ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MASTER OF SCIENCE THESIS

A Heuristic Algorithm for the Fuel Delivery Problem

Author:
Dimitrios BOUKOSIS

Supervisor:
Dr. Konstantinos ANDROUTSOPOULOS

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

[Management Science and Technology]

February 10, 2019


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Abstract

Management of Science and Technology

Master of Science

A Heuristic Algorithm for the Fuel Delivery Problem

by Dimitrios BOUKOSIS

During recent years distribution systems have become increasingly complex. This development is partly due to the high number of company mergers which leave distribution planners with ever bigger and complex problems. Another fact complicating distribution is the increased focus on timeliness in the distribution chains, as intelligent planning offers potential savings in capital bindings in costs related to stock and distribution. In other words, time has become an extremely valuable resource. Nowadays most distribution systems must operate under strict temporal restrictions. This fact has caused an increasing interest in dynamic transportation models and systems in which data are considered to be time-dependent.

In this thesis the multi-compartment counterpart of the conventional vehicle routing problem will be studied. The traditional vehicle routing problem (VRP) consists of constructing minimum cost routes for the vehicles to follow so that the set of customers are visited exactly once. The VRP is an important subproblem in a wide range of distribution systems and a lot of effort has been devoted to research on various aspects of the VRP. However, most of the times different distribution planning problems may arise in the everyday life. Counterparts of the problem need to be solved, because there are different constraints for each business.

The thesis begins by introducing the vehicle routing problem and its counterparts and discussing the differences between them. The existing literature and its counterpart, the Multi-Compartment VRP is explained later, and the problem description is following, along with an extended explanation of the methodology used to tackle the problem. Lastly, there have been many tests with various datasets, and the computational results will be presented.


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Acknowledgements

To my life-coach, my grandfather Menelaos Loulakis: because I owe it all to you. Many Thanks!

I would first like to thank my thesis advisor Prof. Konstantinos Androutsopoulos of the Management Science and Technology at Athens University of Economics and Business. The door to Professor's Androutsopoulos office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my own work, but steered me in the right direction whenever he thought I needed it.

Finally, I must express my very profound gratitude to my parents, Evaggelos Boukosis and Eleni Loulaki, and my brother, Menelaos Boukosis, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author,

Boukosis Dimitrios


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Contents

List of Abbreviations

ACO	**A**nt **C**olony **O**ptimization
ALNS	**A**daptive **L**arge **N**eighborhood **S**earch
BL-VRP	**B**ackhauls and **L**inehauls **V**ehicle **R**outing **P**roblem
CVRP	**C**apacitated **V**ehicle **R**outing **P**roblem
DSS	**D**ecision **S**upport **S**ystem
DVRP	**D**ynamic **V**ehicle **R**outing **P**roblem
ETA	**E**arliest **T**ime of **A**rrival
FSMVRP	**F**leet **S**ize and **M**ix **V**ehicle **R**outing **P**roblem
GRASP	**G**reedy **R**andomized **A**daptive **S**earch **P**rocedure
HFMCVRP	**H**eterogeneous **F**leet and **M**ulti **C**ompartment **V**ehicle **R**outing **P**roblem
HFVRP	**H**eterogeneous **F**leet **V**ehicle **R**outing **P**roblem
ILS	**I**terated **L**ocal **S**earch
IRP	**I**nventory **R**outing **P**roblem
LNS	**L**arge **N**eighborhood **S**earch
MC-IRP	**M**ulti-**C**ompartment **I**nventory **R**outing **P**roblem
MD-VRP	**M**ulti-**D**epot **V**ehicle **R**outing **P**roblem
MC-VRP	**M**ulti-**C**ompartment **V**ehicle **R**outing **P**roblem
PD-VRP	**P**ickup and**D**elivery **V**ehicle **R**outing
PSO	**P**article **S**warm **O**ptimization
RCL	**R**estricted **C**andidate **L**ist
TS	**T**abu **S**earch
TSP	**T**ravelling **S**alesman **P**roblem
TW-VRP	**T**ime **W**indows **V**ehicle **R**outing **P**roblem
VLSN	**V**ery **L**arge **S**cale **N**eighborhood Search
VNS	**V**ariable **N**eighborhood **S**earch
VRP	**V**ehicle **R**outing **P**roblem
VSP	**V**ehicle **S**cheduling **P**roblem


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter 1

Introduction

In recent time, the global antagonism at an operational level, has brought broad development in the fields of management science and technology respectively. Companies have implemented various ways of tackling their problems, in all areas of business.

A large part of business practice is the supply chain management, which includes the design and management of all activities involved in the procurement, conversion and control of the supply chain (Marinakis I., 2008). It also includes the core components of co-ordination and collaboration with corporate channels, which may be suppliers, intermediaries, third-party service providers and customers. In essence, supply chain management integrates the management of supply and demand within and between companies. Its primary responsibility is to link important business functions and business processes to a consistent and high - performing business model.

An important part of the Supply Chain Management is Logistics. Logistics include managing the collection, storage, movement and distribution of materials, within the enterprise. In particular, the transport sector is currently dominated by technologies that support real-time decision making. The use of modern computing tools allows for rapid decision-making on applicable routes, optimal choices, reliable and rapid evaluation of alternative designs, and more general design of services in dynamic business environments.

Transportation costs vary from one sector to another but are on average more than 10% of the total cost of the product (Desrosiers Jacques, 1995). Consequently, it is of utmost importance for each company, to improve performance in order to achieve a proper use of its staff and distribution equipment and to minimize the cost or maximize the profit.

1.1 Outline of the Thesis

This thesis proposes a meta-heuristic algorithm for a fuel delivery VRP. The presentation will be focusing on providing the reader with a theoretical basis for studying the VRP and the targeted problems (Multi-compartment VRP and Fuel Delivery), as well as providing the guidelines of the implementation of the Large Neighborhood Search (LNS) method for solving the problem.

In this present chapter, we introduce the Vehicle Routing Problem (VRP) and its most common counterparts. We give a discussion of the differences between each counterpart, paying particular attention in the multi-compartment VRP.

In Chapter 2, we provide the literature survey, dealing with the multi–compartment delivery problem. The main goal of the chapter is to provide the reader with a consistent overview of the work on the MCVRP and the progress made within this area throughout the past decades.

Chapter 3 is focused mainly in the comprehension of the problem we are facing and the tools used already, in the effort to tackle it.

In Chapter 4, we are describing the problem, its' formulation, how we are tackling and why we are using the current algorithm.

In Chapter 5, we discuss the Solution algorithms used in the problem and how the data are formulated.

In Chapter 6, we are presenting the computational results and the experiments, while the figures created are on the appendix.

1.2 The Capacitated Vehicle Routing Problem

The most well-studied and undoubtedly the most known routing problem in the management science is the Travelling Salesman Problem, in which a salesman starting from a point, e.g. a distribution center or a depot, must return to the same spot after visiting a fixed number of customers just once each. The objective for the TSP is to minimize the total distance traveled by the salesman It first appeared in a paper

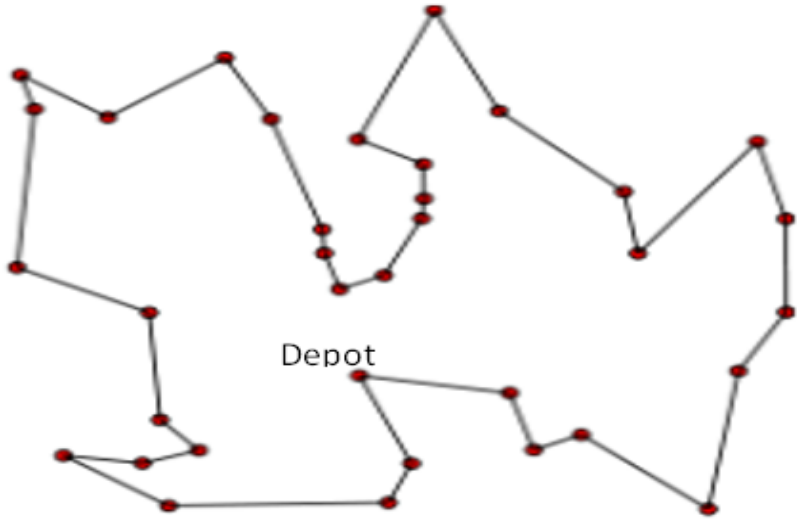


FIGURE 1.1: Common TSP Solution (Source: en.wikipedia.org).

by (Dantzig G. B., 1959) in which first algorithmic approach was written and was

applied to petrol deliveries. Often, the context is that of delivering goods located at a central depot to customers who have placed orders for such goods. The objective of the CVRP is to minimize the total route cost. In 1964, Clarke and Wright improved on Dantzig and Ramser's approach using an effective greedy approach called the savings algorithm.

The CVRP has many obvious applications in industry. In fact, the use of computer optimization programs can give savings of 5% to a company (Geir Hasle, 2007) as transportation is usually a significant component of the cost of a product (10%) (Comtois C., 2013) - indeed, the transportation sector makes up 10% of the EU's GDP. Consequently, any savings created by the VRP, even less than 5%, are significant.

The Capacitated Vehicle Routing Problem (CVRP) is a generalization of the TSP, as the CVRP has the extra constraint that all the customers cannot be visited by only one vehicle, because the total demand exceeds the capacity of the said vehicle. Hence, the VRP consists in determining m vehicle routes, where a route is a tour that begins at the depot, visits a subset of the customers in a given order and returns to the depot. Product distribution involves servicing, over a given period of time, a

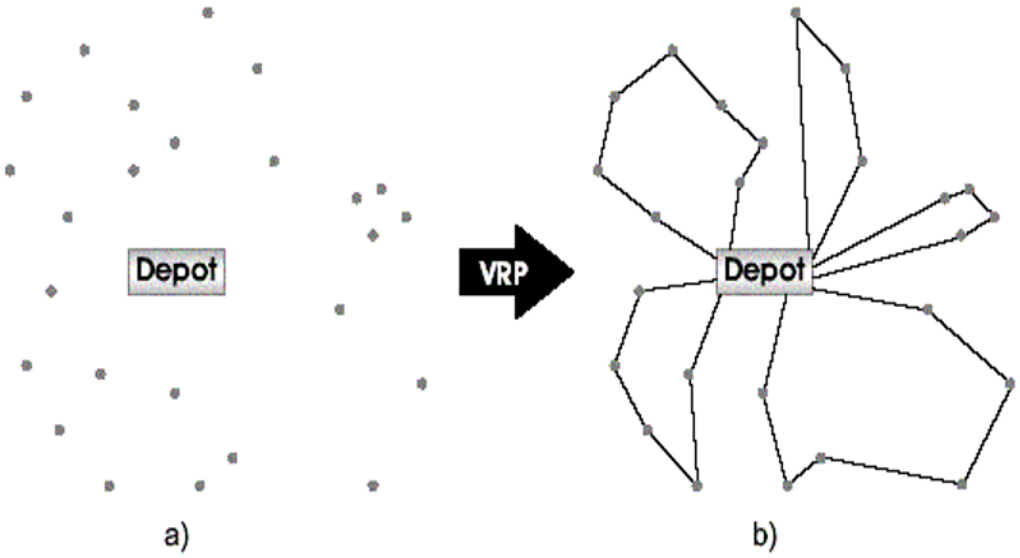


FIGURE 1.2: Common CVRP Solution (Source: neo.lcc.uma.es)

set of customers from a set of vehicles dedicated to a particular storage space, used by a certain number of drivers and making their moves using a particular road network. In general, a proper solution to the vehicle routing problem results in the determination of a set of routes, starting and ending in a depot, satisfying customer requirements, not breaking any of the limitations and minimizing distribution costs. In Figure 1.1 we see that from a cluster of points (a), the vehicle routing problem has defined the routes (b).

The problem can be modelled as a mixed linear programming problem, which is as follows:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{K=1}^{K} x_{ijk} \tag{1.2.1}$$

S.t:

$$\sum_{k} y_{ik} = \begin{cases} 1, & i = 2, \dots, n, \\ m, & i = 1 \end{cases} \tag{1.2.2}$$

$$\sum_{i} q_i y_{ij} \leq Q_{k,} \qquad \forall k \in V \tag{1.2.3}$$

$$\sum_{j} x_{ijk} = \sum_{j} x_{jik} = y_{ik} \qquad \forall i \in V, \forall k \in K \tag{1.2.4}$$

$$\sum_{(i,j) \in S} x_{ijk} \leq |S| - 1 \qquad \forall S \subseteq \{2, \dots, n\} \tag{1.2.5}$$

We have a set of customers V and a set of vehicles K.

C_{ij} : is the unit cost for travelling from customer i to customer j.

x_{ij} : Objective variable. Takes the value 0 if the arc (i, j) is not used, and the value 1 if it is used.

y_{jk} : Takes the value 1 if the customer j is visited from the vehicle k, 0 otherwise.

Q_k : Capacity of vehicle k.

q_i : Demand from customer i.

The Objective function sums the cost for each movement from the customer i to customer j, for each vehicle k. The constraint 1.2.2 helps us ensure, that each customer is being assigned into only one vehicle, besides the depot, which gets visited from each vehicle. The second set of constraints 1.2.3 secures that each vehicle does not serve more than the quantity that it capacitates.

The 1.2.4 set of constraints ensures that a vehicle k visits a customer j and from this customer arrives in another customer i. Finally, the last set of constraints 1.2.5 does not allow sub-routes, besides the sub-routes that contain the depot.

1.3 Time Windows VRP

The VRPTW is an important generalization of the VRP and a basic distribution management problem that can model many real-world problems and which consist of

designing a set of minimum cost routes, originating and terminating at a central depot, for a fleet of vehicles, which services a set of customers with known demands. The customers must be assigned exactly once to vehicles such that the vehicle capacities are not exceeded. The service at a customer must begin within the time window defined by earliest time and the latest time when the customer permits the start of service. Some of the more useful applications of the VRPTW include bank deliveries, postal deliveries, industrial refuse collection, national franchise restaurant deliveries, and school bus routing and security patrol services.

The time windows VRP can be determined as follows: We have a set of customers that are scattered in a geographic area and must be served by a number of vehicles originally placed in a given warehouse. Each customer has a demand to be fulfilled and the customer determines a period of time for the delivery to take place. Customers are served by vehicles with limited capacity so that the total load of each vehicle does not exceed the capacity of the vehicles. The goal is to find a set of routes for vehicles, where each route starts and ends in the depot, serves a subset of customers without violating the capacity and time windows constraints, minimizing the total length of the routes.

In fact, the time windows VRP is an extension of the problem of limited capacity (CVRP) in which capacity constraints apply in the same way as before, but in addition each customer must be served within a time period $[ai, bi]$, which is called a time window. In addition, data is provided for the travel time for each arc (i, j) and of course the time service for each customer. Service for each customer must begin within the time window that the customer states and the vehicle must remain in the customer's place during delivery. In addition, if a vehicle reaches a customer before the specified time, in most cases the vehicle is allowed to remain on the customer's site until the time window starts.

In most cases, the cost and travel matrix coincide, and time windows are determined based on the fact that all vehicles leave the warehouse at time 0. In addition, time windows require a complete orientation of each route even if the original tables are symmetrical. So, most of the time the problem is asymmetrical.

There are two kinds of time windows, the loose that if a vehicle reaches a customer some time outside of the time window it can start its service at that time, as opposed to the hard time windows that the customer service needs to be done strictly within the time window. In these cases, if a vehicle reaches the customer earlier, it will wait for the service to begin.

1.4 VRP with Backhauls and linehauls

This problem is also known as the linehaul-backhaul problem, an extension of the Capacitated VRP (CVRP) where the customer set is partitioned into two subsets. The first subset contains the linehaul customers, each requiring a given quantity of product to be delivered. The second subset contains the backhaul customers, where a given quantity of inbound product must be picked up. This customer partition is extremely frequent in practical situations. A common example is that of the grocery industry, where supermarkets and shops are the linehaul customers and grocery suppliers are the backhaul customers. It has been widely recognized that in this mixed distribution–collection context a significant saving in transportation costs

can be achieved by visiting backhaul customers in distribution routes (Golden B.L., 1985).

In this problem, lies a really fundamental constraint, concerning the two set of customers. Whenever in a route customers of both sets are serviced, all customers of the first set must be serviced before the customers of the second set.

More precisely, the VRPB can be stated as the problem of determining a set of vehicle routes visiting all customers, that:

- each vehicle performs only one route starting and finishing at the depot,
- each customer is visited by only one vehicle,
- for each route the total load associated with linehaul and backhaul customers does not exceed, separately, the vehicle capacity,
- on each route the backhaul customers, if any, are visited after all linehaul customers,
- routes containing only backhauls customers are not allowed and
- the total distance travelled by the vehicles is minimized.

The precedence constraint (iv) is practically motivated by the fact that vehicles are often rear loaded. Hence the on-board load rearrangement required by a mixed service is difficult, or impossible, to carry out at customer locations. Another practical reason is that, in many applications, linehaul customers have a higher service priority than backhaul customers.

1.5 Multi-depot VRP

A very interesting variation of the vehicle routing problem is the variation that uses more than one warehouse. There are two ways to tackle this problem:

- Each of the warehouses has its own number of vehicles and its own customers to serve. In such cases, the company has to deal with a number of simple VRPs.
- In the latter case, a vehicle starts from a warehouse and either ends at another or stops to load, for example, additional products, and then ends the route in any depot.

This particular problem can be considered as a clustering problem, since the goal is to find the routes of the vehicles belonging to each warehouse. The problem can be solved as a two-phase problem, in the first phase we have clients assigned to the warehouses and the second phase creates the routes for each warehouse and for each vehicle.

1.6 Dynamic VRP

The difference in this category of problems lies in the fact that previously, the geographic location of all the customers was known from the beginning. In contrast, in the dynamic VRPs a new customer may come of when the route is carried out. This type of problem is mainly encountered in tasks related to the provision of services,

in jobs that the driver receives real-time commands, such as that of a taxi driver or ambulance driver.

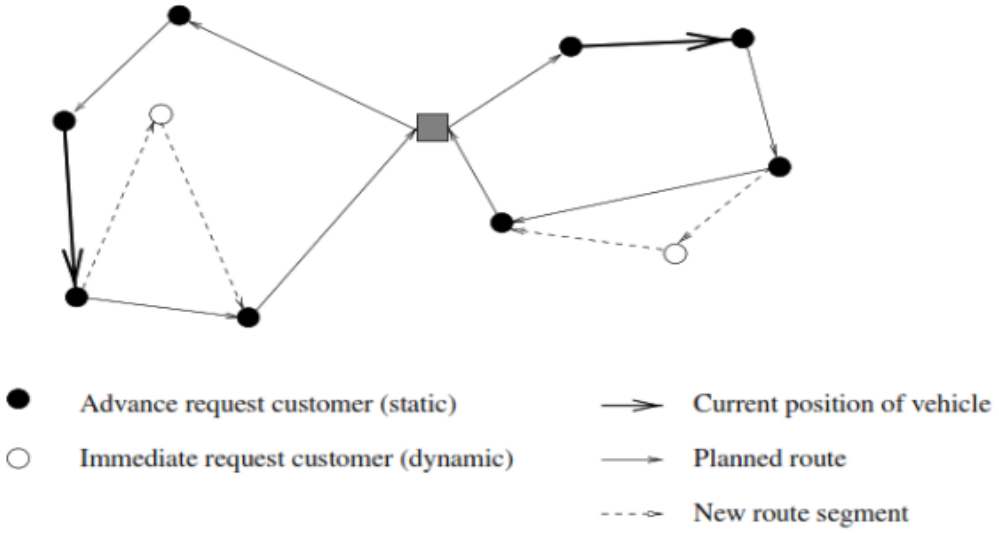


FIGURE 1.3: A dynamic vehicle routing scenario with 8 advance and 2 immediate request customers. (Source google.com)

In figure 1.3 a simple example of a dynamic VRP is shown. In the example, two uncapacitated vehicles must service both advance and immediate request customers without time windows. The advance request customers are represented by black nodes, while those that are immediate requests are depicted by white nodes. The solid lines represent the two routes the company has planned before the vehicles left the depot. The two thick lines indicate the vehicles' positions at the time the requests are received. As we can see the new request, thus the insertion of a new customer in the route, will result to delays. These delays, might be small (right hand side route) or large (left hand side).

1.7 The Vehicle Scheduling Problem

The vehicle scheduling problems are considered Vehicle Routing Problems, but with more constraints, that relate with schedules and times that some tasks should complete. The VRPs pay attention in time characteristics of any problem. However, in VSP, time is related with any activity. Whether a task is feasible or not is influence by time and space constraints. For example, only one vehicle cannot service two different location with the same delivery time. The succession of activities for each vehicle in time and space is the key characteristic in the vehicle scheduling problem (VSP). Three basic constraints of this problem are the following:

- The length of time that each vehicle can serve customers is limited.
- Some customers can be served only by a certain type of vehicle.
- The presence of many depots that the vehicles can be housed.

In this thesis, we have a mix of all these problems, as we are tackling with a multi-compartment with time windows vehicle scheduling problem with loading and unloading constraints. The multi-compartment VRP will be analyzed in the next chapter.

