NATIONAL TECHNICAL UNIVERSITY OF ATHENS

Optimization Methods Applied in a Class of Vehicle Routing and Scheduling Problems

DOCTORAL THESIS

Author

Grigorios D. Konstantakopoulos

Supervisor

Dr. Ilias P. Tatsiopoulos

Professor NTUA

A thesis submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

in the

Industrial Engineering Laboratory

Sector of Industrial Management and Operational Research

Mechanical Engineering School

July 2022

NATIONAL TECHNICAL UNIVERSITY OF ATHENS

Optimization Methods Applied in a Class of Vehicle Routing and Scheduling Problems

DOCTORAL THESIS

Author

Grigorios D. Konstantakopoulos

Supervisor

Dr. Ilias P. Tatsiopoulos

Professor NTUA

A thesis submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

in the

Industrial Engineering Laboratory

Sector of Industrial Management and Operational Research

Mechanical Engineering School

July 2022

National Technical University of Athens
Mechanical Engineering School
Sector of Industrial Management and Operational Research
Industrial Engineering Laboratory

Optimization Methods Applied in a Class of Vehicle Routing and Scheduling Problems

PhD Thesis

Grigorios D. Konstantakopoulos

Examination Committee:

Dr. Tatsiopoulos I., Supervisor
Professor NTUA

Dr. Panayiotou N.
Professor NTUA

Dr. Ponis S.
Professor NTUA

Dr. Aravossis K.
Professor NTUA

Dr. Varvarigou T.
Professor NTUA

Dr. Tolis A.
Associate Professor NTUA

Dr. Zeimpekis
Assistant Professor

Acknowledgements

This PhD thesis, entitled "Optimization Methods Applied in a Class of Vehicle Routing and Scheduling of Deliveries Problems" was conducted in Mechanical Engineering School of National Technical University of Athens during February of 2018 and May 2022. At this point, I would like to thank certain people who have been very supportive of me all this time and have contributed significantly to the completion of the work.

First of all, I would like to express my gratitude to my primary advisor, Professor Ilias Tatsiopoulos, for allowing me to work on this project and guiding my PhD research. I want to thank him for trusting my potential, giving me the appropriate amount of freedom in following my ideas and providing a very supportive working environment as the research group head. Additionally, I would like to thank Dr. Sotiris Gayialis for providing me with precious advice and suggestions and being available anytime to solve questions and give me advice. Our cooperation led to writing multiple research articles and papers that helped me acquire knowledge and experience that will be valuable in the future. Also, I would like to thank Dr. Georgios Papadopoulos for his support.

My sincere thanks go to my other two supervisors, Professors Stavros Ponis and Nikolaos Panayiotou, to participate in my thesis committee and give me valuable feedback on several occasions. Great thanks to my friend and colleague Evripidis Kechagias for his precious assistance, support and valuable collaboration. Many thanks to all the people in the Sector of Industrial Management and Operational Research for creating a friendly and professional environment.

I am also indebted to all the people from Optimum S.A, especially Dimitris Pergamalis, Elias Kanakis and Aris Kolofotias who helped me by supporting the empirical part of this thesis.

Finally, I would like to express my deepest sincere appreciation to my mother, father, and sister for their endless love and support and my friends for helping me relieve the pressure that is sometimes accumulated from the research process. And for the one who has made all the endless hours and effort spent seem to worth, my life partner Matina. You were always on my side, giving me faith and strength to accomplish my targets.

Abstract

The present thesis studies and addresses a class of vehicle routing and scheduling problems while also developing powerful evolutionary and local search metaheuristics that aim to optimize the distribution process in urban areas by proposing a plan of routes and deliveries. The fact that route planning is affected by many variables and constraints that have emerged by the nature of logistics companies, their external environment, and their customers' needs and requirements has led to an increased interest both scientifically and commercially. The current thesis, aiming to bring together research and industry, also studies and proposes the integration of the developed algorithms into a cloud routing system, offering highly efficient route plans.

Moreover, the different variables and constraints of the Vehicle Routing Problem (VRP) correspond to different variants of the problem. In the current thesis, the variants considered have arisen through an analytical literature review and the cooperation with industrial and software development companies operating in this field. Specifically, the most commonly addressed, by researchers and practitioners, VRP variants and those with the potential to contribute financially and environmentally to logistics companies in the future are selected. This methodology ensures that both fields (research and industry) will benefit and be interested in, as they are inextricably linked to each other. The variants of the problem that are selected and studied are:

- the Vehicle Routing Problem with Time Windows (VRPTW),
- the Vehicle Routing Problem with Simultaneous Pickups and Deliveries (VRPSPD),
- the Heterogeneous Fleet Vehicle Routing Problem (HVRP), and
- the Collaborative VRP.

