UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

Diploma Thesis

DEVELOPMENT OF A MATHEMATICAL MODEL AND SOURCE CODES FOR THE SINGLE – DEPOT CAPACITATED VEHICLE ROUTING PROBLEM WITH HETEROGENEOUS

by

IOANNIS GIANNAKOU

Submitted to fulfill part of the requirements for the acquirement of the Diploma of Mechanical Engineer

Volos 2021

Approved by the Members of the Examination Committee:

First Examiner (Supervisor)	Dr. Georgios K.D. Saharidis Assistant Professor, Department of Mechanical Engineering, University of Thessaly
Second Examiner	Dr. Dimitris Pantelis Professor, Department of Mechanical Engineering, University of Thessaly
Third Examiner	Dr. George Liberopoulos Professor, Department of Mechanical Engineering, University of Thessaly

Acknowledgements

First of all, I would like to thank my supervisor Assistant Professor Dr. Georgios K.D. Saharidis, for his valuable guidance and support during my research for this thesis, for the patience he had in me. Moreover, I would like to thank my colleague Christos Papapostolou for their support not only in research but also in personal life. Moreover I want to thank my friend Babis Markos for his help to create the source codes. In addition, I am grateful to all professors of the department of Mechanical Engineering of University of Thessaly for everything that I learned during my studies.

Above all, I am grateful to my parents and my brother, who were by side and support me in every decision I took in all those years as an academic student. Additionally, I would like to thank all the people I met and now I can feel them as very close friends.

Ioannis Giannakou

DEVELOPMENT OF A MATHEMATICAL MODEL AND A SOURCE CODE FOR THE SINGLE – DEPOT CAPACITATED VEHICLE ROUTING PROBLEM WITH HETEROGENEOUS FLEET

IOANNIS GIANNAKOU

University of Thessaly, Department of Mechanical Engineering, 2020

Supervisor: Dr. Georgios K.D. Saharidis, Assistant Professor in Operations Research Methods in Industrial Management

Summary

My thesis statement is about Vehicle Routing Problem and more specifically, is about the Capacitated Vehicle Routing Problem with Heterogeneous Fleet. This is one the most important problems of Operation Research in a logistics procedure.

The main goal of the project is to reduce the travelling distance from the depot to the customers' heterogeneous fleet by comparing to different cases for the solution.

In order to achieve that goal, different algorithms created for its case. In the first case a source code in C++ was created to find the routes via the nearest customer model. For the second case I created an algorithm which I simulated in C++ programming language by using the CPLEX Optimization Studio. The algorithm which created has a suitable objective function and constraints for the given data about customers and vehicles which I receive from EXCEL. In addition, two more source code created in C++ to help us to be as close to optimal solution as possible. The end product of the algorithm is to calculate the route of the given trucks in order to fulfil customers demand.

My thesis statement consisting of five chapters. In the first chapter an introduction of the Vehicle Routing Problem is given describing the definition of the problem, its variations and real-life problems. In the second chapter a definition of the case study is given, by explaining all the elements and the specific features that will be studied. In the third chapter a presentation of the all the models that are used is in this thesis are presented. In the fourth chapter is the application of the two cases in a real-life instance, collecting data which will be useful for the conclusion that will be made in the fifth chapter.

Contents

List of Figures

List of Tables

List of Equations

Acronyms

CVRP	Capacitated Vehicle Routing Problem
DVRP	Dynamic Vehicle Routing Problem
HFVRP	Vehicle Routing Problem with Heterogeneous Fleet
LPR	Location Routing Problem
m – TSP	Multiple Traveling Salesmen Problem
MDVRP	Multi – Depot Vehicle Routing Problem
MILP	Mixed Integer Programming
MSOM	Manufacturing & Service Operations Management
NN	Nearest Neighbour
OR	Operations Research
PVRP	Periodic Vehicle Routing Problem
SDVRP	Vehicle Routing Problem with Split Delivery
TS	Transportation Science
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem
VRPCS	Vehicle Routing Problem with Crew Scheduling

VRPPD	Vehicle Routing Problem with Pick – up and Delivery
VRPSD	Vehicle Routing Problem with Stochastic Demand
VRPSC	Vehicle Routing Problem with Stochastic Customers
VRPTW	Vehicle Routing Problem with Time Windows

Chapter 1 Operation research, Travelling salesman problem and Vehicle routing problem

1.1 Operational Research

Definition of the Operational Research

Operational Research is the application of scientific methods to complex problems that arise in the management of large systems consisting of individuals, machines, materials and funds in industry, business and the military. The approach is to develop a scientific model for the system, which includes measurements for factors such as luck and risk, which will predict and compare the results of various decisions and strategies. The main target is to assist the management to take decisions and shape its policies scientifically.

History of the Operational Research

Officially, Operational Research began shortly before and during World War II. At that time, the first operational research teams were created by British and American forces, consisting of scientists from various fields, in order to address and solve the tactical and strategic problems that arose during the war. The term "Operational Research" was created during this period, as it essentially meant Research in Military Operations.

Specifically, the reference point is the groups that dealt with the more efficient use of a new radar (then "aircraft early detection system"), where not only the system itself, its functions and the correct use of equipment were studied, but also studied and the behavior of the human resources that handle it. The specific studies are considered to have played a decisive role in the outcome of the Battle of Britain (1940).

This successful introduction of the scientific approach to military matters has allowed it to open up to other areas. The general industrial development that followed World War II created new needs, as the complexity of the problems that arose in various organizations, companies and even governments, made it necessary to find new ways to deal with them. The scientists realized that the problems successfully dealt with during the war were similar, simply in a different context. This has resulted in the creation of scientific communities and centers for

Operational Research, academic programs, etc. In fact, many tools of Operational Research that are widely used today, were sufficiently developed at that time.

Operational Research continued to grow in the following decades and with the advent of computers this development was rapid. The reasons for this rapid growth are obvious. Operational Research requires a lot of computation, something that computers can do quickly. From 1980 onwards, a corresponding software was created, a fact that, in combination with the spread of home computers, allowed everyone to have access to this science. Today, software that helps solve operational research problems exists on every computer, even if we do not know it or do not use it.

1.2 The Traveling Salesman Problem

Definition of the Problem

The Traveling Salesman Problem (TSP), which was first formulated in 1930, is considered as the origin of the classical Vehicle Routing Problem (VRP) which is one of the most studied combinatorial optimization problems. The Traveling Salesman Problem (TSP) is widely studied in Computer Science and Linear Programming. It is stated as, given a complete graph, G, with a set of vertices, V, a set of edges, E, and a cost, C_{ij}, associated with each edge in E. The value C_{ij} is the cost incurred when traversing from vertex $i \in V$ to vertex $j \in V$. Given this information, a solution to the TSP must return the Hamiltonian cycle of G with the minimum cost. A Hamiltonian cycle is a cycle that visits each node in a graph exactly once. This is referred to as a tour in TSP terms.

The essence of the traveling salesman problem is evident within many practical applications in real life. From a mail delivery person trying to figure out the most optimal route that will cover all of his/her daily stops, to a network architect trying to design the most efficient ring topology that will connect hundreds of computers. In all of these instances, the cost or distance between each location, whether it is a city, building or node in a network, is known. With this information, the fundamental goal is to find the optimal tour, which is to determine an order in which each location should be visited exactly once, and the total distance traveled, or cost incurred, is minimum. In the general TSP, there are no restrictions on the distance/cost values.

History of the Problem

The origins of TSP range back to the 1800's, when the Irish Mathematician Sir William Rowan Hamilton and the British mathematician Thomas Penyngton Kirkman treated the first mathematical problems related to it. However, the general form of the TSP appears to be first studied in the 1920's, when the mathematician Karl Menger brought it to the attention of his colleagues in Vienna. During the 1930's, the mathematical community of Princeton dealt with the problem, and in the 1940's, mathematician Merrill Meeks Flood, publicized the name, TSP, within the mathematical community. It was the year 1948 that Flood publicized the traveling salesman problem by presenting it at the RAND Corporation, which is a non-profit organization that is the focus of intellectual research and development within the United States.

The TSP soon became very popular due to its connection with the rising combinatorial problems of Linear Programming and its application in many tasks within people's daily lives. In 1950's Dantzig, Fulkerson, and Johnson presented a method for solving the TSP. They showed the effectiveness of their method by solving a 49-city instance.

Variations

Restrictions are set in the general TSP, either to make it easier to solve, or simply because such restrictions allow the problem to reflect certain and more realistic applications. The most popular variations of TSP are listed below.

- The Symmetric TSP: In this variation, all the edges have symmetric costs. This means that, for all nodes in the graph, the cost incurred, when traveling from node a to node b, is the same as cost incurred when traveling backwards (from node b to node a). On the other hand, the asymmetric TSP does not have such constraints. The general TSP is considered asymmetric. An input to the asymmetric TSP would be a directed graph.
- The Metric TSP: In this variation, all of the edge costs are symmetric and also satisfy the triangle inequality. The triangle inequality property means that for any three nodes a, b and c, the cost of going from node a directly to node c is always cheaper than going from node a to node c by passing through node b. In addition, the nodes are points in some space and the edge costs are determined by calculating the metric distance between them.
- The Euclidean TSP: In this variation, all the nodes lie in the plane, which means it is symmetric and the triangle inequality is applied. The cost of each edge e,

connecting nodes a and b, is defined by the Euclidean distance between the nodes a and b. In general, the plane can be d-dimensional, where d > 1.

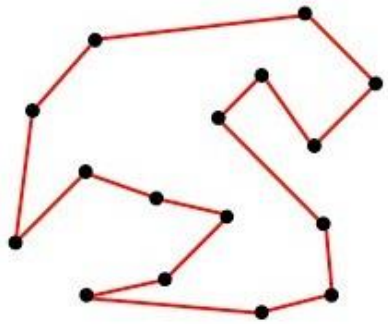

Figure 1 Solve of travelling salesman problem

Real Life Applications

There are many practical real-life uses of the TSP. The most common of which are transportation routing problems.

The most popular application consists of finding a route that a salesman would follow in order to visit every geographical location in a specified list such that minimum total distance is traveled. Considering the case of a salesman traveling from door to door in a certain sub – division of houses, it would be very convenient if the salesperson could obtain a list of all the houses in that sub – division specified in the most optimal order to visit. Furthermore, considering the case of a salesperson that needs to visit hundreds of cities spread throughout an entire country, by knowing the optimal tour that will visit each city, days or even weeks of traveling time could potentially be saved. Moreover, considering a postal delivery person who goes to work in the morning with a truck full of parcels to deliver, in what order should those parcels be delivered in order to minimize the total distance traveled? For all these instances, the nodes in the graph would correspond to the geographical locations, and the distances would be metric values based on the lengths of the roads connecting the locations.

The application which will be analyzed is the multiple salesmen model (m – TSP). The task of this application is to visit a set of cities, where each city has to be visited exactly once by any of the m salesmen. For each salesman j, who is being hires a fixed cost of d_j – the salary – should be paid. Each of the m salesmen must complete a sub tour, and the combination of all the sub tours that each of the m salesmen embark on must result in each city in set being visited exactly once. The salesmen start and end their sub tours at the same home base location. The

object it to determine how many salesmen to hire and what the sub tours should be done in order to minimize the total distance traveled and the cost of hiring the salesmen. This problem can be modeled as the original TSP by adding (m – 1) additional vertices, denoted –1, …, – (m – 1) to the input graph. Now, the m salesmen, numbered 0 to (m – 1), are represented by the source vertex and the new (m – 1) vertices. Then, edges are added to connect these (m – 1) new vertices to the rest of the vertices in the original graph. The costs of the newly added edges are determined by adding the costs d_j, for $0 \leq j \leq (m - 1)$, to the cost of existing edges in the graph.

1.3 The Vehicle Routing Problem

Definition of the Problem

The classical Vehicle Routing Problem (VRP) generalizes the Traveling Salesman problem and is one of the most popular problems in Combinatorial Optimization. It is defined on a complete undirected graph G = (V, E), where V = {0, …, n} is a vertex set and E = {(i, j): i, j ∈ V, i < j} is a set of edges. Each vertex $i \in V \setminus \{0\}$ represents a customer having a nonnegative demand q_i, while vertex 0 corresponds to the depot. Each edge $e \in E$ is associated with a travel cost c_{ij}. A fixed fleet of m identical vehicles, each of capacity Q, is available at the depot. Problem's objective is the determination of a set of at most m vehicle routes whose total travel cost is minimized such that:

- each customer is visited exactly once by one route,
- each route starts and ends at the depot,
- the total demand of the customers served by a route does not exceed the vehicle capacity Q, and
- the length of each route does not exceed a preset limit L.

History of the problem

The Capacitated VRP was formally introduced in 1959 by Dantzig and Ramser. They proposed a simple matching – based heuristic for its solution and illustrated it on a toy – sized example. The following years saw the emergence of several heuristics based on a variety of principles including savings, geographical proximity, customer matchings, as well as intra – route and inter – route improvement steps. Perhaps the most famous heuristic of this category is the Clarke and Wright (1964) savings heuristic, which has resisted the test of time because of its speed, simplicity and reasonably good accuracy.

The development of exact algorithms for the VRP took off in 1981 with the publication of two papers by Christofides, Mingozzi and Toth in Networks. The first one proposed an algorithm based on dynamic programming with state – space relaxation whereas the second one proposed two mathematical formulations making use of q – paths and k – shortest spanning trees. A few years later, Laporte, Desrochers and Nobert proposed the first cutting plane approach for a VRP based on the solution of linear relaxation of an integer model. These seminal concepts have made their way into some of the more recent algorithms.

Since then, a variety of exact algorithms based on mathematical programming formulations have been proposed. Some formulations contain vehicle flow or commodity flow variables and are often solved by branch – and – cut method. The VRP can also be formulated as a set partitioning problem to which some valid inequalities are added. Some of the most successful implementations by Fukasawa (2006) and by Baldacci (2008) are based on this methodology.

The development of modern heuristics for the VRP really started in the 1990s with the advent of metaheuristics. It is fair to say that the study of the VRP has stimulated the growth and understanding of several metaheuristic concepts which are now known. The early research in this area was quite fragmented, with a notable bias towards tabu search-based approaches and some of the algorithms were over engineered, but some rationalization has started to take place in recent years. The best metaheuristics are those that simultaneously perform a wide and deep search of the solution space and can solve several variants of the problem. They generally either apply several operators, as in adaptive large neighborhood search (Pisinger and Ropke 2007), or combine genetic search with local search, as in the hybrid genetic algorithm recently proposed by Vidal (2012).

