;
- *Remove action*: it randomly removes a node belonging to the route of a vehicle;
- *Relocate action*: it randomly relocates a customer that it is removed from the route of a vehicle and inserted to the route of another vehicle;
- *Replace action*: it randomly removes a customer from the route of a vehicle and it is inserted in the route of another vehicle by taking the place of another node;
- *Swap action*: it randomly removes two customers from their routes and inserts them to the respective other routes.

Now that all allowed operations adopted to change the structure of neighbors are presented, different algorithms can be conceived by using them or a combination of them.
Subramanian et al. [39] distinguished between methods that make changes within the same route of neighborhoods and methods in which variations occur between routes of the same neighborhood (see Figure 3.2). Most of these approaches are based on the concept of λ-interchanges that consists in exchanging up to λ nodes between routes. However, a short description of the most common methods is provided:

- *Exchange-based neighborhood*: this approach consists in exchanging customers between routes traveled by different vehicles in order to improve the solution;
- *2-opt neighborhood*: this method selects two nodes within the same routes and their arcs are changed to rebuild the route with a different order;

- *Or-opt neighborhood*: in this method a route is modified by choosing a node and serving it in a different order;
- *Reverse-based neighborhood*: this approach searches for better solution by reversing the route direction that means serving customers of the route in the opposite order;
- *Cross-exchange neighborhood*: it is a method in which segments of different routes are exchanged.

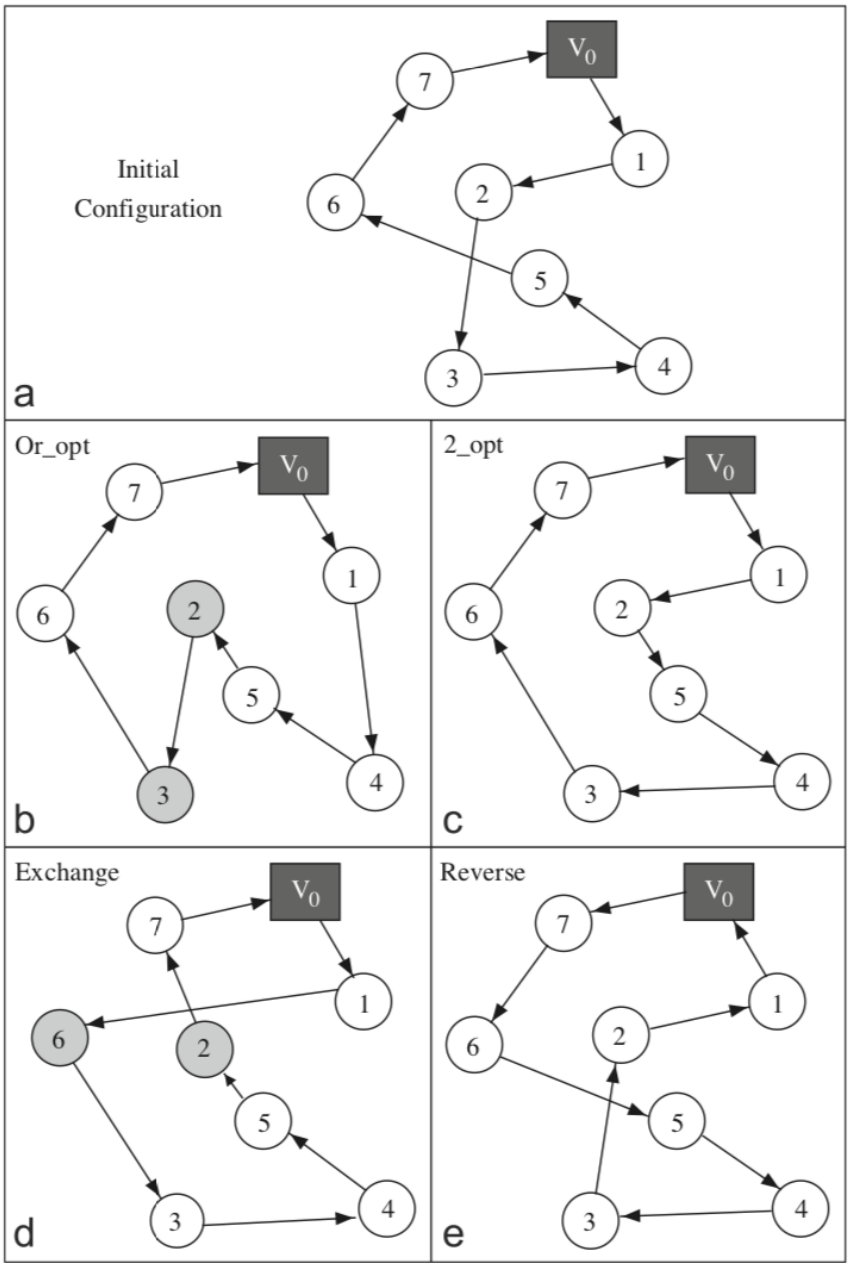

Figure 3.2: Neighborhood methods.

3.3 Metaheuristic methods

The last category of methods used to deal with VRPs is represented by metaheuristic approaches.

They are more advanced algorithms that offer wider optimization to solve VRPs compared to heuristic approaches since they are more versatile methods that can be applied to any combinatorial optimization problem, irrespective of its structural specifics and

constraints. Basically, they combine different heuristic approaches to study a wider set of solutions but significant computational effort is required.
The three distinctive features of these methods are the capability to explore the space of solutions for the VRP in a more efficient way, the flexibility to address different variants of VRPs respecting their specific constraints and, finally, the continuous improvement of current results to reach a highly qualitative solution that gets closer to the optimal one in reasonable timing.

3.3.1 The Tabu Search

The Tabu Search (TS) algorithm has been introduced for the first time by Glover [40]. The original idea behind this method is the construction of a tabu list that collects all operations already used in the improvement of a solution. Through the exploitation of this list, the algorithm avoids the future implementation of worthless actions if it was already proved that they can not lead to a better solution.
At each iteration, the best move for the current solution is selected but it is implemented only if it is not in the tabu list. After this new action is performed, it is inserted in the list and it can not be repeated again. On the other hand, if the best move chosen is already inside the tabu list, it can not be used and the next operation is carried out. Figure 3.3 outlines the main steps followed by this procedure.

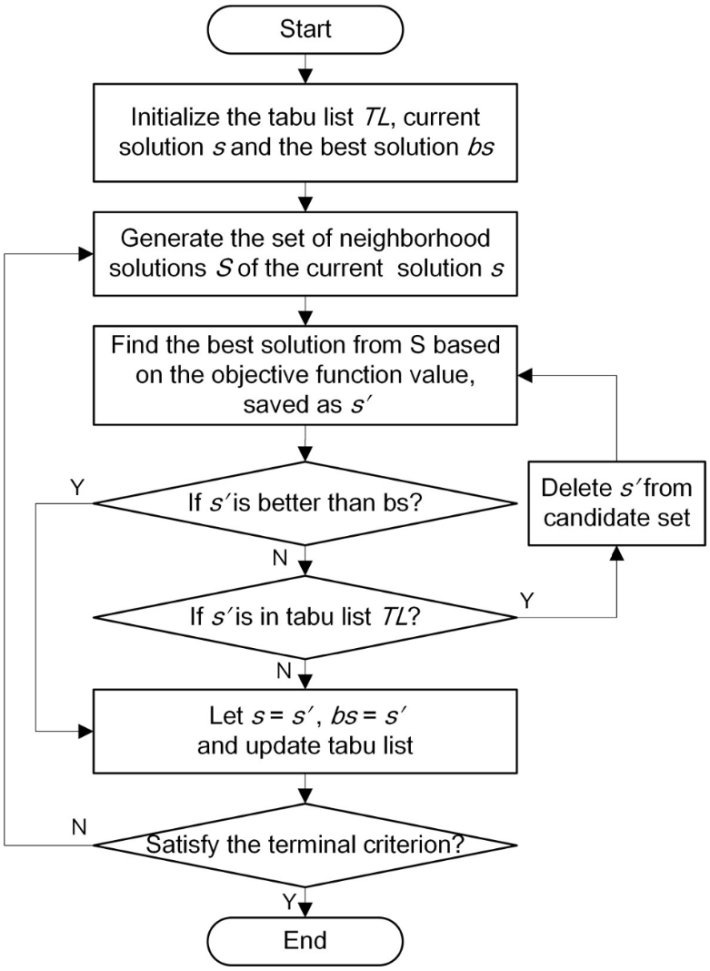

Figure 3.3: The Tabu Search procedure (source: Peng et al. [41])

3.3.2 Simulated Annealing

The application of Simulated Annealing (SA) method derives from the thermodynamic field and, in particular, from the annealing procedure for the modeling of materials. Basically, the first step of this phenomenon consists in heating a metal until it reaches the liquid state while, in a second moment, it is slowly cooled into a recrystallized solid state to reduce imperfections.

At the same way, this method can be adopted in a large number of combinatorial optimization problems and one of the first authors to apply it to VRPs was Osman [42].

The SA technique generates an initial routing solution and, at each iteration, accepts new solutions near to the current one based on a probability of acceptance. At the

beginning it is more likely that poor solutions are permitted but, going forward with the process, the probability to select worse solutions reduces gradually. The crucial metrics for the functioning of this process are the initial temperature which determine how much the algorithm is willing to explore worse solutions at the beginning and the cooling rate that represents the speed with which this temperature is decreased. By reducing this temperature, the probability to accept poor solution lowers and, as a consequence, the process begins to find better and better results.

3.3.3 Guided Local Search

Another broadly diffused metaheuristic approach is the Guided Local Search (GLS) method. The principle of this method is to guide the local search for a qualitative solution in most promising search space through the exploitation of the information known about the VRP. For this purpose, penalties that increase the cost function are introduced in the VRP and the algorithm avoids regions of solution where these penalties are very high. If the local search gets stuck in a locally optimal solution, the algorithm calls again the local search to further improve the cost function reached [43]. This is possible modifying penalties for the solutions found by the local search when the results achieved by the method are not acceptable.
An illustration of the procedure carried out by the GLS technique is shown in the Figure 3.4.

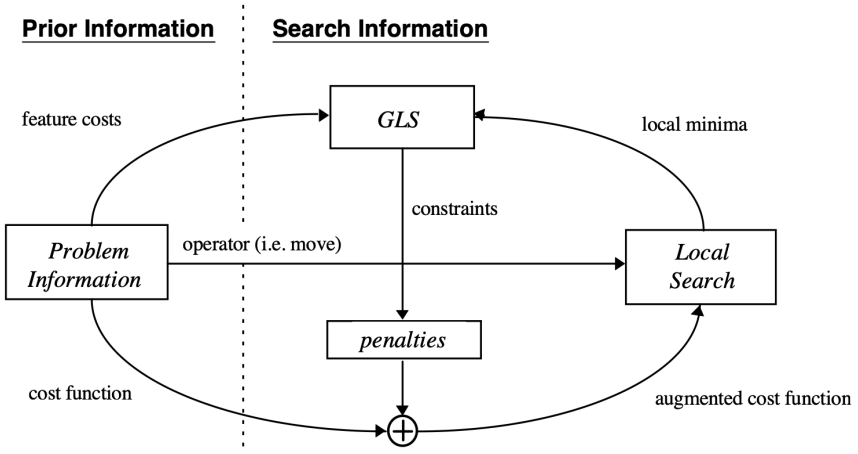

Figure 3.4: The Guided Local Search method (source: Voudoris et al. [43])

3.3.4 The Genetic Algorithm

This Generic Algorithm (GA) takes inspiration from the evolution process of Darwin. The initial computed routes are modified by applying operations of cross and mutation in order to stimulate the evolution of these routes.

The aim is always to find acceptable solutions in terms of distance traveled or other cost metrics. If good solutions are generated, they reproduce to create new improved solutions, while less promising solutions go towards extinction [44]. Once reasonable routes for vehicles are detected, they are kept and labelled as parents since the next generation of routes is created starting from them.