It is obvious that the aforementioned problems can mix with the others, depending on the problem. We can have a Pickup and Delivery problem with time windows, or a Backhauls with Time windows etc. There are many counterparts, that have not been analyzed to a maximum degree. A figure of the VRP's most important counterparts is following.

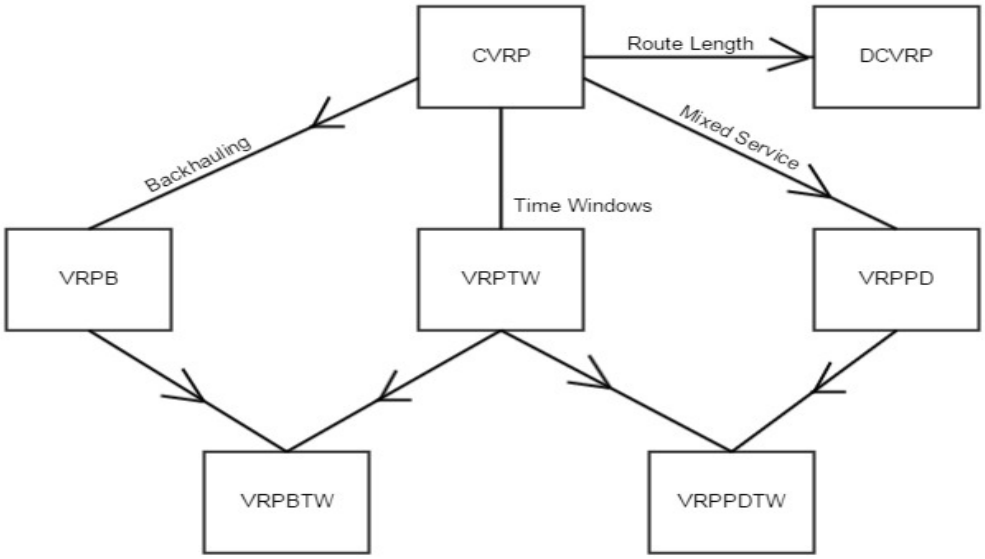


FIGURE 1.4: Map of VRPs most important counterparts

Chapter 2

Multi-Compartment VRP

The problem we are tackling in this thesis is practically the multi-compartment Vehicle Routing problem (MC-VRP), which is a generalization of the classical, capacitated VRP.

In many industries, the products that are to be delivered in a customer are homogeneous, namely they can be transferred together. However, there are many industries, that their goods are not homogeneous. In these cases, a solution was to dedicate one vehicle for one product, but the transportations costs grew incredibly. The solution was to allow transporting inhomogeneous products on the same vehicle together, but in different compartments. The rationale for using vehicles with multiple compartments is to be able to carry products of different temperature or composition in the same vehicle. Thus, we concluded in the multi-compartment VRP.

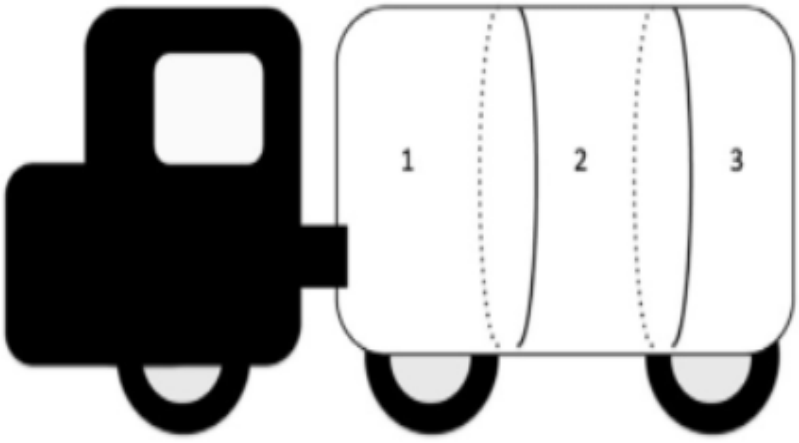


FIGURE 2.1: A vehicle with three compartments (Coelho L, Laporte G., 2015)

The strict MC-VRP is defined on an undirected network, including one depot and a set of n customers. The depot can store a number of products, which must be delivered to customers by a fleet of vehicles that have compartments of limited capacities (Rodrigue J., 2017). The number of products and compartments must be the same, and each compartment is reserved for a product only. The customers can

place orders of one or more products, and each product ordered by a customer must be delivered by only one vehicle.

Although, the literature has begun investigating the former strict version of the MC-VRP, several variants began to emerge. The first change is to whether the type of fleet should only be homogeneous, namely all the vehicle be the same and have the same number of compartments, or heterogeneous with different number of compartments for each vehicle and possible different capacity for each compartment in any vehicle.

Another alteration is the product dedicated compartments. If a compartment is dedicated to only one product, then a serious limitation to a possible solution is introduced. On the other hand, if there are no product-dedicated compartments there is a broad spectrum of solutions to be investigated. There is also the possibility to insert more than one order items per compartment and to split the orders. That means that different products ordered can be brought by different vehicles.

Lastly, a new alteration was recently proposed, and this thesis will primarily target this alteration, where there are unloading constraints. That means, that compartments must be unloaded in a certain way, that are predetermined by management, or law. This variation is frequently seen in industries that are transporting really heavy products, such as petroleum or food, where truck balance and equal force distribution is a crucial part of the routing procedure. The variant will be described thoroughly subsequently.

2.1 Multi Compartment VRP Applications

Despite the vast amount of literature about the Vehicle Routing problems, only very little attention has been paid to vehicles with compartments that allow transportation of inhomogeneous products on the same vehicle, but in different compartment, although this generalized VRP is essential for several industries, like the distribution of food or petrol, as mentioned above.

The main applications, in which vehicles with multiple compartments are used are the transportation of petroleum products tackled by (R. J. Cornillier F., 2008), where trucks transport more than one petroleum products from refineries to customers in a single trip, deliveries of food and grocery items to convenient stores tackled by (Chajakis E., 2003) and even collection of source-separated waste streams (Apotheker, 1990). When the clients to be visited generate different types of waste, collection firms have the flexibility to deploy traditional, in-partitioned vehicles, which collect one type of waste at a time (separate collection), or they may deploy vehicles with multiple compartments to co-collect different waste streams. Co-collection involves the pickup of separated, bagged recyclables at the same time as garbage in a packer truck. The recyclables are bagged to keep them separate from the garbage, hence the multi-compartment VRP. One last application of the multi-compartment VRP is used in the transportation of animals from farms to slaughterhouses, known as the livestock collection problem (Oppen Johan, 2008).

2.1.1 Deliveries of Food and Grocery Items

Convenience stores represent the fastest growing segment of the food retail industry. They are relatively small sized and have extended hours of operation. Their main purpose is to offer groceries and other food items to customers outside business hours, isolated locations and even highways. These stores, unlike the supermarkets, that have typically large storage spaces in the back, have space limitations and need a very tight control of inventories. This is the main reason why convenience store orders must be small, and served by a single distributor that can deliver various types of goods, such as dry, frozen or refrigerated items together in one truck. In contrast, supermarkets place large orders, often to many distributors that deliver products with various properties in different truck.

Another problem that is tackled for convenience stores is clearly business oriented. Supermarkets typically have special doors and facilities and depots in the back, so that trucks can be unloading without interfering with customers or other functions of the business. Most convenience stores do not have such facilities; therefore, all orders have to be delivered through the front door. Clearly, when only one truck delivers instead of more, the inconvenience is minimized. Hence, that makes delivery of mixed orders in a single truck to convenience stores preferable.

Vehicle for deliveries to convenience stores have a cooling unit attached to the top front end of the cargo space behind the driver's cabin. In addition to the main back door, vehicles often have one or more side access doors. They may or may not have separate spaces or compartments to hold items at different temperatures. A compartment can often be as simple as a box filled with dry ice. In larger trucks, compartments are formed by separating spaces using bulkheads that can by fixed or movable. Movable bulkheads slide along special rails in the inside walls of the truck to allow the volume capacity of compartments to vary according to the cubic volume of the items they must hold during a trip. The movement of bulkheads is often limited by the position of side doors

2.1.2 Cattle Food to Farms

As addressed previously in this thesis, MC-VRP is significantly different and studied a handful of times. As opposed before, each compartment of the vehicle is dedicated to only one "product". In fact, a "product" may represent a vast variety of goods that share common characteristics and require a specific compartment.

Another example of a genuine MC-VRP, is found at many farms in Western France, in the distribution of cattle food to these farms. These farms use homogeneous fleet of vehicles, and although the vehicles are almost identical and the compartments don't have differences, sanitary rules recommend to always use the same compartment for each species. For example, due to the sensitivity of rabbits to viruses, it is very risky to transport food in a compartment previously used for the transportation of food for bovines.

This example makes clear of how natural and common the MC-VRP is.

2.1.3 Petroleum Deliveries

In recent years a great deal of research has been carried out on the optimization of the fuel distribution planning and management. All the application that developed for this aspect have shown that distribution planning techniques can save up to 25% of total travel costs (Ballou, 1998). These savings can have a great impact to a company's finance as transportation is present in all phases of business, manufacturing, procurement, selling, and represents about 10 – 25 % of the final product cost.

In the fuel delivery problem, it is necessary to supply a set of clients using a fleet of vehicles that start from one or more warehouses to reach clients using the available transportation network. The solution produced by solving the problem is a set of routes, where each route is associated with a truck that leaves and returns to the warehouse. The set of all routes must satisfy the client orders and the operational constraints set by the organization. The objective is to minimize the total travel cost, expressed either by travel time, fuel usage or more complex generalized cost functions.

We can classify the problem further, according to rules applied on the assignment of order items to vehicle compartments. Many more and of different type problems may arise by enforcing or disregarding the following constraints:

- Each compartment of the vehicle is dedicated to carry a specific type of product. Hence, two variations of the problem arise.
- Each compartment can be assigned up to one order item (compartment split).

Table 1 provides an overview of the major features of the literature and clarifies the variations of the MC-VRP used in the fuel delivery problem. As seen below we can separate the problems according to some of their features. They can either be homogeneous or heterogeneous, have product dedicated compartments, have one or more number of items per compartment, allow or disallow order split and have or not unloading constraints.

If the fleet of vehicles are the same with one another are called homogeneous, and if are different are called heterogeneous. Product dedicated compartments constraints arise from the fact that two or more types of petroleum cannot be mixed together. Due to lack of debit meter (sometimes called flow meters), which implies that whenever a delivery is made, the full content of the compartment must be emptied, the number of order items per compartment must be one or in a generalization of the problem more than one order items are allowed per compartment. The order split associates with the number of times a customer is allowed to be visited, in order to fulfill the order. Lastly, compartments must be unloaded in a specific order.

Type of Fleet	Product Dedicated Compartments	Number of Order items per Compartment	Order Split	Unloading Constraints
Homogeneous	Yes	One	Allowed	Yes
Heterogeneous	No	Many	Not Allowed	No

2.2 Previous Related Work

As previously mentioned, multi-compartments vehicle routing problems arise from the need of industries to use vehicles with more than one compartments, due to some constraints, such as mixed product prohibition. Many problems are tackled

using a MCVRP modelling including the fuel delivery problem ((Brown G., 1981); (Avella P., 2004); (R. J. Cornillier F., 2008); (Derigs U., 2011); (Coelho L., 2015), retail food distribution (Fallahi A., 2008), oil and waste collection (Muyldermans L., 2010); (Lahyani R., 2014); (Henke T., 2017).

The main classification of this special category of vehicle routing problems extends to whether the size of the shipments is specified by the client or the supplier. The former classification leads to Multi-Compartment VRPs and Multi-Compartment Inventory Routing Problems. Inventory Routing problems is the problem where customers have to be served over a discrete time horizon by a fleet of capacitated vehicles starting and ending their routes at a depot. The important is to decide, in each time period, how much to deliver to each customer and the routes of the vehicle in such a way that the sum of inventory and transportation costs is minimized (Bell W., 1983). The main difference between IRP and VRP is that the demand of a period is not requested to be served in that exact period, but can be served during a previous visit to the customer.

Both categories of the MC-VRPs (MC- IRP & MC-VRP) can furtherly be classified according to rules applied on the assignment of order items to compartments, as previously mentioned (product dedicated compartments, number of order items in the compartment and order splits). Several research papers have emerged from the petrol station replenishment problem. Cornilier et al. (2008 and 2009) deal with this MC-IRP problem. The compartments are not equipped with debit meters, no compartment split is allowed and each station must be visited by a single vehicle. In addition, this is the only work on Multi-Compartment Routing Problems that tackles the effect of unloading sequence on the driving stability. They modelled the problem considering unloading the front compartments of each trailer last.

Besides the former, various research has been done on the MC-VRPs and can be classified to two groups, depending on the dedication of the compartment to a specific type of products or not. Split deliveries allowed, namely the orders' items of a customer can be delivered by more than one vehicle, and type product dedication to a compartment, are constraints found in the work of (Fallahi A., 2008) and (Mendoza J.E, 2010). MC-VRPs with no product dedicated compartments are found in the work of (Brown G., 1981), (Brown G. G, 1987), (Avella P., 2004), (Derigs U., 2011) and (Lahyani R., 2014). Brown and Graves (1981 and 1987) presented a DSS (Decision Support System) for a real-time dispatch of petroleum tank trucks. Both papers tackled the dispatching problem, that involves building single or multiple order trips, through assigning orders to the available vehicles. In the extension of their work in 1987, each vehicle is able to deliver items collected from different loading facilities. Their work aimed to maximize the vehicle utilization.

(Avella P., 2004) came across a fuel delivery problem. Each vehicle can perform more that one delivery of fuel per day, under the condition of not exceeding the shift duration set by regulation and management. The fleet is heterogeneous and each compartment can have up to one order item. Order split is not allowed. (Derigs U., 2011) define a multi-compartment VRP, under a homogeneous fleet of vehicles. Order split is allowed, but each customer's order item must be delivered through a single visit.


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter 3

Solution Algorithm

3.1 Exact Methods

In the class of exact methods one can find some classical algorithms, such as dynamic programming, branch and bound, branch and cut etc., developed in the operations research community and constraint programming, developed in the artificial intelligence community. These algorithms are based on finding the exact optimal solution by searching the interesting search space and subdividing the problem into smaller simpler problems.

Dynamic Programming is based on the division of a problem into simpler, smaller subproblems. The procedure is based on Bellman's principle that states that "the sub-policy of an optimal policy, is itself optimal" (Bellman, 1957). This procedure eliminates partial decision sequences that are impossible to lead to optimal solution.

The search space in branch and bound algorithm is explored by dynamically building a tree, whose root node represents the problem being solved and its whole associated search space. The leaf nodes are the potential solutions and the internal nodes the sub-problems of the total solution space. The important feature of branch and bound algorithm is that we may find out which subtrees do not contain the optimal solution; hence we can eliminate them and stop further investigation.

Constraint programming is a language built around concepts of tree search and logical implications (Russell S., 1995). All optimization problems in constraint programming are modeled as a set of variables linked to a set of constraints. These variables take values on a finite domain of integers. The constraints may have any format, whether it is mathematical or symbolic.

Exact methods can be applied to small instance of difficult problems, such as quadratic assignment with up to 30 objects, flow-shop scheduling with up to 100 jobs and 20 machines, or Capacitated VRP with up to 60 customers (Talbi., 2009).

3.2 Heuristic Algorithms

Nowadays, as the size of the problems we are trying to solve and its complexity increase, solving a problem of optimization and particularly combinatorial optimization is becoming more and more difficult, namely finding a totally optimal solution at a reasonable time has become virtually impossible. To solve problems of this kind,

we often resort to different techniques that lead us to an almost optimal, yet satisfactory solution. A solution of a heuristic algorithm is accepted when it meets the criteria for the quality of the solution and its ease of acquisition.

For all optimization problems, there is not only one algorithm that provides the optimal solution, but several algorithms have been developed, that appropriately fit in each different occasion, regarding precision or computational time. With regard to the quality of the solution, in some problems it is impossible to find the optimal solution for a problem in a satisfactory time.

The algorithms are divided into different categories:

• *Greedy algorithms.*

A greedy algorithm is a repetitive process of strictly defined steps, that through it a solution of the problem under consideration is gradually being constructed. In many problems, a greedy strategy generally does not produce an optimal solution, but yet an avid approach can provide local best solutions that reach a global best solution within a reasonable time. However, because these algorithms are characterized as myopic, do not construct the best or a near optimal solution. In most cases, they provide satisfactory initial feasible solution, which is improved by other algorithms. Greedy algorithms are very popular techniques as they are simple to design. Moreover, greedy algorithms have in general a reduced complexity compared to iterative algorithms. However, in most of optimization problems, the local view decreases their performance compared to iterative algorithms.

• *Local search algorithms.*

Local search is based on the earliest optimization method, test method, and error (Papadimitriou C., 1982). Local search has proven to be very successful in practice in a very large number of problems. A local search algorithm does not gradually build a solution at each iteration (such as a construction algorithm) but modifies a current solution through movements, seeking to find a better solution. A movement is a specific execution of a "move type". In the literature the term move type is also referred to as local search operator or neighborhood type.

3.3 Metaheuristics

Metaheuristics algorithms are resolution methods that combine local search processes and higher-level strategies to create a process that is capable of escaping from a local minimum (Marinakis I., 2008). Unlike exact methods, metaheuristics allow to tackle large-size problem instances by delivering satisfactory solutions in a reasonable time. In recent years, most algorithms developed to solve combinatorial optimization problems belong to this category. Metaheuristics typically use more traditional heuristic algorithms as their sub-routines. Several times, it is possible to allow a metaheuristic algorithm steps that lead to an inaccessible intermediate solution, or a worse, concerning cost, solution, at one step of the process. The reason this is allowed is practically the concept behind these algorithms, which is to avoid local minimum. Namely to obtain a better overall solution.

Many classification criteria may be used for metaheuristics (Talbi., 2009):

• **Nature inspired**. Many metaheuristics are inspired by natural processes. Evolutionary algorithms and artificial immune systems from biology. Ants, bees' colonies from swarm intelligence and simulated annealing from physics.

• **Memory usage**. Some metaheuristics are memoryless, namely no information is used during the search. Some examples are Grasp, or simulated annealing. Other metaheuristics, such as tabu search, use memory extracted during the search.

• **Deterministic versus stochastic**. Deterministic algorithms solve problems by making deterministic decision (e.g. selecting the best path), while in stochastic metaheuristics, some random rules are applied during the search (random selection in Large Neighborhood Search).

The elements used by these algorithms and metaphorically are observed in nature are as follows:

• Use a number of iterative tests.

• Include one or more agents (neurons, molecules, etc.).

• They operate (in the case of multi-agents) based on a mechanism of cooperation and competition.

• Include procedures for self-modifying heuristic parameters or even representing the problem.

3.4 Local Search based Metaheuristics

These types of heuristics have as input an initial solution, most of the times from a greedy algorithm, and are moving towards neighboring solutions, while applying changes to the solution, until a stop criterion is met. Some well known types of local search-based metaheuristics are:

• *Iterated Local Search*

This metaheuristic extends the local search method to avoid getting stuck in a local optimum, by iterating the implemented local search with different starting points of the neighborhood, which usually end up in a different solution. The best of these solutions returns when this iterative process ends.

• *Tabu Search*

Tabu search algorithm was proposed by (Glover, 1989). Tabu algorithm, uses a local search operator to search for better solutions, by creating a neighborhood around a starting point, but it accepts non-improving solutions to escape from local optima when all neighbors are non-improving solutions. Usually, the whole neighborhood is explored in a deterministic manner. As in local search, when a better neighbor is found, it replaces the current solution. When a local optimum is reached, the search carries on by selecting a candidate worse than the current solution. The best solution in the neighborhood is selected as the new current solution even if it is not improving the current solution. Tabu search may be viewed as a dynamic transformation of the neighborhood. This policy may generate cycles; that is, previous visited solutions could be selected again.

To avoid cycles, TS discards the neighbors that have been previously visited. It memorizes the recent search trajectory. Tabu search manages a memory of the solutions or moves recently applied, which is called the tabu list. This tabu list constitutes the short-term memory. At each iteration of Tabu Search, the short-term memory is updated. Storing all visited solutions is time and space consuming. Indeed, we have to check at each iteration if a generated solution does not belong to the list of all visited solutions. The tabu list usually contains a constant number of tabu moves. Usually, the attributes of the moves are stored in the tabu list.

• *Simulated Annealing*

This metaheuristic, proposed by (Kirkpatrick S., 1983) is used to solve the type of problems, where finding a good solution in a short time period is important. Simulated Annealing selects a possible move at random, and uses probability to decide whether to accept a solution or not if the quality is poorer than the previous one. The probability factor decreases during the search, which leads to only good quality solutions to be accepted at the end of the search.

• *Variable Neighborhood Search*

The VNS algorithm has been proposed by (Mladenovic M., 1995). The basic idea of VNS is to successively explore a set of predefined neighborhoods to provide a better solution. It explores either at random or systematically a set of neighborhoods to get different local optimum and to escape from them. VNS exploits the fact that using various neighborhoods in local search may generate different local optimum and that the global optima is a local optimum for a given neighborhood.