Variations

As it is mentioned, the VRP arises in several forms because of the variety of constraints encountered in practice. The basic variation of VRP is the Capacitated VRP (CVRP). However, the real – life routing problems usually include much more complications which are not considered by the basic CVRP. Most of the complications are related to the following aspects:

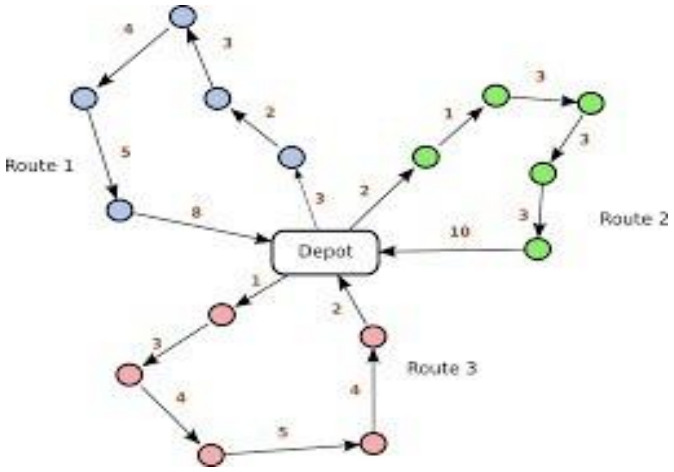

Figure 2 Vehicle Routing Problem

- Planning horizon: In real life, routes are planed for a given planning horizon. This planning horizon can consist of multiple periods.
- Customer: In the basic VRP, each customer has a demand. In more complicated problems, the customers may have requirements on the service time and/ or the vehicle type. There could also be different types of services, e.g., pickup service, delivery service or pickup-and-delivery service. In addition, in some cases, customers are allowed to be visited multiple times by several vehicles instead of just once by one vehicle. Moreover, in some applications with multiple planning days, the customers have demands every day and they can store products for the following days if they have received more than they can consume. In this case, the distributor needs to make a routing plan according to the demands and inventories of the customers.
- Depot: There can be more than one depot in a large distribution network, which may serve different purposes, such as warehousing or crossdocking, to reduce the total cost in the supply chain.
- Vehicle: The vehicles used for distribution can have different capacities and sizes. There are usually a limited number of vehicles available in real – life planning. A vehicle may be used in multiple trips instead of a single trip in a routing plan. In the problem with multi depots, each vehicle may be associated to a base depot. The vehicle must start from and end at its base depot.

- Driver: In most of the real-life problems, distributors need to consider the drivers' working regulations, e.g., the working shift and the break rules.
- Objective: The objective function can be quite complex in practice. It may include the minimization of the total travel cost, the minimization of the difference between the longest and shortest route to balance the workload among drivers, the minimization of the number of vehicles to save the large overhead, and/ or the maximization of the number of served customers to improve the service level.
- Uncertainty: There can be uncertainties in the route planning. For example, the locations and/ or the demands of customers are unknown at the beginning but revealed over time when the vehicles have already been sent out to carry out tasks. In some occasions, the probability distribution of these uncertainties is available, whereas in other cases, it is not.
- Good packing: In some applications, customer demand is formed by a set of two – dimensional or three – dimensional weighted items. A feasible routing implies a feasible packing in the sense of geometrical layout.

These complications and even more real – life restrictions lead to different extensions of the CVRP. The variations of CVRP are:

- Multi – Depot VRP (MDVRP): When there is more than one depot from which the customers can be served, the problem is considered as MDVRP. The MDVRP requires the assignment of customers to depots. A fleet of vehicles is based at each depot. Each vehicle originates from one depot, services the customers assigned to that depot, and returns to the same depot. The objective is to minimize the total travel cost and the number of vehicles used in order to service all customers.

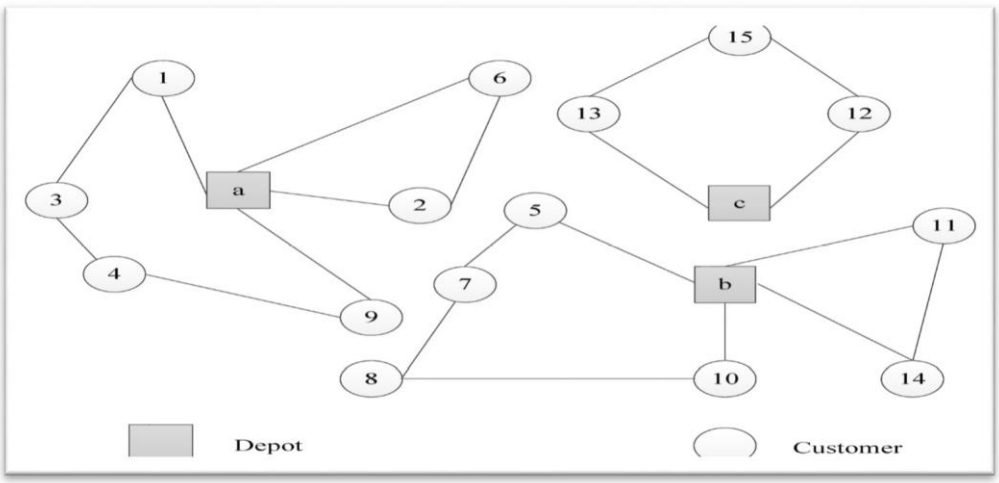

Figure 3 Multi-Depot Vehicle Routing Problem

- VRP with Heterogeneous Fleet (HFVRP): When the available fleet of vehicles for distribution activities characterized by different capacities, types of cargo and/ or costs, the problem is considered as HFVRP. The objective is to minimize the total travel cost and the total cost of vehicles used.

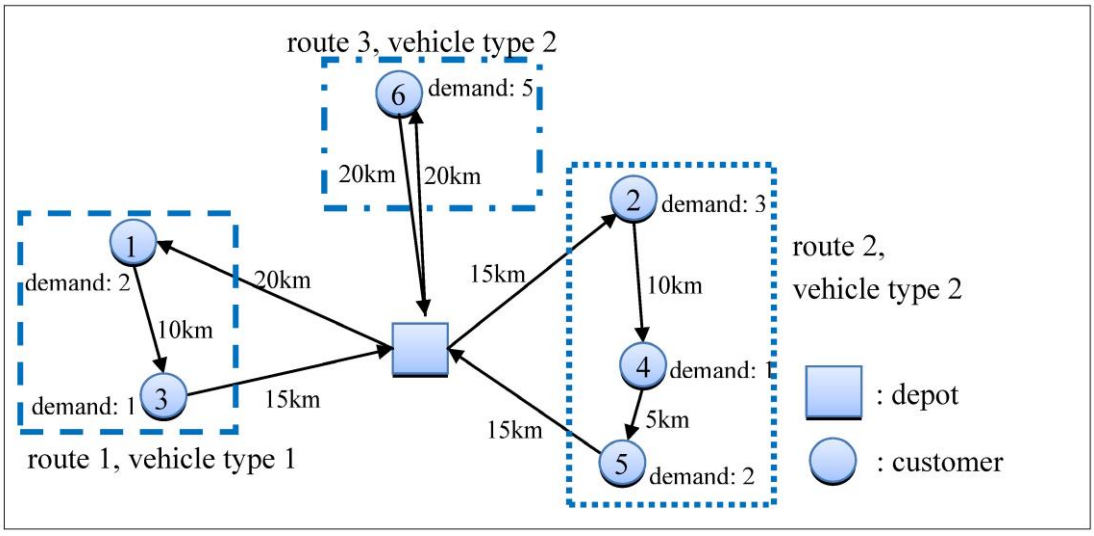

Figure 4 Heterogeneous Fleet Vehicle Routing Problem

- VRP with Time Windows (VRPTW): In VRPTW every customer is characterized by a specific time window [a_i, b_i] within it should be serviced. A vehicle is allowed to arrive before a_i and wait until the customer becomes available. However, arrivals

after b_i are prohibited. The objective is to minimize firstly the number of vehicles used and then the total distance traveled.

- VRP with Split Delivery (SDVRP): In SDVRP each customer can be served by more than one vehicle. This relaxation of the VRP facilitates the service of the customers especially when their size of orders is as big as the capacity of the vehicle. The objective is to minimize the vehicle fleet and the total distance travelled.
- VRP with Pick – up and Delivery (VRPPD): In VRPPD is considered the return of goods to the delivery vehicle. Hence, they have to fit into it. The objective is to minimize the vehicle fleet and the total distance travelled, with the restriction that the vehicle must have enough capacity for transporting the commodities to be delivered and those ones picked – up at customers for returning them to the depot.
- Periodic VRP (PVRP): In PVRP, there is a horizon of M days and a frequency for each customer stating how often within the M – day period each customer should be visited. The objective is to minimize the total cost of all routes over the planning horizon.
- Dynamic VRP (DVRP): DVRP is the extension of the classical VRP in which the uncertainties of real life are taken into consideration. The objective is to minimize the total travel cost subject to constraints related to real life restrictions.
- VRP with Crew Scheduling (VRPCS): VRPCS is the combination of the vehicle routing and the crew scheduling problem. Although it is easier to study these problems separately, due to the dependence between them the combined problem may yield better schedules of the costly manpower and may reduce the total cost significantly.
- VRP with Stochastic Customers (VRPSC): In VRPSC, each customer is present with probability p and absent with probability (1 – p). Two stages are made in order to get a solution. Firstly, a solution is determined before knowing the actual number of customers. In a second stage, a recourse or corrective action can be taken when the customers are determined, in order to define the optimal solution. The objective is to minimize the vehicles used and the total travel cost.
- VRP with Stochastic Demand (VRPSD): When the demand of each customer is a random variable, the problem is considered as VRPSD. As in VRPSC, the optimal solution is determined in two stages and the objective is to minimize the vehicle used and the total travel cost.

- VRP with Two/ Three-Dimensional Loading (2L – VRP/ 3L – VRP): 2L – VRP/ 3L – VRP is the combination of the vehicle routing and the vehicles' loading problem. The objective is to find a partition of the customers into subsets, which are no more than the maximum number of available vehicles and, for each subset, a route starting and ending at the depot such that the total travel cost is minimized.

Real Life Applications

The VRP is rooted in a wide variety of real-life applications worldwide. The key sectors of real life VRP applications are:

- food distribution,
- stock management,
- retail,
- waste collection and management,
- mail and small package delivery, and
- oil, gas and fuel.

Food distribution

Food distribution has its own characteristics, constraints and challenges such as product quality, health and safety. The products often have a limited shelf – life, so that distribution operations must consider temperature, humidity and time in transit considerations, as well as many other constraints related to products. Their use e.g., is a distribution of dairy products in markets, grocery, stores etc. The distribution of soft drinks and alcoholic beverages as well as the transport of edible food etc.

Stock management

In the case of inventory management, we are dealing with customers making inventory information available to distributors who then take responsibility for the decision to serve and which customers. Therefore, the supplier has to choose how often when and in what quantities the customers will be served. This integrated inventory and distribution management offers greater flexibility and ease in designing more efficient corridors for vehicles, while optimizing storage throughout the supply chain.

Retail applications

Retail involves the sales of goods and the provision of services to end – users. In this section we list applications dealing with several final products, and in various sectors such as

supermarkets and consultancy services. These applications generally involve time windows and loading constraints.

Waste collection and management

Trash or waste is defined as residues and discarder items when they have ceased to serve their original purpose for which they were constructed. For this reason, optimizing the routes of garbage trucks is a very important factor for more efficient and economical waste collection. Therefore, the problem of vehicle routing (VRP) on waste / waste collection is very important. As the model creates the appropriate routes which having the lowest possible cost with the shortest possible distances to be travelled considering the density of the population, the capacity of the trucks, the road network and the garbage bins.

Mail and small delivery package

Post and parcel delivery are very important industries. In this section, reviews of real – life applications range from mail delivery to Internet order delivery, touching on many variations of classic VRP, such as time windows and pick-up and deliveries.

Oil, gas and fuel applications

Concerning problems which describe applications related to the supply of oil, gas and fuel to houses, gas stations and companies, they present a number of specific features such as vehicles with capacitated compartments and sometimes the presence of flow meters to control the delivered quantity. The latter feature implies that sometimes the content of the same compartment can be used to satisfy the demands of several customers, whereas when there is no flow meter, the compartment must be completely emptied in a single customer tank. Cleaning operations may be needed between the loading of different products using the same compartment. These problems are generally solved over long – term planning horizons and incorporate mixed inventory and routing decisions.

Chapter 2 Definition of the problem

The problem studied and presented in this dissertation is a well-known variant of the vehicle routing problem. It is the vehicle routing problem that designs and creates the optimal routes to be used by a fleet of vehicles to serve a set of customers by limiting vehicle capacity. The fleet of vehicles is based on a central warehouse and is characterized by different capacities but also different costs.

In this generalization there is no limitation of the time period (time windows), within which the customer service must be achieved. So, a heterogeneous fleet of vehicles with finite capacity, itineraries starting and ending at a specific node – station (depot).

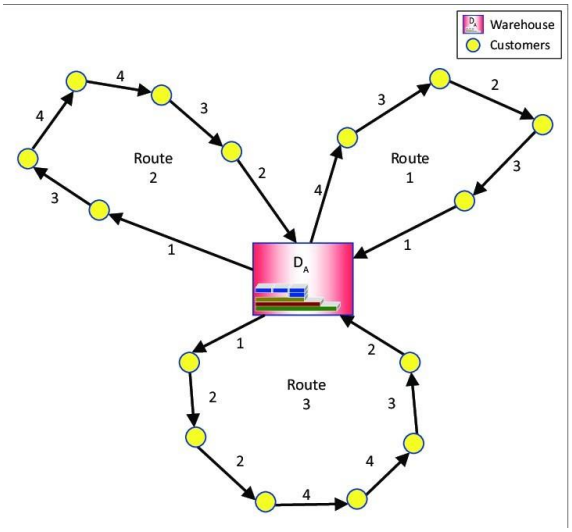

Figure 5 The capacitated Vehicle Routing Problem

The main purpose of this study is to adapt some constraints to the basic model of capacitated vehicle routing problem and compared the solution with a model of the nearest customer solution. The constraints that will be incorporated to the basic model concern the demands of the customers and the utilization of the vehicles. More specifically regarding the constraints on customer demand, the constraint that does not allow the capacity of the vehicles to be exceeded is changed and two more constraints that ensure that each customer will receive exactly his demand will be added. In case of the availability of the trucks one constraint will be included that does not allow the use of trucks that is not available in the warehouse.