The improved solutions are the result of both mutations in which initial routes are randomly changed and cross operations in which characteristics of parents are merged to give rise to new offspring [45]. When the new population of solutions is consistent with the termination criteria set at beginning of the process, that means the minimum acceptable requirements for a solution to be acceptable are reached, the GA ends.

Figure 3.5 provides a graphical representation of this algorithm.

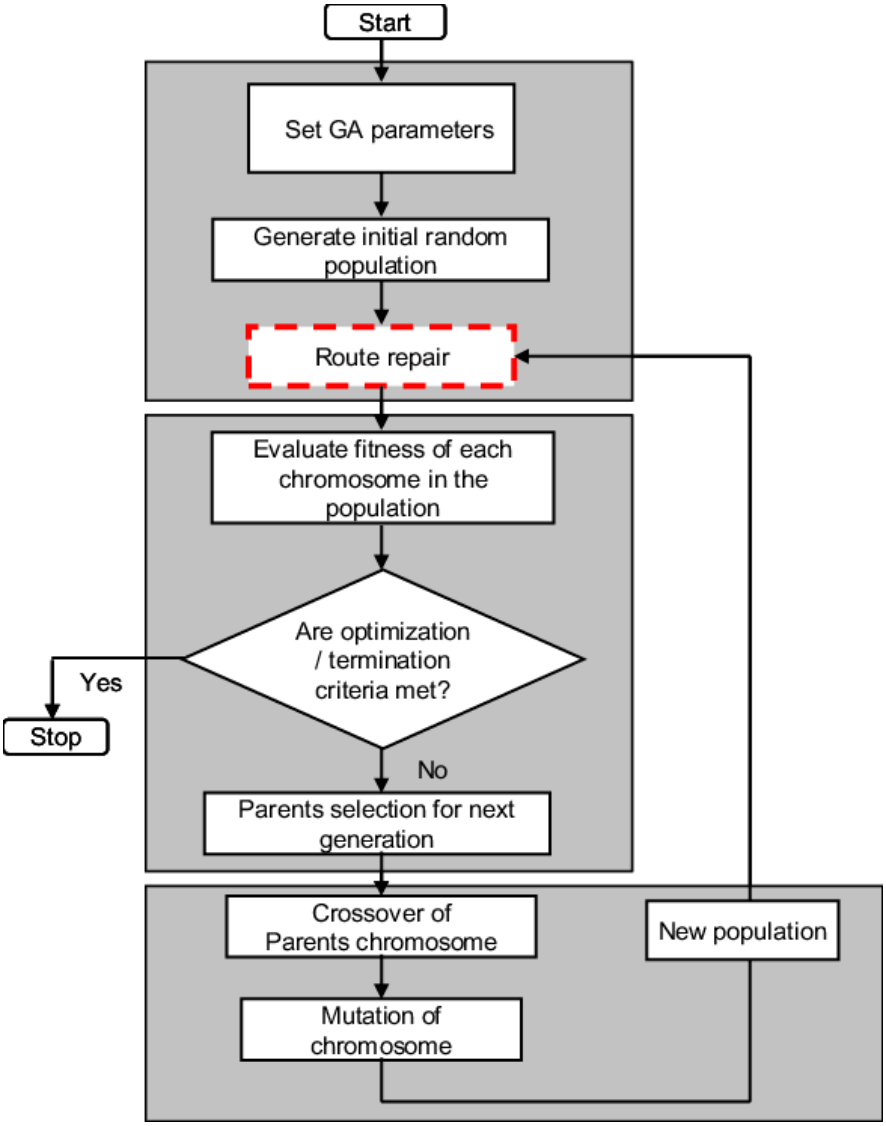

Figure 3.5: The Genetic Algorithm (source: Razali [46])

3.3.5 Ant Colony Optimization

The Ant Colony Optimization (ACO) method is another famous approach adopted to address complex VRPs. Basically, this method is based on the principle of indirect communication between artificial ants that, through the release of pheromones, detect very good solutions near to the optimal one for the VRP under examination. The concept stems from the real behaviour of ant colonies in nature that start from their nest to look for food. On the way, they leave pheromones to attract new ants and mark the efficient routes to reach the food [47].

Obviously, the ants conceived for VRPs are fictitious and simply represent vehicles that try to find the best routes to serve all customers in the network. At each iteration, each ant builds a solution and the shortest routes accumulate more pheromones. In this way, the new ants are only attracted by solutions possessing a large amount of pheromones and focus only on these promising routes in the attempt to enhance the solution. Ants release pheromones along the routes and the quantity of these pheromones is inversely proportional to the distance traveled. This implies that shorter routes are characterized by more pheromones.

However, like other metaheuristic methods, the process goes on until the stopping condition is met and selected routes are the ones that have the highest level of pheromone trail.

Figure 3.6 illustrates the main steps of this algorithm.

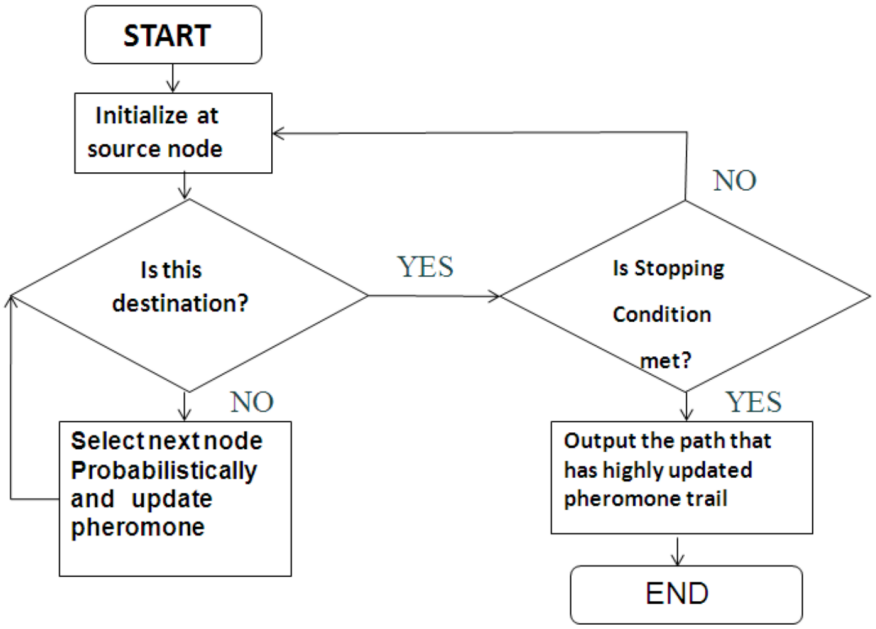

Figure 3.6: The Ant Colony Optimization procedure (source: Ahmmed et al. [48])

3.3.6 Particle Swarm Optimization

The last metaheuristic approach described is the Particle Swarm Optimization (PSO) technique. Like other methods, it is suitable to solve a large number of optimization problems, included the VRP.

Similarly to the ACO method, this algorithm takes inspiration from the collective behaviors of animals such as bird flocking or fish schooling [49]. In particular, PSO leverages a swarm of particles to find a near-optimal solution to the problem. The particles are defined by two main components that are the velocities and the best known position till that moment. The velocity characterizes the movements of particles while the best known position indicates the best solution achieved so far. Through the collaborative search for solutions and the exchange of information about the best position found so far, the particles are able to communicate between each other and converge towards highly qualitative solutions.
Figure 3.7 outlines how the particle swarm optimization works.

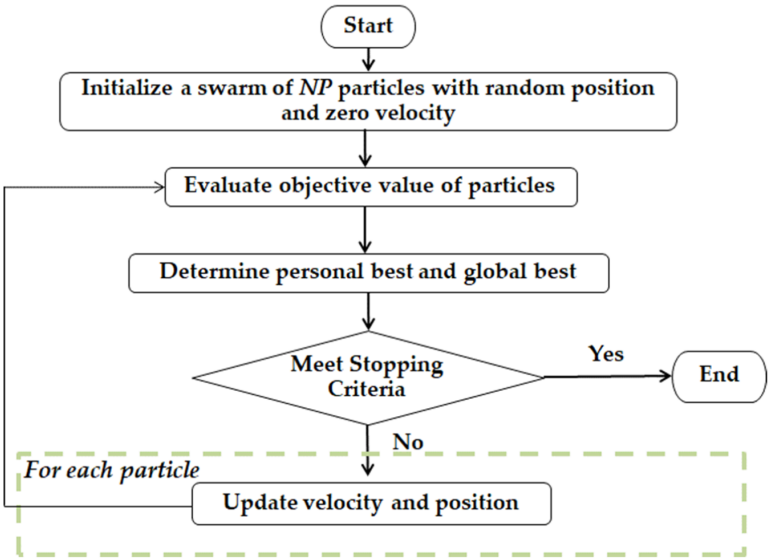

Figure 3.7: The Particle Swarm Optimization method (source: Wisittipanich et al. [50])

In summary, all presented metaheuristic methods became precious tools in dealing with complex VRPs and their constraints. When the scale of the problems is large, although they are not capable to guarantee the optimal solution, they can generate a near-optimal solution in a relatively short time. In addition, the high flexibility allows these algorithms to be deployed and customized in order to address a huge number of VRP variants. Finally, the capability to provide a good solution in fast time makes these methods suitable for real-time applications, like in the case of dynamic VRP.

In the next chapter, the metaheuristic algorithms conceived in this thesis are outlined. As mentioned in the introduction, they apply an initial clustering phase in which customers are grouped based on their geographical proximity and then the routing problem is solved. They differentiate from all other metaheuristic approaches since they find a solution that evolve over time in order to integrate new customers entering the problem. However, the arrival time of new requests is always before the departure time of vehicles from the depot and the VRP is considered static.
Concerning clustering of requests, the affinity propagation method is used to divide customers in groups in two of the four algorithms. Instead, the other two algorithms perform the clustering phase by leveraging the k-means method which is widely popular in VRPs.

4 The algorithms proposed

In this chapter, four metaheuristic approaches to solve static VRPs are proposed. The peculiarity common to all four methods consists in the application of an initial clustering phase prior to the routing phase. Basically, two different clustering techniques (i.e., the k-means method and the affinity propagation method) and two different procedures to build routes (i.e., the insert method and the rebuild method) are implemented by giving rise to four different algorithms.
The k-means method is a widely spread technique in the context of VRP to group customers in groups whereas the application of the affinity propagation method is almost absent in the literature. The four deriving algorithms are called k-means and insert algorithm (K-I), k-means and rebuild algorithm (K-R), affinity propagation and insert algorithm (A-I) and affinity propagation and rebuild algorithm (A-R).
All conceived algorithms are made up by two main sequential phases:

- **Clustering:** Once the orders made by customers are known, methods start the clustering analysis through which the requests are divided into clusters.
- **Routing:** In the second step, the routing construction is performed. All customers inside the same cluster are served by the same vehicle and the VRP inside each cluster is treated as a traveling salesman problem.

In chapter 5, the simulation of a numerical example consisting of a static VRP placed in the city of Turin is conceived in order to test the four proposed algorithms. Then, the results are compared to the solutions found by another metaheuristic approach in order to evaluate their competitiveness in relation to this already established method.

4.1 The clustering phase

The clustering of orders is the first step carried out by the algorithms. In this part the customers are divided into groups based on their geographical positions. The adopted clustering techniques are the k-means method and the affinity propagation method.

4.1.1 The k-means clustering method

The k-means clustering procedure is the first method analyzed to separate nodes in groups. The aim of k-means is clustering data based on similarity by maximizing it for points within the same cluster and minimizing it for points belonging to different clusters. In the case of customers characterized by geographical positions, this method tends to maximize similarity among customers based on the shortest distance between them and the centroid of the cluster [51].