• *GRASP*

The greedy randomized adaptive search procedure (known as GRASP) is a metaheuristic algorithm commonly applied to combinatorial optimization problems.

GRASP typically consists of iterations made up from successive constructions of a greedy randomized solution and subsequent iterative improvements of it through a local search. The greedy randomized solutions are generated by adding elements to the problem's solution set from a list of elements ranked by a greedy function according to the quality of the solution they will achieve. To obtain variability in the candidate set of greedy solutions, well-ranked candidate elements are often placed in a restricted candidate list (also known as RCL), and chosen at random when building up the solution. This kind of greedy randomized construction method is also known as a semi-greedy heuristic, first described in (Hart J.P, 1987). GRASP was first introduced in (Feo T., 1989).

3.5 Evolutionary Algorithms

In the 1980's theories of creation of new species and their evolution from Mendel J. and Darwin C., back from the 19th century, have inspired computer scientists and researchers in designing evolutionary algorithms. Different main schools of evolutionary algorithms have been developed independently during the past 40 years: genetic algorithms, in Michigan, USA, (Holland, 1975), evolutionary programming, (Fogel J. L., 1966), in San Diego and genetic programming, proposed by (Koza, 1992). Each of these of course differ, but share the same approaches and principles of natural evolution.

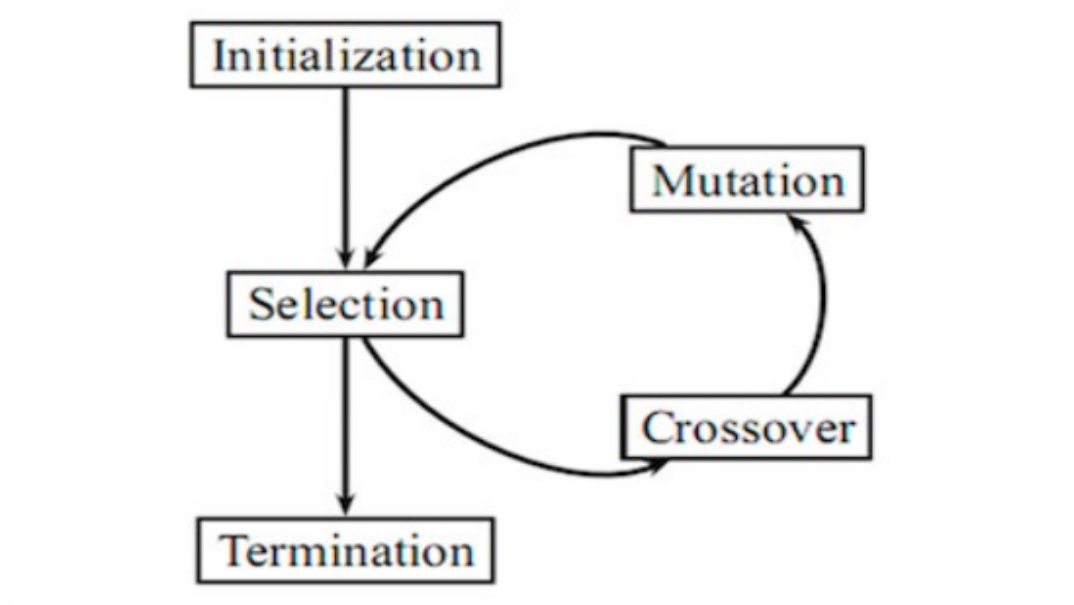


FIGURE 3.1: Process of an evolutionary algorithm

Evolutionary algorithms are stochastic metaheuristics that have been successfully applied to many real complex problems (multi-objective, highly constrained problems etc.). They are the most studied population-based algorithms. Their success in solving difficult optimization problems in various domains, from combinatorial optimization, to data mining and machine learning, has promoted the field known as evolutionary computation.

Evolutionary Algorithms are based on the notion of competition. They represent a class of iterative optimization algorithms that simulate the evolution of species. They are based on the evolution of a population of individuals. Initially, this population is usually generated randomly. Every individual in the population is the encoded version of a tentative solution. An objective function associates a fitness value with every individual indicating its suitability to the problem. At each step, individuals are selected to form the parents, following the selection paradigm in which individuals with better fitness are selected with a higher probability. Then, selected individuals are reproduced using variation operators (e.g., crossover, mutation) to generate new offspring. Finally, a replacement scheme is applied to determine which individuals of the population will survive from the offspring and the parents. This process is iterated until a stopping criterion hold.

In Figure 3.2, we can see a very detailed representation of how many algorithms can be implemented. There is no superior algorithms, due to complexity and time management.

If we want to compare the ideas involved on the various meta heuristics, a good starting point for the comparison is to think of them as Population based vs Trajectory based. Population based meta heuristics works with the concept of *create a solution that mix components of good solutions*. Trajectory based meta heuristics works with the concept of creating a solution and iteratively do improving modifications (moves) on it. With the image below, we can visualize this dichotomy and other possibles parameters for comparison.

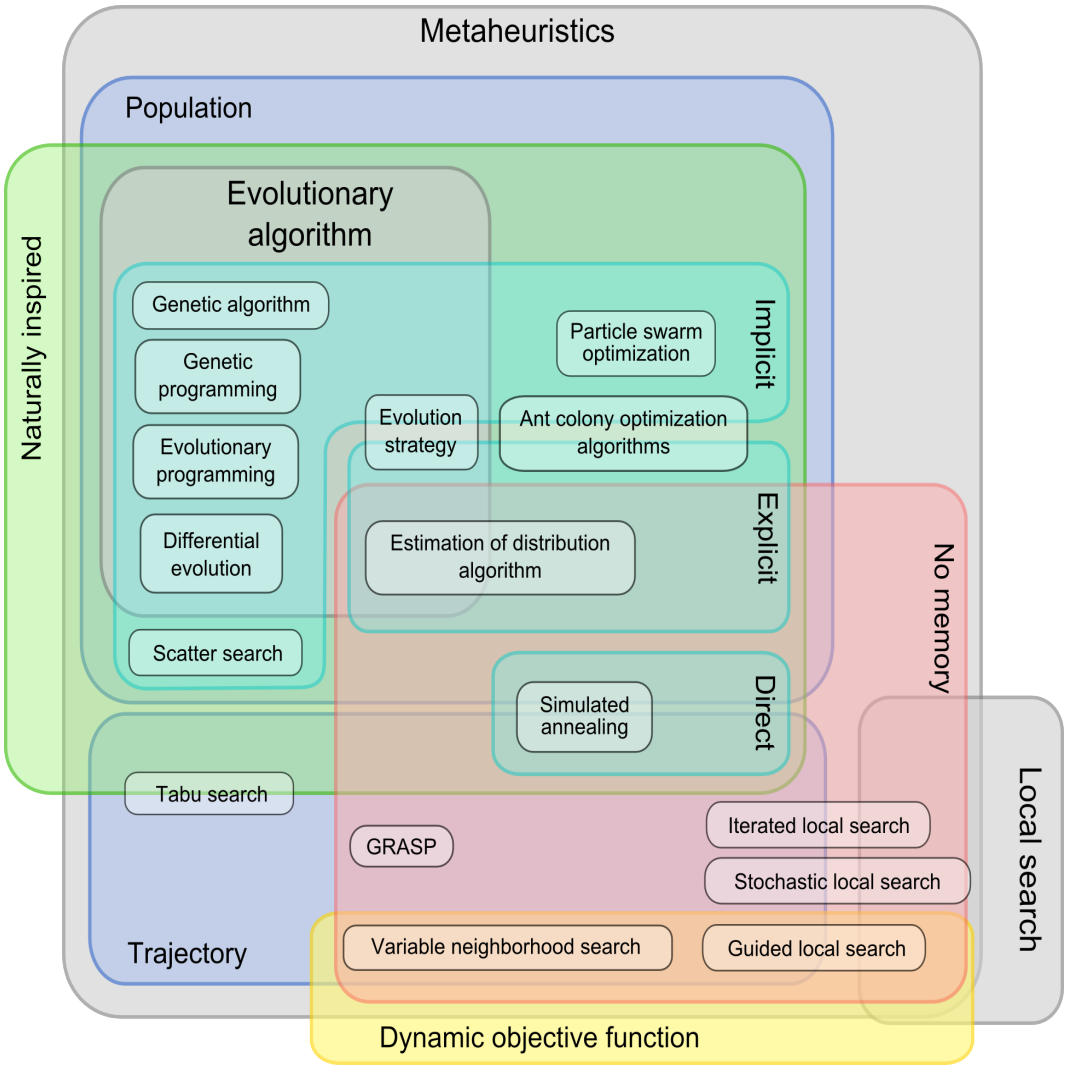


FIGURE 3.2: Grouping of meta-heuristic algorithms (source: Computer Science Stack Exchange)

3.6 Large Neighborhood Search

In this chapter, we will try to explain the Large neighborhood search algorithm. We will do a literature review about the algorithm, and will present a pseudo-code for a generic LNS procedure.

3.6.1 Large Neighborhood Search on Literature

The topic of this chapter is the metaheuristic Large Neighborhood Search (LNS) proposed by (Shaw, 1998). He referred to the technique proposed as Large Neighbourhood Search (LNS), *"which makes moves like local search, but uses a tree-based search with constraint propagation to evaluate the cost and legality of the move"*. As a "heavyweight" technique such as constraint programming is used to evaluate the move, many less moves per second can be evaluated than is normally the case for local search.

LNS is based upon a process of continual relaxation and re-optimization. For the VRP, the positions of some customer visits are relaxed (the visits are removed from the routing plan), and then the routing plan is re-optimized over the relaxed positions (by re-insertion of these visits back into the routing plan). The LNS heuristic belongs to the class of heuristics known as Very Large-Scale Neighborhood search (VLSN) algorithms. All VLSN algorithms are based on the observation that searching a large neighborhood results in finding local optimum of high quality, and hence overall a VLSN algorithm may return better solutions. However, searching a large neighborhood is very time consuming, hence various filtering techniques are used to limit the search. In VLSN algorithms, the neighborhood is typically restricted to a subset of the solutions which can be searched efficiently.In LNS the neighborhood is implicitly defined by methods (often heuristics) which are used to destroy and repair an incumbent solution. The two similar terms LNS and VLSN may cause confusion. We consistently use VLSN for the broad class of algorithms that searches very large neighborhoods and LNS for the particular metaheuristic, belonging to the class of VLSN algorithms, that is described above. In figure 3.2 lies an example of a LNS algorithm implemented on a VRP. As we can see (circled in red) 6 out of 21 (almost 30% of the solution was destroyed) customers were ejected by the initial solution in the destroy method, and then they were re-inserted, better routes were created.

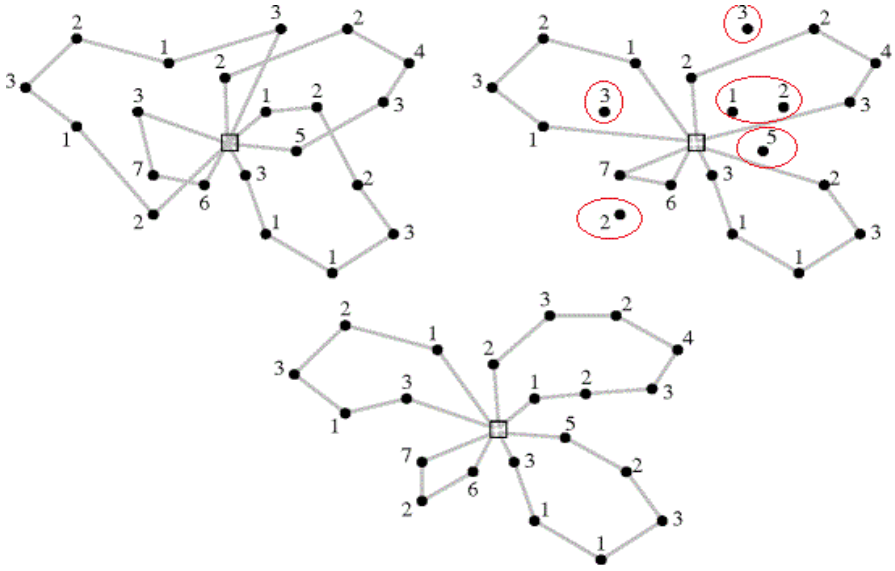


FIGURE 3.3: LNS example (source: Elsevier Science)

Many researchers have been using the algorithm, primarily on problem sets that are big, and they need a tool to rapidly search vast areas of neighborhoods. (Demir E., 2012) used an adaptive LNS heuristic for a Pollution Routing Problem, (Danna E., 2003) used a LNS procedure to study the Job-Shop scheduling type of problems with earliness and tardiness costs. Many more (Godard D., 2005), (Prescott-Gagnon E., 2009), (Masson R., 2013) and more used this technique in order to solve different kind of problems from VRPs to scheduling.

Aside from the literature review, a common LNS procedure is as follows:

In the pseudo code above, x is the current solution, x^d is the partially destroyed solution, x^r is the repaired solution and x^{best} is the best solution. The destroy () and repair () methods are the routines that destroy the current solution and repair the partial solution respectively. Finally, the cost () method is the objective criterion that evaluates the new solution over the current.

Large Neighborhood Search pseudo-code

1: Input: A feasible solution x, iterations n
$x^{best} = x$;
while $(i < n)$ **do**
  x^d = destroy(x);
  x^r = repair(x^d);
  if accept(x^r, x) **then**
    $x = x^r$;
  end if
  if cost(x^r)<cost(x^b) **then**
    $x^{best} = x^r$;
  end if
  $i++$;
end while
return x^b

Table 3.1: A generic LNS pseudo-code

What differs the most in LNS type of algorithms is the destroy and repair methods. We can say that LNS is more of a framework, than an algorithm, as, depending the problem and its characteristics, different destroy and repair methods are used. And of course, the main characteristic in most of applied LNS algorithms is the destroy method, because the way to select visits to exclude from the solution, will determine whether the repair method will produce better results.

3.6.2 Destroy Methods

There are various ways to draw a destroy plan. Shaw, proposed a relatedness measure, for choosing the set of customers that should be removed and re-inserted together. Relatedness has to be suitably defined. Many criterions are used, but a simple and obvious one is that the customers that are geographically close to one another will be more related than customers that are more distanced. If customers close to one another are removed from the route together, there is greater possibility to be re-inserted together. If customers distanced from each other are removed, the best re-insertion point for each will probably be close to where it came from. Shaw used a simple relatedness function:

$$R_{ij} = \frac{1}{c_{ij} + V_{ij}}$$

where c_{ij} is the cost of getting from i to j customer, and V_{ij} is equal to 1 if i and j are on the same route and 0 otherwise. Also, c_{ij} is normalized in the range $[0 \dots 1]$. This measure is used in order to either destroy a whole route and insert the customers in other routes together, as they are related, or find customers in different routes that are highly related and bring them together.

Later came (Hong, 2011), when he proposed an improved LNS algorithm, because *(i) The optimization procedure is time-consuming as its initial solution is poor. (ii)The single strategy of choosing nodes to be removed makes it easily terminate the local minimum points and hard to search global optimal point. (iii) It cannot effectively solve the VRPTW with a relatively wide time window* (Hong, 2011). He proposed and evaluation function, which takes into consideration a current solution, the total distance of all vehicles, the number of violations of time window constraints, and the number of violations of the capacity constraints, all adjusted to what the researcher deems important. This Evaluation function is a composite index for appraising the quality of a solution.

One interesting approach to LNS frameworks was given by (Shifeng Chen, 2018), with the Adaptive Large neighborhood search (ALNS). The algorithm starts from an initial solution containing all static customers that is constructed by the heuristic algorithm. The obtained solution is then optimized by the ALNS. Once the optimized solution is returned, vehicles begin to deliver goods for the customers while allowing a customer to submit an order dynamically. Whenever a new request occurs, the algorithm will check the feasibility of the request. If the request is acceptable, then it is added to the request pool and waits to be inserted into the current solution. This will invoke the ALNS to improve the current optimal solution again, Otherwise, it will be rejected. This dynamic procedure is repeated until there is no new request that is not serviced. Unlike LNS, the ALNS applies several competing destroy and repair heuristics instead of one destroy and one repair heuristic in the search process. The probability of applying a certain heuristic depends on its performance in the past. Each heuristic has a weight that characterizes its probability of be employed during the next search. The heuristic selected lie only on the researcher's eagerness.

It is easy to understand that different relatedness measures can be created to sufficiently determine which solution element is to be destroyed. This arises from the different features that each problem holds. In a TWVRP a relatedness measure could take into account customers with similar time windows, or could be more complex, than that of a CVRP. The destroy methods vary as problems and different constraints change.


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter 4

Problem Description

4.1 Heterogeneous VRP with multiple compartments

The vehicle routing problem (VRP) is one of the most interesting topics in which a lot of research put the effort to develop new methodology for solving a problem in which an efficient solution is obtained. Real-world problem deals with a heterogeneous VRP more than with a homogeneous type. An important variant of the VRP arises when a fleet of vehicles characterized by different capacities and costs, is available for distribution activities. The problem is known as the Mixed Fleet VRP or as the Heterogeneous Fleet VRP. The problem variants such as time windows, split deliveries, etc., appear as additional constraints that can be involved in the model to represent border on the real business.

A homogeneous classification of vehicles, i.e. only one single type of vehicles with the same capacity, is composed in the classic VRP, but in the real world the use of a heterogeneous vehicle fleet is likely to yield better results than homogeneous problem (Osman J.H, 1996). Most research on the VRPs focus on a homogeneous fleet, but the reality of the routing problems involves an aspect of the heterogeneous vehicle fleet in usual. The literature review on heterogeneous vehicle routing can classify the problem into two major groups, the first group studies the heterogeneous fleet vehicle routing problem (HFVRP) and the second one studies the fleet size and mix vehicle routing problem (FSMVRP). Both types are similar, except that the available number of vehicles called fleet size is different. The FSMVRP makes use of an unlimited number of vehicles, the HFVRP is in opposite. Anyhow, few researchers assume infinite vehicle resources in the HFVRP (Choi E., 2007), (Li X., 2010), (Brandao, 2011). Most studies relating to FSMVRP focus on a strategic planning that a decision has to be made for investing or re-sizing the number of vehicles aims at an efficient business plan, while the HFVRP puts more emphasis on optimizing the total cost of the existing available vehicles.

4.1.1 Heuristics and Meta-heuristics for HFVRP and the extensions

The results of the literature review repeat the knowledge that the FSMVRP, HFVRP, and the extensions lack of insight in the study of robustness. The robustness is considered in several homogeneous VRPs, but not for heterogeneous VRPs. The robust solution is the good answer to a question of acquiring vehicle numbers in the long run (Bielli M., 2011).

The fact that most of these problems are complex, with great complexity and of large size, exact algorithms have proven to be unsuccessful. Many papers are written approaching the problem with meta-heuristic algorithms (Sadjadi S.J. Moghaddam B.F. Ruiz R., 2012), (Seyedhosseini S.M. Moghaddam B.F. Sadjadi S.J., 2010), in order to undertake the complex optimization problems. They used ant colony optimization (ACO) and advanced particle swarm optimization (PSO) algorithms.

Literature in the field of robust vehicle routing problem is absolutely not vast, especially in the case of heterogeneous VRPs. The heuristics and meta - heuristics paradigms of the existing FSMVRP, HFVRP, and the extensions with certain data are surveyed in consequence. Osman & Salhi present the route pertubation constructive heuristic and a Tabu search procedure. A generalized insertion heuristic is used for constructing the initial routes and local search for the improvement phase in (Seyedhosseini S.M. Moghaddam B.F. Sadjadi S.J., 2010). Baldacci and Mingozzi, 2009 introduce the exact algorithm based on the set partitioning formulation which is able to solve the problem by an integer linear programming solver. (Penna P.H., 2011) et. al construct routes by iterated local search. Many researchers have tackled different variations of the problems (Repoussis P., 2010), (Belfiore P., 2009), (Kritikos M.N, 2013), (Prins, 2009), (Golden B., 1984), (Renaud J., 2002), (Tarantilis C., 2004).

All above works share the same target aiming to obtain an optimal solution, specifically a total cost to be minimized. The robust problem is excluded that the objective is to obtain the close solution optimization. The VRP and its variants have the general target of finding the routes that optimize an objective function. Although the heuristic and meta-heuristic algorithms proposed by each author as reviewed above are efficient and can yield good solutions, these schemes do not guarantee the optimality.