With the additions and the changes that will take place we expect the model to produce a solution that will be feasible and as close as possible to the optimal solution. Also, the output data will be more useful and practical.

A more detailed description of the problem is the following. A set of customers - nodes which is 192 and the node-station (depot), the distances between them and the distances between each customer – node and the warehouse, as well as the demand of each customer - node in weight are all known.

In addition, the fleet of vehicles is given in this dissertation and consists of a total fleet which is 5 trucks as well as the capacities of the trucks in kilograms to be known.

More specifically, because of the complexity about the big number of costumers it should be a demarcation between them. The solution of the problem will be studied in two different cases in order to compare the results between them. The first case will be an algorithm which is based on the nearest customer solution and an insertion algorithm of customers to the vehicles in order to have solution for all the customers. The second case will be a mixed solution using a separation between the customers, the capacitated vehicle routing problem and an insertion algorithm.

The first case is described below.

From all the 192 customers we will use an algorithm in order to find a route that serves all customers with no vehicles at the beginning. This route will be extracted with a source code based on the solution of nearest customer. In the second stage and after we have found the route for all the customers, we will assign the customers by order to the first available vehicle until it fulfil percentage is close to 100%. Then we will continue with the second, third, fourth and fifth vehicle by order until we have served all the customers. The final solution will be the total travel distance of each vehicle.

Now for the second case. In the first stage from all the customers we will extract the 64 first customers which have the ability of the shortest distance between them and the depot. In the second stage we will extract a solution based on the capacitated vehicle routing problem. From this solution the vehicles which are assigned to a route and do not have a percentage of fulfil of capacity above 95% will be used to an insertion algorithm in order to achieve the maximum available capacity. The insertion of customers to those vehicles will be done with the first criterion to be the distance of the nearest customers between those who have already assigned to the route and the remaining 128 customers that does not selected in the first stage. With the

completion of the above stage, for the new routes for each vehicle we will run a travelling salesman problem formulation in order to get the optimal routes. In the final stage the remaining customers that did not assigned to any of the new routes and the remaining vehicles which are also do not assigned to any route will be routed with the same procedure as in the beginning. In the end we will take the solutions from the above problem and we will add them in order to get the final solution for the 192 customers.

From the solutions of this problem, we should expect to emerge the optimal assignment of nodes - customers to a vehicle route, in order of every node will be served and the optimal routing of the trucks to minimize the total distance between the nodes.

Therefore, is noticed that the decision making of capacitated vehicle routing problem with heterogeneous fleet is consist of two solutions. Firstly, the assignment of node to a vehicle route. Secondly, the demand of its customer will be the one that he will receive. Those results are logical because this thesis is about these alternative solutions of the capacitated vehicle routing problem.

Chapter 3 Models presentation

3.1 Basic characteristics

After identifying the type of problem that was assigned to me as well as its deconstruction, it is time to collect the data and the hypotheses that will be used. They will then be analyzed below through the limitations of the mathematical model. Indicatively, I will mention some constraints that were included in the problem in order to solve it.

1. All customers must be served,
2. Every customer must be visited exactly once,
3. All trucks must start and return at the end of the route to depot,
4. The capacity of the trucks by weight will not be violated.

Also, the specific data of the problem can be divided into two categories:

A. The customers,

B. The Trucks.

Customers Data

1) The demand of offered product of each customer by weight,
2) The distances between the customers.

Truck Data

1. Capacity of each truck by weight.

3.2 Proposed model of single depot capacitated vehicle routing problem with heterogeneous fleet

The proposed model that will be presented consist the basic formulation of a V.R.P. problem plus some constraints that ensure that every customer will receive exactly his demand of products and some differences between the constraints that ensure that the vehicle availability will exceed.

Prior presenting the equations of the model, the parameters of indexes, data and sets and the decision variables are defined.

Indexes

i, j, z: the nodes of network

k: the type of vehicles

Data

N: number of customers

K: number of vehicles

QW_K: capacity of vehicle k in kg

D_{ij}: the distance in m between nodes i and j. Therefore, D_{ij} represents the distances from customer to customer and from depot to customer.

W_i: weight of demand of customer i in kg

Before we proceed with the presentation of sets and decision variable, a fundamental assumption must be made. The depot is the starting point of all vehicles and the weight of demand is 0.

Sets

[0,…,N]: the set of nodes

[1,…,N]: the set of customers

[1,…,K]: the set of vehicle types

Decision Variables

X_{ijk}: binary variable. If $X_{ijk} = 1$, then a vehicle of type k travelling along arc (i, j), otherwise $X_{ijk} = 0$. Where k = 1,…,K, i = j = 1,…,N.

W_{ij}: non-negative continuous variable, which denotes the weight of load defined in terms of kg remaining in the vehicle before reaching node j while traveling along arc (i, j).

o_i: it is an auxiliary variable whose purpose is in solving the well-known sub-routing problem that occurs in a V.R.P.

Mathematical Formulation

A mathematical model consists of the objective function and the constraints, which can be equalities and/ or inequalities.

The objective functions and constraints of proposed mathematical model are presented in this subsection.

Objective function:

$$\text{Minimize } Z = \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} D_{ij} X_{ijk} \qquad (3.2.1)$$

Subject to

$$\sum_{i=1}^{m}\sum_{k=1}^{K} x_{ijk} = d_j \quad , \forall j = 1, \ldots, n (j \neq 1) \tag{3.2.2}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{K} x_{ijk} = s_i \quad , \forall i = 1, \ldots, m (i \neq 1) \tag{3.2.3}$$

$$\sum_{i=1}^{m} x_{ijk} = v_k - r_k \, , \forall k \tag{3.2.4}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} - M * \sum_{i=1}^{m} x_{ijk} \leq 0 \, , \forall k \tag{3.2.5}$$

$$\sum_{i=1}^{m} x_{ijk} = \sum_{i=1}^{m} x_{ijk} \quad , \forall j, \; k = 1, \ldots, K (k \neq 1) \tag{3.2.6}$$

$$u_i - u_j + M \sum_{k=1}^{K} x_{ijk} \leq M - 1 \, , 2 \leq i \neq j \leq n - 1 \tag{3.2.7}$$

$$\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m} x_{ji} \tag{3.2.8}$$

$$x_{ij} \leq \sum_{k=1}^{K} x_{ijk} x_{ijk} \quad , \forall i, j = 1, \ldots, n (i \neq j) \tag{3.2.9}$$

$$\sum_{i=1}^{m} x_{ij} - \sum_{i=1}^{m} x_{ji} = 0 \quad , \forall j = 1, \ldots, n \, (i \neq j \neq 1) \tag{3.2.10}$$

$$x_{ijk} \; x_{ijk} \tag{3.2.11}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} \leq 1$$

$$x_{ijk} = 0\text{-}1 \ , \forall i, j, k (i \neq j) \tag{3.2.12}$$

$$?_{??} \geq 0 \quad , \forall ?, ?(? \neq ?) \tag{3.2.13}$$

$$?_{?} \geq ? \text{ , int} \tag{3.2.14}$$

Explanation of objective function and constraints of initial model

The objective function (3.2.1) represents the total distribution cost, which consists of the running cost $D_{ij,}$ which is the travelling distances from node I to node j.

Constraint (3.2.2) show that each customer is visited exactly once. Specifically, the transition from any node of the network – customer or depot – to a customer will be accomplished exactly once by a vehicle k.

Constraint (3.2.3) guarantee that from each customer exactly one departure is accomplished. Specifically, from every customer a transition to another customer or depot of the network by a vehicle type k will happen.

Constraints (3.2.4) show that each truck must leave from the depot.

Constraint (3.2.5) guarantee that each truck that leaves depot must return to it. Especially when a vehicle is assigned to a route this vehicle must return to depot for the restriction to apply.

Constraint (3.2.6) ensure the flow conservation. The constraints ensure that when a vehicle type k leaves from depot and enters a location, it must leave from this location. This constraint is a pillar of modeling a V.R.P problem.

Constraint (3.2.7) is also a very important restriction for V.R.P. It ensures that the trucks will not create sub-routes but will follow the order of customers service of the routes assigned to them (sub-tour elimination).

Constraint (3.2.8) show that the weight of total load which leaves from all depots is exactly the total weight of customers' demand.

Constraint (3.2.9) guarantee that the capacity defined in terms of kg of any vehicle type is not violated.

Constraint (3.2.10) guarantee that the weight of the remaining load after visiting each customer j, where j = 1…., n, is exactly the weight of the remaining load before visiting this customer minus the weight of its demand.

Constraint (3.2.11) shows that the number of trucks offered should not be violated. It is essentially ensured that the modeling of the system will not require more trucks than are available, in order to possible optimize the solution. It is common for those problems, especially whole programming, to have such constraints, so as not to let the modeling act arbitrarily.

Constraint (3.2.12) ensures that the decision variable X_{ijk} will be binary, (3.2.13) that the decision variable W_{ij} will be positive and lastly (3.2.14) guarantee that the auxiliary variable o_i will be a positive integer variable.

3.3 Nearest customers to depot algorithm

In order to find the nearest customers to the depot we had to produce a source code in C++. As we set as the input of problem the table of distance between all nodes, we start make specific repetitions in order to create an array with the distance from the depot to the customers. In addition, another array is created with the position of customers. As we have searched all the table of distances and have find and create the array with all the important data, we can go the second part. In the second stage with some command lines, we recreate the two arrays having set in ascending order the array with the distances and the array with the position of the customer accordingly.

In the final stage the output of the problem is two, which the first contains the position of customers that have been in order by the distance from the depot and the second table contains the distance between the depot and the customer.

3.4 Insertion algorithm

Firstly, we add the data we have received from the solution of vehicle routing problem, more specifically the route of each truck that was assigned and the remaining capacity of each truck. Secondly, starting with the first truck and taking the first customer served by ascending order, it starts scanning the entire distance table in order to find the two closest customers from that specific customer. Once found, its positions are kept in an array. To avoid choosing a customer that has already been selected or already exists on the route of this particular vehicle or another vehicle route, logical variables are used with conditions that prevent this from happening. This procedure is repeated until we have found the nearest two customers for each customer that is served by this vehicle. The same pattern is followed for all the vehicles routes.

In the next step and after we have found the nearest two customers for each route, we use two arrays in order in the first one to keep the distance from the depot to the customer and in the second the position of the specific customer. This procedure is repeated until the array is foul with distances for all the customer for each route. Now these arrays with the use of repetitions and a specific command line the new array that is formed has the distance from the depot to the customer by ascending order and in the same lines in the other array is the position of the customer.

In the final step, the load of the vehicles is starting to take place. Now having the array with the customers that are the nearest from the route in ascending order we begin to insert them to the corresponding vehicle, until the vehicle cannot include another customer.

The final solution of this problem gives us a table which contains the route of the vehicle which now consists of the customers that was initial in and the customers that were added by this procedure. Additionally, we create two arrays in order to have the total load that has been added to each truck.

3.5 Nearest customer path algorithm

In the beginning, having the warehouse as a reference point (i = j = 0), it starts by scanning the table of distances in order to find the nearest customer from the warehouse. After scanning the whole table and finding the nearest customer, he keeps his position in one array and the distance from the previous customer in another array. This process is repeated until all customers are routed to the shortest path, for the reason not to serve the same customer multiple times we are using a logical variable that prevents scanning already placed customers. Also, when it reaches the last customer, the return to the depot is added.

After finding the route for customer satisfaction and knowing the capacities of the trucks, it starts placing customers on the trucks using appropriate iterations and conditions. In case a customer cannot be added to a vehicle then the customer is added to the next available truck. This procedure continues until all the customers have been routed via a vehicle.

The final solution of the problem contains the information of the route of each vehicle, the route distance of each vehicle and the remaining weight of the truck before it serves the next customer.

Chapter 4 Application of the single depot capacitated vehicle routing problem with heterogeneous fleet

4.1 A real life problem

The proposed mathematical model was applied in a real-life instance of a company that provides the island of Mykonos with products. The company has one depot which is located in the City of the island. The instance concerns a one-day routine of the company's schedule of delivering the products to its customer. All the customers are located in the island, some of them in the City and the other ones scattered across the island. It is necessary to mention again that depot (node 0) has zero demand. The total demand od all customers is 12442 kg. The following table shows the demand of each customer and depot.

Customer	Demand (kg)	Customer	Demand (kg)	Customer	Demand (kg)
0	0	**65**	45	**130**	35
1	35	**66**	40	**131**	45
2	45	**67**	50	**132**	40
3	40	**68**	122	**133**	50
4	50	**69**	144	**134**	122
5	122	**70**	50	**135**	144
6	144	**71**	122	**136**	35
7	80	**72**	144	**137**	45
8	35	**73**	35	**138**	40
9	45	**74**	45	**139**	50
10	40	**75**	40	**140**	122
11	80	**76**	50	**141**	144
12	100	**77**	122	**142**	50
13	35	**78**	144	**143**	122
14	45	**79**	80	**144**	144
15	40	**80**	35	**145**	35
16	80	**81**	45	**146**	45

17	122	**82**	40	**147**	40
18	35	**83**	80	**148**	50
19	45	**84**	100	**149**	122
20	40	**85**	35	**150**	144
21	150	**86**	45	**151**	80
22	50	**87**	40	**152**	35
23	35	**88**	80	**153**	45
24	45	**89**	122	**154**	40
25	40	**90**	35	**155**	80
26	35	**91**	45	**156**	100
27	45	**92**	40	**157**	35
28	40	**93**	150	**158**	45
29	120	**94**	50	**159**	40
30	35	**95**	35	**160**	80
31	45	**96**	45	**161**	122
32	40	**97**	40	**162**	35
33	35	**98**	35	**163**	45
34	45	**99**	45	**164**	40
35	40	**100**	40	**165**	150
36	35	**101**	120	**166**	50
37	45	**102**	35	**167**	35
38	40	**103**	45	**168**	45
39	35	**104**	40	**169**	40
40	45	**105**	35	**170**	35
41	40	**106**	45	**171**	45
42	120	**107**	40	**172**	40
43	35	**108**	35	**173**	120
44	45	**109**	45	**174**	35
45	40	**110**	40	**175**	45
46	50	**111**	35	**176**	40
47	122	**112**	45	**177**	35
48	144	**113**	40	**178**	45

49	35	**114**	120	**179**	40
50	45	**115**	35	**180**	35
51	40	**116**	45	**181**	45
52	50	**117**	40	**182**	40
53	122	**118**	50	**183**	35
54	144	**119**	122	**184**	45
55	50	**120**	144	**185**	40
56	122	**121**	35	**186**	120
57	144	**122**	45	**187**	35
58	35	**123**	40	**188**	45
59	45	**124**	50	**189**	40
60	40	**125**	122	**190**	50
61	50	**126**	144	**191**	122
62	122	**127**	50	**192**	144
63	144	**128**	122		
64	35	**129**	144		

Table 1 Customers Demands in kg

The fleet of vehicles of the company consists of 5 trucks with different capacity each of them. The categories of the truck are: Light Cargo, Medium Cargo and Heavy Cargo. All of the trucks can distribute the product to all customers with no exception and the starting point of them is at the depot. The characteristics of the fleet is shown in the table below.