The k-means method requires that the number of clusters k for the dataset must be specified at the beginning. Once the k number of clusters is chosen, k points in the dataset are randomly selected like the current centroids. At this point, the Euclidean distance between all the points of the dataset and the cluster centroids is computed in order to assign each of them to the nearest centroid. Once all distances are determined, all points are assigned to one centroid and k clusters are formed. Now, the new centroids of created clusters are recomputed and again all points of the dataset are reassigned to the nearest new centroids. Again the algorithm builds new clusters for which new centroids can be detected.

This procedure goes on until convergence when the positions of centroids for new clusters do not change anymore. The steps followed by the k-means clustering are illustrated in Figure 4.1.

The k-means method chooses centroids that minimize the total intra-cluster variation which is called total Within-Cluster Sum of Squares (WCSS). It is expressed as:

$$WCSS = \sum_{i=0}^{n} \min_{\mu_j \in C} (x_i - \mu_j)^2 \tag{4}$$

where x_i is a data point that belongs to the cluster center μ_k. Each point x_i in the dataset is assigned to a specific cluster having μ_k as centroid such that the total sum of squares distance between points and cluster centers is minimized.

However, as mentioned before, in k-means algorithm the best number of clusters must be found at the beginning. A famous graphical method adopted to find the k value is the elbow method. This method shows how the WCSS varies by changing the number of clusters k (Figure 4.2 is a general graphical illustration of how the method works).

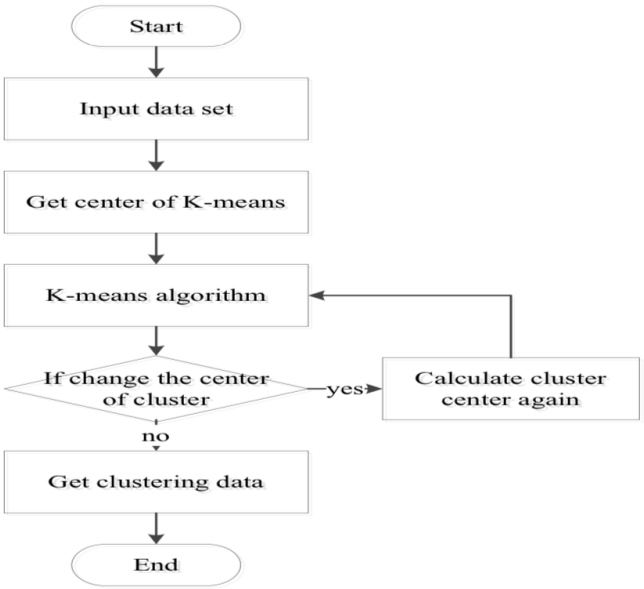

Figure 4.1: K-means algorithm (source: Qiu-yu et al. [52])

In particular, WCSS is inversely proportional to the number of clusters and the WCSS value starts to decrease by increasing k. The graph has the shape of an elbow and the best number of clusters corresponds to the point where the WCSS starts to move parallel to the x-axis.

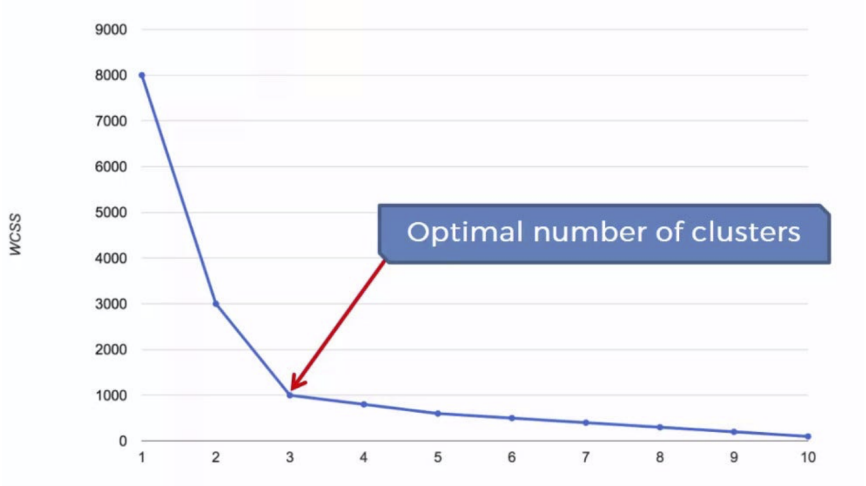

Figure 4.2: The elbow method

4.1.2 The affinity propagation clustering method

The other clustering approach used is the affinity propagation method. The idea behind this clustering algorithm is the passing of 'messages' between points of the dataset in the attempt to discover similarity between points and build clusters in which this similarity is maximized.

The crucial difference with the k-means method is that it does not require the number of clusters in advance. The number is estimated by the algorithm itself based on the data provided [53]. The affinity propagation method chooses centroids, called exemplars, in the dataset to be representative of clusters and, as a consequence, every time a new exemplar is added, also a new cluster is formed.

Clustering is made up by referring to three matrix that are the similarity matrix, the responsibility matrix and the availability matrix. The first matrix shows how much two points are similar, the second matrix contains values that express the responsibility a point has towards another point and, finally, the cells of the third matrix describes how available is one point to be the exemplar of another point.

In particular, the expression that govern responsibilities between points is the following one:

$$r(i,k) \leftarrow s(i,k) - \max[a(i,k') + s(i,k')], \forall k' \neq k \tag{5}$$

where $r(i,k)$ is the responsibility of point i to the cluster centroid k that means how much it is worth to assign the point i to the cluster defined by the exemplar k. $s(i,k)$ is the affinity between point i and the exemplar k. It indicates how much two points are similar and if the point i should be assigned to the cluster of k. Finally, the last part of expression represents the accumulated responsibility of point i towards all other exemplars k' different from k. In other words, this expression assessed if a point i must be assigned to an exemplar k by taking into account its affinity with the cluster centroid k and its similarity with others cluster centroids k'.

Instead, the expression that refers to the availability is written as follows:

$$a(i,k) \leftarrow \min[0, r(k,k) + \sum_{i' \notin i,k} r(i',k)] \tag{6}$$

where $a(i, k)$ shows the availability of the exemplar k to incorporate in its cluster the point i. In other words, it expresses how much the cluster centroid k is available to accept the point i as member of the cluster. For this purpose, the algorithm also evaluates $r(i', k)$ which is the responsibility of all other points i' towards the cluster centroid k.

A simple graphical representation of these expressions is illustrated in Figure 4.3.

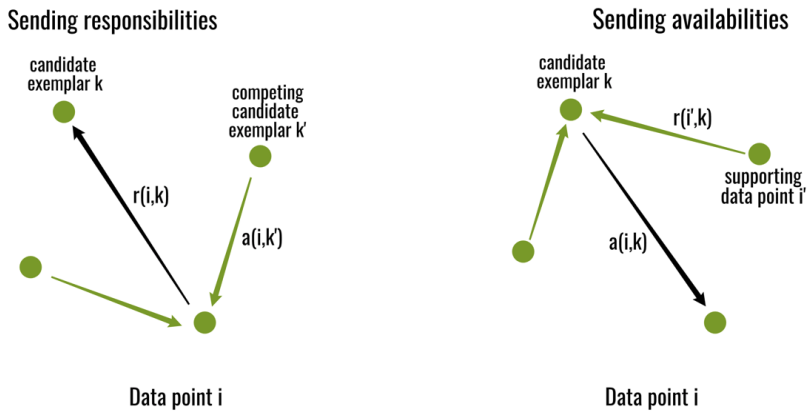

Figure 4.3: The affinity propagation method (source: documentation [54])

4.2 The routing phase

After all the customers have been grouped in clusters, the second step is to build the routes that minimize the total distance traveled by the fleet of vehicles. Each cluster is served by only one vehicle and, as a consequence, the VRP inside each cluster reduces to a TSP.

As already mentioned more than once in this document, these routing methods solve a static VRP. However, they want to reflect the real word in which requests made by customers arrive in different moments in time before the departure of vehicles from depot. For this reason, these algorithms compute intermediate solutions by adding new customers in the problem before finding the final routes that vehicles must follow. In particular, the two routing methods are called the insert procedure and the rebuild procedure.

4.2.1 The insert procedure

The insert procedure is one of the two methods applied to solve the VRP. As it is evident in Figures 4.4 and 4.5 that represent, respectively, the steps followed by the insert procedure and the results of the procedure applied on an example (e.i., this example will be deeply analyzed in chapter 5), the customers are first divided in clusters (according to the clustering method selected) and a first routing solution is obtained by considering only data available at time t_0. When new customers enter the problem in a second moment of time t_1, the initial solution is modified by adding the points to the nearest clusters. At this point, a second routing solution is computed and it is characterized by slightly different routes built to integrate the new customers.
Hence, in the insert method clusters are defined once at the beginning. When new requests arrive, they are included in these already established clusters and routes are recomputed. The final planned routes are the ones followed by vehicles when they leave the depot.

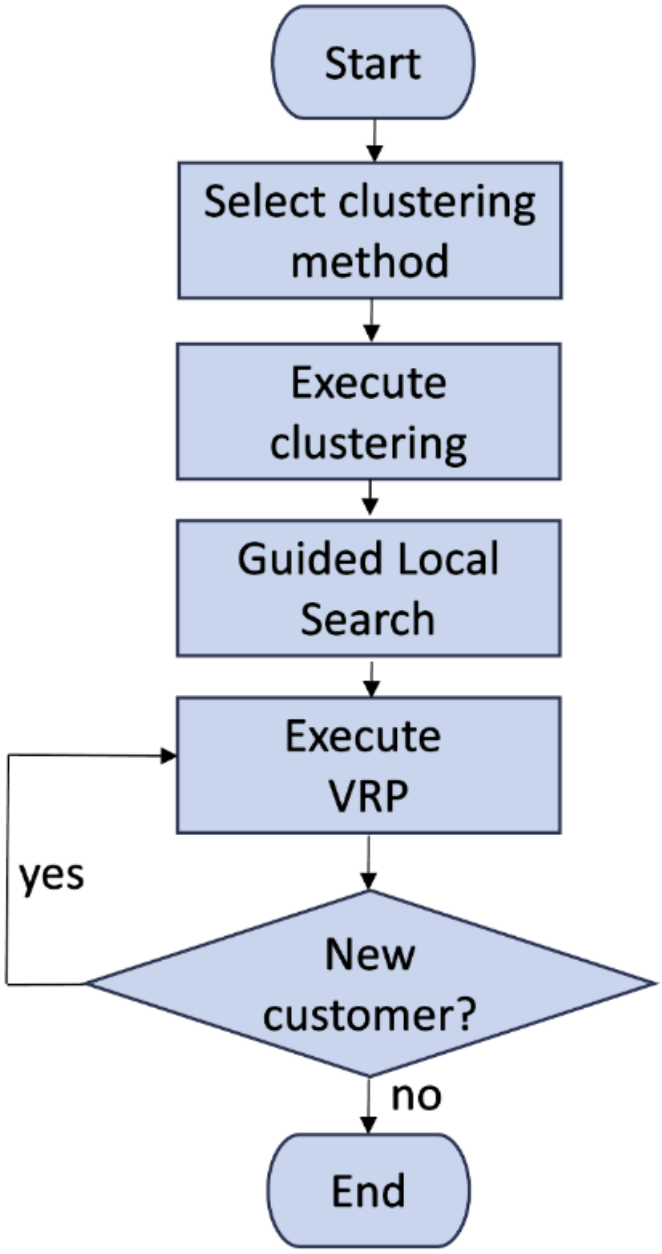

Figure 4.4: The insert procedure.