The VRP variants under study belonging to NP-hard problems are highly complex, primarily when addressed simultaneously. In this thesis, one of the goals is to address all the above-mentioned variants either solely or combined in order to cover most needs and requirements of logistics companies operating on freight distribution field. Each of the above VRP variants corresponds to specific characteristics, variables and restrictions. Specifically, the VRPTW considers the time slots that customers indicate in order to be served by logistics companies. The VRPSPD focuses, as its name indicates, on simultaneous pickups and deliveries, while the HVRP on the heteroginity of vehicles that logistics companies own, in terms of capacity, fixed and variable costs. Finally, the Collaborative VRP focuses on cases that logistics companies that locate and operate at the same areas collaborate in the distribution process. All these VRP variants form the main requirements and challenges that logistics companies face on freight distribution, and by addressing them great benefits can emerge.

Combining multiple VRP variants is complex, along with the need of companies to acquire efficient and cost-saving solutions have led to the need for optimization algorithms. Initially, exact algorithms were proposed by researchers. Their advantage, which is optimally solving the VRP, is accompanied by a significant constrain; the limited number of customers that can be handled and the increased computation time needed, especially when customers exceed 100. This limitation is restrictive, and therefore these algorithms are not integrated into software packages. The disadvantages of exact algorithms resulted in researchers proposing heuristics algorithms that offer good quality solutions that are not optimal in most cases in limited computation time. Finally, researchers developed metaheuristic algorithms that, in most cases, offer the best trade-off between efficiency of solutions and computation time.

In the current thesis, multiple metaheuristic algorithms are studied, developed, and proposed to solve the VRP variants that are mentioned above and correspond to those that logistics companies most commonly address on freight distribution. An Evolutionary Algorithm (EA) and two Local Search (LS) metaheuristics are developed. These algorithms address either separately or simultaneously some of the proposed VRP variants. More precisely, the developed EA addresses the Vehicle Routing Problem with Simultaneous Pickups and Deliveries and with Time Windows (VRPSPDTW). The algorithm is also applied in cases with electric vehicles with a limited driving range, affecting distribution. The novelty in this solution method is related to the applied crossover operator and the fact that the problem is addressed as multiobjective for the first time. In multiobjective optimization, two or more conflicting objectives exist and need to be optimized, which in the current research are (i) the number of vehicles needed for executing the routes, and (ii) the total traveled distance.

Additionally, the first LS algorithm, a Large Neighborhood Search (LNS) metaheuristic, addresses the VRPTW as a multiobjective. This is the first time a multiobjective LNS algorithm is proposed in the specific problem, offering high-quality results. Finally, the second developed LS metaheuristic is an Adaptive Large Neighborhood Search (ALNS) metaheuristic that addresses the Heterogeneous Fleet Vehicle Routing Problem with Simultaneous Pickups and Deliveries and with Time Windows (HVRPSPDTW), which means that the three VRP variants are considered simultaneously. The ALNS algorithm is also applied, after some modifications, in the Collaborative VRP to estimate the economic and environmental benefits from the cooperation of distribution companies. In the context of studying VRP variants and addressing them through optimization algorithms, their mathematical model was formulated, as it is a vital part of efficiently addressing the problem and setting the right variables and constraints.

Finally, the three algorithms have been developed in the Python programming language as libraries containing the necessary functions for solving the VRP. The algorithms are tested in multiple datasets of the literature to ensure their efficiency and ability to be integrated into a cloud routing system. The proposed system is developed in cooperation with a software development company that is responsible for the user interface part, as well as the connection of the cloud system with its individual parts of the system (such as online maps and the

database). Additionally, the requirements of the system regarding the master data, as well as its functionalities have been defined jointly.

On the other hand, the core of the system, which is the algorithms that are implemented, have been developed exclusively by me, and configured to serve the need of the software development company. Since all of the proposed algorithms offer high-quality results, they are all stored in the software development company's server, so that to be accessible by the cloud routing system through the appropriate requests. While the system has the ability to access any of the proposed algorithms, in its current form, the system receives the plan of routes that is extracted by the ALNS algorithm, covering multiple freight distribution cases. The system that handles and addresses multiple variables and constraints needs to be fed with the correct data to offer efficient routes and deliveries. Therefore, it exploits advanced technologies that can ensure the quality and accuracy of data. Online maps are exploited concerning the geocoding procedure and the data related to the distance and the travel time between order points.

The benefits of using such a routing system provided through the cloud, and thus no installation of the system is required on each company's computers, are many and are not limited only to the financial aspect. Besides eliminating installation and maintenance costs, the users can extract efficient and high accuracy plans that are visualized on online maps, modify the proposed routes and schedules, and execute rerouting strategies after the distribution process has started. In addition to these functionalities, when fed with real-time data from online maps, the system can handle the dynamic nature of the problem and address unexpected events that are occurred after the plan of routes have begun. Specifically, data may change during the route execution, affecting the distribution process. All the technologies mentioned above and system functionalities can significantly enhance logistics companies' efforts and decision-makers before and after the distribution process started.