Vehicles	Capacity in kg	Type
1	3700	Medium Cargo
2	2800	Light Cargo
3	4250	Heavy Cargo
4	2500	Light Cargo
5	3700	Medium Cargo

Table 2 Vehicle Capacity and Type

4.2 Simulation and Results for the 1st case

4.2.1Simulation

The mathematical model of nearest customer was simulated in C++ programming language. The sources are available in Annex.

The characteristics of the program and the software and hardware of the computer which were used for the simulation of models are.

Program edition

Visual Studio 2019

Windows edition

Windows 10 Home

System

Processor: Intel® Core™ i7-2670QM CPU @ 2.20GHz

Installed memory (RAM): 6.00 GB DDR3

System type: 64-bit Operating System x64

4.2.2Results of the proposed model of nearest customer algorithm

The proposed source code for the 192 customers gives us a final solution with value of 59426m. The routes that each vehicle was assigned will be given in the following tables.

Vehicle 1

The route that vehicle 1 was assigned is presented in the following table.

Vehicle	Route
1	0→12→61→55→17→70→78→44→2→122→50 →137→163→69→96→191→154→41→174→92 →155→48→113→184→90→178→54→63→153 →86→95→115→185→182→30→183→56→4 →11→71→102→169→114→66→64→75→3→189 →58→53→103→181→152→37→135→100→127

→65→170→35→0

Table 3 First Vehicle Route

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (Wij)	Customer Demand
0→12	3684	0
12→61	3584	100
61→55	3534	50
55→17	3484	50
17→70	3362	122
70→78	3312	50
78→44	3168	144
44→2	3123	45
2→122	3078	45
122→50	3033	45
50→137	2988	45
137→163	2943	45
163→69	2898	45
69→96	2754	144
96→191	2709	45
191→154	2587	122
154→41	2547	40
41→174	2507	40
174→92	2472	35
92→155	2432	40
155→48	2352	80
48→113	2208	144
113→184	2168	40
184→90	2123	45
90→178	2088	35
178→54	2043	45
54→63	1899	144
63→153	1755	144
153→86	1710	45
86→95	1665	45
95→115	1630	35
115→185	1595	35
185→182	1555	40
182→30	1515	40
30→183	1480	35
183→56	1445	35

56→4	1323	122
4→11	1273	50
11→71	1193	80
71→102	1071	122
102→169	1036	35
169→114	996	40
114→66	876	120
66→64	836	40
64→75	801	35
75→3	761	40
3→189	721	40
189→58	681	40
58→53	646	35
53→103	524	122
103→181	479	45
181→152	434	45
152→37	399	35
37→135	354	45
135→100	210	144
100→127	170	40
127→65	120	50
65→170	75	45
170→35	40	35
35→0	0	40

Table 4 Remaining weight in each transition of vehicle 1

According to the table the vehicle is loaded with 3684 kg and it serves a total of 59 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 2

The route that vehicle 2 was assigned is presented in the following table.

Vehicle	**Route**
2	0→160→34→25→9→131→142→26→81 →57→46→52→60→82→98→43→117→123 →192→79→59→116→143→15→19→14→99 →27→89→62→187→72→141→18→119→179 →94→32→162→180→10→148→107→38→110 →171→51→145→0

Table 5 Second Vehicle Route

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (Wij)	Customer Demand
0→160	2769	0
160→34	2689	80
34→25	2644	45
25→9	2604	40
9→131	2559	45
131→142	2514	45
142→26	2464	50
26→81	2429	35
81→57	2384	45
57→46	2240	144
46→52	2190	50
52→60	2140	50
60→82	2100	40
82→98	2060	40
98→43	2025	35
43→117	1990	35
117→123	1950	40
123→192	1910	40
192→79	1766	144
79→59	1686	80
59→116	1641	45
116→143	1596	45
143→15	1474	122
15→19	1434	40
19→14	1389	45
14→99	1344	45
99→27	1299	45
27→89	1254	45
89→62	1132	122
62→187	1010	122
187→72	975	35
72→141	831	144
141→18	687	144
18→119	652	35
119→179	530	122
179→94	490	40
94→32	440	50
32→162	400	40
162→180	365	35
180→10	330	35

10→148	290	40
148→107	240	50
107→38	200	40
38→110	160	40
110→171	120	40
171→51	75	45
51→145	35	40
145→0	0	35

Table 6 Remaining weight in each transition of vehicle 4

According to the table the vehicle is loaded with 2769 kg and it serves a total of 47 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 3

The route that vehicle 3 was assigned is presented in the following table.

Vehicle	**Route**
3	0→126→144→108→166→106→147→151 →111→49→173→177→172→168→1→21 →176→8→112→136→175→149→130→22 →85→101→29→190→165→164→87→84 →80→31→42→23→128→20→7→39→91 →188→73→167→28→150→124→68→156 →77→33→134→13→133→6→24→186→120 →157→5→36→0

Table 7 Third Vehicle Route

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	**Remaining Weight (Wij)**	**Customer Demand**
0→126	4232	0
126→144	4088	144
144→108	3944	144
108→166	3909	35
166→106	3859	50
106→147	3814	45
147→151	3774	40
151→111	3694	80
111→49	3659	35

49→173	3624	35
173→177	3504	120
177→172	3469	35
172→168	3429	40
168→1	3384	45
1→21	3349	35
21→176	3199	150
176→8	3159	40
8→112	3124	35
112→136	3079	45
136→175	3044	35
175→149	2999	45
149→130	2877	122
130→22	2842	35
22→85	2792	50
85→101	2757	35
101→29	2637	120
29→190	2517	120
190→165	2467	50
165→164	2317	150
164→87	2277	40
87→84	2237	40
84→80	2137	100
80→31	2102	35
31→42	2057	45
42→23	1937	120
23→128	1902	35
128→20	1780	122
20→7	1740	40
7→39	1660	80
39→91	1625	35
91→188	1580	45
188→73	1535	45
73→167	1500	35
167→28	1465	35
28→150	1425	40
150→124	1281	144
124→68	1231	50
68→156	1109	122
156→77	1009	100
77→33	887	122
33→134	852	35
134→13	730	122
13→133	695	35

133→6	645	50
6→24	501	144
24→186	456	45
186→120	336	120
120→157	192	144
157→5	157	35
5→36	35	122
36→0	0	35

Table 8 Remaining weight in each transition of vehicle 3

According to the table the vehicle is loaded with 4232 kg and it serves a total of 60 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 4

The route that vehicle 3 was assigned is presented in the following table.

Vehicle	**Route**
4	0→47→161→105→83→16→118→74→125 →40→132→138→109→129→67→88→97→45→121 →159→139→140→158→104→93→76→146→0

Table 9 Fourth Vehicle Route

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	**Remaining Weight (Wij)**	**Customer Demand**
0→47	1757	0
47→161	1635	122
161→105	1513	122
105→83	1478	35
83→16	1398	80
16→118	1318	80
118→74	1268	50
74→125	1223	45
125→40	1101	122
40→132	1056	45
132→138	1016	40
138→109	976	40
109→129	931	45

129→67	787	144
67→88	737	50
88→97	657	80
97→45	617	40
45→121	577	40
121→159	542	35
159→139	502	40
139→140	452	50
140→158	330	122
158→104	285	45
104→93	245	40
93→76	95	150
76→146	45	50
146→0	0	45

Table 10 Remaining weight in each transition of vehicle 4

According to the table the vehicle is loaded with 1757 kg and it serves a total of 26 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 5

The solution of the problem does not include the assignment of vehicle 5.

4.3 Simulation and results for the 2nd case

4.3.1 Simulation

The mathematical model of S.D.C.V.R.P.H.F was simulated in C++ programming language by using the CPLEX Optimization Studio. The sources are available in Annex.

The characteristics of the program and the software and hardware of the computer which were used for the simulation of models are

Program edition

IBM CPLEX Optimization 12.6

Windows edition

Windows 10 Home

System

Processor: Intel® Core™ i7-2670QM CPU @ 2.20GHz

Installed memory (RAM): 6.00 GB DDR3

System type: 64 bit Operating System x64

The optimization libraries used from CPLEX ILOG IBM in Microsoft Visual Studio 2010 C++.

The mathematical model of closest customer to depot and the insertion model were simulated in C++ programming language. The sources are available in Annex.

The characteristics of the program which was used for the simulation of models is

Program edition

Visual Studio 2019

The software and hardware of the computer is the same as it referred in the above characteristics.

4.3.2 Results of the proposed model of the single depot capacitated vehicle routing problem with heterogeneous fleet

The above real-life problem was tested for a service of 192 customers plus the depot which gives us a total of 193 nodes. All the distances of the network from node to node are the real distances and the capacities of the vehicles are as it stands in real life.

Due to the size and the complexity of the problem some restrictions were important to be made in order to achieve a solution. The main issue was the memory space. In order to achieve a feasible solution but not optimal the limitation that we had to take were a time limit of running the simulation. The time limit is before the simulation collapse due to the memory space limitation. In this period the sources were able to give us feasible solutions.

64 Customers Nearest to the Depot

The proposed mathematical model for the case of 64 customers which are those who have the minimum distance from the depot gives us this assignment of vehicles. The routes that each vehicle was assigned will be given in the following tables.

Vehicle 1

The route that vehicle 1 was assigned is presented in the following table.

Vehicle	Route
1	0→56→57→23→18→35→49→43→58→10→20→55→42 →25→59→38→41→60→37→50→13→64→53→33→7→ 19→22→46→52→32→11→48→45→26→9→54→30→62 →36→12→0

Table 11 First Vehicle Route

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (W_{ij})	Customer Demand
0→56	2618	0
56→57	2583	35
57→23	2538	45
23→18	2394	144
18→35	2250	144
35→49	2215	35
49→43	2170	45
43→58	2135	35
58→10	2095	40
10→20	2060	35
20→55	1938	122
55→42	1898	40
42→25	1778	120
25→59	1738	40
59→38	1703	35
38→41	1668	35
41→60	1628	40
60→37	1583	45
37→50	1543	40
50→13	1503	40
13→64	1463	40
64→53	1341	122
53→33	1296	45

33→7	1152	144
7→19	1030	122
19→22	980	50
22→46	930	50
46→52	808	122
52→32	708	100
32→11	586	122
11→48	551	35
48→45	407	144
45→26	357	50
26→9	235	122
9→54	195	40
54→30	160	35
30→62	125	35
62→36	80	45
36→12	35	45
12→0	0	35

Table 12 Remaining weight in each transition of vehicle 1

According to the table the vehicle 1 is loaded with 2618 kg and it serves a total of 39 customers before it returns to the depot. From the above data the vehicle capacity does not exceed.

Vehicle 2

The route that vehicle 2 was assigned is presented in the following table.

Vehicle	**Route**
2	0→28→16→14→47→44→27→39→15→61→5→24 →17→63→21→2→31→40→29→3→34→51→1→8 →4→6→0

Table 13 Second Vehicle Route

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	**Remaining Weight (W_{ij})**	**Customer Demand**
0→28	1592	0
28→16	1542	50
16→14	1497	45
14→47	1452	45

47→44	1407	45
44→27	1362	45
27→39	1218	144
39→15	1173	45
15→61	1029	144
61→5	989	40
5→24	909	80
24→17	874	35
17→63	752	122
63→21	712	40
21→2	677	35
2→31	637	40
31→40	597	40
40→29	562	35
29→3	440	122
3→34	390	50
34→51	345	45
51→1	265	80
1→8	220	45
8→4	180	40
4→6	100	80
6→0	0	100

Table 14 Remaining weight in each transition of vehicle 2

According to the table the vehicle 2 is loaded with 1592 kg and it serves a total of 25 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 3

The solution of the problem does not include the assignment of vehicle 3.

Vehicle 4

The solution of the problem does not include the assignment of vehicle 4.

Vehicle 5

The solution of the problem does not include the assignment of vehicle 5.

Due to the fact the total load of the vehicles is not optimal and the percentage of fulfil, which is shown on the table below, is not close to 100% we had to apply a customer import algorithm to the existing routes in order to reach the 100% capacity of each truck.

Vehicle	Vehicle Capacity	Route Capacity	Remaining Capacity	Percentage of Fulfil
1	3700	2618	1082	70,75%
2	2800	1592	1208	56,86%

Table 15 Percentage of fulfil by vehicle

64 Customers Nearest to the Depot with Insertion of customers and TSP application

The new routes of each truck with the insertion of customers are presented below.

Vehicle 1

The route that vehicle 1 was assigned is presented in the following table.

Vehicle	Route
1	0→9→44→19→41→35→4→42→53→1→34 →2→7→21→32→37→5→20→54→24→13→3→22→55→1 0→47→25→11→52→23→16→31→50→14→30→28→8→3 6→27→46→17→33→38→49→18→45→51→6→26→39→1 2→48→29→40→43→15→0

Table 16 First Vehicle Route with numbering by ascending order

The above numbering of customers it is not the real position of each customer. The route with the real position of the customers is presented in the following table.

Vehicle	Route
1	0→39→167→68→156→134→24→157→186→6 →133→13→33→77→124→150→28→73→188 →91→55→17→78→191→41→174→92→48→184 →90→63→115→182→56→114→103→37→135→100 →170→65→127→152→181→66→169→183→30→ 95→153→54→178→113→154→163→61→0

Table 17 First Vehicle Route with original numbering of customers

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (Wij)	Customer Demand
0→39	3675	0
39→167	3640	35
167→68	3605	35
68→156	3483	122
156→134	3383	100
134→24	3261	122
24→157	3216	45
157→186	3181	35
186→6	3061	120
6→133	2917	144
133→13	2867	50
13→33	2832	35
33→77	2797	35
77→124	2675	122
124→150	2625	50
150→28	2481	144
28→73	2441	40
73→188	2406	35
188→91	2361	45
91→55	2316	45
55→17	2266	50
17→78	2144	122
78→191	2000	144
191→41	1878	122
41→174	1838	40
174→92	1803	35
92→48	1763	40
48→184	1619	144
184→90	1574	45
90→63	1539	35
63→115	1395	144
115→182	1360	35
182→56	1320	40
56→114	1198	122
114→103	1078	120
103→37	1033	45
37→135	988	45
135→100	844	144
100→170	804	40
170→65	769	35
65→127	724	45
127→152	674	50
152→181	639	35

181→66	594	45
66→169	554	40
169→183	514	40
183→30	479	35
30→95	444	35
95→153	409	35
153→54	364	45
54→178	220	144
178→113	175	45
113→154	135	40
154→163	95	40
163→61	50	45
61→0	0	50

Table 18 Remaining weight in each transition of vehicle 1

According to the table the vehicle 1 is loaded with 3675 kg and it serves a total of 55 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 2

The route that vehicle 2 was assigned is presented in the following table.