4.2 Multi-Compartment VRP Formulation

The MC-VRP is defined on a complete undirected graph with a node set $N = \{0, 1, \ldots, n\}$ including the depot (node 0) and a set of n customers C. Each edge (i, j) has a transport cost $c_{ij} = c_{ji}$. The depot stores a set of $P=\{1, 2, \ldots, m\}$ of m products (diesel, unleaded, etc.), which must be delivered by a fleet of vehicles $V = \{1, 2, \ldots, v\}$ with a set of m compartments. Compartment k of each vehicle is dedicated to product p with a known capacity of Q_p. Each customer i has a known request $q_{ip} \leq Q_p$ for each product and zero for a product not ordered by the customer.

$$\min \quad \sum_{i,j \in N} \sum_{v \in V} c_{ij} x_{ijv}$$

$$s.t. \quad \sum_{i \in N} x_{ijv} \leq 1, \quad \forall j \in C, \forall v \in V \ [1]$$

$$\sum_{i \in N} x_{ijv} - \sum_{i \in N} x_{jiv} = 0, \quad \forall j \in C, \forall v \in V \ [2]$$

$$\sum_{i,j \in S} x_{ijv} \leq |S| - 1, \quad \forall v \in V, \forall S \subseteq C, |S| \geq 2 \ [3]$$

$$y_{jvp} \leq \sum_{i \in N} x_{ijv}, \quad \forall j \in C, \forall v \in V, \forall p \in P \ [4]$$

$$\sum_{i \in N} y_{jvp} = 1, \forall j \in C, \forall p \in P \ [5]$$

$$\sum_{j \in C} y_{jvp} q_{jp} \leq Q_p, \forall v \in V, \forall p \in P \ [6]$$

$$x_{ijv} \in \{0,1\}, \forall i, j \in C, i \neq j, \forall v \in V \ [7]$$

$$y_{jvp} \in \{0,1\}, \forall j \in C, \forall v \in V, \forall p \in P, q_{jp} \neq 0 \ [8]$$

This formulation is a primitive one (Fallahi A., 2008), that does not encode the precedence constraints and the inhomogeneous fleet of vehicles. The 0-1 variables x_{ijv} [7] are equal to 1 only if the edge (i, j) is used by a vehicle v, and zero otherwise. The 0-1 variables y_{jvp} [8] take the value of 1 if a customer j receives a product p from vehicle v, and zero otherwise.

The objective function represents the total cost of the routes, which is a sum for each vehicle and every active traverse between customers. If a variable x_{ijv} is equal to 1, namely an edge in the graph is used, times the cost for this traverse, we have the cost for the transportation between two customers i, j. We need to minimize this cost. Eq. [1], states that each customer may be visited at most once by each vehicle, hence no split deliveries allowed. Eq. [2] implies that if a vehicle v enters a customer's location, it has to leave the customer location as well. Eq. [3] are classical

sub route elimination constraints. Constraints [4] set y_{jvp} to zero for each product p if customer j is not visited by vehicle v. The next set of constraints [5] ensure that when a customer is served, he is served by only one vehicle. The last set of constraints [6] prevents compartment capacity violations.

This model is a generic form of the real problem solved in this thesis. The real problem is way more constrained and harder, concerning complexity.

4.3 Tackling the Problem

In this thesis, we study, as previously mentioned, a vehicle routing and scheduling problem that is encountered in fuel distribution industry. It is a heterogeneous Vehicle Routing Problem with multiple compartments (HFMCVRP). It involves a Greek fuel distribution company using a heterogeneous fleet of multi-compartment vehicles and a fixed number of pumps located at the depot (the refinery) of the company. The heterogeneous fleet delivers the customers' orders for multiple types of fuel, named as order items. The problem is in the category of in-homogeneous fleet of vehicles, all order items (of any given customer) must be delivered by only one vehicle in only one visit, for example the company cannot fulfill an order by sending a split delivery with two vehicles. That means that an order of any given customer, containing order items, cannot exceed the total capacity of the largest vehicle of the fleet, and also the demand of the order items cannot exceed the individual capacities of the compartment. Furthermore, each compartment may be assigned up to one order item, each vehicle may perform multiple routes within its shift.

Another important feature of the problem is that due to regulations, as fuel is a hazardous material, certain safety measures must be held. In the fuel distribution industry of Greece, the measures held is the vehicle stability considerations. Each vehicle compartment that is assigned with an order item, must have a payload weight distribution on the axles, so that the fuel weight will not affect the stability of the vehicle. The vehicle stability considerations are crucial in designing the delivery routes. These constraints are implemented with what we call loading/ unloading constraints, that explain with what way the compartments are loaded and unloaded. In this thesis we retain the prevailing practice of the industry in which the middle compartments of the vehicle must be first unloaded, followed by the rear compartments, leaving the front compartments at the end.

It is important to understand how the scheduling problem emerges. Since there is a fixed number of pumps at the company's refinery and each vehicle requires a given (and fixed) loading time for getting its allocated orders on board, when the number of routes exceeds the number of available pumps, then a scheduling problem arises. Each vehicle can perform multiple routes in its' time-line. Hence, it has a unique estimated time of arrival (ETA) from its' route, which is the time that the vehicle needs to transport the order items from the depot to the customer, in addition to the estimated service time and the time to return to the depot. Each vehicle route must be allocated to a single pump for a given period of time. The earliest possible departure time of each vehicle is the allocated start time of loading on the available pump plus the loading time, which we assume fixed. Eventually, the allocated time slot to a vehicle route may affect the earliness or lateness in delivering the associated orders, due to time windows.

4.4 Assumptions

The fuel distribution company offers a set of fuel products P, which are nine. Each customer places an order defined by a set of items Π_i. Each item $r \in \Pi_i$ of an order is associated to a specific product and the total requested volume quantity of the order π_r. By this information, the weight of an order item can be computed by multiplying the volume of the item by its specific weight. The number of order items in a set of items Π_i is declared as k_i and the maximum number of these order items cannot surpass the number of products that the company offers. The delivery must be performed between the opening and closing time of a customer and be done by a single vehicle.

The opening (a_0) and closing time (b_0) of the company's premises are 5 : 30 - 23 : 00. All vehicles are loaded at the depot on the day of the delivery through nine pumps. The loading duration per vehicle is approximately 45 minutes, but in this thesis considered as fixed. All the products may be supplied be any of the pumps to vehicles, so it is convenient to load a vehicle via only one pump. This fuel delivery problem involves assigning order items to vehicle compartments, allocating loading slots to routes, and finding the best program to service all orders of customers, with respect to minimizing the total travelled distance and maximizing the average capacity utilization.

The problem is solved lexicographicaly. This arises from the fact that this problem is formulated as a tri-objective problem, namely has three objective functions, as stated above. One for the serviced customers, the second one for the total travelled distance and the third for the capacity utilization. What needs to be achieved is to have a solution with minimum number of un-serviced customers, minimum distance travelled by all vehicles and the maximum capacity utilized by the loading of the vehicles respectively. Objectively, if we have two solutions, the first thing to do is check the serviced customers. If the two solutions have the same number of serviced customers, we proceed to check the travelled distance and select as superior the one with the less total travelled distance. And in the rare case of having both the serviced customers and the travelled distance the same, then we proceed to check the final criterion, the loading factor. We select the solution with the maximum loading factor.

Concerning feasibility, the Greek fuel distribution company deemed necessary to set some ground rules to what was acceptable as a solution. The fuel company wanted to have as much serviced customers as possible, with the minimum total distance travelled, but all vehicles must have an average loading factor more than 80%. This stands because the dispatchers are accepting routes with a loading factor of such plentitude. If the loading factor was less than 80&, then the dispatcher will not accept the route, as they are getting paid by volume and not per km travelled.

The average loading factor per vehicle is the ratio of the weight of the orders served by the route over the vehicle maximum net weight. For example if we had a vehicle with 3 compartments with maximum net weight per compartment C_i = [2000, 1500, 1000], and the weight of the orders per compartment was the following vector v_i = [1800, 1450, 950], the loading factor would be: lf = $\frac{1800+1450+950}{2000+1500+1000}$ = 93.33% > 80%, thus an acceptable loading factor.

4.5 Loading and un-loading constraints

The most significant feature showed in this problem is the following loading/ unloading rules that dominate throughout the assignment of order items to vehicle compartments.

- The most common constraint in a fuel delivery problem, especially when the compartments are not equipped with a flow meter, is that each compartment can have up to one order item, namely it is not allowed to have more than one order of the same product in the same compartment if it is placed by different customers.
- Each order item Π_i of customer i must be assigned to compartments of the same vehicle.
- An order item can be split and assigned to more than one compartment of the same vehicle, only when none of the available compartments has sufficient capacity to carry the full quantity of an order item.
- As previously said each vehicle compartment that is assigned with an order item, must have a payload weight distribution on the axles, so that the fuel weight will not affect the stability of the vehicle.

There are different ways to apply the last rule in the Greek oil distribution. In order to create such rules, there has been a classification of the vehicle compartments, depending on their physical ranking, to three groups: front, middle and rear. Many companies use the *rear, middle, front* rule, as in the rear, middle and front set of compartments. In this thesis, the compartments must be unloaded in the following order: *middle, rear, front*. This is easily explained. If the front compartments get unloaded first, then the weight distribution is transferred to the back of the vehicle, with the fear of unhooking. In another occasion if the rear compartments get unloaded first, then the weight distribution is transferred to the front and in the event of a severe breaking, it would be very difficult to stop the vehicle. Hence we use the aforementioned unloading precedence.

Next follows an example for these loading and unloading constraints. Let's assume we have one vehicle, that has 6 compartments (two front, two middle and two rear) C_i, where $i = 1, \dots 6$. Also, we have two products $p_j = \{p_1, p_2\}$ and three customers $\{1, 2, 3\}$. We have already chose a plan to route them and the best route is $1 \rightarrow 2 \rightarrow 3$. The demand for each product of each customer is:

Customer 1:

$$\begin{bmatrix} 1000lt/p_1 \\ 1500lt/p_2 \end{bmatrix}$$

Customer 2:

$$\begin{bmatrix} 3000lt/p_1 \\ 2000lt/p_2 \end{bmatrix}$$

Customer 3:

$$\begin{bmatrix} 4000lt/p_1 \end{bmatrix}$$

Figure 4.1 illustrates a way to load the vehicle with a correct way.

- The order items of the 1^{st} customer are assigned to the middle compartments (C_3 and C_4).

- The order items of the 2^{nd} customer are assigned to the rear compartments (C_5 and C_6).
- The order items of the 3^{rd} customer are assigned to the front compartments (C_1 and C_2).

First, we arrive to the premises of the 1^{st} customer and unload the $1000lt$ of p_1 and the $1500lt$ of p_2. The middle compartments are emptied first. Then, we arrive to the premises of the 2^{nd} customer and unload the $3000lt$ of p_1 and the $2000lt$ of p_2. The rear compartments are empty. Finally, we arrive to the premises of the last customer and unload the $4000lt$ of p_1 that he ordered and the front compartments are empty too. The compartments are emptied with the correct order *middle, rear, front*. In

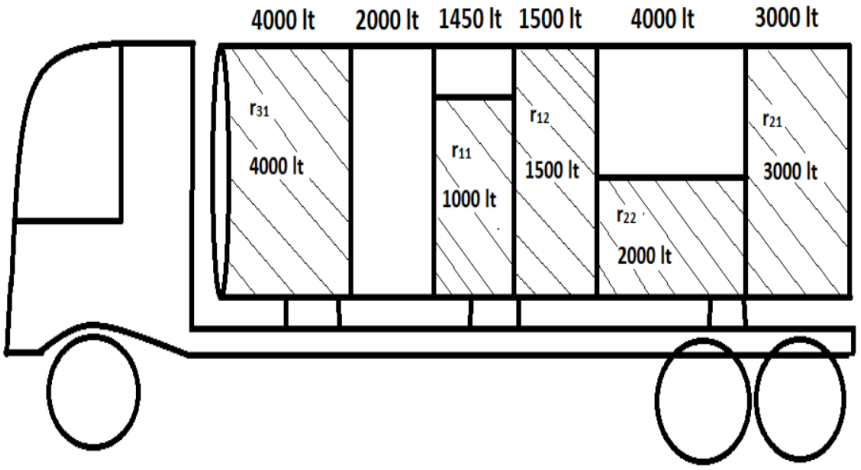


FIGURE 4.1: Right vehicle loading

Figure 4.2, appears a false way to load a vehicle, if we want to follow the loading and unloading constraints. As we can see, this way to load the vehicle produces a better, in this case optimal, way to load it, since the wastage of empty space is minimized, but safety comes first. In this example, first we arrive to the premises of the 1^{st} customer and unload the $1000lt$ of p_1 and the $1500lt$ of p_2. The middle compartments are emptied. Then we arrive to the premises of the 2^{nd} customer and unload the $3000lt$ of p_1 and the $2000lt$ of p_2. As we see we emptied the front compartments completely, when one compartment (C_5) remained full. This is not compatible with the rule applied in these constraints.

4.6 Discussion

After establishing the basics about the algorithms used in chapter 3 and the problem description in this chapter, we will try to explain the reasons the Large neighborhood search algorithm was selected to tackle this problem.

First of all, these kind of problems, are really big, concerning the data and the calculations that need to be made, in order to just obtain a feasible solution. A heterogeneous fleet of vehicles, rather than a homogeneous fleet, can increase the complexity of the problem by a vast amount. Hence, the problem we are tackling has not only the heterogeneous fleet of vehicles, but many parameters, such as the pumps, the

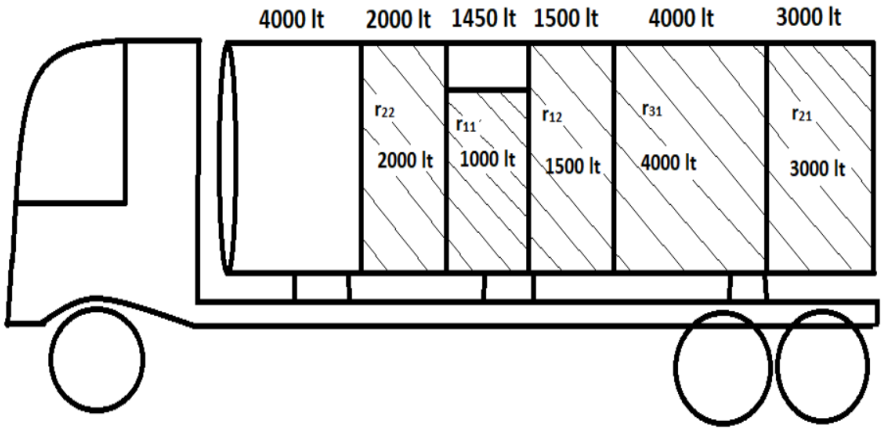


FIGURE 4.2: Wrong vehicle loading

order items, the time windows, and because we are tackling it as a scheduling problem with time windows and many constraints, this problem is considered a NP-hard problem (non deterministic polynomial time hardness), which means that it cannot be solved in polynomial time. Because of its' complexity, huge neighborhoods are created in order to check a feasible solution.

We decided to use an LNS procedure, because of how complex the problem was. As we discussed earlier, if the heuristic approach obtained a relatively acceptable solution, or even feasible, the LNS algorithms could make a great impact, as it would destroy and repair the "foul" neighborhoods.

Chapter 5

Solution Algorithms and Data

As it is easily understood, the problem was not tackled from the beginning until the end by the author of this thesis. A previous work was made by Dr. Androutsopoulos Konstantinos and Nikolaos Kamaras. This thesis is containing the LNS procedure implemented, in order to solve some problems explained earlier in chapter 4.

First, we need to explain the techniques evolved until the stage the LNS was implemented. The algorithm involves a series of GRASP iterations. The solution being taken by GRASP is then further processed through the LNS routine that aims to improve the solution, in terms of total distance travelled by all vehicles. The GRASP algorithm is the construction routine for allocating orders to vehicles, make a first semi-randomized solution and design of the delivery routes. Of course the initial solution given by GRASP is not of high quality. Hence, the need of local search procedures, in order to obtain an acceptable overall solution is almost obligatory.

GRASP, as previously said involves two phases. The first is a semi-randomized route construction, in order to obtain an initial solution quickly. The second phase comprises of local search routines. In this problems, several operators have been developed, since the objective criteria to optimize are multiple. We will continue to explain briefly each of them and then we will explain the LNS procedure and the way it was implemented in this algorithm.

5.1 Basic Operators

These two fundamental operators are used throughout the algorithm to ensure scheduling and capacity feasibility of a route during construction.

- *Route Scheduling*

  This routine is made to secure that, if a sequence of routes are assigned to a vehicle, no time constraints will be violated. The schedule of any route is defined by its loading start time, because knowing just the loading start time, we know that the ETA for each vehicle would be: $ETA = loading\ start\ time + loading\ time + duration\ of\ visit$. The loading time is set to 45 minutes for each order and the duration is the time to and from the customer plus 45 minutes for the unloading of the vehicle.

  Let there be a new route that we are currently examining to insert to the proposed plan. This route will have assigned a new start time. The objective of route scheduling is to check the schedule feasibility of assigning the route in

the schedule, given the set of available slots left available for the loading facilities of the depot. This routine aims to determine the loading start time for all routes, so that the schedule of any current route is not overlapping with the schedules of the already assigned routes.

- *Order Loading*

  This routine is the key factor to assigning the items of an order to compartments of a vehicle, so that all loading and unloading constraints are met, while the "wastage", namely the remaining unused capacity of the selected compartments is minimized.

  Let there be an order and a vehicle match, which is where we want to load the order. In order to load we need to examine whether the capacity of the idle compartments is enough for the weight of the orders, if the number of order items exceeds the number of idle compartments and if the total volume of the order items is lower than the total volume capacity of the compartments.

  Then the routine selects and matches the order item to load in a compartments using a method called *FindBestMatching*, that finds the best fit for each couple order item - compartment, concerning the capacity utilization.

5.2 Local Search Operators

The algorithm uses three different types of local search procedures to improve the objective criteria that need to be obtained, the coloading, reloading and repositioning routines.

- *Co-Loading Routine*

  After the construction algorithm, the solution created is not of the best quality. That happens, because of the semi-randomization and the needed speed of the GRASP algorithm, in order to obtain a feasible solution fast. This situation ends up in creating solutions, with solutions that under-utilize the vehicle capacity. Co-Loading routine helps to reduce the total number of routes, by canceling out the routes with a low loading factor. The routine starts by ranking the routes, in respect of their loading factor, and then one by one try to load the order items of a route in other neighboring routes. After the route reloading of existing orders of route in another vehicle, a route scheduling takes place, for determining a feasible loading time of the updated route. If the order items of a vehicle cannot fit anywhere, a new vehicle, of greater capacity is used.

- *Re-Loading Routine*

  The Re-loading routine aims to improve the average capacity utilization of the solution by re-allocating routes from vehicles where low capacity utilization (e.g., the total weight of the orders in a route divided by the maximum net weight of the vehicle) is attained to lower-capacity vehicles. Hence, if the capacity utilization of a route on a vehicle is below a threshold value, then the procedure runs Route Assignment for the route. If the capacity utilization of route on the new vehicle returned by Route Assignment is higher than the current utilization rate of the route, then it is re-assigned to the new vehicle.

- *Re-Positioning Routine*

  Given a route , the re-positioning routine applies a series of exchanges of the customers' positions (swaps) in the route aiming to reduce the travelled distance. This procedure is making all possible exchanges of the customers in the route, if and only if they are feasible. Hence, all neighborhoods are feasible. This process is iterated until no further distance reduction can be achieved and selects the one that attains the maximum reduction of the traveled distance.

5.3 Phase 3: LNS

After the construction phase and the local search operators, in order to obtain a better solution, since the construction algorithm does not return a solution of high quality, we proceed to the third phase of the algorithm, in which we are implementing the LNS procedure, in order to improve the solution by the total travelled distance of all the vehicles.

Because of the vastness of data, and the combinatorial complexity, the searching space is really big. Hence we need to search efficiently and quickly a lot of neighborhoods. The LNS procedure works better when we use many iterations (50 - 100 - 150). That happens because in each iteration, a different starting point is chosen and we have greater probability to achieve lowest total travelled distance and find a better solution. Our implementation of the LNS algorithm comprises of 4 steps, i) root selection, ii) solution destroy, iii) solution repair and iv) implementation. We will continue with a detailed description of each step.

- *Root selection*

  In order to begin each iteration of LNS from a different starting point (root), we need to implement a randomization in the process. A method has been created, called *SelectRouteStop*, which is the one that selects the first customer, from which we will build the neighborhood we will destroy. Given all the routes, this method selects a customer completely randomly. As stated before, this helps search efficiently and quickly vast neighborhoods, because we will start, each iteration, from a different starting point.

  After this method and the selection of the root customer, we need to create the neighborhood to destroy. The neighborhood is created with an iterative procedure, starting from the selected customer and finding each customer in a radius of 2 kilometers. If this neighborhood overcomes the limit of destruction we set (e.g. 20% of the solution to be destroyed), it stops and the neighborhood is set; else, the radius broadens by 2 kilometers and we find each customer in this circle and so forth, until a radius of 20 kilometers is reached, if needed. If we search all the customers in a radius of 20 kilometers and we don't reach the necessary percentage, we select the customers already selected as the neighborhood to be destroyed.

- *Destroy solution*

  Since the neighborhood to be destroyed is set by the previous routine, we proceed to the implementation of destruction. In order to successfully destroy a part of solution, we need to first of all clone our initial solution, since we don't know if the new solution will be better. Then we must first of all dismiss all

customers from the vehicles, all order items from compartments and all time slots, used by each vehicle of the neighborhood, from the schedule. Then, these customers are, for now, un-serviced and the vehicles, that were previously used to deliver the order items to these customers, idle. Everything, except the partial solution that is already built, is back to its' initial position, vehicles are unloaded and customers are un-serviced.