Concluding, the research focuses on multiple aspects, starting with the study of the above-mentioned VRP variants which are considered either solely or combined, along with their mathematical formulation. In addition, the presented thesis proposes and develops optimization algorithms that address these distribution cases and are tested in both datasets of the literature and in real-life cases in order to evaluate the ability of the algorithms to be implemented in a routing system. The last part of the research concerns the ability of algorithms to be suitable for integration into a cloud routing system, as well as their connection with technologies that are necessary for system development.

Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation	Definition
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
HVRP	Heterogeneous Fleet Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
VRPSPD	Vehicle Routing Problem with Simultaneous Pickups and Deliveries
VRPSPDTW	Vehicle Routing Problem with Simultaneous Pickups and Deliveries and Time Windows
HVRPSPDTW	Heterogeneous Fleet Vehicle Routing Problem with Simultaneous Pickups and Deliveries and Time Windows
Coll-VRP	Collaboration Vehicle Routing Problem
DVRP	Dynamic Vehicle Routing Problem
PS	Population Search
GA	Genetic Algorithm
MA	Memetic Algorithm
EA	Evolutionary Algorithms
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
SS	Scatter Search
LS	Local Search
LNS	Large Neighborhood Search
ALNS	Adaptive Large Neighborhood Search
ILS	Iterated Local Search
GRASP	Greedy Randomized Adaptive Search Procedure
GLS	Guided Local Search
MO	Multi-objective
MOLNS	Multi-objective Large Neighborhood Search
MOEA	Multi-objective Evolutionary Algorithms
SaaS	Software as a Service
GIS	Geographic Information System
DSS	Decision Support System
3PL	Third-Party Logistics

Chapter 1: Introduction

This introductory Chapter aims to provide the background, motivation, contribution and overview of this thesis. The research topic, the Vehicle Routing Problem (VRP), is initially introduced, while the motivation that led to extensively studying the VRP is presented. Consequently, the contribution of this work is also given, followed by the overview of the thesis.

1.1 Motivations

Supply chain networks include multiple operations such as the supply of raw materials to manufacturers, the warehousing of products, and the distribution of goods to retailers and consumers. Regardless it is perceived or not, all these actions play an essential role in our lives. Efficient operations across the entire supply chain network can provide high-quality products delivered on time and at minimum cost. The current thesis enhances this effort by focusing on one of the last parts of supply chain networks, specifically on distributing goods, where multiple variables and constraints affect the process.

The study's significance can also be enhanced by numerical data (Statistics Explained 2019), as the road freight transport, which includes national, international, cross trade, and cabotage, increased by 0.2% from 2017 to 2018 in the countries of the European Union. More precisely, the total freight transportation in 2018 reached 1,925 billion tonne-kilometers. Concerning Greece, the total freight transport reached 28.4 billion tonne-kilometers, from which 54.5% correspond to national transportation. Additionally, the transport and logistics sector in Greece (including road, rail, air and shipping) is rapidly growing and contributes around 9% of gross domestic product (GDP) (19 billion). Despite the importance of the sector in the national economy, another research (EEL 2018) that included 181 companies operating in this field indicated that only 29% of the companies use routing systems and only 12% telematic systems. This fact is not limited only to Greece, as other studies such as Leyerer et al. (2019) indicate that most companies do not use routing software yet, despite the considerable saving potentials regarding driving distances, costs, and resulting emissions. These data can be interpreted either as that companies are mistrustful concerning the use of systems or that systems and software packages constitute expensive solutions for small-size companies.

Additionally, in the past decade, e-commerce, online purchases, and urbanization have increased. This fact has led to an enlarged number of end customers and retailers, especially in city centers. This fact significantly affects the distribution process as logistics companies need to serve well over 2,000 orders daily. Moreover, logistics companies need to handle and respect the requirements of their customers, such as their time windows and their needs for simultaneously receiving and returning goods, as well as the company's characteristics such as the different vehicles that distribute items. As a result, advanced algorithmic approaches integrated into routing systems seem to be the only solution for companies. All these needs of

logistics companies have driven us to study the VRP and its variants, develop algorithmic approaches that can handle large amounts of data while also considering multiple variables and constraints involved. Moreover, their integration into a cloud routing system provided through the internet and the Software as a Service (SaaS) model is also a field of study in the current thesis.

However, besides the needs of logistics companies and the benefits that can emerge from this research, the primary motivation originates from the gap that exists in the research community. Specifically, only a few algorithmic approaches proposed and published by researchers are also integrated into routing software packages (Drexl 2012), let alone to cloud routing systems, as it emerges from the state of the art analysis regarding the VRP in Chapter 2. This is a significant case since researchers study the VRP for over 60 years as it was introduced by Dantzig and Ramser (1959). This is the simplest version of the VRP and is characterized by a fleet of vehicles, based at one or several warehouses, that need to serve geographically scattered customers with known demands. All vehicles start from a central depot and need to return after executing the routes (Máximo and Nascimento 2021). The aim is to serve these customers at minimum cost, by defining the routes of vehicles originating and terminating at the depot. Over the years, different variants of the VRP that are correlated with specific real-life distribution parameters or constraints have been considered, formulated, and studied (Konstantakopoulos et al. 2020a).