Vehicle	**Route**
2	0→25→16→1→35→24→31→39→30→42 →3→7→26→32→23→27→2→43→22→21 →40→14→13→11→6→36→18→28→12→38 →37→20→8→10→5→34→4→19→41→15→33 →29→9→17→0

Table 19 Second Vehicle Route with numbering by ascending order

The above numbering of customers it is not the real position of each customer. The route with the real position of the customers is presented in the following table.

Vehicle	**Route**
2	0→70→44→2→122→69→96→155→86→185 →4→11→71→102→64→75→3→189→58→53 →161→35→34→25→9→131→46→81→26→142 →137→50→12→20→7→120→5→47→161→36→105 →83→16→45→0

Table 20 Second Vehicle Route with original numbering of customers

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (Wij)	Customer Demand
0→70	2797	0
70→44	2747	50
44→2	2702	45
2→122	2657	45
122→69	2612	45
69→96	2468	144
96→155	2423	45
155→86	2343	80
86→185	2298	45
185→4	2258	40
4→11	2208	50
11→71	2128	80
71→102	2006	122
102→64	1971	35
64→75	1936	35
75→3	1896	40
3→189	1856	40
189→58	1816	40
58→53	1781	35
53→161	1659	122
161→35	1579	80
35→34	1539	40
34→25	1494	45
25→9	1454	40
9→131	1409	45
131→46	1364	45
46→81	1324	40
81→26	1279	45
26→142	1244	35
142→137	1194	50
137→50	1149	45
50→12	1104	45
12→20	1004	100
20→7	964	40
7→120	884	80
120→5	740	144
5→47	618	122
47→161	496	122
161→36	374	122
36→105	339	35

105→83	304	35
83→16	224	80
16→45	144	80
45→0	0	144

Table 21 Remaining weight in each transition of vehicle 2

According to the table the vehicle 2 is loaded with 2797 kg and it serves a total of 43 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Remaining Customers

The proposed mathematical model for 50 customers gives us a final solution with value of 20784m and the gap between our solution and optimal is 57,30%. The routes that each vehicle was assigned will be given in the following tables.

Vehicle 3

The route that vehicle 3 was assigned is presented in the following table.

Vehicle	**Route**
3	0→13→39→40→25→22→17→38→29→9→36→19→33→5 →27→16→6→31→14→32→35→10→12→26→15→21→0

Table 22 Third Vehicle Route with numbering by ascending order

The above numbering of customers it is not the real position of each customer. The route with the real position of the customers is presented in the following table.

Vehicle	**Route**
3	0→57→132→138→97→88→67→129→109→40→125→74 →118→19→99→62→27→116→59→117→123→43→52 →98→60→82→0

Table 23 Third Vehicle Route with original numbering of customers

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (Wij)	Customer Demand
0→57	1540	0
57→132	1396	144
132→138	1361	35
138→97	1326	35
97→88	1286	40
88→67	1186	100
67→129	1136	50
129→109	1014	122
109→40	969	45
40→125	924	45
125→74	802	122
74→118	757	45
118→19	707	50
19→99	662	45
99→62	617	45
62→27	495	122
27→116	445	50
116→59	400	45
59→117	355	45
117→123	315	40
123→43	280	35
43→52	160	120
52→98	110	50
98→60	75	35
60→82	35	40
82→0	0	35

Table 24 Remaining weight in each transition of vehicle 3

According to the table the vehicle 3 is loaded with 1540 kg and it serves a total of 25 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 4

The route that vehicle 4 was assigned is presented in the following table.

Vehicle	Route
4	0→42→20→50→2→49→18→7→48→44→1→28→8 →46→30→11→37→43→45→47→34→41→4→24 →23→3→0

Table 25 Fourth Vehicle Route with numbering by ascending order

The above numbering of customers it is not the real position of each customer. The route with the real position of the customers is presented in the following table.

Vehicle 4	Route
4	0→143→79→192→14→187→72→32→180→148→10 →107→38→171→110→51→126→144→162→179→11 9→141→18→94→89→15→0

Table 26 Fourth Vehicle Route with original numbering of customers

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (Wij)	Customer Demand
0→143	1750	0
143→79	1666	122
79→192	1616	50
192→14	1566	50
14→187	1521	45
187→72	1486	35
72→32	1342	144
32→180	1222	120
180→148	1187	35
148→10	1137	50
10→107	1097	40
107→38	1057	40
38→171	1017	40
171→110	972	45
110→51	932	40
51→126	892	40
126→144	748	144
144→162	604	144
162→179	559	45
179→119	519	40
119→141	397	122
141→18	347	50
18→94	312	35
94→89	162	150
89→15	40	122
15→0	0	40

Table 27 Remaining weight in each transition of vehicle 4

According to the table the vehicle 4 is loaded with 1750 kg and it serves a total of 25 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 5

The solution of the problem does not include the assignment of vehicle 5.

Due to the fact the total load of the vehicles is not optimal and the percentage of fulfil, which is shown on the table below, is not close to 100% we had to apply a customer import algorithm to the existing routes in order to reach the 100% capacity of each truck.

Vehicle	Vehicle Capacity	Route Capacity	Remaining Capacity	Percentage of Fulfil
3	4250	1540	2710	36,23%
4	2500	1750	750	70,00%

Table 28 Percentage of fulfil by vehicle

Remaining Customers with Insertion of customers and TSP application

The new routes of each truck with the insertion of customers are presented below.

Vehicle 3

The route that vehicle 3 was assigned is presented in the following table.

Vehicle	Route
3	0→15→14→17→23→30→12→40→43→16 →39→3→31→7→18→53→35→62→34→55 →57→37→13→68→65→64→63→1→4→67 →2→38→49→66→56→47→5→25→32→8 →69→61→60→26→24→22→9→11→6→45 →54→27→29→42→59→51→52→33→28 →21→58→19→46→36→50→48→10→44 →20→41→0

Table 29 Third Vehicle Route with numbering by ascending order

The above numbering of customers it is not the real position of each customer. The route with the real position of the customers is presented in the following table.

Vehicle	Route
3	0→57→52→60→82→98→43→117→123→59 →116→19→99→27→62→145→108→166→106

→147→151→111→49→177→173→172→168
→1→21→176→8→112→136→175→149→130
→22→85→101→29→190→165→164→87→84
→80→31→42→23→128→146→88→93→121
→159→139→140→104→93→76→158→67→129
→109→138→132→16→125→74→118→0

Table 30 Third Vehicle Route with original numbering of customers

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (Wij)	Customer Demand
0→57	4220	0
57→52	4104	144
52→60	4054	50
60→82	4014	40
82→98	3974	40
98→43	3939	35
43→117	3904	35
117→123	3864	40
123→59	3824	40
59→116	3779	45
116→19	3734	45
19→99	3689	45
99→27	3644	45
27→62	3599	45
62→145	3477	122
145→108	3442	35
108→166	3407	35
166→106	3357	50
106→147	3312	45
147→151	3272	40
151→111	3192	80
111→49	3157	35
49→177	3122	35
177→173	3087	35
173→172	2967	120
172→168	2927	40
168→1	2882	45
1→21	2847	35
21→176	2697	150
176→8	2652	45
8→112	2622	30

112→136	2577	45
136→175	2542	35
175→149	2497	45
149→130	2375	122
130→22	2340	35
22→85	2290	50
85→101	2255	35
101→29	2135	120
29→190	2015	120
190→165	1965	50
165→164	1815	150
164→87	1775	40
87→84	1735	40
84→80	1635	100
80→31	1600	35
31→42	1555	45
42→23	1435	120
23→128	1400	35
128→146	1278	122
146→88	1233	45
88→93	1153	80
93→121	1113	40
121→159	1078	35
159→139	1038	40
139→140	988	50
140→104	866	122
104→93	826	40
93→76	676	150
76→158	626	50
158→67	581	45
67→129	531	50
129→109	387	144
109→138	342	45
138→132	302	40
132→16	262	40
16→125	217	45
125→74	95	122
74→118	50	45
118→0	0	50

Table 31 Remaining weight in each transition of vehicle 3

According to the table the vehicle 3 is loaded with 4220 kg and it serves a total of 69 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 4

The route that vehicle 4 was assigned is presented in the following table.

Vehicle	Route
4	0→17→10→24→8→4→14→22→11→23 →7→18→15→21→13→6→12→19→1→20 →5→16→2→3→9→25→0

Table 32 Fourth Vehicle Route with numbering by ascending order

The above numbering of customers it is not the real position of each customer. The route with the real position of the customers is presented in the following table.

Vehicle	Route
4	0→143→89→187→72→14→119→179→94 →180→51→144→126→171→110→38→107 →148→10→162→15→141→14→15→79→192→0

Table 33 Fourth Vehicle Route with original numbering of customers

The remaining weight of each transaction plus the demand of the cluster is demonstrated in the table below.

Transition	Remaining Weight (Wij)	Customer Demand
0→143	1750	0
143→89	1666	122
89→187	1544	122
187→72	1510	35
72→14	1460	144
14→119	1358	35
119→179	1236	122
179→94	1196	40
94→180	1146	50
180→51	1111	35
51→144	1071	40
144→126	927	144
126→171	783	144
171→110	738	45
110→38	698	40

38→107	658	40
107→148	618	40
148→10	568	50
10→162	528	40
162→15	493	35
15→141	453	40
141→14	309	144
14→15	264	45
15→79	224	40
79→192	144	80
192→0	0	144

Table 34 Remaining weight in each transition of vehicle 4

According to the table the vehicle 4 is loaded with 1750 kg and it serves a total of 25 customers before it returns to the depot. From the above data it is clear that the vehicle capacity does not exceed.

Vehicle 5

The solution of the problem does not include the assignment of vehicle 5.

Chapter 5 Conclusions and future research

5.1 Comparison of the results

After the execution of the two cases for the capacitated vehicle routing problem with heterogeneous fleet algorithm it is clear from the collection of data that there are differences between solutions. In the following table is presented the summary of results in each solution.

Case	Solution
1st	59426m
2nd	45876m

Table 35 Summary of the Results

As we analyze the data from the table, the solution of the second case is better than the solution of the first case. Moreover, the second solution is by 22,8% better, as far as the total travel distance that the vehicles have to travel in order to serve all the customers.

As regard the fullness of the trucks during the routing, this is presented in the following tables. In the first table the fulfil of capacity of each is presented.

Vehicle	Capacity	Route Capacity	Remaining Capacity	Percentage of fulfil
1	2800	3684	16	99,58%
2	3700	2769	31	98,89%
3	4250	4232	18	99,58%
4	2500	1757	743	70,28%
5	3700	0	3700	00,00%

Table 36 Percentage of fulfil of vehicle in 1st Case

In the table below the fulfil of capacity for the second case is presented.

Vehicle	Capacity	Route Capacity	Remaining Capacity	Percentage of fulfil
1	2800	3675	25	99,30%
2	3700	2797	3	99,90%
3	4250	4220	30	99,29%
4	2500	1750	750	70,00%
5	3700	0	3700	00,00%

Table 37 Percentage of fulfil of vehicle in 2nd Case

From the above we can extract that the vehicles one, two and three for each case have a percentage very close to 100% and especially no more customer can be added to them because the minimum demand for a customer is 35 kg. For the vehicle 4 for both cases the percentage of fulfil is close to 70% this is mainly because on the route of vehicle our are added the customers that remain from vehicles one, two and three.

On another import data that can be collected from the solutions is the travel distance that each vehicle had to travel in order to serve the customers of the route. The distance for each vehicle of each case is shown in the table below.

Vehicle	1st Case Travel Distance	2nd Case Travel Distance
1	3820m	4936m
2	9289m	7127m
3	14186m	27723m
4	32131m	6090m
5	0m	0m

Table 38 Distance Travel by vehicle in each case

As we analyze the data from the table above, we can extract a very important note. For the vehicles one and two for each the travel distance as added is close. More specifically the travel distance of vehicle one in the first case is by 1100m smaller than in case two and the travel distance for vehicle two is greater by 2100m in case one than in case two. But, in the travel distance in vehicles three and four we can notice a very significant difference. The vehicle's three travel distance is better in the first case by 13000m, but on the opposite site the travel distance of vehicle four is better in second case by 26000m. This happen because of the simulation that in its case we are running. In the first case the route depends on the nearest customer, so in the final stage of the algorithm the customers that remain are the most distant between them and the vehicle four has to travel long distances to serve them. In the second case the third vehicle has a large travel distance since the customers that are in the route has no criterion between them and are the remaining customers from the first stage of the algorithm.

Lastly, an important factor that we can analyze is the number of customers that a vehicle is serving in the route that it is assigned. The number of customers of every route is shown in the table below.

Vehicle	Customers Served Case 1	Customers Served Case 2
1	59	55
2	47	43
3	60	69
4	26	25
5	0	0

Table 39 Customers Served per Route in both cases

The table below show us that the number of customers served by its route is close. In the first pair of vehicles in the second case the number of customers that is served 98 and in the second case is 106. And in the second pair the difference is again 8 customers but now in favor of the second case. The difference in the first pair of vehicles is covered by vehicle three in the second case.

In conclusion, we see that the case with the vehicle routing problem as part of the solution gives us a very good final solution compared to the solution of the problem without using it.

5.2 Future research

In a future research, more constraints based on real – life restrictions can be added to this model in order to extent it. One constraint that is important in a real-life instance is the fact that its vehicle have a limit in the hours of operation that can produce. In a real life the driver's working hour is limited to 8 and with the use of a table which consists the travelling time between each customer this constraint would be a critical factor. Furthermore, a basic constraint that could be added concerns the fact that many customers have specific time windows in which their service should be accomplished. This leads to the fact that a vehicle might arrive earlier than the earliest permitted time of service and thus should wait until the customer is available for service, but is prohibited to arrive later than the latest permitted time of service.