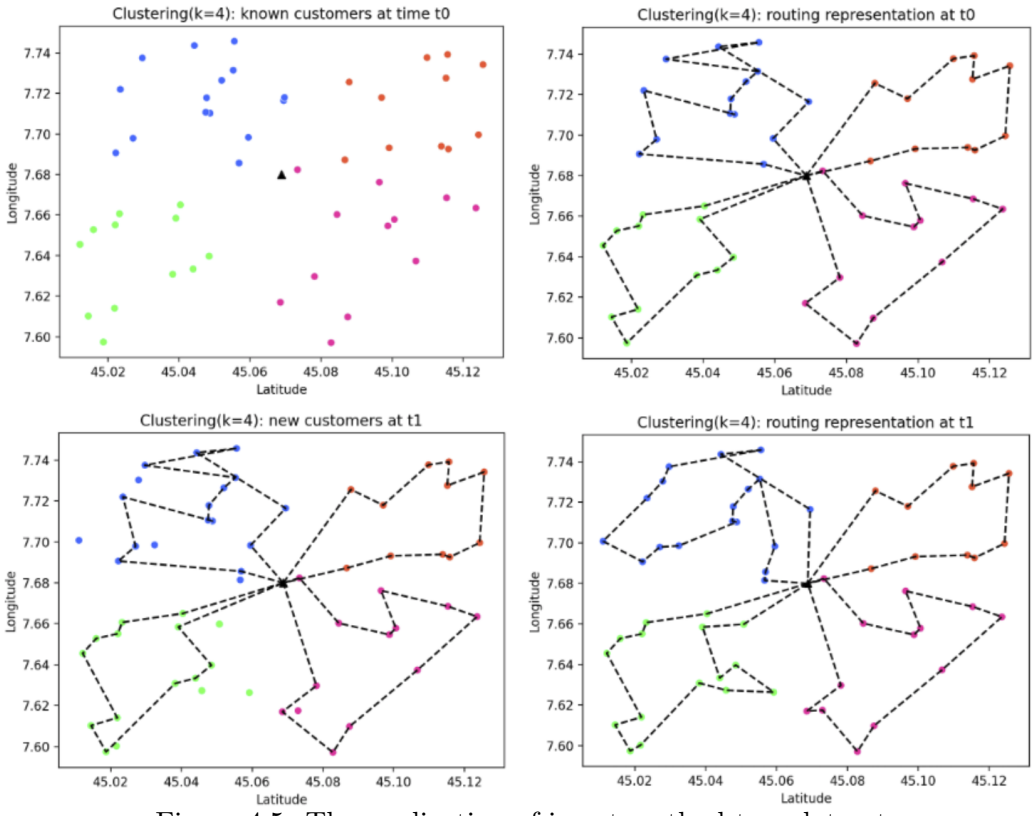

Figure 4.5: The application of insert method to a dataset.

4.2.2 The rebuild procedure

The alternative technique for the construction of routes is the rebuild procedure. Again, Figure 4.6 illustrates how this resolution method works while Figure 4.7 shows the results of the procedure applied on an example (e.i., this example will be deeply analyzed in chapter 5). Now, like in the insert method, the first step is selecting the clustering method, then customers available at time t_0 are grouped in different clusters and an initial routing solution for them is obtained. But, now differently from before, every time new requests enter the system, clusters are formed again and it may happen that customers belonging before to a cluster, now are part of another cluster. Then routes are recomputed on new clusters.

Hence, in rebuild method clusters are defined every time new requests appear in the system and routes are reformulated according to new clusters. When the moment to serve customers arrives, the final solution found by the algorithm is the one followed by the fleet of vehicles and routes can not be modified anymore.

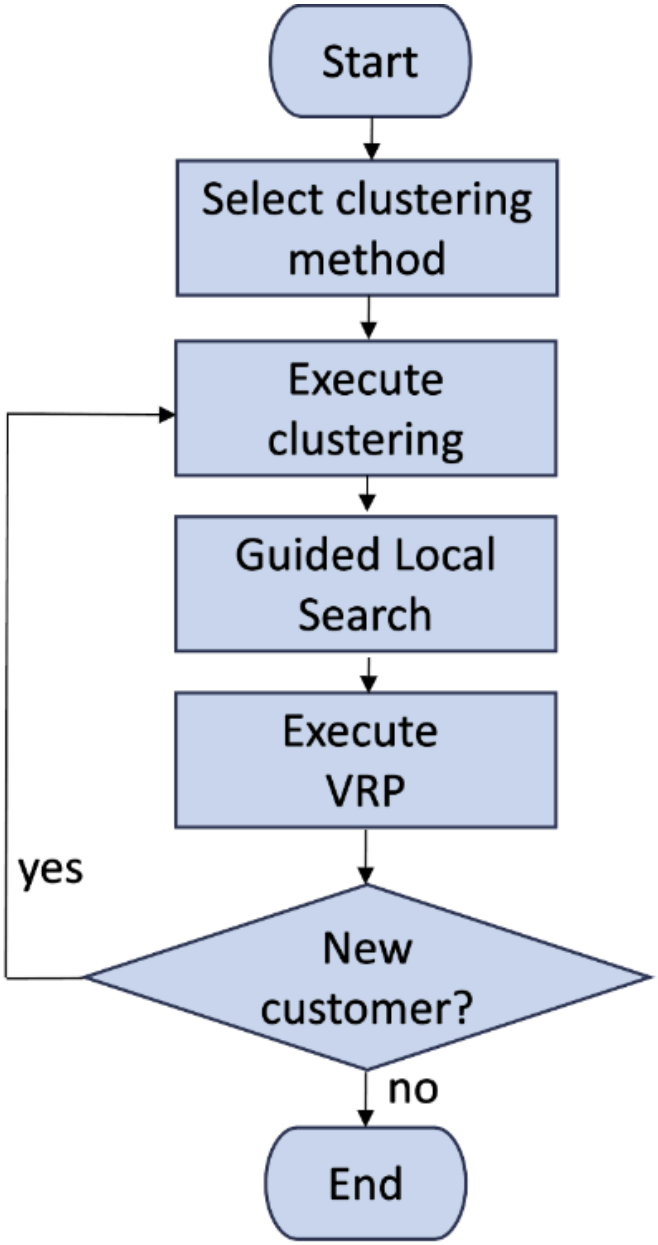

Figure 4.6: The rebuild procedure.

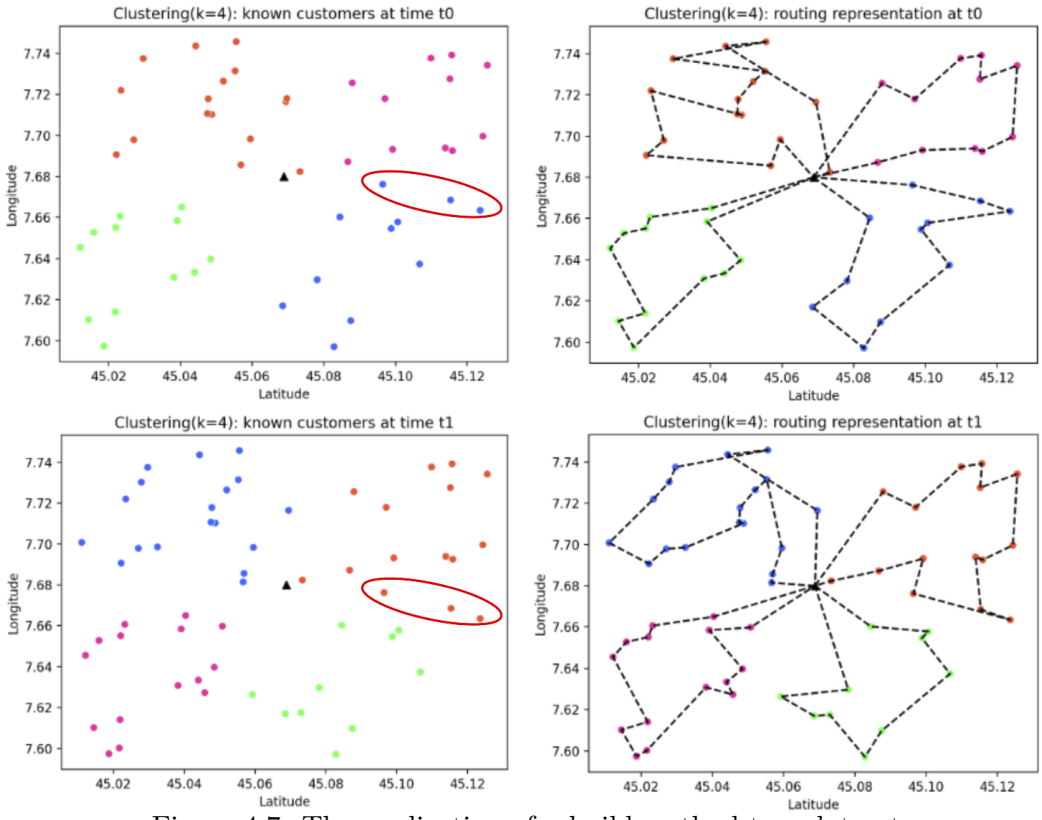

Figure 4.7: The application of rebuild method to a dataset.

In sections B.1 and B.2 appendix B, the Python code written to execute respectively the k-means clustering in the insert procedure and the k-means clustering in the rebuild procedure is illustrated. Instead, in sections B.3 and B.4 appendix B the Python code written to execute respectively the affinity propagation clustering in the insert procedure and the affinity propagation clustering in the rebuild procedure is outlined. Finally, in section C.1 of Appendix C, it is shown the code that solves the vehicle routing problem.

However, these main components must be connected in the correct order to build algorithms. Moreover, they are applied to the specific example of the next chapter 5. Therefore, before reading the appendix, it is suggested to have a look to the next section in order to comprehend how the code works.

5 The simulation of a numerical example

In this chapter, the simulation of a fictional VRP placed in the city of Turin is conceived. Let suppose that a logistics company must plan routes for its vehicles. It accepts orders made by customers until a certain moment in time, after which vehicles leave the depot and follow the last routes computed. Requests by customers occur in different moments in time and, every time, the company changes routes to satisfy new orders. However, the re-computation of routes is possible till the departure time of vehicles from the depot.

The aim is to analyze the performance of the four different algorithms (i.e., K-I, K-R, A-I, A-R) explained in the previous section to assess which is the best method in terms of shortest distance traveled, expressed in kilometers. Moreover, a comparison with solutions found by a known metaheuristic approach (GLS), that does not divide customers in clusters, is shown to evaluate the competitiveness of the algorithms. However, it is important to remember that these algorithms are metaheuristic approaches and, as such, their application finds a near-optimal solution for the fleet of vehicles.

The assumptions of the example are as follows:

- each vehicle starts and ends the route at the depot;
- customers appears in the routing problem in different moments of time;
- the arrival time of all customer requests is before the departure time of vehicles from the depot and therefore they are considered all known customers;
- each customer is visited by only one vehicle;
- each cluster is served by only one vehicle.

First, a short description about how the several datasets of customers have been created is provided. The number of generated datasets is equal to 30 in order to study the behaviour of the algorithms in customers datasets of different size. Hence, each of the five algorithms (i.e., K-I, K-R, A-I, A-R, GLS) has been applied to each dataset and the total number of performed test is 150.

Then, the obtained experimental results from the application of algorithms are detailed.

However, before starting with the analysis of this example, it is important to clarify that all 150 tests have been coded in Python on a personal computer equipped with a Intel Core i5 processor 2.7 GHz and 500 MB of memory.

5.1 The generation of the dataset

As mentioned before, the four adopted algorithms adjust the routes for vehicles every time a new request appears. However, although all these customers enter the problem in different moments, it arrives a moment in time when vehicles leave the depot and no new request is considered. This means that the routing problem is static and surely all requests are known before the departure time of vehicles from the depot. In real world, new orders calling in for service occur continuously till the last moment and, consequently, every time algorithms should recalculate the routes to incorporate new orders.

In this example it is supposed that the arrival process is governed by a Poisson distribution. The main property of this distribution is that events occur independently within a specified interval of time. The parameter λ represents the average number of requests that arrive within a specified time interval. In this experiment, the time is divided in three intervals of equal length in which requests can occur. The exact length of one interval of time is not relevant and the only needed parameter is the mean arrival rate of orders λ.