One of the most known variants of the VRP is the Vehicle Routing Problem with Time Windows, where geographically scattered customers need to be served within a predetermined time interval (time window), only once and by a single vehicle. The total quantity of goods distributed in each route cannot exceed the vehicle's capacity, while each vehicle starts and ends its route at a specific depot. A vehicle may arrive at a customer before the opening of the time window and wait until the agreed service time, but is not allowed to deliver goods if it arrives after the time window closes (Nagata et al. 2010; Ait Haddadene et al. 2019; Juárez Pérez et al. 2019). Another variant of the VRP is the Vehicle Routing Problem with Simultaneous Pickups and Deliveries (VRPSPD), where customers can both receive and return goods (Nagy and Salhi 2005). This creates an extra constraint as at any stage of the distribution, whether goods are delivered or collected, the capacity of the vehicle cannot be violated. In the Heterogeneous Fleet Vehicle Routing Problem (HVRP), one of the most encountered variants, the vehicles are considered different in terms of capacity, variable and fixed cost. This is quite common in the distribution network as few companies own and operate exclusively identical vehicles. Finally, in the Collaborative VRP, logistics companies that locate and distribute goods on the same areas cooperate in the distribution process, and consequently in the routing. A mixture of these variants can represent a practical and realistic routing problem that most logistics companies that distribute freight face daily.

Moreover, as the VRP variants increased over the years and newly emerged, the need for more efficient algorithms that could handle multiple variants at the same time also arose. Initially,

exact algorithms were proposed most often by researchers. However, the fact that these methods can handle only a limited number of customers (around 100) due to the computational complexity of the VRP was a determining factor for not being studied and applied so often. Consequently, researchers focused on developing approximate methods that offer the best trade-off between quality of solutions and computation time. This was necessary since, in real-life distribution cases, the number of orders may exceed 2,000. This becomes increasingly important, especially in today's challenging market where e-commerce and urbanization have increased. Heuristics constitute such methods that produce good quality solutions in minimum computation time. The first heuristics that was developed focused on constructing a feasible solution, while in a later face focused on the improvement of the solution by relocating one customer to another route. After heuristics, the attention was given to metaheuristics considered sophisticated heuristics based on mathematical, biology, and nature concepts. Their ability to provide improved solutions compared to heuristics at a similar computation time made them more popular.

Additionally, the fact that researchers have intensely studied metaheuristic algorithms over the years is also a motivation for creating even more efficient metaheuristics that can outperform the existing algorithms and have the ability to tackle multiple VRP variants simultaneously. The developed algorithms focus on specific attributes to be optimized and offer the best possible results.

To conclude, while the main gap between research and logistics industry is observed on the limited algorithms that are developed and integrated into real-life systems, the presented thesis aims to study the VRP from multiple perspectives. Specifically, the objectives of the research are to study the VRPTW, VRPSPD, HVRP and Collaborative VRP either solely or combined, formulate the complex problems, propose and develop efficient metaheuristic algorithms, and finally study how the proposed algorithms can be integrated into a cloud routing system and connect with latest technological advances. The benefits that can be offered to consumers and logistics companies by addressing the VRP through efficient algorithmic approaches integrated into real-life applications constitute an important motivating force of the current research. In addition, the study of a cloud routing system that integrates novel algorithmic approaches and exploits advanced technologies that offer the necessary data fast and with high accuracy constitute another key pillar of the research. Finally, this thesis covers a wide range of operations to develop a cloud routing system, aiming to bring the research community and industry closer.

1.2 Research Methodology

Over the years an increasing number of problems arise, leading applied research to become more and more common, as it aims to solve practical problems of the modern world. On the other hand, pure research is driven by scientists' curiosity and interest on a specific research/question, aiming to expand knowledge and not to create or invent something. The

current research is classified in the applied research category and propose practical solutions on a problem that is of great interest for practitioners and city inhabitants. That is the Vehicle Routing Problem (VRP) that is related to the distribution of freight to consumers and retailers. Specifically, in the current research a number of optimization algorithms are developed for addressing the VRP and its variants. The developed algorithms propose novel operators based on advanced mathematical models, some of which formulated for the first time. Consequently, while the research belongs on the applied research category, multiple elements of innovations are proposed.

Additionally, besides classifying the research, either as pure or applied, it is essential to identify the followed research methodology, as well as, the rationale behind the decision. The current research follows the quantitative approach, as the research focuses on the generalization of experimental results and adopts structured, and highly planned approaches that will give specific results (Fee J.F 2012). In general, these methods require large sample sizes and use of statistical analysis techniques to make sense of the data (Rokou E. 2013). In the current research, multiple datasets of the literature are exploited for testing the proposed algorithms along with case studies arose from logistics companies.