Annex

In this section of the thesis the nearest customers to depot model, the insertion of customers model and the nearest customer model that is used in order to demarcate the data of the problem and the source code that is used to solve the problem of vehicle routing with the simulation in CPLEX are presented.

Section A Nearest customers to depot

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
#include <cstdlib>
#include <vector>
#include <fstream>
#include <stdlib.h>
#include <algorithm>
#include <iterator>
#include<conio.h>

using std::vector;


int main() {

    // Data

    int             i, j, count
    int             D[193][193];

    vector<int>     distanceFromDepot; // Distance nearest customers
from depot vehicle one
    vector<int>     customerFromDepot;  // Ascending order nearest
customers vehicle one

    // Import of Data

    ifstream ifs_1;

    // Import of Distance Data

    ifs_1.open("Distances192.txt");
    for (i = 0; i < 193; i++) {
        for (j = 0; j < 193; j++) {
            ifs_1 >> D[i][j];
        }
    }


    // Main

    // Finding the distance from the depot to the customer
```

```
    for (i = 1; i < 193; i++) {
        int T;
        T = i;
        double K;
        K = D[0][T];
        distanceFromDepot.push_back(K);
        distanceandCustomer.push_back(T);
        distanceandCustomer.push_back(K);
        customerFromDepot.push_back(T);
    }


    // Classification by ascending order of customer

    int temp = 0;
    int tempB = 0;
    for (i = 0; i < distanceFromDepot.size(); i++) {
        for (j = i + 1; j < distanceFromDepot.size(); j++) {
            if (distanceFromDepot[i] > distanceFromDepot[j]) {
                temp = distanceFromDepot[i];
                distanceFromDepot[i] = distanceFromDepot[j];
                distanceFromDepot[j ] = temp;
                tempB = customerFromDepot[i];
                customerFromDepot[i] = customerFromDepot[j];
                customerFromDepot[j] = tempB;
            }
        }
    }

    // Data outpout

    ofstream myfile("Distance_Nearest_Customer_From_Depot.txt");
    if (myfile.is_open()) {
        for (count = 0; count < distanceFromDepot(); count++) {
            myfile << distanceFromDepot[count] << " ";
        }
        myfile.close();
    }
    else cout << "Unable to open file";

    ofstream myfile1("Nearest_Customers_To_Depot.txt");
    if (myfile1.is_open()) {
        for (count = 0; count < customerFromDepot.size(); count++) {
            myfile1 << customerFromDepot[count] << "    ";
        }
        myfile1.close();
    }
    else cout << "Unable to open file";

    system("pause");
    return 0;
}
```

Section B Insertion of customers

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
#include <cstdlib>
#include <vector>
#include <fstream>
#include <stdlib.h>
#include <algorithm>
#include <iterator>
#include<conio.h>

using std::vector;


int main() {

    // Data

    int             i, j, count, wA, wB, QWA, QWB;
    double          MAX = 1500000;
    double          w[193];
    int             D[193][193];
    int             VA[39]; // Vehicle one route
    int             VB[25]; // Vehicle two route

    vector<int>     vehicleOneRoute; // Vehicle one initital route
    vector<int>     vehicleTwoRoute; // Vehicle two initital route
    vector<int>     closestCustomerandFirst; // Nearest customers and
the refference vehicle one
    vector<int>     closestCustomerandFirstB; // Nearest customers
and the refference vehicle two
    vector<int>     closestCustomer; // Nearest customer vehicle one
    vector<int>     closestCustomerB; // Nearest customer vehicle two
    vector<int>     distanceFromDepot; // Distance nearest customers
from depot vehicle one
    vector<int>     distanceFromDepotB; // Distance nearest customers
from depot vehicle two
    vector<int>     distanceandCustomer; // Customer positition
Distance nearest customers from depot vehicle one
    vector<int>     distanceandCustomerB; // Customer positition
Distance nearest customers from depot vehicle two
    vector<int>     customerFromDepot;  // Ascending order nearest
customers vehicle one
    vector<int>     customerFromDepotB; // Ascending order nearest
customers vehicle two
    vector<int>     additionalCustomerA; // Additional customers
vehicle one
    vector<int>     additionalCustomerB; // Additional customers
vehicle two
    vector<int>     demandAdditionalCustomer; // Demand of additional
customers vehicle one
    vector<int>     demandAdditionalCustomerB; // Demand of
additional customers vehicle two
    vector<int>     totalLoad; // Total load added vehicle one
    vector<int>     totalLoadB; // Total load added vehicle two
```

```
    // Import of Data

    ifstream ifs_1;
    ifstream ifs_2;
    ifstream ifs_3;
    ifstream ifs_4;

    // Import of Distance Data

    ifs_1.open("Distances192.txt");
    for (i = 0; i < 193; i++) {
        for (j = 0; j < 193; j++) {
            ifs_1 >> D[i][j];
        }
    }

    // Import of Customer weight demand

    ifs_2.open("CustomerDemand192.txt");
    for (i = 0; i < 193; i++) {
        ifs_2 >> w[i];
    }

    // Import Vehicle Route

    // Vehicle One
    ifs_3.open("VehicleOneRoute.txt");
    for (i = 0; i < 39; i++) {
        ifs_3 >> VA[i];
    }

    for (i = 0; i < 39; i++) {
        vehicleOneRoute.push_back(VA[i]);
    }

    // Vehicle Two
    ifs_4.open("VehicleTwoRoute.txt");
    for (i = 0; i < 25; i++) {
        ifs_4 >> VB[i];
    }

    for (i = 0; i < 25; i++) {
        vehicleTwoRoute.push_back(VB[i]);
    }

    // Main

    // Finding the nearest customers to the already customers of the
route for vehicle one

    for (i = 0; i < 192; i++) {
        int P = i;
        bool exists = std::find(std::begin(vehicleOneRoute),
std::end(vehicleOneRoute), P) != std::end(vehicleOneRoute);
        if (exists) {
```

```
            bool exists = std::find(std::begin(vehicleTwoRoute),
std::end(vehicleTwoRoute), P) != std::end(vehicleTwoRoute);
            if (!exists){
                int firstJ = 0;
                int secJ = 0;
                double firstmin = MAX;
                double secmin = MAX;
                for (j = 0; j < 192; j++) {
                    int T = j;
                    if (T != 0) {
                        bool exists =
std::find(std::begin(vehicleOneRoute), std::end(vehicleOneRoute), T)
!= std::end(vehicleOneRoute);
                        if (!exists) {
                            bool exists =
std::find(std::begin(vehicleTwoRoute), std::end(vehicleTwoRoute), T)
!= std::end(vehicleTwoRoute);
                            if (!exists) {
                                bool exists =
std::find(std::begin(closestCustomer), std::end(closestCustomer), T)
!= std::end(closestCustomer);
                                if (!exists) {
                                    if (D[i][j] < firstmin && D[i][j]
!= 0) {

                                        secmin = firstmin;
                                        firstmin = D[i][j];
                                        secJ = firstJ;
                                        firstJ = j;

                                    }
                                    else if (D[i][j] < secmin &&
D[i][j] != 0) {

                                        secmin = D[i][j];
                                        secJ = j;
                                    }
                                }
                            }
                        }
                    }
                    closestCustomerandFirst.push_back(P);
                    closestCustomerandFirst.push_back(firstJ);
                    closestCustomerandFirst.push_back(secJ);
                    closestCustomer.push_back(firstJ);
                    closestCustomer.push_back(secJ);
                }
            }
        }

    // Finding the nearest customers to the already customers of the
route for vehicle two
    for (i = 0; i < 192; i++) {
        int P = i;
        bool exists = std::find(std::begin(vehicleTwoRoute),
std::end(vehicleTwoRoute), P) != std::end(vehicleTwoRoute);
        if (exists) {
            bool exists = std::find(std::begin(vehicleOneRoute),
std::end(vehicleOneRoute), P) != std::end(vehicleOneRoute);
            if (!exists) {
```

```
                int firstB = 0;
                int secB = 0;
                double firstminB = MAX;
                double secminB = MAX;
                for (j = 0; j < 192; j++) {
                    int T = j;
                    if (T != 0) {
                        bool exists =
std::find(std::begin(vehicleTwoRoute), std::end(vehicleTwoRoute), T)
!= std::end(vehicleTwoRoute);
                        if (!exists) {
                            bool exists =
std::find(std::begin(vehicleOneRoute), std::end(vehicleOneRoute), T)
!= std::end(vehicleOneRoute);
                            if (!exists) {
                                bool exists =
std::find(std::begin(closestCustomerB), std::end(closestCustomerB),
T) != std::end(closestCustomerB);
                                if (!exists) {
                                    if (D[i][j] < firstminB &&
D[i][j] != 0) {
                                        secminB = firstminB;
                                        firstminB = D[i][j];
                                        secB = firstB;
                                        firstB = j;
                                    }
                                    else if (D[i][j] < secminB &&
D[i][j] != 0) {
                                        secminB = D[i][j];
                                        secB = j;
                                    }
                                }
                            }
                        }
                    }
                }
                closestCustomerandFirstB.push_back(P);
                closestCustomerandFirstB.push_back(firstB);
                closestCustomerandFirstB.push_back(secB);
                closestCustomerB.push_back(firstB);
                closestCustomerB.push_back(secB);
            }
        }
    }

    // Finding the distance from the depot to the customer vehicle
one
    for (i = 0; i < closestCustomer.size(); i++) {
        int T;
        T = closestCustomer[i];
        double K;
        K = D[0][T];
        distanceFromDepot.push_back(K);
        distanceandCustomer.push_back(T);
        distanceandCustomer.push_back(K);
        customerFromDepot.push_back(T);
    }
```

```
    // Finding the distance from the depot to the customer vehicle
two
    for (i = 0; i < closestCustomerB.size(); i++) {
        int T;
        T = closestCustomerB[i];
        double K;
        K = D[0][T];
        distanceFromDepotB.push_back(K);
        distanceandCustomerB.push_back(T);
        distanceandCustomerB.push_back(K);
        customerFromDepotB.push_back(T);
    }

    // Classification by ascending order of customer vehicle one
    int temp = 0;
    int tempB = 0;
    for (i = 0; i < distanceFromDepot.size(); i++) {
        for (j = i + 1; j < distanceFromDepot.size(); j++) {
            if (distanceFromDepot[i] > distanceFromDepot[j]) {
                temp = distanceFromDepot[i];
                distanceFromDepot[i] = distanceFromDepot[j];
                distanceFromDepot[j ] = temp;
                tempB = customerFromDepot[i];
                customerFromDepot[i] = customerFromDepot[j];
                customerFromDepot[j] = tempB;
            }
        }
    }
    // Classification by ascending order of customer vehicle two
    temp = 0;
    tempB = 0;
    for (i = 0; i < distanceFromDepotB.size(); i++) {
        for (j = i + 1 ; j < distanceFromDepotB.size(); j++) {
            if (distanceFromDepotB[i] > distanceFromDepotB[j]) {
                temp = distanceFromDepotB[i];
                distanceFromDepotB[i] = distanceFromDepotB[j];
                distanceFromDepotB[j] = temp;
                tempB = customerFromDepotB[i];
                customerFromDepotB[i] = customerFromDepotB[j];
                customerFromDepotB[j] = tempB;
            }
        }
    }

    // Insert of customers of  vehicle one
    wA = 1082;
    for (i = 0; i < customerFromDepot.size(); i++) {
        if (wA > 0 || wA < 35) {
            QWA = customerFromDepot[i];
            wA = wA - w[QWA];
            if ( wA >= 0 ) {
                additionalCustomerA.push_back(customerFromDepot[i]);
                demandAdditionalCustomer.push_back(w[QWA]);
                vehicleOneRoute.push_back(customerFromDepot[i]);
            }
            else {
                wA = wA + w[QWA];
            }
```

```
        }
    }
    totalLoad.push_back(1082 - wA);

    // Insert of customers of  vehicle two
    wB = 1208;
    for (i = 0; i < customerFromDepotB.size(); i++) {
        if (wB > 0 || wB < 35) {
            QWB = customerFromDepot[i];
            bool exists = std::find(std::begin(additionalCustomerA),
std::end(additionalCustomerA), QWB) != std::end(additionalCustomerA);
            if (!exists) {
                wB = wB - w[QWB];
                if (wB >= 0) {

additionalCustomerB.push_back(customerFromDepotB[i]);
                    demandAdditionalCustomerB.push_back(w[QWB]);
                    vehicleTwoRoute.push_back(customerFromDepotB[i]);
                }
                else {
                    wB = wB + w[QWB];
                }
            }
        }
    }
    totalLoadB.push_back(1208 - wB);

    // Data outpout

    ofstream myfile("Vehicle_One_Route.txt");
    if (myfile.is_open()) {
        for (count = 0; count < vehicleOneRoute.size(); count++) {
            myfile << vehicleOneRoute[count] << "   ";
        }
        myfile.close();
    }
    else cout << "Unable to open file";

    ofstream myfile1("Vehicle_Two_Route.txt");
    if (myfile1.is_open()) {
        for (count = 0; count < vehicleTwoRoute.size(); count++) {
            myfile1 << vehicleTwoRoute[count] << "  ";
        }
        myfile1.close();
    }
    else cout << "Unable to open file";

    system("pause");
    return 0;
}
```

Section C Nearest Customer Routing

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
#include <cstdlib>
#include <vector>
#include <fstream>
#include <stdlib.h>
#include <algorithm>
#include <iterator>
#include<conio.h>

using std::vector;


int main() {

    // Data

    int             i, j, count;
    int             VWA, VWB, VWC, VWD, VWE; // Vehicle capacities
    int             VWAA, VWBB, VWCC, VWDD, VWEE; // Vehicle route
capacity
    double          MAX = 1500000;
    double          w[193];
    int             D[193][193];

    vector<int>     shortestPath; // Shortest Path
    vector<int>     distanceShortestPath;   // Total distance of
shortest path
    vector<int>     totalDistanceShortestPath; // Total distance of
shortest path
    vector<int>     additionalCustomerVA; // Customers that is added
in vehicle one
    vector<int>     additionalCustomerVB; // Customers that is added
in vehicle two
    vector<int>     additionalCustomerVC; // Customers that is added
in vehicle three
    vector<int>     additionalCustomerVD; // Customers that is added
in vehicle four
    vector<int>     additionalCustomerVE; // Customers that is added
in vehicle five
    vector<int>     demandAdditionalCustomerVA; // Demand of the
customer that is added in vehicle one
    vector<int>     demandAdditionalCustomerVB; // Demand of the
customer that is added in vehicle two
    vector<int>     demandAdditionalCustomerVC; // Demand of the
customer that is added in vehicle three
    vector<int>     demandAdditionalCustomerVD; // Demand of the
customer that is added in vehicle four
    vector<int>     demandAdditionalCustomerVE; // Demand of the
customer that is added in vehicle five
    vector<int>     totalLoadVA; // Total load of vehicle one
    vector<int>     totalLoadVB; // Total load of vehicle two
    vector<int>     totalLoadVC; // Total load of vehicle three
    vector<int>     totalLoadVD; // Total load of vehicle four
```