In each of 30 datasets, it is supposed that the first 50 customers known at time t_0 are always the same. Their position is defined by the latitude and longitude coordinates. These coordinates have been generated randomly by constraining them to the geographic area of Turin and its surroundings. However, once these coordinates have been defined, they are saved in a file because this is the fraction of customers that never change in each dataset.

Concerning the following three moments in time (i.e., t_1, t_2, t_3), the number of new customers depends on the chosen parameter λ. In all datasets, this parameter varies in order to generate datasets of customers that assume different dimensions. In this way, algorithms are tested on datasets of different size. In particular, tests with a small

size of total customers in which the λ parameter ranges between 3 and 7, tests with a medium size of requests in which the λ parameter varies between 10 and 25 and, finally, big size tests characterized by a λ parameter that fluctuates between 40 and 65 are considered. These values of λ have been selected because they allow to study datasets of customers that approximately ranges between 50 and 200 customers in total.

Hence, each of 30 customer datasets comprises a different total number of customers depending on the value of λ specific for the dataset under examination. Instead, what is common to all datasets is the initial fraction of customers known at time t_0 which always corresponds to the same amount of 50 customers.

Concerning the positions of customers, each request is represented by the latitude and longitude coordinates that have been constrained to the geographic area of Turin and its surroundings. Again, in all datasets the initial customers at time t_0 are described always by the same latitude and longitude coordinates. Hence, both the amount and the locations of customers known at time t_0 remain the same in all 30 datasets. Instead, similarly to what happens in the case of new customers appearing in the following three moments in time in which the number depends on the specific value of λ for the dataset, the latitude and longitude coordinates for new customers are always generated randomly and different in each dataset.

Hence, all 30 datasets include a fraction of 50 customers that is always the same in terms of amount and coordinates about their position. Instead, what makes datasets unique is the specific value of the λ parameter that generates an additional amount of customers that varies in each dataset, and their changing latitude and longitude coordinates.

In sections A.1 and A.2 of Appendix A, an example of the *File.csv* with known customers at time t_0 common to all datasets and one of 30 *UpdatedFile.csv* with additional new customers generated by the Poisson distribution is outlined. In the specific test shown in Appendix A, the *UpdatedFile.csv* including the total amount of customers has been generated with a λ parameter equal to 15.

5.2 Experimental results

In this section, results with the shortest distance computed by each of the proposed algorithms are illustrated.

In particular, Table 1 provides distances traveled by vehicles in final routes at time t_3, when all customers are known and vehicles are going to leave the depot. The table outlines the results provided by each of the five algorithms for each of 30 customer datasets. The final row shows the average distance considering all tests.

The bold values in Table 1 represents distances computed by clustering-based algorithms that overcome the solution provided by the GLS approach without clustering. The purpose of the comparison is not to choose one algorithm over another but rather to understand whether the methods based on clustering provide acceptable solutions or not. All metaheuristic approaches stand out for their peculiarity in addressing the VRP differently from exact techniques. What matters is finding admissible solutions that satisfy the stopping criteria for the VRP under examination, considering time and resource constraints. However, if the GLS method is one of the most used metaheuristic techniques and its solutions are given for good, the comparison allows to realize if the clustering-based solutions get closer to the GLS solutions and whether techniques based on clustering methods can be assessed competitive in terms of performances.

In 13 out of the 30 datasets, the K-I algorithm finds a cheaper solution than the GLS. A similar trend is obtained also in the case of the K-R algorithm where, in 15 out of 30 datasets, it offers a better solution that overcomes the one provided by the GLS technique. Concerning methods based on the affinity propagation, both A-I and A-R algorithms always perform worse than the GLS approach, except for 3 tests in which the A-I algorithm gives more competitive solutions. By taking into account only clustering-based algorithms, in all tests K-I and K-R algorithms alternate in providing the most promising solutions, while A-I and A-R techniques finds more expensive routes in terms of distance traveled, with the A-R algorithm that is confirmed to be always the worst. All these considerations are also proven by the average distances in the last row of the table.

Table 2 shows the best number of clusters formed by each of the four clustering-based

Total distance traveled [km] at time t_3					
test	K-I	K-R	A-I	A-R	GLS
1	12.54	12.57	15.19	15.32	11.42
2	12.02	12.19	14.54	15.2	11.72
3	12.14	12.07	14.41	15.86	11.06
4	13.07	12.92	14.96	17.01	11.43
5	13.41	13.35	15.95	15.28	12.18
6	11.58	11.36	13.78	13.79	10
7	13.07	13.19	15.2	16.94	11.8
8	11.77	11.82	14.6	14.78	11.03
9	11.3	11.3	13.66	13.85	10.7
10	11.94	11.94	14.81	14.67	10.96
11	15.17	14.89	17.02	17.36	14.35
12	16.02	15.17	16.6	17.98	14.56
13	19.09	**18.15**	19.96	22.72	18.94
14	**19.64**	**18.94**	21.22	23.79	21.07
15	14.74	**14.4**	16.2	17.83	14.42
16	14.37	14.35	17.19	18.28	13.84
17	17.65	16.74	19.47	22.76	16.33
18	**17.26**	**17.2**	19.68	21.44	18.43
19	**18.41**	**18.36**	**18.41**	23.34	18.94
20	16.48	16.52	18.64	20.34	15.97
21	**23.42**	**23.22**	25.9	29.72	24.15
22	**20.54**	**19.97**	23.08	27.49	20.96
23	**24.42**	**24.61**	**25.03**	29.75	26.21
24	**22.7**	**21.83**	24.66	29.04	24.27
25	**21.54**	**21.49**	23.78	27.34	23.64
26	**21.16**	**21.36**	23.06	26.63	22.86
27	**23.76**	**24.68**	25.51	30.22	25.12
28	**23.34**	**22.84**	24.34	28.41	23.43
29	**26**	**26.01**	**27.45**	32.84	28.12
30	**25.49**	**25.27**	27.36	32.13	26.19
average	**17.47**	**17.29**	19.39	21.74	17.47

Table 1: Total distance traveled [km] at time t_3

algorithms at time t_3 when all requests are known. The algorithms characterized by the insert procedure (i.e. K-I and A-I) detects the number of clusters with only customers known at time t_0. Then, new customers in following moments of time are simply added to these already formed clusters. The K-I procedure applies the elbow method (see Figure 5.1) and finds that the best number of clusters is equal to 4 when considering only 50 customers known at time t_0. Instead, in the A-I algorithm, in which the best number of clusters is computed by the algorithm itself, customers known at time t_0 are grouped within 6 clusters. Instead, concerning algorithms that rebuild clusters every time new customers are included, in the K-R algorithm the value of k ranges between 3 and 4 clusters while the A-R technique provides values of k that increase as the total number of customers in the problem grows.

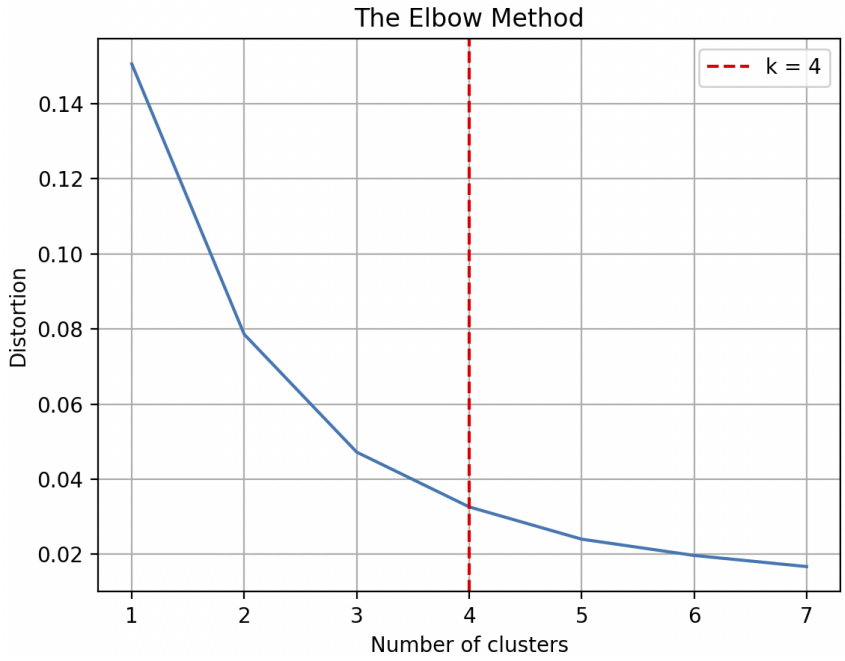

Figure 5.1: The optimal number of clusters at time t_0 according to the elbow method.

The results observed in these two tables are consistent with each other. The best clustering-based algorithms to solve the VRP placed in Turin are those that rely on the k-means method.

This clustering method, regardless of the resolution procedure chosen (i.e., the insert or the rebuild method), finds a smaller best number of clusters in each test. This implies

Number of clusters at time t_3				
test	K-I	K-R	A-I	A-R
1	4	4	6	6
2	4	4	6	7
3	4	4	6	7
4	4	4	6	8
5	4	4	6	6
6	4	4	6	6
7	4	4	6	8
8	4	4	6	7
9	4	4	6	6
10	4	4	6	7
11	4	4	6	7
12	4	4	6	8
13	4	4	6	9
14	4	3	6	10
15	4	4	6	8
16	4	4	6	8
17	4	3	6	10
18	4	4	6	9
19	4	3	6	10
20	4	4	6	9
21	4	4	6	11
22	4	4	6	12
23	4	3	6	11
24	4	3	6	12
25	4	4	6	13
26	4	3	6	11
27	4	3	6	12
28	4	4	6	12
29	4	3	6	13
30	4	3	6	12
average	4	3.7	6	9.2

Table 2: Number of clusters at time t_3

p-value				
K-I vs K-R	K-I vs A-I	K-I vs A-R	K-I vs GLS	K-R vs A-I
0.013	2.561e-06	1.862e-09	1.0	1.862e-09
K-R vs A-R	K-R vs GLS	A-I vs A-R	A-I vs GLS	A-R vs GLS
1.862e-09	0.477	1.024e-07	1.419e-09	1.862e-09

Table 3: Statistical test of Wilcoxon between each pair of algorithms

a smaller amount of vehicles used to serve all customers and a consequent reduction in the distance traveled to visit them.

Instead, the algorithms based on the affinity propagation tend to select a higher value of k for the same amount of customers. This translates in a larger number of vehicles committed to serve all customers and a consequent increase in the distance traveled to satisfy them. Indeed, for instance, the A-R algorithm is the technique that creates on average the highest number of clusters and, at the same time, the one that provides on average the less promising solutions in terms of distance traveled.

However, a statistical analysis using the test of Wilcoxon is performed to investigate whether differences in results provided by algorithms are statistically significant. The comparison is made two by two for each pair of algorithms. The null hypothesis states that differences are not statistically significant and the aim is to accept or reject this hypothesis through the statistic test.

Table 3 shows the p value computed at statistical level of $\alpha = 0.05$ for each pair of algorithms. All these p values are less than α, except for the comparison between K-I versus GLS and K-R versus GLS. The statistical test proves that differences between algorithms are statistically significant, except for the comparison between K-I versus GLS and K-R versus GLS in which the null hypothesis can not be rejected. This means that for comparisons characterized by $p < \alpha$ there is enough statistical evidence to assert that the difference between algorithms is due to a real cause and not to chance.

6 Conclusions and future perspectives

The document, after having provided a definition of the VRP with its main extensions and a description of all resolution methods studied over the years, focused on four particular algorithms used to solve the VRP. The common concept behind these methods is the execution of an initial phase of clustering to divide customers in groups based on their geographical locations. In addition, in these techniques the concept of time is fundamental since, every time new customers enter the VRP, clusters are updated or formed again depending on the algorithm selected.