An important operator in this procedure is the Rescheduling. Since we destroyed a part of the solution, several time slots, loading starting times and schedules are now idle. Hence, a pretty useful technique, in order to save some time or try to make some extra free time, is to reschedule the partial solution. In this operator, every vehicle used in the partial solution is trying to fit in the schedule, even just one route, as early as possible.

- *Solution Repair*

  Thereafter, begins the repair operator. Given a set of idle vehicles V, with a known capacity and schedule and the neighborhood of un-serviced customers N, with a known demand, we try to repair the partially destroyed solution, and return solutions with less number of un-serviced customers, or solutions with equal or less number of un-serviced customers, that the total distance travelled by the vehicles of the new repaired solution is less than the total distance travelled by the current solution. The total demand of the customers is D_{tot} and their order items are l.

  With that information, knowing the total demand of all un-serviced customers and the number of order items, we can select vehicles appropriately, namely vehicles with number of compartments a little more than l, and at the same time with a total weight capacity a little more than D_{tot}. What we are trying to achieve is, if we manage to load the customers and their order items inside the vehicles, then the capacity utilization would be almost perfect.

  The building procedure begins with a vehicle, which tries to be loaded with the order items of a random customer, in order to have more than 80% capacity utilization. If this condition is not met, it tries to be loaded again, up to 8 times, with a different starting customer each time. If a vehicle fails to be loaded these 8 times it is set as banned and is unused for the rest of the procedure. The same process is applied for each vehicle selected. If under these conditions a solution is found, it is guaranteed that the entire given neighborhood will be serviced and we will have a different solution. Although, the routine might not find a feasible solution, where all customers return serviced. In this occasion we replace the banned vehicles, namely the ones that could not load a feasible route more than 8 times, with those that have slightly more capacity and same or more number of compartments and retry to find a feasible solution. The process stops if a feasible solution is found, or the vehicles that are not banned cannot serve the remaining customers.

- *Implementation*

  After iterating the previously mentioned procedure, we encounter two possibilities. The first is to find a feasible solution, in which case if the total distance travelled by all the vehicles at the new solution is less than the distance travelled at the previous solution we set as the current solution, namely the best so far, the LNS solution. On the other hand the repair method might return a non

feasible solution. In this case, the current solution returns and we continue the LNS iterations from a different starting customer.

LNS Routine pseudo-code

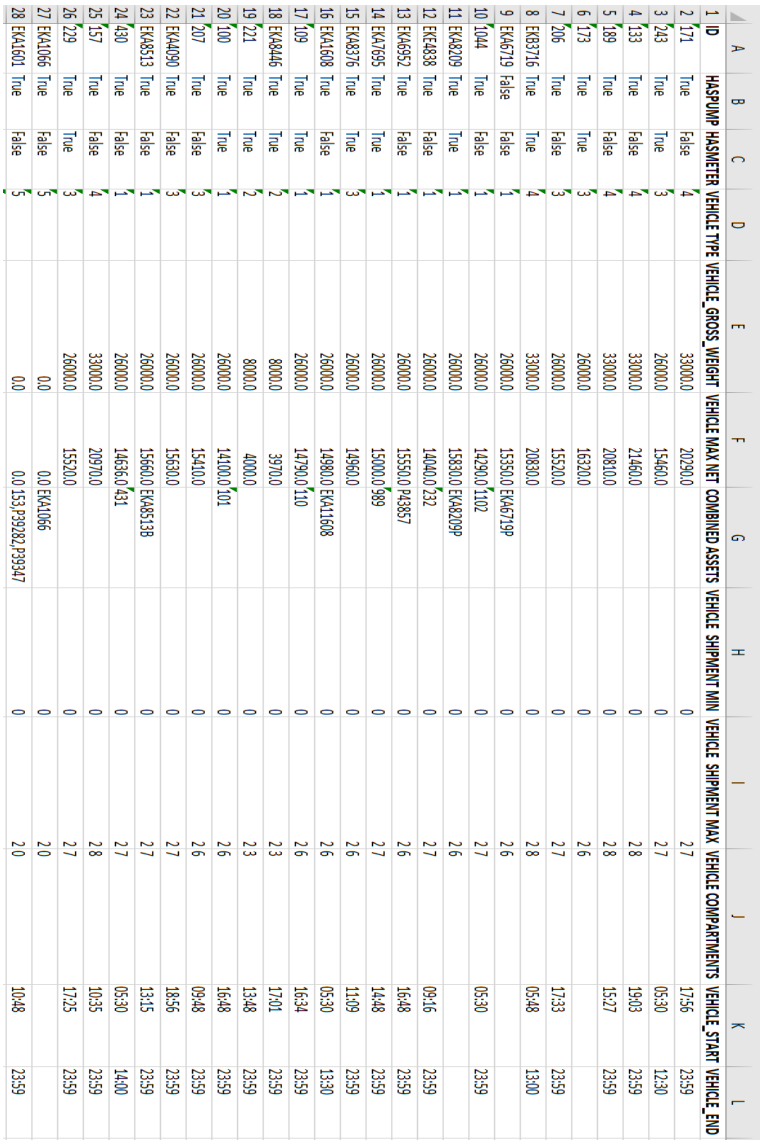
1: Input: A feasible overall solution Os, iterations n
$Os^{best} = Os$;
while ($i < n$) **do**
  RS = **selectRouteStop**(Os);
  customersToBeRemoved = *FindNeighborhood*(RS, *DestroyPercent*);
  PartiallyDestroyOs;
  newOs = *LNSRepair*(Os);
  if (*newOs.unservicedStops()* < Os.*unservicedStops()*)
  or (*newOs.unservicedStops()* == Os.*unservicedStops()* **and** kmsNew<kmsOld)) **then**
    $Os = newOs$;
  end if
  $i++$;
end while
$Os^{best} = newOs$;
return Os^{best};

Table 5.1: LNS Routine pseudo-code

5.4 Data

In this section we will explain the format our data has and the different aspects of the constraints, through them.

The Greek fuel distribution company sends an excel spreadsheet with the necessary data, in order to implement the algorithm. In the first sheet we get all the information about the vehicles. Their registration plate, whether or not they have a pump and a meter, their type (1, 2, 3, 4 is for tri-axial vehicles and 5 and 6 for trucks with a trailer). Also has information about the vehicle's gross weight and max net weight, the pairings with each asset that can be done, its' minimum shipments per day (usually zero) and its' maximum shipments (usually two). Lastly, this tab has information about the number of compartments each vehicle has, the starting hour and finish hour.



	A	B	C	D	E	F	G	H	I	J	K	L
1	ID	HASPUMP	HASMETER	VEHICLE TYPE	VEHICLE_GROSS_WEIGHT	VEHICLE MAX NET	COMBINED ASSETS	VEHICLE SHIPMENT MIN	VEHICLE SHIPMENT MAX	VEHICLE COMPARTMENTS	VEHICLE_START	VEHICLE_END
2	171	True	False	4	33000.0	20290.0		0	2 7	17:56	23:59	
3	243	True	True	3	26000.0	15460.0		0	2 7	05:30	12:30	
4	133	True	False	4	33000.0	21460.0		0	2 8	19:03	23:59	
5	189	True	False	4	33000.0	20810.0		0	2 8	15:27	23:59	
6	173	True	True	3	26000.0	16320.0		0	2 6			
7	206	True	False	3	26000.0	15520.0		0	2 7	17:33	23:59	
8	EKB3716	True	True	4	33000.0	20830.0		0	2 8	05:48	13:00	
9	EKA6719	False	False	1	26000.0	15350.0	EKA6719P	0	2 6			
10	1044	True	False	1	26000.0	14290.0	1102	0	2 7	05:30	23:59	
11	EKA8209	True	True	1	26000.0	15830.0	EKA8209P	0	2 6			
12	EKE4838	True	False	1	26000.0	14040.0	232	0	2 7	09:16	23:59	
13	EKA6952	True	False	1	26000.0	15550.0	P43857	0	2 6	16:48	23:59	
14	EKA7695	True	True	1	26000.0	15000.0	989	0	2 7	14:48	23:59	
15	EKA8376	True	True	3	26000.0	14960.0		0	2 6	11:09	23:59	
16	EKA1608	True	False	1	26000.0	14980.0	EKA11608	0	2 6	05:30	13:30	
17	109	True	True	1	26000.0	14790.0	110	0	2 6	16:34	23:59	
18	EKA8446	True	True	2	8000.0	3970.0		0	2 3	17:01	23:59	
19	221	True	True	2	8000.0	4000.0		0	2 3	13:48	23:59	
20	100	True	True	1	26000.0	14100.0	101	0	2 6	16:48	23:59	
21	207	True	False	3	26000.0	15410.0		0	2 6	09:48	23:59	
22	EKA4090	True	False	3	26000.0	15630.0		0	2 7	18:56	23:59	
23	EKA8513	True	False	1	26000.0	15660.0	EKA8513B	0	2 7	13:15	23:59	
24	430	True	False	1	26000.0	14636.0	431	0	2 7	05:30	14:00	
25	157	True	False	4	33000.0	20970.0		0	2 8	10:35	23:59	
26	229	True	True	3	26000.0	15520.0		0	2 7	17:25	23:59	
27	EKA1066	True	False	5	0.0	0.0	EKA1066	0	2 0			
28	EKA1601	True	False	5	0.0	0.0	153,P39282,P39347	0	2 0	10:48	23:59	

FIGURE 5.1: Vehicles tab

There is also two other tabs containing information about the compartments and assets of each vehicle. In the compartments tab, for each ID of vehicle a compartment

is numbered and the capacity of each compartment is given. In the asset tab, given a vehicle that the asset matches, the information about said asset is given (if it has pump or meter, about its' gross weight and max net, number of compartments etc.). The asset tab is accompanied by a tab, called asset compartments, with the same information that the tab compartments have.

	A	B	C
1	**ID**	**Vehicle_Compartment_AA**	**Vehicle_Compartment_Capacity**
2	171	1	2950.0
3	171	2	3900.0
4	171	3	3900.0
5	171	4	4350.0
6	171	5	4450.0
7	171	6	3450.0
8	171	7	3700.0
9	243	1	2700.0
10	243	2	2700.0
11	243	3	2800.0

FIGURE 5.2: Vehicle compartments tab

Next, the tab premises contains the premise's ID, the premise's name, its' geographical coordinates (latitude and longitude), the number of pumps it has, its starting and closing time and the loading time. There is a tab of customers, containing for each customer, the premise that is subjected to and the customers geographical coordinates.

Of course, for each customer the Greek fuel distribution company sends a tab that contains IDs, latitudes and longitudes, customer's opening and closing time, the average unloading time, whether it is in land, what vehicles can access its' premises etc. Each customer, holds an order and each order comprises of some order items. The tab including the order items gives information about which order item each customer has ordered, what is the volume of the order in litres and what is the specific weight of each product, in order to evaluate the weight of each product.

	A	B	C	D	E
1	ORDER_ID	ORDER_IT	ORDER_IT	ORDER_IT	ORDER_ITEM_SPECIFIC_WEIGHT
2	5172497	10	18310500	5000	0.8365
3	5172497	20	28214500	10000	0.7455
4	5172164	10	18310500	3500	0.8365
5	5172164	20	18320500	2500	0.8365
6	5172164	30	28214500	13000	0.7455
7	5172164	40	28215500	4000	0.7544
8	5172520	10	28214500	9000	0.7455

FIGURE 5.3: Order Items tab

Last but not the least, the tab parameters contains the parameters needed to iterate the algorithm. The number of iterations for the construction algorithm (GRASP), the

swaps performed from the local search operators, the capacity utilization, namely the number, from which the solutions are inadequate. It also has some basic numbers concerning distances for the construction, and local search operators and the number of LNS iterations and percent of destruction.

In figures 5.1, 5.2, 5.3, you can see indicatively three tabs (order items, compartments and vehicles) that have the data used for this thesis.

Chapter 6

Computational Results

After the coding and implementation of the LNS procedure, we should create various instances, in order to evaluate the solutions given by said algorithm. Next, we present the computational results and describe the experiments done, for that matter.

The proposed heuristic algorithm was assessed under three criteria: i) computational time performance and ii) convergence performance. This assessment has been chosen, because there is not a known work done in this specific problem. Hence, there was no way to compare our results with the work of another researcher. The assessment of the algorithm was based on measuring its performance for solving a set of test problems (instances). All the test problems were generated on the basis of actual data provided by the Greek Fuel Distribution Company. Hence, the delivery locations of the test problems are located in South Greece. The orders comprise from 1 to 9 items, while the quantity of each order item ranges from 1000 to 32000 Liters. The fleet of vehicles used for servicing demand consists of 159 vehicles each one consisting from 3 to 12 compartments. The compartments volume capacity ranges from 1000 to 9200 liters. The max net weight of the vehicles ranges from 4000 to 30030 kilograms. The following data are common in all test problems developed:

- The opening and closing time of the depot are 05:00 and 23:00 respectively
- Loading facility pumps: 9
- Average customer service duration:45 min
- Vehicle average loading time: 45 min

Thirty-two different data-sets were created, comprising from 32 to 165 different customer orders to test the convergence and solution performance of the algorithm. The 32 different data-sets were implemented for 20, 40 and 60 GRASP iterations, having set the number of LNS Iterations to 50, and then having set the number of GRASP Iterations to 30, the sets were run with 50, 100 and 150 LNS Iterations. In what follows on the appendix, we present the experiments that were performed for the assessment of the algorithm and the associated results. It is worth noting that the proposed heuristic algorithm was coded in Java 1.8 while all experiments were performed on an Intel Core 3.60 GHz computer with 64-bit Windows operating system and 16GB RAM.

6.1 Computational Time Experiments

We tried to evaluate the algorithm under the time criterion. Due to vast complexity, the time that the algorithm needed to return a good solution should be exponentially growing, as customers were increasing. In this occasion, we observe exactly that.

Computational time was assessed under the following two criteria: i) the relationship of the average computational time and the number of customers and ii) the relationship between the average computational time of the construction algorithm (GRASP) and the LNS. Figure 7.11 presents the average computational time per instance, required for solving the sets of test problems by the proposed GRASP heuristic algorithm and the LNS procedure. Moreover, Figure 7.11 presents the average computational time per iteration required for solving 15 of the 32 problem tests. For the convenience of appearance of the figure we chose to include 5 instances of 50 customers, 5 instances of 100 and 5 of 150 customers respectively. In addition, figure 7.11 indicates that the relationship between the number of customers of a problem test and the average computational time per iteration is not linear.

For assessing the second criterion, we selected 7 different problem sets, comprising of 32, 50, 95, 120, 140, 160 and 165 customers, throughout all the spectrum of datasets. The data used to create the following figure are taken from the given solution with 30 GRASP iterations and 150 LNS iterations. Figure 7.12 indicates that the increase of the number of customers leads to an increase of the average computational time required.

6.2 Convergence Experiments

In numerical analysis, the speed at which a convergent sequence approaches its limit is called the rate of convergence. Although strictly speaking, a limit does not give information about any finite first part of the sequence. The concept of rate of convergence is of practical importance when working with a sequence of successive approximations for an iterative method, as then typically fewer iterations are needed to yield a useful approximation if the rate of convergence is higher. This may even make the difference between needing ten or a million iterations.

Similar concepts are used in combinatorial optimization and algorithms. In our case we need to categorize the convergence we are searching in to two categories:

- Convergence to a point

  For the first category (also known as stability analysis) we need to note that convergence has nothing to do with the quality of the solutions at the end of the run. It is only involved with a sequence that will end up inside the search space rather than going to infinity. The idea is if is there any guarantee that the sequence of the generated solutions by the algorithm converges to a single point, i.e., the generated sequence is not divergent. There are many types of such convergence: first order, second order, etc. First order tells us if the generated solution converges in expectation. Meaning that if we run the algorithm for infinite number of steps, the average of the positions of the generated points converge. This is not enough to guarantee that the positions generated end up

inside the search space though. However, second order stability guarantee that the variance (rather than the expected value) of the generated solutions is also zero, meaning that positions converge to a point inside the search space and do not move. In simple manners if we run the algorithm infinite numbers of iterations, in one instance, then the algorithm will converge to one point/ value, whether it is optimum or not.

- Convergence to an optimum

  For this category, we are after a proof that the generated solution will converge to a local optimum (or global optimum) eventually (note the difference with 1, where no assumption was made for the quality of the solution). This also may have many different types, but one frequently used is convergence in probability (almost surely, or asymptotically). The idea is: the generated solutions will be arbitrarily close to a local optimum when time goes to infinity, (Mohammad Reza Bonyadi, 2014). We should note that there are meta-heuristic methods that do guarantee local convergence. Hence, in this category, we aim for a convergence in a value, which value can be prooved to be local (or global) optimum.

The convergence capability assessment (i.e. the improvement of the quality of the solutions of the algorithm as iterations increase) was based on the kilometric distance travelled by the vehicles for each of the solutions of the algorithm. The convergence capability was assessed under the two following criteria: i) the decrease of the travelled distance, as the GRASP iterations increase, assuming that if the first GRASP constructed solution will be of good quality then the LNS procedure will find a better one more easily. Hence there are more possibilities to find a better solution, if the iterations increase and ii) the decrease of the travelled distance, as the LNS iterations increase.

Figures 7.9, 7.10 and 7.11 in the appendix, indicate that the increase of the number of iterations leads to an improvement of total travelled distance of the solution. Lastly, we can see familiar results in figure 7.12 and 7.13, while in these only the number of LNS iterations increased, while the construction algorithm had the same number of iterations. For the instances with a lower number of customers, an optimum, probably global, was achieved for each case of 50, 100 or 150 LNS iterations, hence we believe it has become obsolete to include it.

Last but not least, we can see in figures 7.7 to 7.10, and it is very obvious that the algorithm converges to a point/ value, almost every time. We could make hundreds of these diagrams, but is obsolete, as in all instances such a point has been approach, either from the construction algorithms (for the small problems), or from the LNS procedure.


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Chapter 7

Conclusions

In this thesis various aspects of the Vehicle Routing Problems, especially the heterogeneous multi-compartment VRP were discussed. In this final section we give a short summary of our findings we believe have been provided in this thesis.

First of all, we needed to make a literature review, in order to embrace as good as possible the Vehicle Routing Problems area. We spoke about the simple VRPs (Capacitated VRP and TW-VRP), but even about the more complex ones (BL-VRP, Multi-depot VRP, DVRP, VSP), that have increasing complexity, as they have more constraints.

We provided an overview of the challenged general problem, the Multi-Compartment VRP (MCVRP). We explained its applications, how it can be very important at deliveries of Food and Groceries, cattle food to farms and of course petroleum deliveries. Along with the problem's application, we also discussed its formulation on chapter 4. In chapter 4 we presented a more specific outlook of the fuel delivery problem we were tackling. We did a literature review for the specific problem tackled, the heterogeneous fleet VRP with multi - compartment vehicles, we explained the problem's specifics, the assumptions we had to make, the loading and unloading constraints and how these constraints may have an impact on solutions.

Furthermore, we explained the different types of algorithms and their differences, and the algorithm we are using, the Large Neighborhood Search. On chapter 3 we provided a more general view about the selected algorithm.

On chapter 5 we explained the operators used for the solution of this fuel delivery problem until this time. Also, there was a further explanation and targeting of the LNS procedure implemented for this problem.

Last but not least on chapter 6 we presented the computational results of the created instances and discussed how we approached the evaluation of the implemented method. On the Appendix (8) you can find the figures we deemed important for the assessment.

7.1 Further Research Perspectives

Even though the rapid development in computer hardware along with a long line of improvements in optimization software has pushed the boundaries of what is possible to solve in a reasonable amount of time, it is still computationally hard to solve real-life VRP's. This is mainly due to the wide variety of side constraints that

are often present in a real-life routing problems. Naturally, this fact makes it almost absurd to encompass a mathematical programming based VRP model within a real-time environment. Although, it is of great importance to obtain the mathematical program, because having the mathematical formulation, almost always provides some ground rules for research. And who knows, maybe in the future, we will be able to solve the NP hard problems in an instance.

Meta-heuristics usually also require relatively long computation times in order to provide good quality solutions. However, in some real-world applications meta-heuristics like tabu search and simulated annealing might prove to be another good choice of method. This is due to the fact that these methods for the most part will be able to find a feasible solution within a reasonable amount of time. This is an important issue since a real-life fuel delivery problem should be able to provide the dispatcher with either a feasible solution generally quickly or a message saying that the request is being rejected due to in-feasibility so that the customer could be informed whether or not will be serviced the next day.

There are, also, many occasions in real-life fuel delivery problems where constraints are different. Many fuel distribution companies exist and for each one, the operations and ways of business vary. Some use dispatchers, some use homogeneous fleet of vehicles, others don't use the precedence constraints used in this thesis. Others use IRP, namely the distribution company has a vast knowledge of the customer's inventory, hence uses this information for routing and distribution. Thereafter, due to so many alternative problems, the research perspective are huge.