The steps followed on the current research are clearly defined bellow in order to clarify the aims and contributions of the thesis. Specifically, the first step of the research methodology is the literature review of the VRP regarding freight distribution, focusing on the multiple variables and constraints that have been formed over the years, as well as the optimization algorithms that have been developed for addressing the problem. Through this approach, the multiple VRP variants that have been formed over the years are studied and analyzed, offering an overall knowledge in the specific research subject. Additionally, through the literature review analysis, the most commonly studied VRP variants are identified. This analysis directs the research in selecting the VRP variants that concern most logistics companies and stakeholders, as the trends in the research community comfort with those of the industry. In addition, an identical analysis is executed in the optimization algorithms, so that to identify those that are most commonly applied in the VRP.

Since the most commonly addressed VRP variants are identified in the review process, along with the optimization methods applied, multiple algorithms are developed for addressing these variants. Each developed algorithm addresses different variants of the problems, but may has similarities with the other algorithms. For each problem addressed, the mathematical model is formulated, as the description of a system using mathematical concepts and language and is mandatory before developing an algorithm.

Consequently, two multiobjective algorithms are developed, a Large Neighborhood Search (LNS) and an Evolutionary Algorithm (EA). The first one addresses solely the VRP with Time Windows, while the former the VRP with Simultaneous Pickups and Deliveries, and Time Windows. Both algorithms aim to minimize both the total travelled distance and the number of

vehicles. Both objectives are significant and shape the total distribution cost. The aim for developing two algorithms that address similar problems and are tested in the same dataset is to evaluate a Population Search metaheuristic (as the EA) and a Local Search Metaheuristic (as the LNS) in order to identify their efficiency both in terms of solution quality and in time complexity. The choice for selecting metaheuristic algorithms over heuristics is based on the review process.

Following the evaluation of the two algorithms and its outcome, another local search metaheuristic algorithm is developed. The proposed algorithm aims to address more variables and constraints, compared to the previously mentioned algorithms, so that to enhance the effort of logistics and distribution companies operating in city environments. Considering that the aim of the developed algorithms is also to be integrated into a cloud system, the selection of the variables and constraints constitutes an important decision. Therefore, the considered variants are not limited to those that originate from the review process, but are strengthened by other variants that are proposed from stakeholders of the field. The developed algorithm is an Adaptive LNS algorithms that considers only a single objective which is the total distribution cost.

The common thread between all developed algorithms is the process followed after their development which is the testing procedure. All algorithms are tested in datasets of the literature in order to estimate their efficiency.

Following the testing procedure, the integration of the developed algorithms in a cloud routing system is performed. The integration of algorithms into the system makes the evaluation of the system possible. Only through this way the benefits by using a system for the distribution process can be identified. Therefore, real-life distribution cases from multiple sectors (e-commerce, supermarket chain, door to door deliveries) are solved, so that to enhance the evaluation process and the statements of the current study.

Concluding, the methodology followed in the current research is multifaceted as it focuses in multiple aspects, so that to be complete and eliminate research gaps. The steps of the research methodology describe the process for developing algorithms that are implemented in a cloud routing system. On the contrary, most research studies focus on some of these steps, leaving aside in most cases the implementation of the algorithm in a decision support system. Finally, the research methodology is described in Figure 1.1.