The benefit of clustering consists in simplifying the planning of routes to serve customers. Without clustering, the number of possible routes grows exponentially with the number of requests, making the problem computationally burdensome. Clustering permits to reduce the complexity since routes are built for each cluster and not for all customers in the problem. On the other hand, the limitation of algorithms proposed, like all metaheuristic approaches, is that finding the optimal solution for a NP-hard problem like the VRP is not feasible when its size is large.

However, the experimental results achieved for the example placed in the city of Turin give little space to the emergence of possible doubts. In each test performed, the distance traveled by vehicles is directly proportional to the number of clusters k. Algorithms that perform better are the ones that detect a smaller value of k. Indeed, regardless the solution procedure selected, in k-means algorithms solutions are quite similar since the value of k is always equal to 4 for the K-I procedure while it ranges between 3 and 4 for the K-R method. Instead, the performances of algorithms based on affinity propagation are poorer since the value of k is higher for these algorithms. In particular, there is an evident difference between the A-I and A-R methods because the value of k for the A-I algorithm is stable to 6 while the number of clusters in the A-R procedure ranges between 6 and 13.

So, in conclusion if an algorithm should be selected, this would be one of the two based on the k-means method in which the optimal number of clusters is detected by the elbow method. Algorithms based on affinity propagation tend to overestimate the necessary number of clusters and this leads to a larger amount of vehicles used that

translates in a higher number of kilometers traveled. Surely each client is visited faster but the solution is not efficient in terms of distance traveled.

However, this work wants to be a starting point for future implementations. Possible extensions could be the analysis of the algorithms behaviours in dealing with more complex scenarios of VRPs. Additional constraints such as the integration of time windows to serve customers or the limitation of vehicles capacity could be included. Moreover, the possibility to accept new customer requests occurring after the departure time of vehicles from the depot shifts the analysis of the proposed algorithms in the context of dynamic VRP. Finally, another option could be represented by the comparison of the algorithms based on clustering with the algorithms conceived by other authors in other cases of study.

References

[1] Alessandra Riccobono. La logistica dei trasporti e il vehicle routing problem. 2015.

[2] Allan Larsen and OB Madsen. *The dynamic vehicle routing problem*. PhD thesis, Institute of Mathematical Modelling, Technical University of Denmark, 2000.

[3] Karla L Hoffman, Manfred Padberg, Giovanni Rinaldi, et al. Traveling salesman problem. *Encyclopedia of operations research and management science*, 1:1573–1578, 2013.

[4] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[5] Gilbert Laporte, Paolo Toth, and Daniele Vigo. Vehicle routing: historical perspective and recent contributions. *EURO Journal on Transportation and Logistics*, 2:1–4, 2013.

[6] Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[7] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & industrial engineering*, 99:300–313, 2016.

[8] Gilbert Laporte. What you should know about the vehicle routing problem. *Naval Research Logistics (NRL)*, 54(8):811–819, 2007.

[9] Ashima Gupta and Sanjay Saini. An enhanced ant colony optimization algorithm for vehicle routing problem with time windows. In *2017 ninth international conference on advanced computing (ICOAC)*, pages 267–274. IEEE, 2017.

[10] Ted K Ralphs, Leonid Kopman, William R Pulleyblank, and Leslie E Trotter. On the capacitated vehicle routing problem. *Mathematical programming*, 94:343–359, 2003.

[11] Imdat Kara. Arc based integer programming formulations for the distance constrained vehicle routing problem. In *3rd IEEE International Symposium on Logistics and Industrial informatics*, pages 33–38. IEEE, 2011.

[12] Brian Kallehauge, Jesper Larsen, Oli BG Madsen, and Marius M Solomon. *Vehicle routing problem with time windows*. Springer, 2005.

[13] Roberto Baldacci, Maria Battarra, and Daniele Vigo. Routing a heterogeneous fleet of vehicles. *The vehicle routing problem: latest advances and new challenges*, pages 3–27, 2008.

[14] Parida Jewpanya, Yu-Wei Chen, and VF Yu. The pickup and delivery multi-depot vehicle routing problem. In *The 17th APIEMS Conference*, 2016.

[15] Paolo Toth and Daniele Vigo. An exact algorithm for the vehicle routing problem with backhauls. *Transportation science*, 31(4):372–385, 1997.

[16] Jairo R Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, and Nilson Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129, 2015.

[17] Inmaculada Rodríguez-Martín and Hande Yaman. Periodic vehicle routing problem with driver consistency and service time optimization. *Transportation Research Part B: Methodological*, 166:468–484, 2022.

[18] Bruce L Golden, Subramanian Raghavan, Edward A Wasil, et al. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer, 2008.

[19] Warren B Powell, Patrick Jaillet, and Amedeo Odoni. Stochastic and dynamic networks and routing. *Handbooks in operations research and management science*, 8:141–295, 1995.

[20] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[21] Jose Caceres-Cruz, Pol Arias, Daniel Guimarans, Daniel Riera, and Angel A Juan. Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR)*, 47(2):1–28, 2014.

[22] Citra Dewi Purnamasari and Amelia Santoso. Vehicle routing problem (vrp) for courier service: A review. 2018.

[23] Anke Fabri and Peter Recht. On dynamic pickup and delivery vehicle routing with several time windows and waiting times. *Transportation Research Part B: Methodological*, 40(4):335–350, 2006.

[24] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153:29–46, 2007.

[25] Victor Pillac, Christelle Guéret, and Andrés Medaglia. Dynamic vehicle routing problems: state of the art and prospects. 2011.

[26] Dimitris Bertsimas, Garrett Van Ryzin, et al. The dynamic traveling repairman problem. 1989.

[27] Brenner Humberto Ojeda Rios, Eduardo C Xavier, Flávio K Miyazawa, Pedro Amorim, Eduardo Curcio, and Maria João Santos. Recent dynamic vehicle routing problems: A survey. *Computers & Industrial Engineering*, 160:107604, 2021.

[28] GH Dastghaibifard, E Ansari, SM Sheykhalishahi, A Bavandpouri, and E Ashoor. A parallel branch and bound algorithm for vehicle routing problem. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 2, pages 19–21, 2008.

[29] Antonio De Guzman, Geneva Valenton, Kimberly Somera, and Hazel Mae Diza. A comparison of clarke-wright method and branch & bound method for vehicle routing problem in pangasinan. *Southeast Asian Journal of Science and Technology*, 6(2 (SI)), 2021.

[30] Jens Lysgaard, Adam N Letchford, and Richard W Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical programming*, 100:423–445, 2004.

[31] Jacques Desrosiers, François Soumis, and Martin Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984.

[32] Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *International conference on integration of constraint programming, artificial intelligence, and operations research*, pages 190–213. Springer, 2022.

[33] Martina Jakara, Jasmina Pašagić Škrinjar, and Nikolina Brnjac. Vehicle routing problem-case study on logistics company in croatia. *International Journal for Traffic & Transport Engineering*, 9(4), 2019.

[34] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[35] Colin R Reeves. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., 1993.

[36] İ K Altınel and Temel Öncan. A new enhancement of the clarke and wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 56:954–961, 2005.

[37] Gunadi W Nurcahyo, Rose Alinda Alias, Siti Mariyam Shamsuddin, and Mohd Noor Mohd Sap. Sweep algorithm in vehicle routing problem for public transport. *Jurnal Antarabangsa Teknologi Maklumat*, 2:51–64, 2002.

[38] Asvin Goel and Volker Gruhn. A general vehicle routing problem. *European Journal of Operational Research*, 191(3):650–660, 2008.

[39] Anand Subramanian, Lúcia MA Drummond, Cristiana Bentes, Luiz Satoru Ochi, and Ricardo Farias. A parallel heuristic for the vehicle routing problem with simul-

taneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911, 2010.

[40] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[41] Jiajie Peng, Hongxiang Li, Qinghua Jiang, Yadong Wang, and Jin Chen. An integrative approach for measuring semantic similarities using gene ontology. *BMC systems biology*, 8:1–12, 2014.

[42] Ibrahim Hassan Osman. Metastrategy: simulated annealing and tabu search for combinatorial optimization problems. 1991.

[43] Christos Voudouris, Edward PK Tsang, and Abdullah Alsheddy. Guided local search. In *Handbook of metaheuristics*, pages 321–361. Springer, 2010.

[44] Vickie Dawn Wester. A genetic algorithm for the vehicle routing problem. 1993.

[45] Joanna Ochelska-Mierzejewska, Aneta Poniszewska-Marańda, and Witold Marańda. Selected genetic algorithms for vehicle routing problem solving. *Electronics*, 10(24):3147, 2021.

[46] Noraini Mohd Razali. An efficient genetic algorithm for large scale vehicle routing problem subject to precedence constraints. *Procedia-Social and Behavioral Sciences*, 195:1922–1931, 2015.

[47] John E Bell and Patrick R McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced engineering informatics*, 18(1):41–48, 2004.

[48] Ashek Ahmmed, Md Ali Ahsan Rana, Abul Ahsan Md Mahmudul Haque, and Md Al Mamun. A multiple ant colony system for dynamic vehicle routing problem with time window. In *2008 Third International Conference on Convergence and Hybrid Information Technology*, volume 2, pages 182–187. IEEE, 2008.

[49] Fatma Pinar Goksal, Ismail Karaoglan, and Fulya Altiparmak. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & industrial engineering*, 65(1):39–53, 2013.

[50] Warisa Wisittipanich, Khamphe Phoungthong, Chanin Srisuwannapa, Adirek Baisukhan, and Nuttachat Wisittipanit. Performance comparison between particle swarm optimization and differential evolution algorithms for postman delivery routing problem. *Applied Sciences*, 11(6):2703, 2021.

[51] Rena Nainggolan, Resianta Perangin-angin, Emma Simarmata, and Astuti Feriani Tarigan. Improved the performance of the k-means cluster using the sum of squared error (sse) optimized by using the elbow method. In *Journal of Physics: Conference Series*, volume 1361, page 012015. IOP Publishing, 2019.

[52] Zhang Qiu-yu, Lu Jun-chi, Zhang Mo-Yi, Duan Hong-xiang, and Lv Lu. Hand gesture segmentation method based on ycbcr color space and k-means clustering. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8(5):105–116, 2015.

[53] Clustering — scikit-learn 1.3.1 documentation. `https://scikit-learn.org/stable/modules/clustering.html`.

[54] Affinity propagation in machine learning. `https://www.geeksforgeeks.org/affinity-propagation-in-ml-to-find-the-number-of-clusters/`, 2019.

A The dataset of customers

The arrival of new customers is governed by a Poisson distribution. In particular, the Poisson arrival process has been implemented by passing the parameter λ in input to the method *random.poisson* of the Python library *numpy*. At time t_0, in all performed tests customers are always the same and equal to 50, while in the following three moments in time (i.e., t_1, t_2, t_3), the number of new customers depends on the chosen parameter λ.

Concerning the positions of customers, each request is represented by the latitude and longitude coordinates that have been constrained to the geographic area of Turin and its surroundings. Again, the generation of the location for each customer has been implemented by passing the constrained latitude and longitude coordinates of Turin to the method *numpy.random.uniform*.

Then, the generated list of customers with their position and arrival time is saved on a *File.csv* so that it can be used by all the algorithms. In sections A.1 and A.2 of Appendix A, the *File.csv* with known customers at time t_0 common to all datasets and one of 30 *UpdatedFile.csv* with additional new customers generated by the Poisson distribution are outlined. In particular, in section A.1 of Appendix A, the *File.csv* containing the initial fraction of customers common to all datasets with their correspondent positions and arrival time (i.e., it is t_0 for all requests) is provided.

Then, in section A.2 of Appendix A, one of 30 *UpdatedFile.csv* including the complete amount of customers with their correspondent positions and arrival time (i.e., it is t_0 for the first 50 customers and t_1, t_2 or t_3 for the remaining customers depending on their arrival time) is shown. In the specific test illustrated in this appendix, the λ parameter is equal to 15 and, from the initial customers equal to 50 at time t_0, the total amount of requests at time t_3 is equal to 82.