Chapter 8

Appendix A: Figures

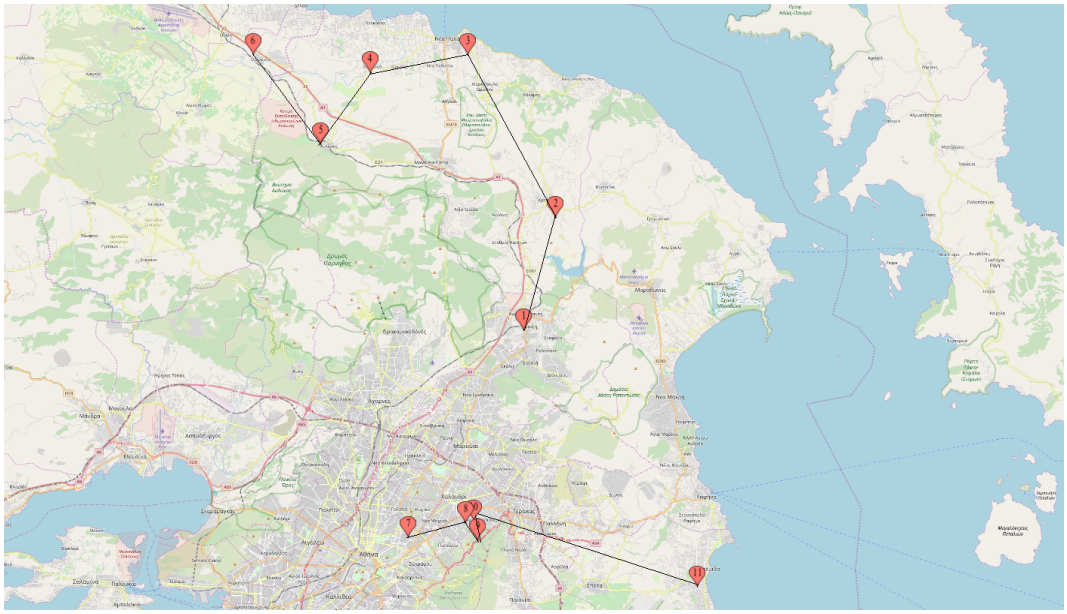


FIGURE 8.1: Two routes, containing 6 and 5 customers.

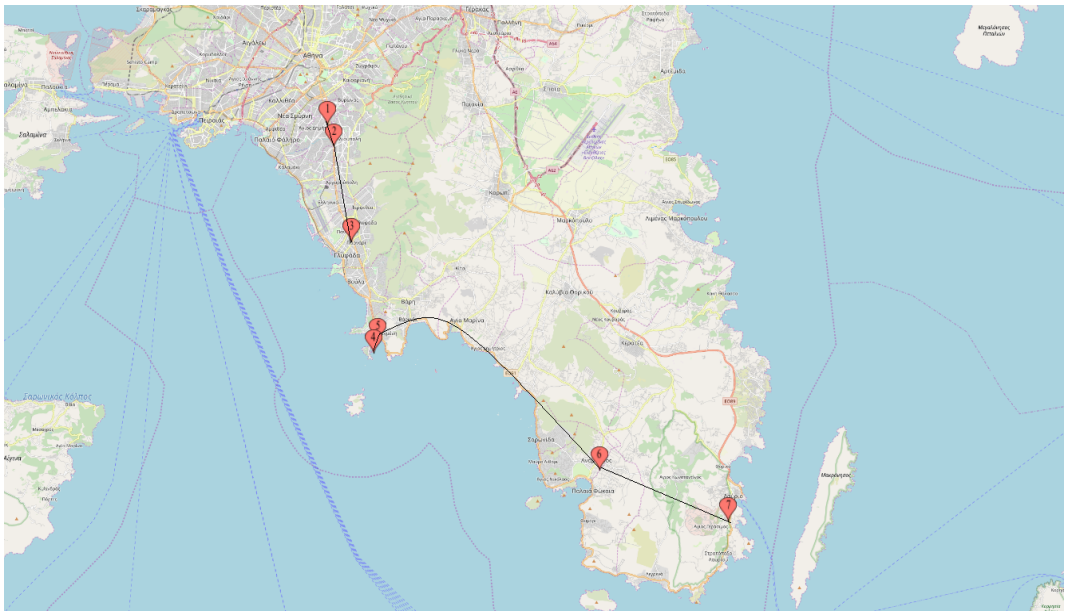


FIGURE 8.2: Two routes, containing 3 and 4 customers.

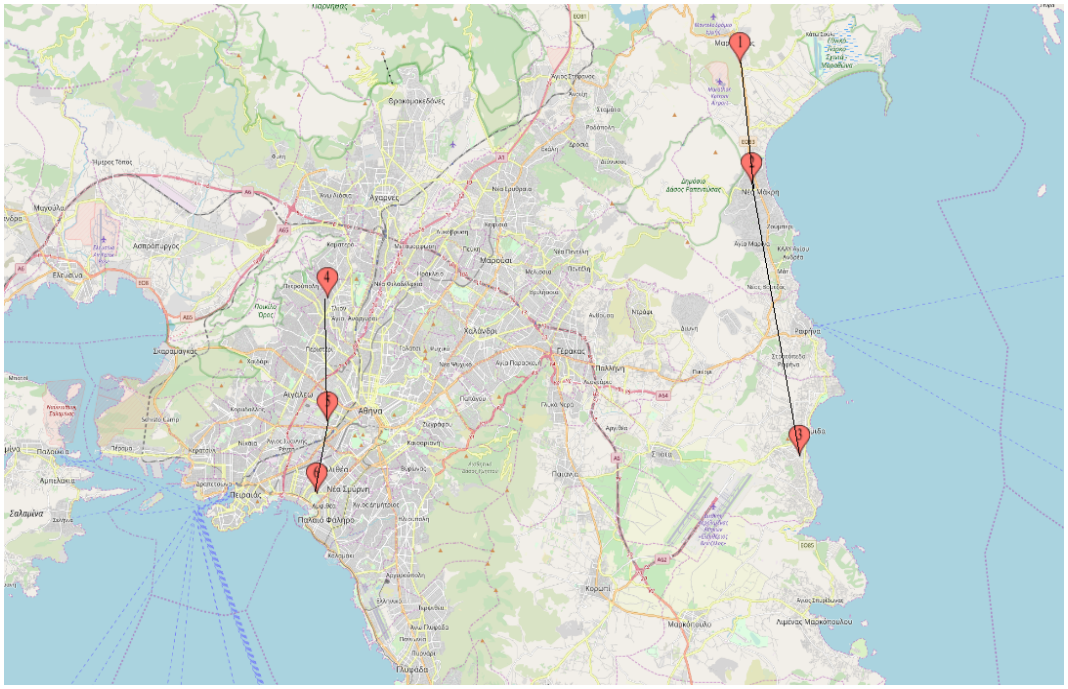


FIGURE 8.3: Two routes, containing 3 customers.

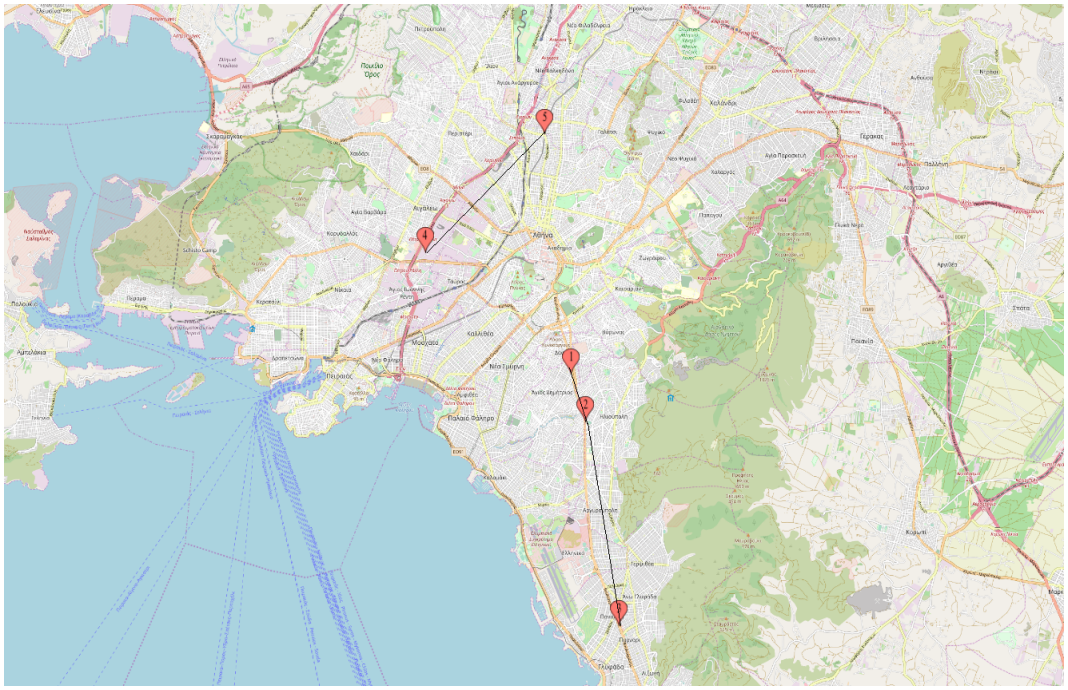


FIGURE 8.4: Two routes, containing 3 and 2 customers.

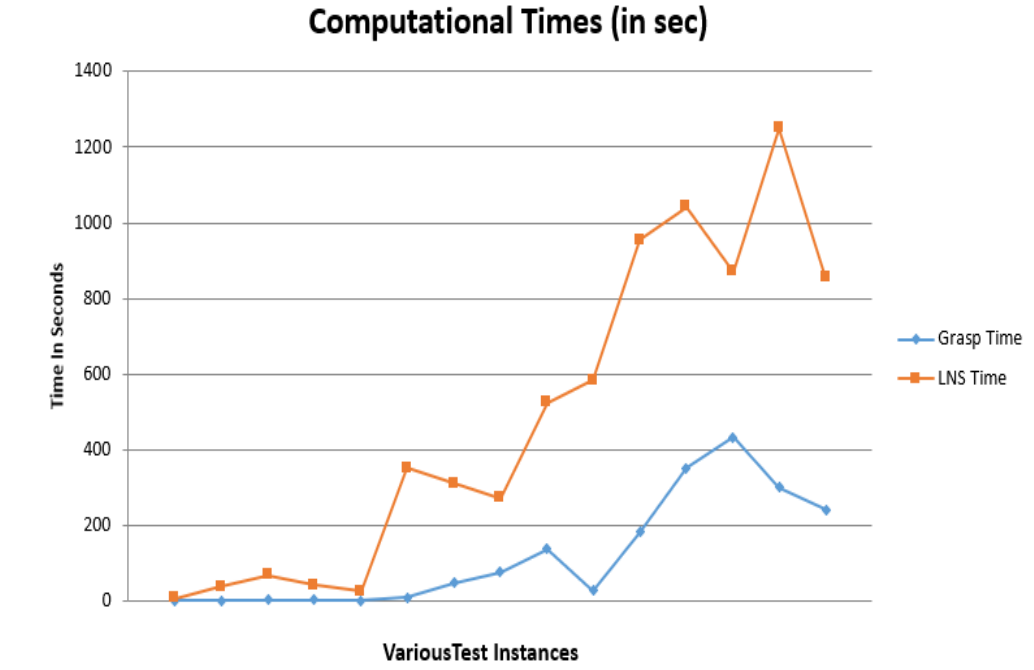


FIGURE 8.5: Computational time throughout the algorithm.

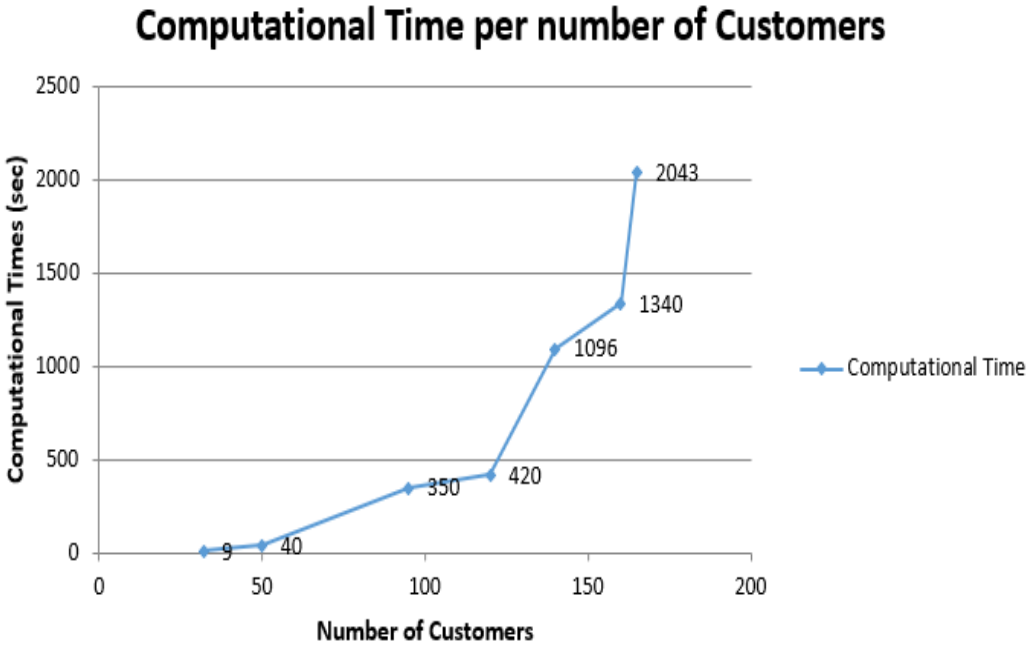


FIGURE 8.6: Computational time function to the LNS iterations.

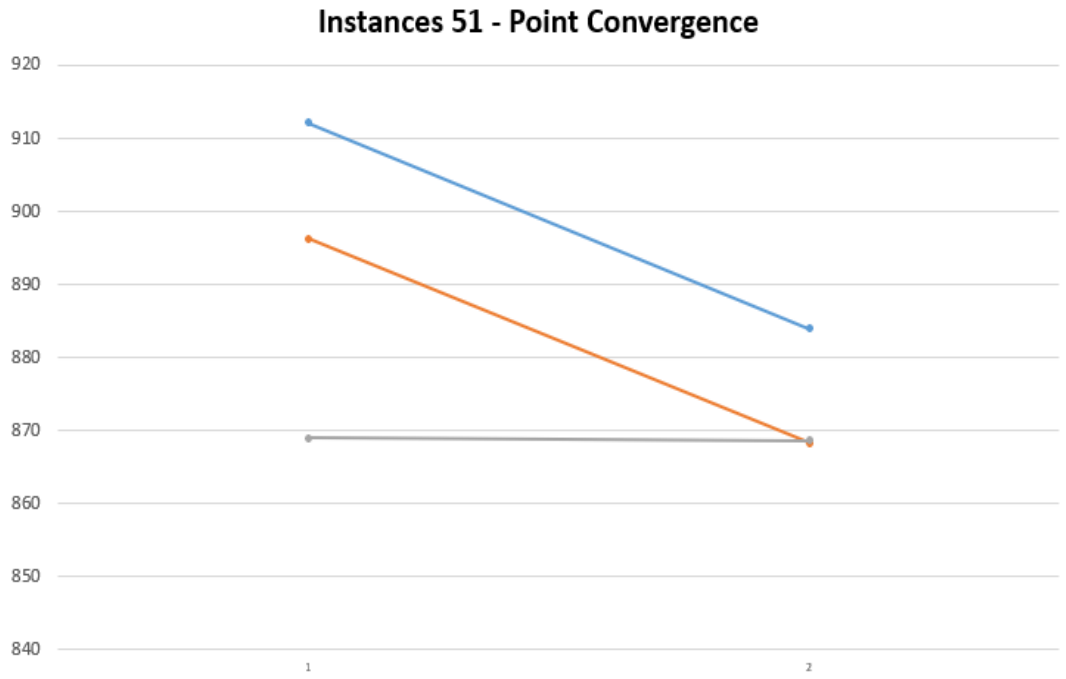


FIGURE 8.7: Point Convergence for 51 customer instance.

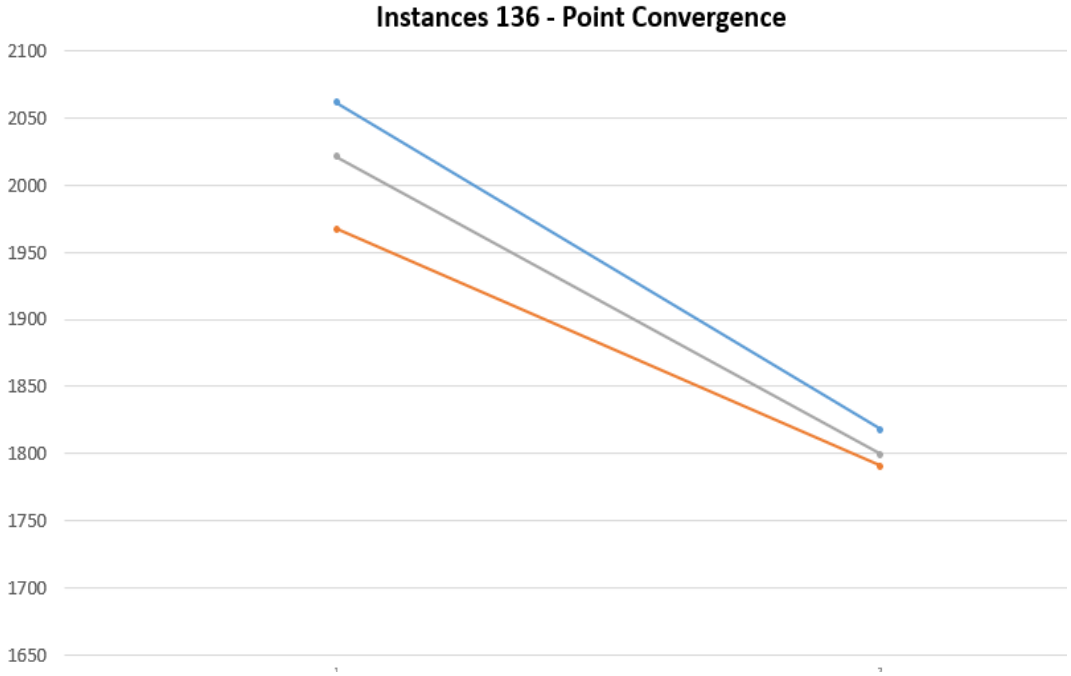


FIGURE 8.8: Point Convergence for 136 customer instance.

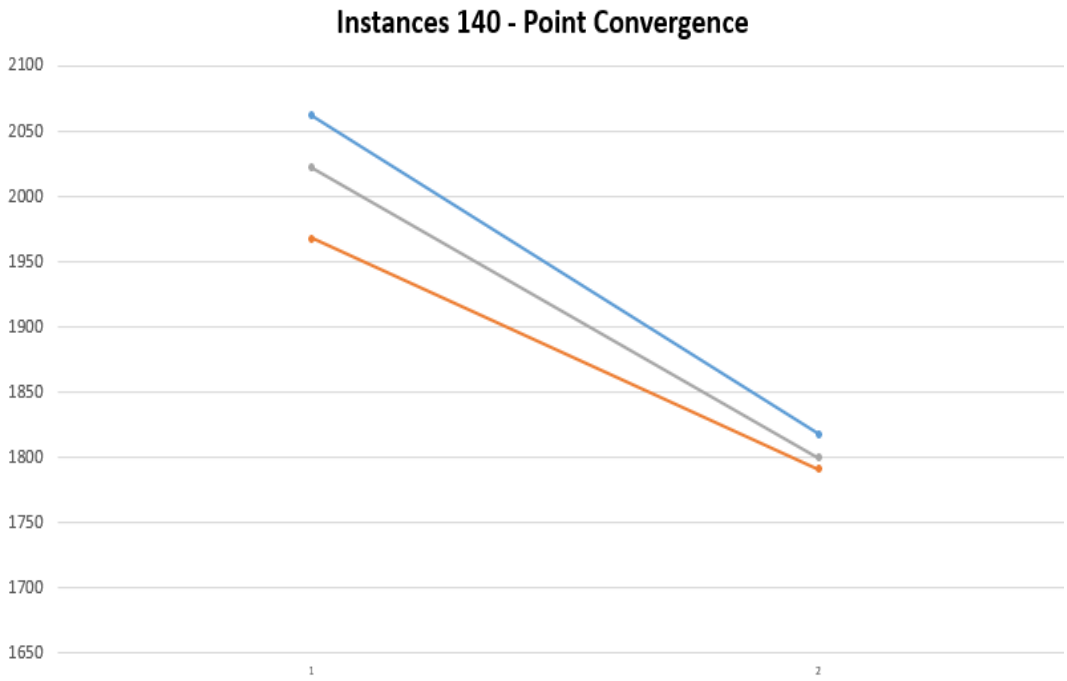


FIGURE 8.9: Point Convergence for 140 customer instance.