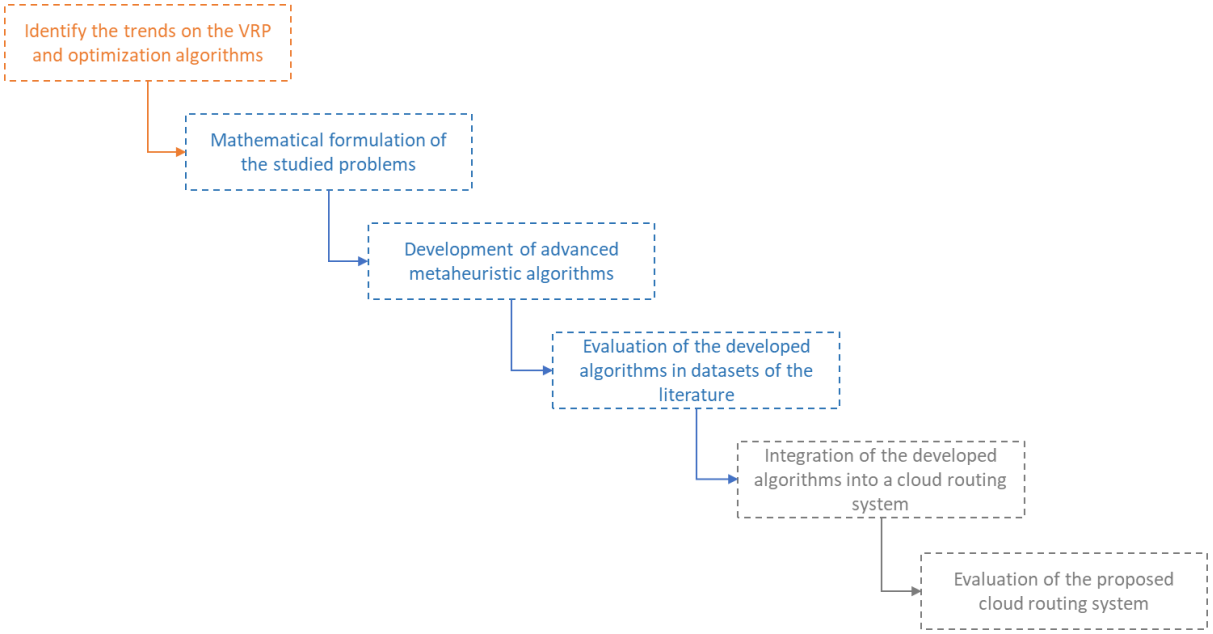


Figure 1.1 Research Methodology

1.3 Purpose and Contribution of Thesis

The main purpose of the current study is to fill the research gap, which is the limited algorithms that are developed by researchers and address simultaneously multiple variables and constraints involved in the distribution process, as well as that are integrated into real-life applications. To fulfill this objective and offer advanced technological solutions, it is necessary to initially identify the variables and constraints that are most commonly addressed by logistics companies, and consequently define the mathematical formulation of the problem. The most commonly addressed variables and constraints correspond to the needs of most logistics companies and that is the main reason for focusing on them. Thereafter, the research focuses on developing advanced metaheuristic algorithms that solve the complex distribution cases that have been formed and offer decreased distribution costs. Finally, another critical objective of the current study is the integration of the proposed metaheuristics in a cloud routing system that can be applied in real distribution operations.

As for the contributions of the current study, they are inextricably linked to the aims of the research. More specifically, the contributions of the current study are separated into five main pillars which are:

i. the literature review for identifying the trends of the Vehicle Routing Problem (VRP) variants and optimization algorithms, their correlation, as well as for the algorithms that lead to real-life applications,

ii. the optimization methods that are developed for addressing a wide range of VRP variants that correspond to multiple variables and constraints of the distribution process,

iii. the mathematical formulations that are developed for the addressed problems, even in cases that multiple variables and constraints of urban freight distribution are met,

iv. the computational results, as the developed algorithms are tested both in multiple datasets of the literature, offering new optimal solutions, as well as in real-life distribution cases, offering decreased distribution costs to companies, and

v. the practical application and implementation of the developed algorithms into a cloud routing system.

1.4 Overview of the PhD thesis

The remainder of this thesis is organized as follows:

Chapter 2 presents state of the art in VRP and the multiple VRP variants that exist, and how these variants are connected to real-life distribution cases regarding urban freight transportation. This is a fundamental part of the research as the most frequently addressed VRP variants are defined and compared with the variables and constraints that companies of the industry consider most important. Additionally, the literature review focuses on the algorithmic approaches that exist, along with the trends that the algorithms follow. Moreover, a correlation and connection between VRP variants algorithmic approaches are defined, while the publications that develop algorithms and integrate them into real-life applications are analyzed.

Chapter 3 addresses the Vehicle Routing Problem with Time Windows (VRPTW) solely. More specifically, a literature review concerning the VRPTW is conducted. Consequently, the mathematical formulation of the problem is presented, and the proposed algorithmic approach (Multiobjective Large Neighborhood Search - MOLNS) addresses the specific problem. Finally, the algorithm is tested in Solomon's benchmark instances, and its results are compared to the best-published in the literature. The algorithm indicates high-quality results, and even new best results are proposed.

Chapter 4 addresses the Vehicle Routing Problem with Simultaneous Pickups and Deliveries and Time Windows (VRPSPDTW). The structure of the current Chapter follows Chapter 3 as initially a literature review in the VRPSPD is presented. Consequently, the mathematical formulation of the VRPSPDTW is presented, along with the proposed algorithmic approach, i.e., the Multiobjective Evolutionary Algorithm (MOEA). The algorithm is tested in two datasets of the literature to define its efficiency, indicating high-quality results in both datasets, as in both cases, new optimal solutions are proposed.

Chapter 5 addresses the Heterogeneous Fleet Vehicle Routing Problem with Simultaneous Pickups and Deliveries and Time Windows (HVRPSPDTW), which combines the VRPTW, the VRPSPD, and the HVRP. The literature review of this problem is followed by the mathematical formulation and description of the problem. The proposed Adaptive Large Neighborhood Search (ALNS) algorithm is initially presented, while its efficiency is defined by testing it in multiple literature datasets. Additionally, the algorithm is tested in a

collaborative case to estimate the benefits in terms of cost and emissions when three Third-Party Logistics (3PL) companies cooperate in the distribution process.