Before showing the two files csv, it is important to clarify how a csv file is represented. A csv file is characterized by data in the form of text and the values of each cell are separated by commas. However, for ease of representation, the dataset of customers is shown inside a table.

Each row, except for the first one, is a specific customer while each column describes a characteristic for that customer. The first column denotes the type of request (i.e., 0 if the customer is known at time t_0 and 1 if the customer is known at time t_1, t_2 or t_3,), the second and the third consist in the latitude and longitude coordinates of the request, and the last column shows the arrival time of the customer. The first row represents the geographical coordinates of the depot and it has NaN value in New request and Time interval columns since, obviously, it is not a request.

A.1 the dataset of customer at time t_0

In this table, the initial fraction of known customers common to all 30 datasets is reported. They are equal to 50 at time t_0 and have the following latitude and longitude coordinates in all tests performed.

New request	Latitude	Longitude	Time interval
Nan	45.0688	7.68	Nan
0	45.0120873378955	7.64552613150164	0
0	45.0520083467709	7.72652983631041	0
0	45.0568951740094	7.68568255803922	0
0	45.1235974207845	7.66349414065841	0
0	45.0879276661641	7.72567134778513	0
0	45.0232216157975	7.66067835894532	0
0	45.0867109369781	7.68725084793321	0
0	45.1067040686691	7.63733836569737	0
0	5.1255929708042	7.73428354454981	0
0	45.015892657567	7.65280337957878	0
0	45.0296639806245	7.73757376380078	0
0	45.0987964409615	7.6546727535446	0
0	45.014438398235	7.61019079554931	0
0	45.0694666275897	7.71653675132062	0
0	45.0403792523293	7.66504937272296	0
0	45.1243313011729	7.69965635904461	0

New request	Latitude	Longitude	Time interval
0	45.0875103197832	7.60972765695222	0
0	45.0555748186703	7.74582988305926	0
0	45.0487545622969	7.71028679252593	0
0	45.1098499839172	7.73776810247859	0
0	45.0685436661552	7.61695929634866	0
0	45.0970365058818	7.7180245392484	0
0	45.0391118403773	7.65846066692718	0
0	45.1138724949252	7.69396394765023	0
0	45.0733702947586	7.68239801093924	0
0	45.0270365602449	7.69793638524497	0
0	45.0221754859492	7.69069672527829	0
0	45.0991781275793	7.69321701156751	0
0	45.1006261336581	7.65785520936468	0
0	45.1151816626905	7.72764385877252	0
0	45.0439685550232	7.63337104487315	0
0	45.0475297912567	7.71073809743528	0
0	45.0844798660921	7.66027750832561	0
0	45.0552698110098	7.7315267657797	0
0	45.0443121164434	7.74371781383988	0
0	45.0484745506056	7.63971395240225	0
0	45.0220276904403	7.65514497199478	0
0	45.0828323900947	7.59705666542283	0
0	45.0186928701289	7.59735897745475	0
0	45.0218487633904	7.614032626916	0
0	45.0964299069134	7.67623866158172	0
0	45.0477736798047	7.71791950578131	0
0	45.0595291787738	7.69834795890168	0
0	45.069705816995	7.71808073848439	0
0	45.1158582955046	7.69260894360755	0
0	45.1156436889776	7.7392685952947	0
0	45.0382193314983	7.63081653108776	0
0	45.0234548965359	7.72206715206364	0
0	45.0781593070208	7.62969061914881	0
0	45.1153474489998	7.66852829101108	0

A.2 the dataset of customer at time t_3

In this table, one of 30 datasets with all the customers up to time t_3 is shown. After time t_3, new requests are not considered because final routes are computed and the vehicles are leaving the depot. In particular, in this test the final dataset of customers has been created by using a λ parameter equal to 15. From the initial customers equal

to 50 at time t_0, the total amount of requests at time t_3 is equal to 82.

New request	Latitude	Longitude	Time interval
Nan	45.0688	7.68	Nan
0	45.0120873378955	7.64552613150164	0
0	45.0520083467709	7.72652983631041	0
0	45.0568951740094	7.68568255803922	0
0	45.1235974207845	7.66349414065841	0
0	45.0879276661641	7.72567134778513	0
0	45.0232216157975	7.66067835894532	0
0	45.0867109369781	7.68725084793321	0
0	45.1067040686691	7.63733836569737	0
0	5.1255929708042	7.73428354454981	0
0	45.015892657567	7.65280337957878	0
0	45.0296639806245	7.73757376380078	0
0	45.0987964409615	7.6546727535446	0
0	45.014438398235	7.61019079554931	0
0	45.0694666275897	7.71653675132062	0
0	45.0403792523293	7.66504937272296	0
0	45.1243313011729	7.69965635904461	0
0	45.0875103197832	7.60972765695222	0
0	45.0555748186703	7.74582988305926	0
0	45.0487545622969	7.71028679252593	0
0	45.1098499839172	7.73776810247859	0
0	45.0685436661552	7.61695929634866	0
0	45.0970365058818	7.7180245392484	0
0	45.0391118403773	7.65846066692718	0
0	45.1138724949252	7.69396394765023	0
0	45.0733702947586	7.68239801093924	0
0	45.0270365602449	7.69793638524497	0
0	45.0221754859492	7.69069672527829	0
0	45.0991781275793	7.69321701156751	0
0	45.1006261336581	7.65785520936468	0
0	45.1151816626905	7.72764385877252	0
0	45.0439685550232	7.63337104487315	0
0	45.0475297912567	7.71073809743528	0
0	45.0844798660921	7.66027750832561	0
0	45.0552698110098	7.7315267657797	0
0	45.0443121164434	7.74371781383988	0
0	45.0484745506056	7.63971395240225	0
0	45.0220276904403	7.65514497199478	0
0	45.0828323900947	7.59705666542283	0
0	45.0186928701289	7.59735897745475	0

New request	Latitude	Longitude	Time interval
0	45.0218487633904	7.614032626916	0
0	45.0964299069134	7.67623866158172	0
0	45.0477736798047	7.71791950578131	0
0	45.0595291787738	7.69834795890168	0
0	45.069705816995	7.71808073848439	0
0	45.1158582955046	7.69260894360755	0
0	45.1156436889776	7.7392685952947	0
0	45.0382193314983	7.63081653108776	0
0	45.0234548965359	7.72206715206364	0
0	45.0781593070208	7.62969061914881	0
0	45.1153474489998	7.66852829101108	0
1	45.011013153103995	7.65984763476217	1
1	45.04576824585266	7.6814795437522	1
1	45.02787692744397	7.617479201605071	1
1	45.02161893839841	7.626309527829987	1
1	45.032419924308435	7.718513919007357	1
1	45.05073948360995	7.744144021085067	1
1	45.05662825953653	7.643953899258364	1
1	45.07296392441039	7.7019253601974045	1
1	45.05920736915638	7.730087540301294	1
1	45.08980024254563	7.732874819516089	1
1	45.034512008719126	7.609011764339576	1
1	45.11198350518496	7.601975381834631	1
1	45.014149573217765	7.621984054193379	1
1	45.08810376367052	7.7303558030247	1
1	45.12115729596731	7.710469345120115	2
1	45.072314007771894	7.638907930785854	2
1	45.090565868104306	7.716759737253078	2
1	45.047284297565696	7.611793579006379	2
1	45.08994760668338	7.664527709504913	2
1	45.1069819522682	7.735015111973244	2
1	45.01310315189458	7.640922964701173	2
1	45.09726659621867	7.640029626803711	2
1	45.124719025224245	7.615894371534097	2
1	45.06753091331724	7.677992150105073	3
1	45.01713669268847	7.697560580718627	3
1	45.077023524631585	7.674778034144921	3
1	45.02787378611417	7.740522997666594	3
1	45.07877013674388	7.6857429211968045	3
1	45.091472211402404	7.734220493039046	3
1	45.0227684593152	7.617033629734374	3
1	45.05861643859925	7.617309281129366	3

New request	Latitude	Longitude	Time interval
1	45.09085601813869	7.719530867172557	3
1	45.0586306159956	7.656844556058786	3
1	45.0167446477788	7.621299192158891	3

B The clustering phase

Once the *UpdatedFile.csv* containing the information about all customers to be served in Turin is built, each of the proposed algorithms accesses it in order to start with the clustering phase. Both k-means and affinity propagation methods have been implemented through the Python machine learning library *Scikit-learn*. In this appendix, the written code for the execution of the two clustering methods is shown. Depending on which solution method used (i.e., the insert or the rebuild procedure), the clustering is made respectively either only at the beginning or every time new customers enter the problem.

B.1 K-means clustering in the insert procedure

In this section, the code for the k-means clustering in the insert procedure is provided. In particular, clusters are formed by taking into account only customers know at time t_0. The optimal number of clusters is detected by the elbow method. Then, each new customers is simply added to the nearest already established clusters.

```
# CLUSTERING BY CONSIDERING ONLY KNOWN CUSTOMERS AT t=0
X = dtf[dtf["base"]==0][dtf["Time␣Interval"]==0][["Lat","
   Lng"]]
# FIND THE OPTIMAL NUMBER K OF CLUSTERS
max_k = 7
# iterations
distortions = []
for i in range(1, max_k+1):
    if len(X) >= i:
        model = cluster.KMeans(n_clusters=i, init='k-means++
           ',
        max_iter=300, n_init=10, random_state=0)
        model.fit(X)
        distortions.append(model.inertia_)
```

```
# best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i
   *100 for i
     in np.diff(distortions,2)]))
# plot the elbow method
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters
   ',
       ylabel="WCSS")
ax.legend()
ax.grid(True)
plt.show()

# DIVIDE CUSTOMERS IN K CLUSTERS
k = 4 # it is found by the elbow method
model = cluster.KMeans(n_clusters=k, init='k-means++')
X = dtf[dtf["base"]==0][dtf["Time Interval"]==0][["Lat","
   Lng"]]
dtf_X = X.copy()
dtf_X["cluster"] = model.fit_predict(X)
dtf["cluster"] = dtf_X["cluster"]
dtf.sample(5)
# plot customers divided in clusters found
fig, ax = plt.subplots()
palette_personalizzata = ["#FF5733", "#33FF57", "#3366FF",
   "#FF33A1",
    "#33FFFF", "#FF3366", "#FFFF33", "#9933FF", "#FF9933",
      "#33FF99"]
```

```
sns.scatterplot(x="Lat", y="Lng", data=dtf,
                palette=sns.color_palette(
                   palette_personalizzata,k),
                hue='cluster', legend=False,
                ax=ax).set_title('Clustering(k='+str(k)+'):
                   known customers at time t=0')
ax.set_xlabel('Latitude')
ax.set_ylabel('Longitude')
ax.scatter(start[0], start[1], c='black', marker='^')
plt.show()
```

Once the division of known customers at time t_0 in k clusters is performed, an initial solution to serve these customers is computed. Then, every time new customers enter the system, this initial solution is adjusted to incorporate them in the routes. The code to add following customers in the established clusters is outlined below. In this piece of code new customers appearing at time t_1 are included. The same procedure is also used to add new customers at time t_2 and t_3, and for this reason is not repeated.

```
# ADDING NEW CUSTOMERS ARRIVED AT t=1 IN CLOSEST CLUSTERS
new_customers = dtf[dtf["base"]==0][dtf["Time Interval"
   ]==1][["Lat","Lng"]]
dtf_new_cust = new_customers.copy()
dtf_new_customers["cluster"] = model.predict(new_customers)
# update clusters with new customers
dtf.loc[dtf["Time Interval"] == 1, "cluster"] =
   dtf_new_customers["cluster"]
# plot customers divided in clusters
fig, ax = plt.subplots()
palette_personalizzata = ["#FF5733", "#33FF57", "#3366FF",
   "#FF33A1",
    "#33FFFF", "#FF3366", "#FFFF33", "#9933FF", "#FF9933",
```

```
        "#33FF99"]
k = dtf["cluster"].nunique()
sns.scatterplot(x="Lat", y="Lng", data=dtf,
                palette=sns.color_palette(
                   palette_personalizzata,k),
                hue='cluster', legend=False,
                ax=ax).set_title('Clustering(k='+str(k)+'):
                   ␣new␣customers␣at␣t=1')
ax.scatter(start[0], start[1], c='black', marker='^')
for k,v in dic_routes_clusters.items():
    route_coordinates = dtf.loc[v, ["Lat", "Lng"]]
    ax.plot(route_coordinates["Lat"], route_coordinates["
       Lng"], linestyle='--', color='black')
ax.set_xlabel('Latitude')
ax.set_ylabel('Longitude')
plt.show()
```

B.2 K-means clustering in the rebuild procedure

In this section, the code for the k-means clustering in the rebuild procedure is described. In this algorithm, clusters are rebuilt every time new customers enter the routing problem. This implies that only the first block of code seen in the section B.1 of Appendix B is run. In particular, the first block is run four times, one for every time requests appear (i.e., t_0, t_1, t_2 and t_3). At any moment of time, the best number of clusters is detected by the elbow method.