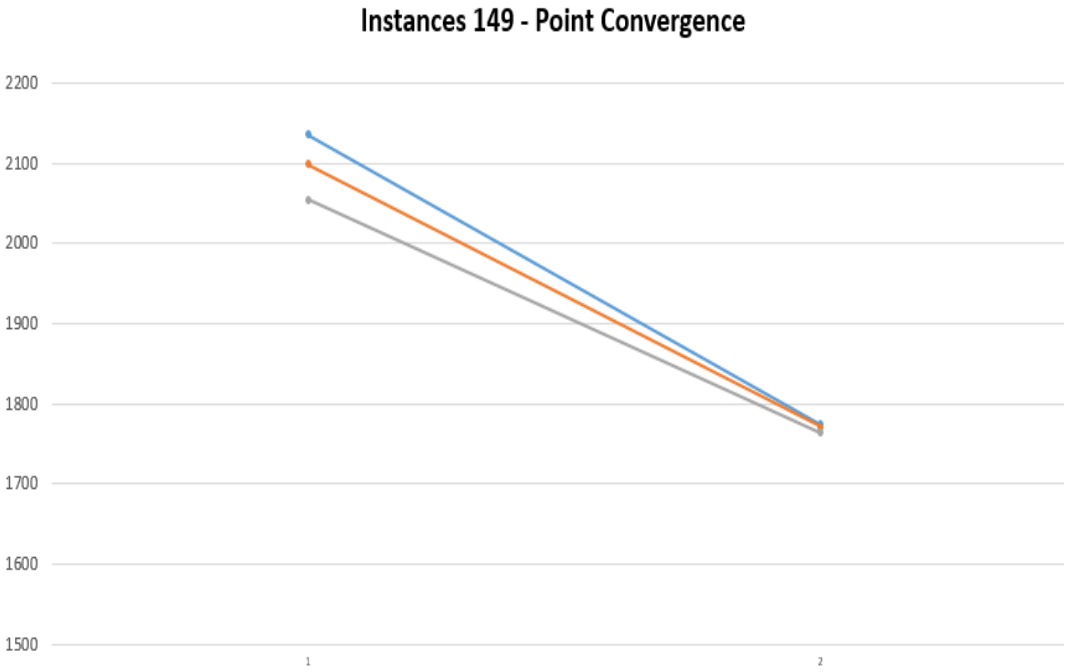


FIGURE 8.10: Point Convergence for 149 customer instance.

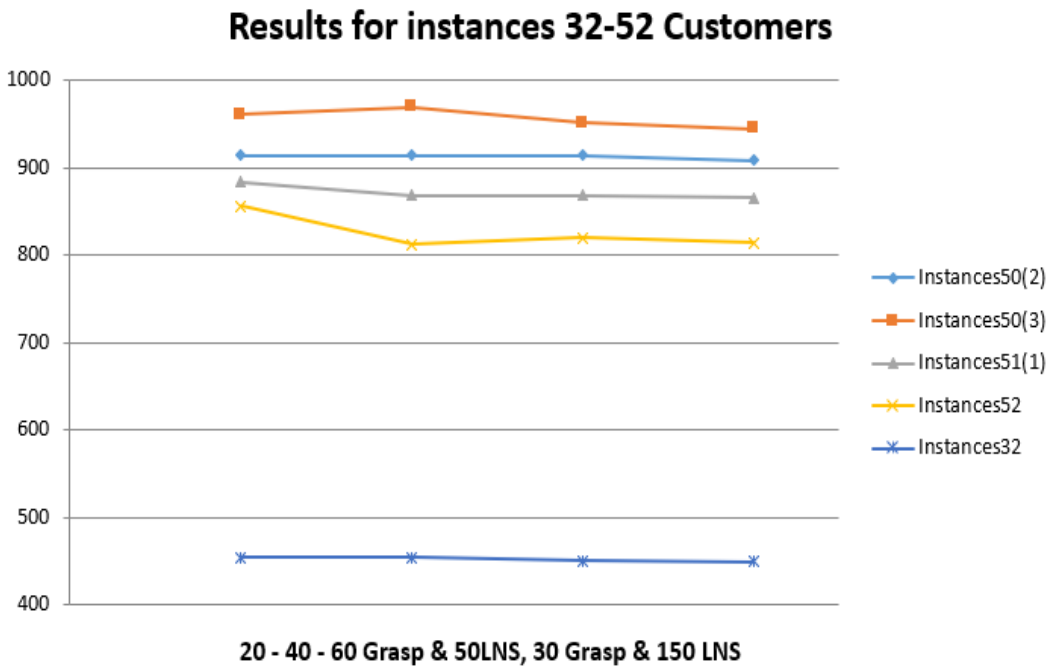


FIGURE 8.11: Total distance travelled as a GRASP-LNS function for "small" instances.

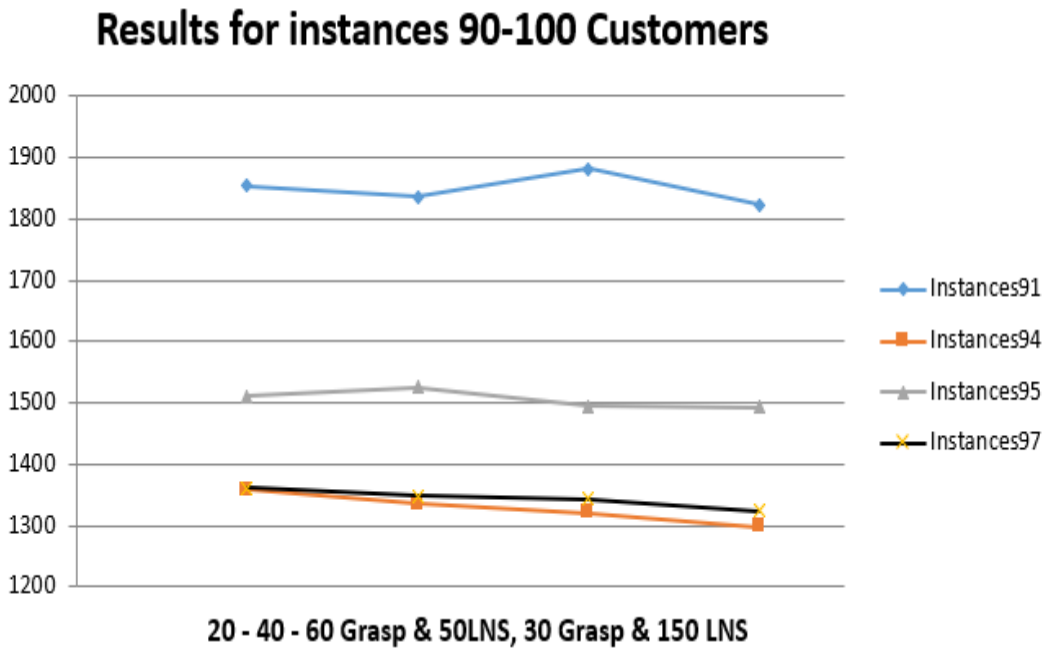


FIGURE 8.12: Total distance travelled as a GRASP-LNS function for "medium" instances.

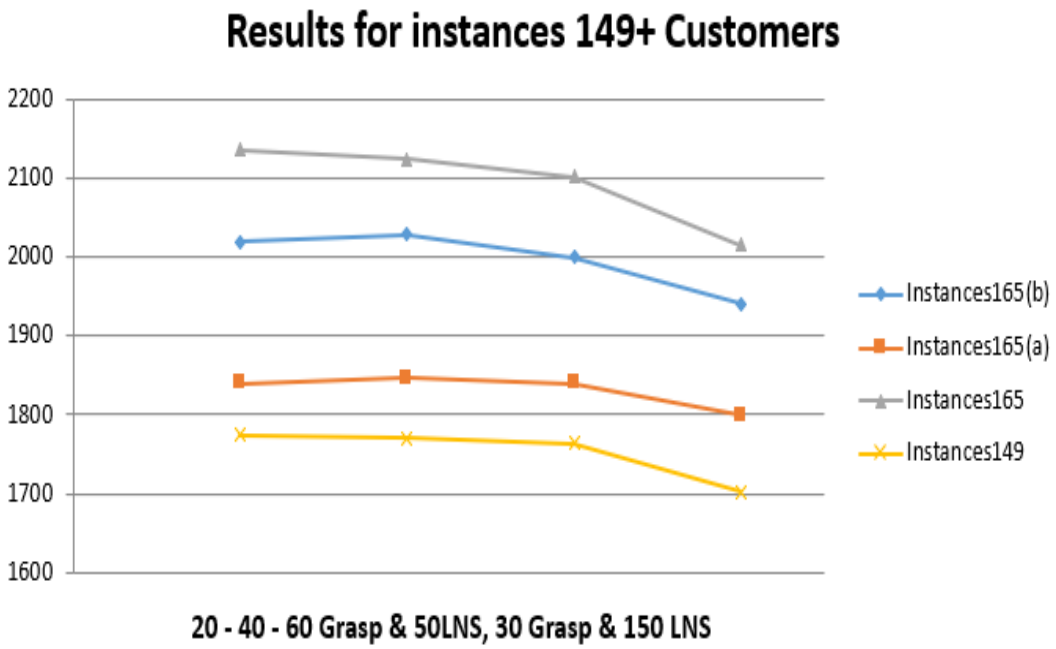


FIGURE 8.13: Total distance travelled as a GRASP-LNS function for "big" instances.

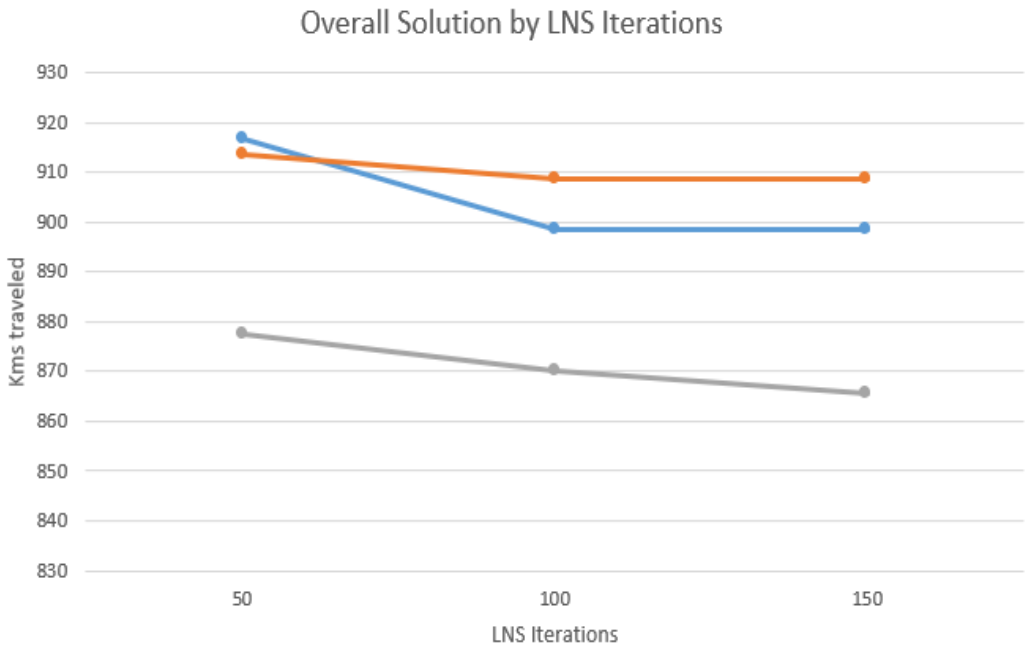


FIGURE 8.14: Total distance travelled as a LNS function for "small" instances.

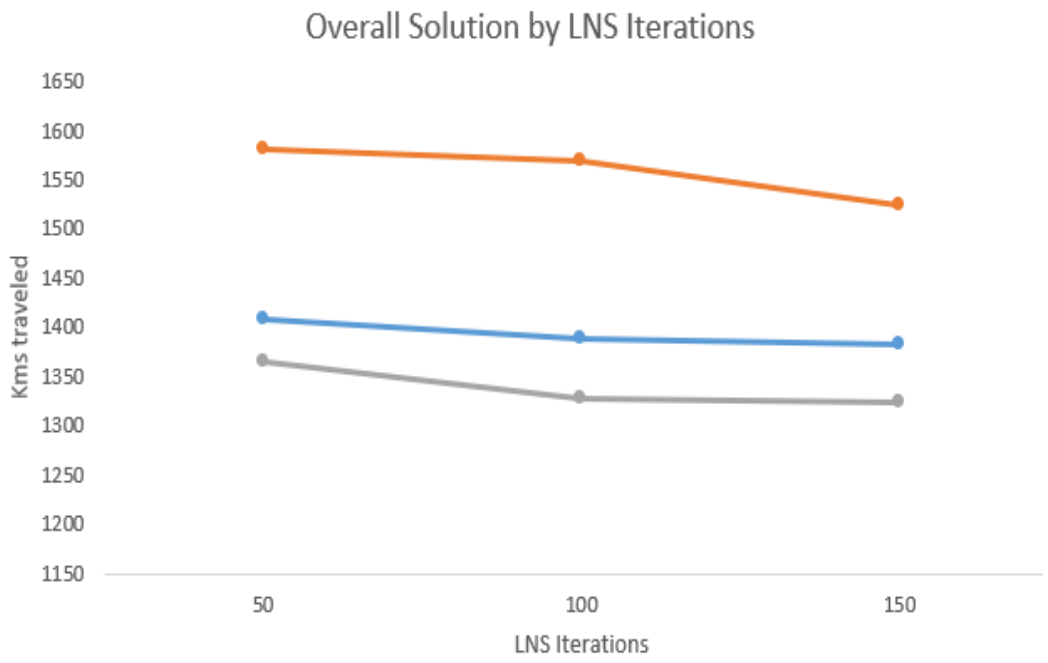


FIGURE 8.15: Total distance travelled as a LNS function for "medium" instances.

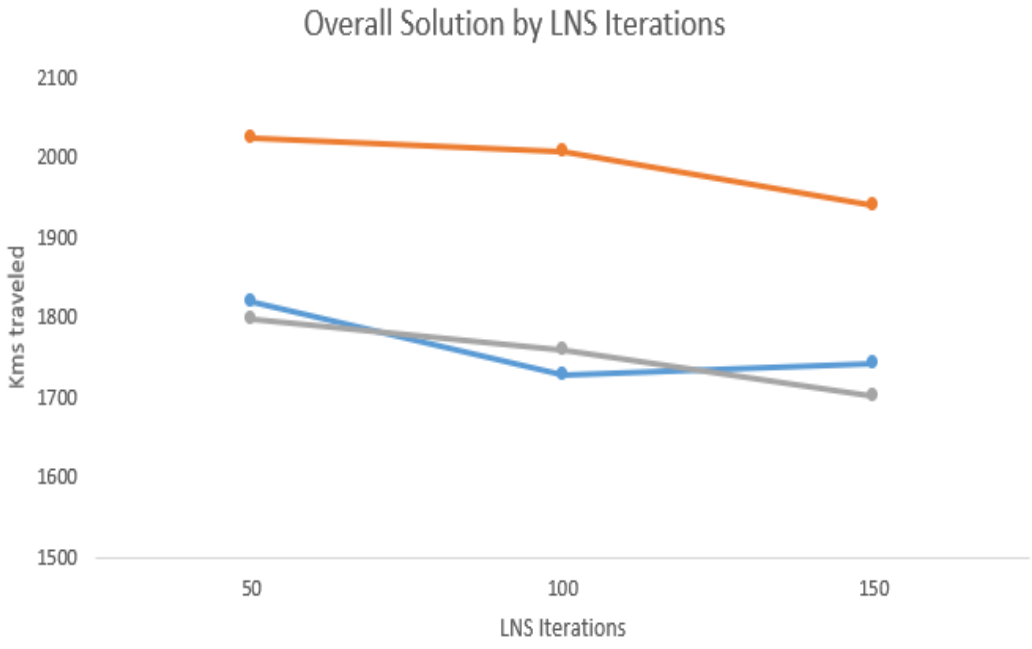


FIGURE 8.16: Total distance travelled as a LNS function for "big" instances.

Chapter 9

Appendix B: Code

This is the method that selects randomly one customer

```
public static ELPERouteStop selectRStoDestroy(ArrayList<Vehicle>
    all_vehicles,ArrayList<shadowVehicle> all_svehicles)
{
   Random n = new Random();
   Vehicle veh = all_vehicles.get(n.nextInt(all_vehicles.size()));
   shadowVehicle sveh = Routing_Procedure_Shadow.getshadowVehicle(veh,
       all_svehicles);
   ArrayList<ELPEVehicleLoaded> Vls = sveh.getVLs();
   while(Vls.isEmpty())
   {
      veh = all_vehicles.get(n.nextInt(all_vehicles.size()));
      sveh = Routing_Procedure_Shadow.getshadowVehicle(veh, all_svehicles);
      Vls = sveh.getVLs();
   }
   ArrayList<ELPERouteStop> ArrayRouteStop =
       Vls.get(n.nextInt(Vls.size())).getRouteStops();
   ELPERouteStop routeStop =
       ArrayRouteStop.get(n.nextInt(ArrayRouteStop.size()));
   return routeStop;
}
```

This is the method that creates the neighborhood that LNS is going to destroy and repair

```
public static ArrayList<ELPERouteStop> getNeighborRSFromSC(SolutionCreated
    sc,ELPERouteStop rs, double neighborPercentage,
    ArrayList<ELPEVehicleLoaded> VLsToDismiss, ArrayList<Distance> distances
    , ArrayList<shadowVehicle> all_svehicles){
     ArrayList<ELPERouteStop> neighborRS = new ArrayList<ELPERouteStop>();
     /* Given a solution, a customer and a destruction percentage
      * this method finds and returns the appropriate and needed
      * customers to destroy and extract from the current routing plan
      */

     if (VLsToDismiss==null){
        VLsToDismiss = new ArrayList<ELPEVehicleLoaded>();
     }
     double minDist = 0;
     double maxDist = 0;
```

```
    int maxNumberRS=Math.min((int)
        Math.round(sc.getAssignedRSList().size()*neighborPercentage),
        sc.getAssignedRSList().size()-1);
    double DistStep = 0;

    //Implementation of step distance
    if (rs.getELPELandMark().getUrban()){
       DistStep = 2000;
       maxDist = 20000;
    }
    else{
       DistStep = 10000;
       maxDist = 150000;
    }
    double fromDist = minDist;
    double toDist = fromDist+DistStep;
    boolean found = false;

     while (neighborRS.size()<maxNumberRS && toDist<=maxDist){
    int i = 0;
    while (i<sc.getUsedVehicleList().size() && !found){
       Vehicle veh = sc.getUsedVehicleList().get(i);
       shadowVehicle sveh = getshadowVehicle(veh,all_svehicles);

       for (int j=0;j<sveh.getVLs().size();j++){

          if (!VLsToDismiss.contains(sveh.getVLs().get(j))){
             ArrayList<ELPERouteStop> neighborVL_RS = getNeighborRS
                 (rs,sveh.getVLs().get(j).getRouteStops(),fromDist,
                 toDist,distances);
             if (neighborVL_RS.size()>0){
                VLsToDismiss.add(sveh.getVLs().get(j));
                ELPEVehicleLoaded vl = sveh.getVLs().get(j);
                neighborRS.addAll(sveh.getVLs().get(j).getRouteStops());
             }
             if (neighborRS.size()>=maxNumberRS){
                found=true;
             }
          }
       }
       i++;
     }
     fromDist=toDist;
     toDist = fromDist+DistStep;
     }
    return neighborRS;
  }
```

```
public OverallSolution LNSRoutine_Distance(OverallSolution
    os,ArrayList<Premise> loading_facilities,ArrayList<ELPERouteStop>
    all_stops,ArrayList<Vehicle> all_vehicles,ArrayList<Distance>
    distances,VLRepository vlrep,ArrayList<Parameters>
    parameter,ArrayList<Double> parameters, ArrayList<Rotation>
    rotation_list,ArrayList<Compartment>
    all_compartments,ArrayList<OrderLineItem>
    orderelements,ArrayList<Integer> ranks,int niterations, int
    number_of_intervals){
```

```
    OverallSolution bestSol = null;
    SolutionCreated sc= null;
    OverallSolution new_os = null;
    OverallSolution current_os = os;
    ELPERouteStop rs_out = null;

    /*
    This method is the one called from the original application. It
        initiates all the information needed and calls the
        LNSiteration_Distance
    */
    for (int iprem=0;iprem<loading_facilities.size();iprem++){

       Premise vprem = loading_facilities.get(iprem);
       ArrayList<ELPERouteStop> unservicedRS = new
           ArrayList<ELPERouteStop>();

       for (int k=0;k<current_os.get_unserviced_stops_list().size();k++){
          if (current_os.get_unserviced_stops_list().get(k).
              getELPEORDER().getDepotId(). equals(vprem.getPremise_ID())){
             unservicedRS.add(current_os.get_unserviced_stops_list().get(k));
          }
       }

       int icount=0;
       niterations=parameter.get(0).getLns_iterations();
       while (icount<niterations ){
          System.out.println("LNS ITERATION:"+icount);
          rs_out=selectRStoDestroy(current_os,vprem);
          sc=current_os.getSolutionCreatedByPremiseLogo(vprem, "NOLABEL");
          if (sc!=null){
             new_os = LNSiteration_Distance(number_of_intervals,current_os,
                 sc, rs_out, loading_facilities, all_stops,
                 all_vehicles,current_os.get_all_svehicles(), distances,
                 vlrep, parameter, parameters, rotation_list,
                 all_compartments, orderelements, ranks);

             if (new_os!=null){
                current_os=new_os;
             }
             icount++;
          }
       }
       bestSol = current_os;
    }

    return bestSol;
}
```