Chapter 6 describes a cloud routing system that exploits the developed in Chapter 3, Chapter 4, and Chapter 5 algorithms. Additionally, the technologies that are exploited for the development of the cloud system are also discussed, along with the system's functionalities. Finally, the system is tested and validated in three real-life distribution cases to estimate its efficiency and the benefits offered to logistics companies and planners.

Chapter 7 summarizes the contribution of the current research, the goals that have been achieved, and the future research steps that will be followed for addressing even more variants that concern researchers and practitioners, as well as the improvements that can be made to the mathematical formulation and the algorithms for extracting even more efficient route plans.

Finally, the analytical results produced by the testing of the proposed algorithms are presented in the **Appendices**. More specifically, the solutions that improve the best-published results in the datasets of the literature are thoroughly analyzed and presented. Finally, the developed algorithms are also given in the Python programming language.

The interactions between the Chapters of the thesis are depicted in Figure 1.2. Specifically, the research process starts with state-of-the-art analysis concerning the vehicle routing problem and related optimization methods. Then, the research continues with developing three metaheuristic algorithms addressing multiple VRP variants and ends with their integration into a cloud routing system that offers multiple benefits to logistics and distribution companies that are the potential users.

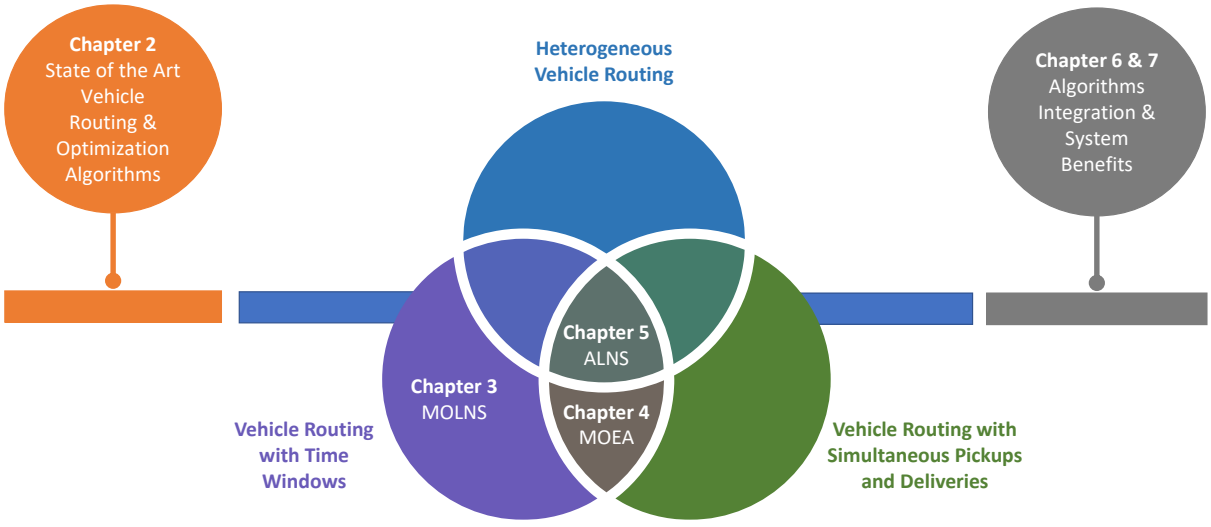


Figure 1.2 Thesis Outline

References

Ait Haddadene S, Labadie N, Prodhon C (2019) Bicriteria Vehicle Routing Problem with Preferences and Timing Constraints in Home Health Care Services. Algorithms 12:152. https://doi.org/doi.org/10.3390/a12080152

Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manage Sci 6:80–91. https://doi.org/10.1287/mnsc.6.1.80

Drexl M (2012) Rich vehicle routing in theory and practice. Logist Res 5:47–63. https://doi.org/10.1007/s12159-012-0080-2

EEL (2018) Logistics in Greece Today

Fee, J.F. (2012). Quantitative Methods in Action Research. In: Klein, S.R. (eds) Action Research Methods. Palgrave Macmillan, New York. https://doi.org/10.1057/9781137046635_8

Juárez Pérez M., Pérez Loaiza RE, Quintero Flores PM, et al (2019) A Heuristic Algorithm for the Routing and Scheduling Problem with Time Windows: A Case Study of the Automotive Industry in Mexico. Algorithms 12:111. https://doi.org/10.3390/a12050111

Konstantakopoulos GD, Gayialis SP, Kechagias EP (2020) Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. Oper Res Int J. https://doi.org/10.1007/s12351-020-00600-7

Leyerer M, Sonneberg M-O, Heumann M, et al (2019) Individually Optimized Commercial Road Transport: A Decision Support System for Customizable Routing Problems. Sustainability 11:. https://doi.org/10.3390/su11205544