B.3 Affinity propagation clustering in the insert procedure

In this section, the code for the affinity propagation clustering in the insert procedure is provided. In particular, clusters are formed by taking into account only customers known at time t_0. The best number of clusters is detected by the algorithm itself.

Then, each new customers is simply added to the nearest already established clusters.

```
model = cluster.AffinityPropagation()
X = dtf[dtf["base"]==0][dtf["Time␣Interval"]==0][["Lat","
   Lng"]]
dtf_X = X.copy()
dtf_X["cluster"] = model.fit_predict(X)
dtf["cluster"] = dtf_X["cluster"]
dtf.sample(5)
k = dtf["cluster"].nunique()
## plot
fig, ax = plt.subplots()
palette_personalizzata = ["#FF5733", "#33FF57", "#3366FF",
   "#FF33A1", "#33FFFF", "#FF3366", "#FFFF33", "#9933FF", "
   #FF9933", "#33FF99"]
# create a graph using the sns library
sns.scatterplot(x="Lat", y="Lng", data=dtf,
                palette=sns.color_palette(
                   palette_personalizzata,k),
                hue='cluster', legend=False, ax=ax).
                   set_title('Clustering(k='+str(k)+'):␣
                   known␣customers␣at␣time␣t=0')
ax.set_xlabel('Latitude')
ax.set_ylabel('Longitude')

ax.scatter(start[0], start[1], c='black', marker='^')
plt.show()
```

Once the division of known customers at time t_0 in k clusters is performed, an initial solution to serve these customers is computed. Then, every time new customers enter the system, this initial solution is adjusted to incorporate them in the routes. The code

to add following customers in the established clusters is outlined below. In this piece of code new customers appearing at time t_1 are included. The same procedure is also used to add new customers at time t_2 and t_3, and for this reason is not repeated.

```
# ADDING NEW CUSTOMERS ARRIVED AT t=1 IN CLOSEST CLUSTERS
k = dtf["cluster"].nunique()
new_customers = dtf[dtf["base"]==0][dtf["Time␣Interval"
   ]==1][["Lat","Lng"]]
dtf_new_customers = new_customers.copy()
dtf_new_customers["cluster"] = model.predict(new_customers)
# update clusters with new customers
dtf.loc[dtf["Time␣Interval"] == 1, "cluster"] =
   dtf_new_customers["cluster"]
# plot customers divided in clusters
fig, ax = plt.subplots()
palette_personalizzata = ["#FF5733", "#33FF57", "#3366FF",
   "#FF33A1", "#33FFFF", "#FF3366", "#FFFF33", "#9933FF", "
   #FF9933", "#33FF99"]
sns.scatterplot(x="Lat", y="Lng", data=dtf,
                palette=sns.color_palette(
                   palette_personalizzata,k),
                hue='cluster', legend=False, ax=ax).
                   set_title('Clustering(k='+str(k)+'):␣new
                   ␣customers␣at␣t=1')
ax.scatter(start[0], start[1], c='black', marker='^')
for k,v in dic_routes_clusters.items():
    route_coordinates = dtf.loc[v, ["Lat", "Lng"]]
    ax.plot(route_coordinates["Lat"], route_coordinates["
       Lng"], linestyle='--', color='black')
ax.set_xlabel('Latitude')
```

```
ax.set_ylabel('Longitude')
plt.show()
```

B.4 Affinity propagation clustering in the rebuild procedure

In this section, the code for the affinity propagation clustering in the rebuild procedure is described. In this algorithm, clusters are rebuilt every time new customers enter the routing problem. This implies that only the first block of code seen in the section B.3 of Appendix B is run. In particular, the first block is run four times, one for every time requests appear (i.e., t_0, t_1, t_2 and t_3). At any moment of time, the best number of clusters is detected by the algorithm itself.

C The routing phase

In this appendix, the written code for the execution of the routing phase is illustrated. In all algorithms the vehicle routing problem is solved by using *ortools*. It is a Python library developed by Google and it is capable to solve a large number of linear programming and optimization problems. The two main components that must be defined to solve the VRP are the *manager* and the *routingmodel*. The former needs in input the length of the distance matrix, the number of vehicles and the index of the depot. The latter takes in input the *manager* object and it is the component that defines and solves the routing problem.
Concerning the distance matrix passed as input parameter to the *manager* object, it is an array in which the cell (i, j) represents the distance from customer i to customer j. These distance values inside the distance matrix are computed by adopting the Python library *osmnx* that provides the real street distances for each pair of nodes.
Once the *routingmodel* has been defined, the Guided Local Search is the metaheuristic approach called to solve the VRP in all algorithms.

C.1 The creation of routes

In this section, the code used to solve the routing problem within each cluster is provided. In particular, it shows the creation of routes for clusters at time t_0. This means that the following code is run four times, one for every time requests appear (i.e., t_0, t_1, t_2 and t_3), and for this reason is not repeated. The final solution computed at time t_3 represents the final routes that vehicles travel to serve all customers.

```
# create network graph
G = ox.graph_from_point(start, dist=10000,network_type="
   drive")
G = ox.add_edge_speeds(G)
G = ox.add_edge_travel_times(G)
# get the node for each location (both depot and customers)
```

```
# create a new column 'node' in dtf with the node value for
    each location
dtf["node"] = dtf[["Lat","Lng"]].apply(lambda x: ox.
   nearest_nodes(G, x[1], x[0]), axis=1)
dtf = dtf.drop_duplicates("node", keep='first')
dtf.head()
# this function computes the distance shortest path between
    each node
def shortest_distance(a,b):
    try:
        d = nx.shortest_path_length(G, source=a, target=b,
           method='dijkstra', weight='travel_time')
    except:
        d = np.nan
    return d

# CLUSTERING WITH ONLY KNOWN CUSTOMERS AT t=0
dic_routes_clusters = {}
total_distance = 0
total_load = 0
for cluster in range(k):

    lst_for_dist_matrix = dtf[dtf["base"] == 1]["node"].
       tolist()
    lst_for_dist_matrix += dtf[dtf["cluster"] == cluster]["
       node"].tolist()
    lst_id_nodes = dtf[dtf["base"] == 1]["id"].tolist()
    lst_id_nodes += dtf[dtf["cluster"] == cluster]["id"].
       tolist()
    my_dict = dict(zip(lst_id_nodes, lst_for_dist_matrix))
```

```
size = list(range(0, len(lst_for_dist_matrix)))
dict_route_nodes = dict(zip(size, lst_id_nodes))

distance_matrix = np.asarray([[shortest_distance(a, b)
   for b in lst_for_dist_matrix] for a in
   lst_for_dist_matrix])
distance_matrix = (np.rint(distance_matrix)).astype(int
   )

# Parameters
driver = 1
# we need the equivalent node in the graph
start_node = ox.nearest_nodes(G, start[1], start[0])
print("start␣node:", start_node, "|␣total␣locations␣to␣
   visit␣in␣the␣cluster:", len(lst_for_dist_matrix) -
   1,
      "|␣drivers:", driver, "\n")
driver_capacity = [100]
demands = [0] + [1] * (len(lst_for_dist_matrix) - 1)
max_distance = 100000

# Create the routing index manager
manager = pywrapcp.RoutingIndexManager(len(
   lst_for_dist_matrix), driver, lst_for_dist_matrix.
   index(start_node))
# Create routing model.
routing = pywrapcp.RoutingModel(manager)

def distance_callback(from_index, to_index):
    """Returns the distance between the two nodes."""
```

```
    # Convert from routing variable Index to distance
       matrix NodeIndex.
    from_node = manager.IndexToNode(from_index)
    to_node = manager.IndexToNode(to_index)
    return distance_matrix[from_node][to_node]

transit_callback_index = routing.
   RegisterTransitCallback(distance_callback)

# Define cost of each arc.
routing.SetArcCostEvaluatorOfAllVehicles(
   transit_callback_index)

# The constraint about capacity
def get_demand(from_index):
    return demands[from_index]

demand = routing.RegisterUnaryTransitCallback(
   get_demand)

routing.AddDimensionWithVehicleCapacity(demand,
   slack_max=0, vehicle_capacities=driver_capacity,
   fix_start_cumul_to_zero=True, name='Capacity')

# The constraint about distance
name = 'Distance'
routing.AddDimension(transit_callback_index,
    slack_max=0, capacity=max_distance,
       fix_start_cumul_to_zero=True, name=name)
distance_dimension = routing.GetDimensionOrDie(name)
```

```
distance_dimension.SetGlobalSpanCostCoefficient(100)

# Initial solution that minimizes costs
parameters = pywrapcp.DefaultRoutingSearchParameters()
parameters.first_solution_strategy = (routing_enums_pb2
   .FirstSolutionStrategy.PATH_CHEAPEST_ARC)
# Metaheuristic optimization of initial solution
parameters.local_search_metaheuristic = (
   routing_enums_pb2.LocalSearchMetaheuristic.
   GUIDED_LOCAL_SEARCH)
parameters.time_limit.FromSeconds(1)
solution = routing.SolveWithParameters(parameters)

index = routing.Start(0)
route_idx = []
route_distance = 0
route_load = 0
while not routing.IsEnd(index):
    node_index = manager.IndexToNode(index)
    route_idx.append(manager.IndexToNode(index))
    previous_index = index
    index = solution.Value(routing.NextVar(index))
    route_distance += distance_callback(previous_index,
        index)
    #update load
    route_load += demands[node_index]  ## for data
route_idx.append(manager.IndexToNode(index))
my_route = [dict_route_nodes[x] for x in route_idx]
print(my_route)
dic_routes_clusters[cluster] = my_route
```

```
    print(f'distance: {round(route_distance / 1000, 2)} km'
        )
    print(f'load: {round(route_load, 2)}', "\n")
    total_distance += route_distance
    total_load += route_load

print(f'Total distance: {round(total_distance / 1000, 2)} 
    km')
print(f'Total load: {total_load}')

# ROUTING REPRESENTATION FOR ALL CLUSTERS AT t=0
fig, ax = plt.subplots()
palette_personalizzata = ["#FF5733", "#33FF57", "#3366FF",
    "#FF33A1", "#33FFFF", "#FF3366", "#FFFF33", "#9933FF", "
    #FF9933", "#33FF99"]

# Scatter plot for clusters
sns.scatterplot(x="Lat", y="Lng", data=dtf,
                palette=sns.color_palette(
                    palette_personalizzata, k),
                hue='cluster', legend=False, ax=ax).
                    set_title('Clustering(k=' + str(k) + '):
                    routing representation at t=0')

# Add the start point (depot)
ax.scatter(start[0], start[1], c='black', marker='^')

ax.set_xlabel('Latitude')
ax.set_ylabel('Longitude')
```

```
for k,v in dic_routes_clusters.items():
    route_coordinates = dtf.loc[v, ["Lat", "Lng"]]
    ax.plot(route_coordinates["Lat"], route_coordinates["
       Lng"], linestyle='--', color='black')

plt.show()
```