```
    vector<int>     totalLoadVE; // Total load of vehicle five
    vector<int>     totalTravelDistance; // Total travel distanve for
all vehicles
    vector<int>     remainingWeightVA; // Remaining weight in each
transition of vehicle one
    vector<int>     remainingWeightVB; // Remaining weight in each
transition of vehicle two
    vector<int>     remainingWeightVC; // Remaining weight in each
transition of vehicle three
    vector<int>     remainingWeightVD; // Remaining weight in each
transition of vehicle four
    vector<int>     remainingWeightVE; // Remaining weight in each
transition of vehicle five

    // Import of Data

    ifstream ifs_1;
    ifstream ifs_2;

    // Import of Distance Data

    ifs_1.open("Distances193.txt");
    for (i = 0; i < 193; i++) {
        for (j = 0; j < 193; j++) {
            ifs_1 >> D[i][j];
        }
    }

    // Import of Customer weight demand

    ifs_2.open("CustomerDemand192.txt");
    for (i = 0; i < 193; i++) {
        ifs_2 >> w[i];
    }

    // Main

    // Finding the shortest route
    i = 0;
    j = 0;
    int R = 0;
    int firstJ = 0;
    shortestPath.push_back(0);
    while (R < 192) {
        i = firstJ;
        double firstmin = MAX;
        for (j = 0; j < 193; j++) {
            bool exists = std::find(std::begin(shortestPath),
std::end(shortestPath), j) != std::end(shortestPath);
            if (!exists) {
                if (D[i][j] < firstmin && D[i][j] != 0) {
                    firstmin = D[i][j];
                    firstJ = j;
                }
            }
        }
        shortestPath.push_back(firstJ);
        distanceShortestPath.push_back(D[i][firstJ]);
```

```
    R = R + 1;
}
shortestPath.push_back(0);

// Finding the trvel distance of the shortest travel route
int totalDistance = 0;
for (i = 0; i < distanceShortestPath.size(); i++) {
    totalDistance = totalDistance + distanceShortestPath[i];
}
totalDistance = totalDistance + D[146][0];
totalDistanceShortestPath.push_back(totalDistance);

// Procedure of insert customers to vehicles
VWA = 3700;
VWB = 2800;
VWC = 4250;
VWD = 2500;
VWE = 3700;
additionalCustomerVB.push_back(0);
additionalCustomerVC.push_back(0);
additionalCustomerVD.push_back(0);
demandAdditionalCustomerVB.push_back(0);
demandAdditionalCustomerVC.push_back(0);
demandAdditionalCustomerVD.push_back(0);
for (i = 0; i < shortestPath.size(); i++) {
    VWA = VWA - w[shortestPath[i]];
    VWB = VWB - w[shortestPath[i]];
    VWC = VWC - w[shortestPath[i]];
    VWD = VWD - w[shortestPath[i]];
    VWE = VWE - w[shortestPath[i]];
    if (VWA >= 0) {
        additionalCustomerVA.push_back(shortestPath[i]);
        demandAdditionalCustomerVA.push_back(w[shortestPath[i]]);
        VWB = VWB + w[shortestPath[i]];
        VWC = VWC + w[shortestPath[i]];
        VWD = VWD + w[shortestPath[i]];
        VWE = VWE + w[shortestPath[i]];
    }
    else if (VWB >= 0) {
        additionalCustomerVB.push_back(shortestPath[i]);
        demandAdditionalCustomerVB.push_back(w[shortestPath[i]]);
        VWA = VWA + w[shortestPath[i]];
        VWC = VWC + w[shortestPath[i]];
        VWD = VWD + w[shortestPath[i]];
        VWE = VWE + w[shortestPath[i]];
    }
    else if (VWC >= 0) {
        additionalCustomerVC.push_back(shortestPath[i]);
        demandAdditionalCustomerVC.push_back(w[shortestPath[i]]);
        VWA = VWA + w[shortestPath[i]];
        VWB = VWB + w[shortestPath[i]];
        VWD = VWD + w[shortestPath[i]];
        VWE = VWE + w[shortestPath[i]];
    }
    else if (VWD >= 0) {
        additionalCustomerVD.push_back(shortestPath[i]);
        demandAdditionalCustomerVD.push_back(w[shortestPath[i]]);
        VWA = VWA + w[shortestPath[i]];
```

```
                VWB = VWB + w[shortestPath[i]];
                VWC = VWC + w[shortestPath[i]];
                VWE = VWE + w[shortestPath[i]];
            }
            else {
                additionalCustomerVE.push_back(shortestPath[i]);
                demandAdditionalCustomerVE.push_back(w[shortestPath[i]]);
                VWA = VWA + w[shortestPath[i]];
                VWB = VWB + w[shortestPath[i]];
                VWC = VWC + w[shortestPath[i]];
                VWD = VWD + w[shortestPath[i]];
            }
        }
        additionalCustomerVB.push_back(0);
        additionalCustomerVC.push_back(0);
        additionalCustomerVD.push_back(0);
        demandAdditionalCustomerVB.push_back(0);
        demandAdditionalCustomerVC.push_back(0);
        demandAdditionalCustomerVD.push_back(0);
        totalLoadVA.push_back(3700 - VWA);
        totalLoadVB.push_back(2800 - VWB);
        totalLoadVC.push_back(4250 - VWC);
        totalLoadVD.push_back(2500 - VWD);

        // Finding the travel distance of each vehicle
        int distanceVA = 0;
        for (i = 0; i < additionalCustomerVA.size(); i++) {
            distanceVA = distanceVA +
D[additionalCustomerVA[i]][additionalCustomerVA[i + 1]];
        }

        int distanceVB = 0;
        for (i = 0; i < additionalCustomerVB.size(); i++) {
            distanceVB = distanceVB +
D[additionalCustomerVB[i]][additionalCustomerVB[i + 1]];
        }

        int distanceVC = 0;
        for (i = 0; i < additionalCustomerVC.size(); i++) {
            distanceVC = distanceVC +
D[additionalCustomerVC[i]][additionalCustomerVC[i + 1]];
        }

        int distanceVD = 0;
        for (i = 0; i < 27; i++) {
            int P = D[additionalCustomerVD[i]][additionalCustomerVD[i +
1]];
            distanceVD = distanceVD +
D[additionalCustomerVD[i]][additionalCustomerVD[i + 1]];
        }

        totalTravelDistance.push_back(distanceVA + distanceVB +
distanceVC + distanceVD);

        // Finding the remaining weight in transition of each vehicle
        VWAA = 3700 - VWA;
        VWBB = 2800 - VWB;
        VWCC = 4250 - VWC;
```

```
    VWDD = 2500 - VWD;
    VWEE = 3700 - VWE;
    for (i = 0; i < demandAdditionalCustomerVA.size(); i++) {
        VWAA = VWAA - demandAdditionalCustomerVA[i];
        remainingWeightVA.push_back(VWAA);
    }
    for (i = 0; i < demandAdditionalCustomerVB.size(); i++) {
        VWBB = VWBB - demandAdditionalCustomerVB[i];
        remainingWeightVB.push_back(VWBB);
    }
    for (i = 0; i < demandAdditionalCustomerVC.size(); i++) {
        VWCC = VWCC - demandAdditionalCustomerVC[i];
        remainingWeightVC.push_back(VWCC);
    }
    for (i = 0; i < demandAdditionalCustomerVD.size(); i++) {
        VWDD = VWDD - demandAdditionalCustomerVD[i];
        remainingWeightVD.push_back(VWDD);
    }

    // Export of file

    ofstream myfile("Shortest_Path.txt");
    if (myfile.is_open()) {
        for (count = 0; count < shortestPath.size(); count++) {
            myfile << shortestPath[count] << "   ";
        }
        myfile.close();
    }
    else cout << "Unable to open file";

    ofstream myfile1("Vehicle_One_Route.txt");
    if (myfile1.is_open()) {
        for (count = 0; count < additionalCustomerVA.size(); count++)
{
            myfile1 << additionalCustomerVA[count] << " ";
        }
        myfile1.close();
    }
    else cout << "Unable to open file";

    ofstream myfile2("Vehicle_Two_Route.txt");
    if (myfile2.is_open()) {
        for (count = 0; count < additionalCustomerVB.size(); count++)
{
            myfile2 << additionalCustomerVB[count] << " ";
        }
        myfile2.close();
    }
    else cout << "Unable to open file";

    ofstream myfile3("Vehicle_Three_Route.txt");
    if (myfile3.is_open()) {
        for (count = 0; count < additionalCustomerVC.size(); count++)
{
            myfile3 << additionalCustomerVC[count] << " ";
        }
        myfile3.close();
    }
```

```
    else cout << "Unable to open file";

    ofstream myfile4("Vehicle_Four_Route.txt");
    if (myfile4.is_open()) {
        for (count = 0; count < additionalCustomerVD.size(); count++)
{
            myfile4 << additionalCustomerVD[count] << " ";
        }
        myfile4.close();
    }
    else cout << "Unable to open file";

    ofstream myfile5("Vehicle_Five_Route.txt");
    if (myfile5.is_open()) {
        for (count = 0; count < additionalCustomerVE.size(); count++)
{
            myfile5 << additionalCustomerVE[count] << " ";
        }
        myfile5.close();
    }
    else cout << "Unable to open file";

    ofstream myfile6("Total_Travel_Distance.txt");
    if (myfile6.is_open()) {
        myfile6 << distanceVA << "  ";
        myfile6 << distanceVB << "  ";
        myfile6 << distanceVC << "  ";
        myfile6 << distanceVD << "   ";
        myfile6 << distanceVA + distanceVB + distanceVC + distanceVD
<< "   ";
        myfile6.close();
    }
    else cout << "Unable to open file";

    ofstream myfile7("Remaining_Weight_Vehicle_One.txt");
    if (myfile7.is_open()) {
        for (count = 0; count < remainingWeightVA.size(); count++) {
            myfile7 << remainingWeightVA[count] << "     ";
        }
        myfile7.close();
    }
    else cout << "Unable to open file";

    ofstream myfile8("Remaining_Weight_Vehicle_Two.txt");
    if (myfile8.is_open()) {
        for (count = 0; count < remainingWeightVB.size(); count++) {
            myfile8 << remainingWeightVB[count] << "     ";
        }
        myfile8.close();
    }
    else cout << "Unable to open file";

    ofstream myfile9("Remaining_Weight_Vehicle_Three.txt");
    if (myfile9.is_open()) {
        for (count = 0; count < remainingWeightVC.size(); count++) {
            myfile9 << remainingWeightVC[count] << "     ";
        }
        myfile9.close();
```

```
    }
    else cout << "Unable to open file";

    ofstream myfile10("Remaining_Weight_Vehicle_Four.txt");
    if (myfile10.is_open()) {
        for (count = 0; count < remainingWeightVD.size(); count++) {
            myfile10 << remainingWeightVD[count] << "    ";
        }
        myfile10.close();
    }
    else cout << "Unable to open file";