This method is the iterator of the method

```
public static OverallSolution LNSiteration_Distance
                  (int number_of_intervals,OverallSolution
                      os,SolutionCreated sc,ELPERouteStop
                      rs_out,ArrayList<Premise> loading_facilities,
                  ArrayList<ELPERouteStop> all_stops,ArrayList<Vehicle>
                      all_vehicles,ArrayList<shadowVehicle> all_svehicles,
```

```
                ArrayList<Distance> distances,VLRepository
                    vlrep,ArrayList<Parameters>
                    parameter,ArrayList<Double> parameters,
                ArrayList<Rotation> rotation_list,ArrayList<Compartment>
                    all_compartments,ArrayList<OrderLineItem>
                    orderelements,
                ArrayList<Integer> ranks){

/*
 This method implements each iteration of the LNS_Distance. Takes an
     OverallSolution, RouteStops to remove, destroys partially the
     solution
 and then repairs it.
*/
SwapAlgorithmShadow sa = new SwapAlgorithmShadow();
OverallSolution response = null;
OverallSolution new_os = new OverallSolution(os,vlrep);
ArrayList<shadowVehicle> newall_svehicles = new_os.get_all_svehicles();
double destroyPercent = parameter.get(0).getDestruction_percent();
SolutionCreated trial_sc =
    new_os.getSolutionCreatedByPremiseLogo(sc.getPremise(), sc.getLogo());
ArrayList<ELPERouteStop> RSs_to_be_removed = null;
ArrayList<ELPERouteStop> UnservicedRSs_to_be_routed = new
    ArrayList<ELPERouteStop>();
ArrayList<ELPEVehicleLoaded> VLsToDismiss = new
    ArrayList<ELPEVehicleLoaded>();
  double
      rDestroyPercent=0.1+Math.random()*parameter.get(0).getDestruction_percent();

//Find the neighboring route stops to destroy.
RSs_to_be_removed=Routing_Procedure_Shadow.getNeighborRSFromSC(trial_sc,rs_out,
    rDestroyPercent, VLsToDismiss, distances ,
    new_os.get_all_svehicles());

if (!RSs_to_be_removed.contains(rs_out))
{
   RSs_to_be_removed.add(rs_out);
}

//Call the Partially Destroy Method
ArrayList<ELPERouteStop> RSs_to_be_routed =
    PartiallyDestroySC(number_of_intervals,trial_sc,distances,RSs_to_be_removed,
    all_stops,new_os.get_all_svehicles(),new_os.get_SlotsTableByPrem(sc.getPremise().getPre

//Call the repair Method
LNSRepair_Distance(sa,new_os,trial_sc,UnservicedRSs_to_be_routed,
    RSs_to_be_routed,all_stops,all_vehicles,loading_facilities,distances,vlrep,
    parameter, parameters, rotation_list , all_compartments,
    orderelements, ranks);

//Update the repaired OverallSolution
new_os.UpdateUsedVehiclesOS();
new_os.UpdateUnservicedStopsOS(all_stops);

double kmsnew =
    Routing_Procedure_Shadow.ComputeDistanceKilometersEnhanced(new_os,
    distances, new_os.get_all_svehicles());
```

```
   double kmsold =
       Routing_Procedure_Shadow.ComputeDistanceKilometersEnhanced(os,
       distances, os.get_all_svehicles());


   /* If the unserviced customers of the new created solution are less than
       the original solution
    or the kms of the new solution are less than the old, having the same
        unserviced stops,
    then the new solution is better than the old one, and we keep it and
        update it as our current.
    Else, we redo the process.
   */
   if ((new_os.get_unserviced_stops_list().size() <
       os.get_unserviced_stops_list().size()) ||
       (new_os.get_unserviced_stops_list().size() ==
       os.get_unserviced_stops_list().size() && kmsnew < kmsold))
   {
      System.out.println("New Overall Solution was found by LNS. --- Old
          Solution Kilometers: "+kmsold +" and unrouted: "+
          os.get_unserviced_stops_list().size()+ " New Solution Kilometers:
          "+kmsnew + " and unrouted: "+
          new_os.get_unserviced_stops_list().size());
      response = new_os;
      os.releaseOverallSolution(vlrep);
   }

   else
   {
      System.out.println("No Overall Solution was found by LNS");
      new_os.releaseOverallSolution(vlrep);
   }
   return response;
}
```

This is the Destroy method

```
public static ArrayList<ELPERouteStop> PartiallyDestroySC(int
    number_of_intervals,SolutionCreated sc,ArrayList<Distance>
    distances,ArrayList<ELPERouteStop>
    RS_list_to_remove,ArrayList<ELPERouteStop>
    all_stops,ArrayList<shadowVehicle> all_svehicles,ELPESlotTable st,
    VLRepository vlrep,ArrayList<Vehicle> all_vehicles){

   ArrayList<ELPEVehicleLoaded> all_used_vehicles_VLs = new
       ArrayList<ELPEVehicleLoaded>();
   ArrayList<ELPERouteStop> RS_removed= new ArrayList<ELPERouteStop>();
   if (RS_list_to_remove.size()>0){
      Premise prem = sc.getPremise();
      int number_of_pumps = prem.getPremise_Pumps();
      // MAKE A LIST WITH ALL Routes
      for (int i=0;i<sc.getUsedVehicleList().size();i++){
         Vehicle veh = sc.getUsedVehicleList().get(i);
         shadowVehicle sveh =
             Routing_Procedure_Shadow.getshadowVehicle(veh,all_svehicles);
         all_used_vehicles_VLs.addAll(sveh.getVLs());
      }
```

```
      //Find routes that contain at least one customer in RS_list_to_remove
          (These are the routes that will be destroyed)
      ArrayList<ELPEVehicleLoaded> VLs_to_destroy = new
          ArrayList<ELPEVehicleLoaded>();
      for (int i=0;i<RS_list_to_remove.size();i++){
         ELPERouteStop rs_out = RS_list_to_remove.get(i);
         if (!RS_removed.contains(rs_out)){
            RS_removed.add(rs_out);
         }
         ELPEVehicleLoaded new_vl =
             Routing_Procedure_Shadow.getVL_of_RS_fromUsedVehicles(rs_out,sc,
             all_svehicles);

         if (!VLs_to_destroy.contains(new_vl)){
            VLs_to_destroy.add(new_vl);
         }
      }
      ArrayList<Vehicle> VehiclesToReschedule = new ArrayList<Vehicle>();
      // DESTROY VLs FOUND ABOVE
      for (int i=0;i<VLs_to_destroy.size();i++){
         ELPEVehicleLoaded vl = VLs_to_destroy.get(i);
         Vehicle veh =
             all_vehicles.get(VLs_to_destroy.get(i).getVehicleRefId());
         //Remove them
         RemoveVLfromSC(sc,vl,all_svehicles);
         //Erase their time slots
         Routing_Procedure_Shadow.EraseTimeSlot(vl, st.getSlotsTable(),
             st.getSlotsTable().length, st.getSlotsTable()[0].length);
         for (int j=0;j<vl.getRouteStops().size();j++){
            ELPERouteStop rs_out = vl.getRouteStops().get(j);
            if (!RS_removed.contains(rs_out)){
               RS_removed.add(rs_out);
            }
         }
         vlrep.EraseVehicleLoaded(vl);
         if (sc.getUsedVehicleList().contains(veh)){
            if (!VehiclesToReschedule.contains(veh)){
               VehiclesToReschedule.add(veh);
            }
         }
         else
         {
            VehiclesToReschedule.remove(veh);
         }
      }
      for (int i=0;i<VehiclesToReschedule.size();i++){
         Vehicle veh = VehiclesToReschedule.get(i);
         //Reschedule the vehicles
         Routing_Procedure_Shadow.RescheduleVehicleRoutes (number_of_pumps,
             number_of_intervals,veh, st.getSlotsTable(),
             all_svehicles,sc.getUsedVehicleList(),prem, all_vehicles,
             distances,all_stops);
      }
    }
   return RS_removed;
}
```

```
public static void LNSRepair_Distance(SwapAlgorithmShadow
    sa,OverallSolution os,SolutionCreated partial_sc,
   ArrayList<ELPERouteStop> UnservicedRS_list,ArrayList<ELPERouteStop>
       ServicedRS_list,
   ArrayList<ELPERouteStop> all_route_stops,ArrayList<Vehicle> all_vehicles,
   ArrayList<Premise> loading_facilities,ArrayList<Distance>
       distances,VLRepository vlrep,
   ArrayList<Parameters> parameter,ArrayList<Double> parameters,
       ArrayList<Rotation> rotation_list,
   ArrayList<Compartment> all_compartments,ArrayList<OrderLineItem>
       orderelements,ArrayList<Integer> ranks){

   //Parameter Initialization
   ArrayList<shadowVehicle> all_svehicles = os.get_all_svehicles();
    Premise loading_facility1=partial_sc.getPremise();
    ArrayList<Vehicle> vehicles = new ArrayList<Vehicle>();
    vehicles.addAll(partial_sc.getUnused_Vehicles());

   //If the premises are correct
    if(loading_facility1!=null)
   {
       ArrayList<SolutionCreated> created_solution_list=new
           ArrayList<SolutionCreated>();
       ELPESlotTable st1 =
           os.get_SlotsTableByPrem(loading_facility1.getPremise_ID());
       int premise1_pumps=loading_facility1.getPremise_Pumps();
       int[][] all_slots=st1.getSlotsTable();
       int number_of_intervals = all_slots.length;
       int npumps =all_slots[0].length;

       if(UnservicedRS_list.size()!=0 || ServicedRS_list.size()!=0)
       {
         //Here we call the repair function
         FleetRouting_Shipments shipment_MC =
             Routing_Procedure_Shadow.Fleet_CompleteRouting_Distance
             (os,partial_sc, all_route_stops, UnservicedRS_list,
             ServicedRS_list, null, vehicles, all_vehicles, all_svehicles,
             null, loading_facility1, distances, all_compartments,
             orderelements, ranks,all_slots, "0", loading_facilities,
             premise1_pumps, number_of_intervals, false, false, false, vlrep,
             parameter, rotation_list);
         if(shipment_MC.getSolutionCreated().getUsedVehicleList().size()!=0)
         {
             int
                 iterations=shipment_MC.getSolutionCreated().getNumberOfIterations();
         }
         created_solution_list.add(shipment_MC.getSolutionCreated());

       }

       os.UpdateUsedVehiclesOS();
       os.UpdateUnservicedStopsOS(all_route_stops);
       //After the feasible repair, we change the vehicles' load times
       Routing_Procedure_Shadow.ChangeRouteLoadTime (os, all_vehicles,
           all_slots, number_of_intervals, premise1_pumps,
           loading_facility1, distances, parameter.get(0));
```

```
    }
}
```

Bibliography

Apotheker, S. (1990). "Construction and Demolition Debris - The invincible waste stream". In: *Resource Recycling* 9.

Avella P. M. Boccia, A. Sforza (2004). "Solving a fuel delivery problem by heuristic and exact approaches". In: *European Journal of Operational Research* 152, pp. 170–179.

Back T., Fogel D. B. and Michalewicz Z. (1997). *Handbook of Evolutionary Computation.* Oxford University Press.

Baldacci R., Mingozzi A. (2009). "A unified exact method for solving different classes of vehicle routing problems." In: *Mathematical Programming* 120, pp. 347–380.

Belfiore P., Yoshizaki H.T. (2009). "Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil." In: *European Journal of Operational Research* 199, pp. 750–758.

Bell W., P. Prutzman (1983). "Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer." In: *INFORMS Journal on Applied Analytics*.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bertazzi L., G. Speranza (2012). "Inventory Routing Problems: an introduction." In: *Euro J Transp Logist* 1, pp. 307–326.

Bielli M. Bielli A., Rossi R. (2011). "Trends in models and algorithms for fleet management." In: *Procedia Social and Behavioral Sciences* 20, pp. 4–18.

Brandao, J. (2011). "A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem." In: *Computers & Operations Research* 38, pp. 140–151.

Brown G. G, Glenn W.G (1987). "Real Time Dispatch Of Petroleum Tank Trucks". In: *Management Science* 27.

Brown G., G. Graves (1981). "Real time dispatch of petroleum tank trucks". In: *Management Science* 27.

Cerny, V. (1985). "A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm". In: *Journal of Optimization Theory and Applications* 45, pp. 41–51.

Chajakis E., M. Guinard (2003). "Scheduling Deliveries in Vehicles with Multiple Compartments". In: *Journal of Global Optimization* 26, pp. 43–78.

Choi E., Tcha D.W. (2007). "A column generation approach to the heterogeneous fleet vehicle routing problem." In: *Computers & Operations Research* 34, pp. 2080–2095.

Clarke G., Wright J. W. (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points." In: *Operations Research.* 12, pp. 519–643.

Coelho L., Laporte G. (2015). "Classification, model and exact algorithms for multi-compartment delivery problems". In: *European Journal of Operational Research*.

Comtois C. Slack B., Rodrigue J.P. (2013). *The geography of transport systems (3rd ed.* London: Routledge, Taylor & Francis Group.

Cornillier F., G. Laporte et al. (2009). "The petrol station replenishment problem with time windows". In: *Computers and Operations Research* 36, pp. 915–935.

Cornillier F., Renaud J. (2008). "An exact algorithm for the petrol station replenishment problem". In: *Journal of the Operational Research Society* 59, pp. 607–615.

Danna E., Perron L. (2003). "Structured vs. Unstructured Large Neighborhood Search: A Case Study on Job-Shop Scheduling Problems with Earliness and Tardiness Costs." In: 2883.

Dantzig G. B., Ramser J. H. (1959). "The Truck Dispatching Problem." In: *Management Science.* 6, pp. 1–140.

Demir E. Bektas T., Laporte G. (2012). "An adaptive large neighborhood search heuristic for the Pollution-Routing Problem." In: *European Journal of Operational Research* 223, pp. 346–359.

Derigs U., U. Vogel et al. (2011). "Vehicle Routing with compartments: applications, modelling and heuristics". In: *OR Spectrum* 33.

Desrosiers Jacques, Francois Soumis et al. (1995). "Handbooks in Operations Research and Management Science". In: *Elsevier Science*, pp. 35–140.

Fallahi A. Prins C., R. W. Calvo (2008). "A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem". In: *ScienceDirect, Elsevier Science* 8, pp. 885–914.

Feo T., Resende M.G.C (1989). "A probabilistic heuristic for a computationally difficult set covering problem." In: *Operations Research Letters.* 8, pp. 67–71.

Fischer, Marshall L. (1995). "Network Routing". In: *Elsevier Science* 8, pp. 1–33.

Fogel J. L. Owens J. A., Walsh J. M. (1966). *Artificial Intelligence Through Simulated Evolution*. Wiley.

Geir Hasle Knut-Andreas Lie, Ewald Quak (2007). *Geometric Modelling, Numerical Simulation, and Optimization*. Applied Mathematics at SINTEF. Berlin: Springer Verlag.

Glover, F. (1989). "Tabu Search: Part I". In: *ORSA Journal on Computing.* 1, pp. 190–206.

Godard D. Laborie P., Nuijten W. (2005). "Randomized Large Neighborhood Search for Cumulative Scheduling." In: *American Association for Artificial Intelligence.*

Golden B., Assad A. (1984). "The fleet size and mix vehicle routing problem." In: *Computers & Operations Research* 11, pp. 49–66.

Golden B.L. Baker E., Schaffer J. (1985). *The vehicle routing problem with backhauling: Two approaches*. R. Hammesfahr, editor, Proceedings of the XXI Annual Meeting of S.E. TIMS, Myrtle Beach, SC, pp. 90–92.

Hart J.P, Shogan A.W. (1987). "Semi-greedy heuristics: An empirical study." In: *Operations Research Letters.* 6, pp. 107–114.

Henke T. Speranza G., Wäscher G. (2017). *A Branch-and-Cut Algorithm for the Multi Compartment vehicle Routing Problem with Flexbile Compartment Sizes*. FEMM Working Papers 170004. Otto-von-Guericke University Magdeburg, Faculty of Economics and Management. URL: https://ideas.repec.org/p/mag/wpaper/170004.html.

Holland, H. (1962). "Outline for a logical theory of adaptive systems". In: *Journal of the ACM* 3, pp. 297–314.

— (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Hong, Lianxi (2011). "An improved LNS algorithm for real-time vehicle routing problem with time windows." In: *Computers and Operations Research.* 39, pp. 151–163.

Kirkpatrick S. Gelatt C.D., Vecchi M.P. (1983). "Optimization by simulated annealing." In: *Science* 220, pp. 671–680.

Koza, R. (1992). *Genetic Programming*. MIT Press, Cambridge, MA.

Kritikos M.N, Ioannou G. (2013). "The heterogeneous fleet vehicle routing problem with over loads." In: *International Journal of Production Economics*.

Lahyani R. Coelho L., Semet F. (2014). "A Multi-Compartment Vehicle Routing problem Arising in the collection of Olive oil in Tunisia". In: *Interuniversity Research Center on Enterprise Network, Logistics and Transportation*.

Li X. Tian P., Y. Aneja (2010). "An adaptive memory programming metaheuristic for the heterogeneous fixed fleet vehicle routing problem." In: *Transportation Research Part E* 46, pp. 1111–1127.

Marinakis I., Mygdalas A. (2008). *Design and Optimization of the Supply Chain*. Sophia.

Masson R. Leheude F., Peton O. (2013). "An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers." In: 47, pp. 295–454.

Mendoza J.E, Velasco N. (2010). "A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands". In: *Computers and Operations Research* 37, pp. 1886–1898.

Mladenovic M., Hansen P. (1995). "Variable Neighborhood Search". In: *Computers and Operations Research* 24, pp. 1097–1100.

Moghaddam B.F. Ruiz R., Sadjadi S.J. (2012). "Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm." In: *Computers & industrial Engineering* 62, pp. 306–317.

Moghaddam B.F. Sadjadi S.J., Seyedhosseini S.M. (2010). "Comparing mathematical and heuristic methods." In: *International Journal of Research and Reviews in Applied Sciences* 2, pp. 108–116.

Mohammad Reza Bonyadi, Zbigniew Michalewicz (2014). "A locally convergent rotationally invariant particle swarm optimization algorithm." In: *Swarm Intelligence* 8.

Muyldermans L., G. Pang (2010). "On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm". In: *European Journal of Operational Research* 206, pp. 93–103.

Oppen Johan, Lokketangen A. (2008). "A tabu search approach for the livestock collection problem". In: *Computers and Operations Research* 35, pp. 3213–3229.

Osman J.H, Salhi S. (1996). *Local search strategies for the vehicle fleet mix problem. In Modern heuristic search methods.* Chichester: John Wiley & Sons, pp. 131–154.

Papadimitriou C., K. Steiglitz (1982). *Combinatorial Optimization - Algorithms and Complexity.* New Jersey: Prentice - Hall.

Penna P.H. Subramanian A., Ochi L.S. (2011). "An iterated local search heuristic for the heterogeneous fleet vehicle routing problem." In: *Journal of Heuristics* 19, pp. 201–232.

Prescott-Gagnon E. Desaulniers G., Rousseau L.M (2009). "A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows." In: *Networks* 54.

Prins, C. (2009). "Two memetic algorithms for heterogeneous fleet vehicle routing problems." In: *Engineering Applications of Artificial Intelligence* 22, pp. 916–928.

Renaud J., F. F. Boctor (2002). "A sweep-based algorithm for the fleet size and mix vehicle routing problem." In: *European Journal of Operational Research* 140, pp. 618–628.

Repoussis P., Tarantilis C. (2010). "Solving the fleet size and mix vehicle routing problem with time windows via adaptive memory programming." In: *Transportation Research Part C* 18, pp. 695–712.

Rodrigue J., Notteboom T. (2017). *The Geography of Transport Systems*. URL: https://transportgeography.org/?page_id=5268.

Russell S., P. Norvig (1995). *Artificial Intelligence A Modern Approach*. New Jersey: Prentice - Hall.

Shaw, P. (1998). "Using constraint programming and local search methods to solve vehicle routing problems". In: *Lecture Notes in Computer Science* 1520, pp. 417–431.

Shifeng Chen Rong Chen, Arun Kumar Sangaiah (2018). "An adaptive large neighborhood search heuristic for dynamic vehicle routing problems." In: *Computers and Electrical Engineering.* 67, pp. 596–607.

Stephen Smith, Frank Imeson (2017). "GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem." In: *Computers and Operations Research.* 87, pp. 1–19.

Talbi., El-Ghazali (2009). *Metaheuristics, from design to implementation.* John Wiley and Sons, Inc., Hoboken, New Jersey.

Tarantilis C. Kiranoudis C., Vassiliadis V. (2004). "A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem." In: *European Journal of Operational Research* 152, pp. 148–158.

Toth P., Vigo D. (1996). *A heuristic algorithm for the vehicle routing problem with backhauls*. L. Bianco and P. Toth, editors, Advanced Methods in Transportation Analysis, Springer-Verlag,Berlin.