    system("pause");
    return 0;
}
```

Section D Mathematical Model of Vehicle Routing Problem

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN

#include <vector>
#include <fstream>
#include <stdlib.h>
using std::vector;

int main() {

// Data

int             i,j;                // i,j: pointer of customer and
depot k: pointer of vehicles
int             k;                  // k  : pointer of vehicles
int             N=64;               // N  : number of customers
const int       imax=65;            // max nodes (customer and depot)
const int       jmax=65;            // max nodes (customer and
1depot)
const int       Kmax=5;             // max number of vehicles
const int       M=100000;           // M  : auxiliary variable
float           w[imax];            // Table of demand of customers
defined in kg
float           QW[Kmax];           // Table of capacity of vehicle
type k defined in kg
float           D[imax][jmax];      // Table of the distances from
customer to customer and from depot to customers defined in m

//---------------------------------------------------------------------
-------------------------------------------------------

// Initialization:Set table of distances and tables of capacity to
zero

for (i=0;i<imax;i++){
    for (j=0;j<jmax;j++){
        D[i][j]=0;
    }
}

for (k=0;k<Kmax;k++){
    QW[k]=0;
}

for (i=0;i<imax;i++){
    w[i]=0;
}

//---------------------------------------------------------------------
-------------------------------------------------------

// Import of Data

ifstream ifs_1;                 //Distances
```

```
ifstream ifs_2;                  //Demand in kg
ifstream ifs_3;                  //Capacity of vehicles in kg

//------------------------------------------------------------------
-------------------------------------------------------

// Import of Distance Data

ifs_1.open("Distances.txt");
for (i=0;i<imax;i++){
    for ( j=0;j<jmax;j++){
        ifs_1 >> D[i][j];
    }
}

//------------------------------------------------------------------
-------------------------------------------------------

//Import of Weight of Demand Data

ifs_2.open("CustomerDemand.txt");
for (i=0;i<imax;i++){
    ifs_2 >> w[i];
    //cout<<"w["<<i<<"]="<<w[i]<<endl;
}

//------------------------------------------------------------------
-------------------------------------------------------

// Import of Vehicles' Capacity

// Capacity in kg
ifs_3.open("VehicleCapacity.txt");
for (k=0;k<Kmax;k++){
    ifs_3 >> QW[k];
    //cout<<"QW["<<k<<"]="<<QW[k]<<endl;
}

//------------------------------------------------------------------
-------------------------------------------------------

// Building Model

IloEnv env;
try{
IloModel model (env);
typedef IloArray<IloNumVarArray> IloNumVarMatrix2x2;
typedef IloArray<IloNumVarMatrix2x2> IloNumVarMatrix3x3;
typedef IloArray<IloRangeArray> IloRangeMatrix2x2;
typedef IloArray<IloRangeMatrix2x2> IloRangeMatrix3x3;
IloCplex cplex(env);

//------------------------------------------------------------------
-------------------------------------------------------------------
-----------------------------

// Decision Variables
```

```
// Xijk
char Path[35];
IloNumVarMatrix3x3 Xijk(env,0);
for (i=0;i<imax;i++){
    IloNumVarMatrix2x2 Xjk(env,0);
    for (j=0;j<jmax;j++){
        IloNumVarArray Xk(env,0);
        for (k=0;k<Kmax;k++){
            sprintf(Path,"Xijk(i%d,j%d,k%d)",i,j,k);
            IloNumVar X(env,0,1,ILOBOOL,Path);
            Xk.add(X);
        }
        Xjk.add(Xk);
    }
    Xijk.add(Xjk);
}

// Wij
char RemainingWeight[35];
IloNumVarMatrix2x2 Wij(env,0);
for (i=0;i<imax;i++){
    IloNumVarArray Wj(env,0);
    for (j=0;j<jmax;j++){
        sprintf(RemainingWeight,"Wij(i%d,j%d)",i,j);
        IloNumVar W(env,0,10000,ILOFLOAT,RemainingWeight);
        Wj.add(W);
    }
    Wij.add(Wj);
}

//Oi
char Order[35];
IloNumVarArray Oi(env,0);
for (i=0;i<imax;i++){
    sprintf(Order,"Oi(i%d",i);
    IloNumVar O(env,0,100,ILOINT,Order);
    Oi.add(O);
}

//-----------------------------------------------------------------------
-------------------------------------------------------------

int UB, LB;
float UB1, LB1;

// Constraints

// Constraint (1):  Starting from i node we have to travel to another
j node

LB=1;
UB=1;
IloRangeArray Sum1_Xi(env,0);
for (i=1;i<imax;i++){
    IloExpr expr(env,0);
    for (j=0;j<jmax;j++){
        if (i!=j)
            for (k=0;k<Kmax;k++){
```

```
                expr+=Xijk[i][j][k];
            }
        }
        IloRange Sum1_X(env,LB,expr,UB);
        expr.end();
        model.add(Sum1_X);
        Sum1_Xi.add(Sum1_X);
}

// Constraint (2):  Each customer must be visited exactly once

IloRangeArray Sum2_Xj(env,0);
for (j=1;j<jmax;j++){
        IloExpr expr(env,0);
        for (i=0;i<imax;i++){
            if (i!=j)
                for (k=0;k<Kmax;k++){
                    expr+=Xijk[i][j][k];
                }
        }
        IloRange Sum2_X(env,LB,expr,UB);
        expr.end();
        model.add(Sum2_X);
        Sum2_Xj.add(Sum2_X);
}

// Constraint (3):  Each vehicle leaves depot one time max

LB1=-IloInfinity;
char leave[30];
IloRangeArray leave_oncek(env, 0);
for (k=0;k<Kmax;k++){
        IloExpr expr(env, 0);
        for (j=1;j<jmax;j++){
            expr += Xijk[0][j][k];
        }
        sprintf(leave,"leave_oncek(k%d)",k);
        IloRange leave_once(env,LB1,expr,UB,leave);
        model.add(leave_once);
        leave_oncek.add(leave_once);
        expr.end();
        }

// Constraint (4):  Each vehicle leaves from depot and return to
depot

UB=0;
char apodepot[30];
IloRangeArray DEPOTijk(env, 0);
for (k = 0; k<Kmax; k++){
        IloExpr expr(env, 0);
        IloExpr expr1(env, 0);
        IloExpr expr2(env, 0);
        for (i=1;i<imax;i++){
            for (j=0;j<jmax;j++){
                expr1 += Xijk[i][j][k];
            }
        }
```

```
    for (j=1;j<jmax;j++){
        expr2 -= Xijk[0][j][k];
    }
    expr = expr1 + expr2 * M;
    sprintf(apodepot,"DEPOTij(k%d)",k);
    IloRange DEPOTij(env,LB1,expr,UB,apodepot);
    model.add(DEPOTij);
    DEPOTijk.add(DEPOTij);
    expr.end();
}

// Constraint (5):  Flow conservation

LB=0;
IloRangeMatrix2x2 Equality_Xjk(env,0);
for (k=0;k<Kmax;k++){
    IloRangeArray Equality_Xk(env,0);
    for (j=0;j<jmax;j++){
        IloExpr expr(env,0);
            for (i=0;i<imax;i++){
                if (i!=j)
                    expr+=Xijk[i][j][k]-Xijk[j][i][k];
            }
        IloRange Equality_X(env,LB,expr,UB);
        expr.end();
        model.add(Equality_X);
        Equality_Xk.add(Equality_X);
    }
    Equality_Xjk.add(Equality_Xk);
}

// Constraint (6):  Sub-tour elimination

char Order1[35];
IloRangeMatrix2x2 Subtourij(env,0);
for (i=1;i<imax;i++){
    IloRangeArray Subtourj(env,0);
    for (j=1;j<jmax;j++){
        IloExpr expr(env,0);
        if (i!=j)
        for (k=0;k<Kmax;k++){
            expr+=Xijk[i][j][k];
        }
        expr+=Oi[i]-Oi[j]+N*expr-N+1;
        sprintf(Order1,"Subtourij(i%d,j%d)",i,j);
        IloRange Subtour(env,LB1,expr,UB,Order);
        expr.end();
        model.add(Subtour);
        Subtourj.add(Subtour);
    }
    Subtourij.add(Subtourj);
}

// Constraint (7):  The total load which starts from depot must be
equal to total demand in kg

float sum_w=0;
for (j=0;j<jmax;j++){
```

```
    sum_w+=w[j];
}
//cout<<"sum_w="<<sum_w<<endl;

IloExpr exprw(env,0);
for (j=0;j<jmax;j++){
     exprw+=Wij[0][j];
}
char Sum_W[100];
sprintf(Sum_W,"Sum_W");
LB1=sum_w, UB1=sum_w;
IloRange SumW(env,LB1,exprw,UB1,Sum_W);
exprw.end();
model.add(SumW);

// Constraint (8):  The vehicle cacpacity in kg must not exceeded

LB1=0;
UB1=IloInfinity;
char WeightCapacity[35];
IloRangeMatrix2x2 CapacityWeight_Xij(env,0);
for (i=0;i<imax;i++){
    IloRangeArray CapacityWeight_Xj(env,0);
    for (j=1;j<jmax;j++){
        IloExpr expr(env,0);
        if (i!=j)
        for (k=0;k<Kmax;k++){
            expr+=QW[k]*Xijk[i][j][k];
        }
        expr-=Wij[i][j];
        sprintf(WeightCapacity,"CapacityWeight_Xij(i%d,j%d)",i,j);
        IloRange CapacityWeight_X(env,LB1,expr,UB1,WeightCapacity);
        expr.end();
        model.add(CapacityWeight_X);
        CapacityWeight_Xj.add(CapacityWeight_X);
    }
    CapacityWeight_Xij.add(CapacityWeight_Xj);
}

// Constraint (9): Each customer receives exactly his demands in kg

char Sumw[35];
IloRangeArray Sum_Wj(env,0);
for (j=1;j<jmax;j++){
    IloExpr expr(env,0);
    for (i=0;i<imax;i++){
        if (i!=j)
        expr+=Wij[i][j]-Wij[j][i];
    }
    sprintf(Sumw,"SumW_Wj(j%d)",j);
    LB1=w[j], UB1=w[j];
    IloRange Sum_W(env,LB1,expr,UB1,Sumw);
    expr.end();
    model.add(Sum_W);
    Sum_Wj.add(Sum_W);
}

UB=Kmax;
```

```
LB=0;

// Constraint (10): Violation of number of vehicles

IloRangeArray NumberofVehicles_Xi(env,0);
for (i=0;i<1;i++){
IloExpr expr(env);
    for (k=0;k<Kmax;k++){
        for (j=0;j<jmax;j++){
            expr+=Xijk[i][j][k];
        }
    }
char Vehicles_Number[100];
sprintf(Vehicles_Number,"NumberofVehicles_Xi(i%d)",i);
int LB=0,UB=Kmax;
IloRange NumberofVehicles_X(env,LB,expr,UB,Vehicles_Number);
expr.end();
model.add(NumberofVehicles_X);
NumberofVehicles_Xi.add(NumberofVehicles_X);
}

//-------------------------------------------------------------------
-----------------------------------------------------

//Objective Function: Minimization of the total travel distance

IloExpr expr_obj(env);
for (i=0;i<imax;i++){
    for (j=0;j<jmax;j++){
        for (k=0;k<Kmax;k++){
            //cout<< i <<"     "<< j <<"    "<< k<<endl;
            expr_obj+=D[i][j]*Xijk[i][j][k];
        }
    }
}
model.add(IloMinimize(env, expr_obj));
expr_obj.end();

// For Out of memory problem

cplex.setParam(IloCplex::WorkMem,4000.0);
cplex.setParam(IloCplex::WorkDir,"c:/cplex/");
cplex.setParam(IloCplex::NodeFileInd,3);
cplex.setParam(IloCplex::EpGap,0.0000);

// Solve

cplex.extract(model);
cplex.exportModel("onoma.lp");

// Print Results

if (!(cplex.solve())){
    env.error()<<"Faild to optimize LP."<<endl;
    throw(-1);
}

env.out()<<"Solution status = " <<cplex.getStatus()<<endl;
```

```
env.out()<<"Solution value = " <<cplex.getObjValue()<<endl;

cplex.solve();

//Print Results

ofstream Solution;
Solution.open("Sol.txt");
Solution<<cplex.getObjValue()<<endl;
Solution<<endl;

// Print Xijkd

cout<<endl;
cout<<"The optimal routes are:"<<endl;
for (i=0;i<imax;i++){
    for (j=0;j<jmax;j++){
        for (k=0;k<Kmax;k++){
            //cout<< i <<"     "<< j <<"    "<< k<<endl;
            if (D[i][j]!=0){
            int g = cplex.getValue(Xijk[i][j][k]);

if(g!=0)cout<<"Xijk"<<"("<<i<<","<<j<<","<<k<<")"<<"="<<g<<endl;
            if(g!=0)
Solution<<"Xijk"<<"("<<i<<","<<j<<","<<k<<")"<<"="<<g<<endl;
            }
        }
    }
}
cout<<endl;

// Print Wij

cout<<"The remaining weight in each transition is:"<<endl;
for (i=0;i<imax;i++){
    for (j=0;j<jmax;j++){
        float g = cplex.getValue(Wij[i][j]);
        if(g!=0 && abs(g)>0.01)
cout<<"Wij"<<"("<<i<<","<<j<<")"<<"="<<g<<endl;
        if(g!=0)Solution<<"Wij"<<"("<<i<<","<<j<<")"<<"="<<g<<endl;
    }
}
cout<<endl;

Solution<<endl;
Solution.close();
}
catch ( IloException& e){
    cerr <<"concert exception caught:"<<e<<endl;
}
catch (...){
    cerr <<"Unknown exception caught"<<endl;
}

//------------------------------------------------------------------
-------------------------------------------------------

// End of env
```

```
env.end();

system("pause");
return 0;

}
// End main
```

Bibliography

1) George L. Nemhauser, Laurence A. Wolsey. (1998). Integer and combinatorial optimization, John Wiley & Sons, Inc
2) Alexander Schrijver. (1998). Theory of Linear and Integer Programming, John Wiley & Sons, Inc.
3) Zambito, Leonardo. The Traveling Salesman Problem: A Comprehensive Survey. s.l. : Submitted as a project for CSE 4080, Fall 2006
4) AppleGate, R. Bixby, V. Chvatal and W. Cook. Chapter 3, pp. 645-656. On the Solution of the Traveling. s.l. : Documenta Mathematica - Extra Volume ICM, 1998.
5) L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. s.l. : John Wiley & Sons, 1985.
6) Wikipedia, the free encyclopedia - RAND. [Online] November 10, 2006. http://en.wikipedia.org/wiki/RAND.
7) G.B. Dantzig, R. Fulkerson, and S.M. Johnson. Solution of a large-scale traveling salesman problem. Operations Research 2. 1954.
8) Gilbert Laporte, Paolo Toth, Daniele Vigo. Vehicle routing: historical perspective and recent contributions. s.l. : Springer-Verlag Berlin Heidelberg and EURO - The Association of European Operational Research Societies, March 2013.
9) Roberto Baldacci, Maria Battarra and Daniele Vigo. Technical Report: Routing a Heterogeneous Fleet of Vehicles. Bologna, Italy : DEIS, University Bologna, January 2007.
10) Ralphs, T., Kopman, L., Pulleyblank, W.*et al.*On the capacitated vehicle routing problem.*Math. Program.* (2003)
11) Lysgaard, J., Letchford, A. & Eglese, R. A new branch-and-cut algorithm for the capacitated vehicle routing problem.*Math. Program* (2004)
12) Achuthan, N., Caccetta, L., Hill, S.: An improved branch-and-cut algorithm for the capacitated vehicle routing problem. Transportation. (2003)
13) Augerat, P, Naddef, D, Belenguer, J M, Benavent, E, Corberan, A, and Rinaldi, G. Computational results with a branch and cut code for the capacitated vehicle routing problem. France: N. 1995.
14) Naddef, D., Rinaldi, G.: Branch-and-cut algorithms for the capacitated VRP. In: Toth, P., Vigo, D. (eds.) The Vehicle Routing Problem. SIAM, 2002.
15) Laporte, C. Barnhart and G. Chapter 6 Vehicle Routing. Handbook in OR & MS, Vol. 14. s.l. : Elsevier B.V., 2007. =
16) Kohl, N., Desrosiers, J., Madsen, O., Solomon, M., Soumis, F.: 2-Path cuts for the vehicle routing problem with time windows. Transportation Science (1999)
17) Sofie Coene, Arent Arnout and Frits C.R. Spieksma. The Periodic Vehicle Routing Problem: A Case Study. Katholieke Universiteit Leuven (KBI) : Department of Decision Sciences and Information Management, October, 2008.
18) Fisher, M.: Optimal solution of vehicle routing problem using minimum k-trees. Oper. Res. (1994)
19) Wen, Min. Rich Vehicle Routing Problems and Applications. s.l. : PhD Thesis, DTU Management Engineering, July 2010.
20) Christofides, N., Mingozzi, A., Toth, P.: Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Math. Prog. (1981)

21) Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. Oper. Res. Q. (1969)
22) Manuel Iori, Silvano Martello. Routing problems with loading constraints. PAGE 18
23) Leandro C. Coelho, Jacques Renaud, Gilbert Laporte. Road - Based Goods Transportation: A Survey of Real - World Applications from 2000 to 2015. s.l. : CIRRELT, August 2015.