Máximo VR, Nascimento MC V (2021) A hybrid adaptive iterated local search with diversification control to the capacitated vehicle routing problem. Eur J Oper Res. https://doi.org/10.1016/j.ejor.2021.02.024

Nagata Y, Bräysy O, Dullaert W (2010) A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Comput Oper Res 37:724–737. https://doi.org/https://doi.org/10.1016/j.cor.2009.06.022

Nagy G, Salhi S (2005) Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. Eur J Oper Res 162:126–141. https://doi.org/https://doi.org/10.1016/j.ejor.2002.11.003

Rokou E. (2013) Decision making in project management: multi objective extended resource constrained project scheduling. National Technical University of Athens

Statistics Explained (2019) Road freight transport statistics

Chapter 2: State of the Art

2.1 Introduction

The Vehicle Routing Problem (VRP) is one of the most critical challenges that logistics companies face nowadays. Researchers have been studying the routing of vehicles and scheduling of deliveries since 1959, when Dantzig and Ramser (1959) introduced the Truck Dispatching Problem. It is considered the paradigmatic case of the vehicle routing problem (VRP) and refers to the distribution of goods from a central depot to geographically scattered customers.

Since then, several factors influencing the problem have been introduced, such as the variance of vehicles' capacities, time-related restrictions, i.e., time windows set by customers, and the existence of multiple depots involved in the distribution. These and other characteristics and requirements are transformed either to constraints framing the problem or variables defining the problem. This challenge creates a complex problem in which multiple criteria and limitations must be taken into account at the same time, including the requirements of each client. In any case, the variables and the constraints that originate from real-life cases that logistics companies face are transformed from researchers to VRP variants. Therefore, the VRP variants that are analyzed below are connected to real-life cases. The objective, in most cases, both in practice and in theory, remains the same and is the minimization of the total distribution costs while the distribution services remain at a high level.

The distribution cost constitutes a big part of the final selling price of a product and consists of both fixed and variable costs. Therefore, companies need to reduce either or both of these costs to achieve higher product demand. The fixed costs mainly originate from the driver's wage or the vehicle usage costs and burden the distribution company just by using the vehicle, irrespectively of the route and the number of served customers. On the other hand, the variable costs mainly originate from the fuel costs or the traveling time of each route. As a result, variable costs are affected by the length and the duration of the route. The restrictions and parameters included in real-life routing problems related to the different VRP variants determine the routes, length, and duration. Therefore, it is primarily needed to define the VRP variants of a real-life case, the formulation of the problem, and the objective function that compose the total cost. All these factors will enable searching for the appropriate optimization method to address the problem most cost-effectively.

In the past, the entire process of vehicle routing and scheduling was complex for business executives to execute because it was almost exclusively based on maps and the person's empirical knowledge in charge. Today, with the use of computers, the process has become more accessible, using vehicle routing and scheduling software, although challenges continue to arise mainly due to the large volume of data needed to be managed and the increasing demands and requirements of the external environment. Of course, this does not mean that the

experience of those responsible for planning deliveries is questioned. The experience of market leaders is the basis on which every computer developer will rely to develop efficient algorithms integrated into routing software. Today, the demand and the challenges faced by logistics companies are more intense than ever. Therefore, it is becoming increasingly necessary to use advanced systems for the routing and execution of deliveries.

In this direction, it is vital for companies to adequately recognize the many variants and parameters that influence their daily operations and algorithms. On this premise, the thesis's main objective is to first define the most common VRP variants in logistics distribution and then present the algorithms used for solving the specific variants. The algorithms are necessary for logistics companies in today's challenging and constantly changing environment. The number of customers and their needs and requirements is increasing, and optimization algorithms are a key component for effective customer service and efficient operations.

This Chapter recognizes the trends characterizing the VRP variants, the algorithms proposed, and their correlation. The different VRP variants are classified into sixteen (16) categories representing the most real-life cases in logistics. Some of the variants with similar characteristics are classified and discussed together in one of the 16 categories. Next, the algorithms solving the VRP variants are presented using a systematic literature review, and the relations between VRP variants and the various algorithms are discussed. Additionally, while other literature reviews of VRP algorithms categorize them just in the high-level: exact, heuristic, and metaheuristics, in the current research the analysis extends to lower levels of algorithms' categorization. Marinakis and Migdalas (2007) were the first to conduct qualitative research of the VRP variants and the algorithms solving them, being a good starting point for the current research. These VRP variants were further enhanced by the other variants that have either presented in the publications of Eksioglu et al. (2009), Lahyani et al. (2015), and Braekers et al. (2016) either have been proposed the last few years and are considered significant in the VRP.

Finally, besides focusing on the VRP variants and the algorithms that solve them, the current Chapter also presents studies and papers that lead to practical applications (software packages and information systems) in vehicle routing. It is vital to estimate how much of the proposed and developed algorithms are employed in information systems and software packages.

The remaining part of this Chapter is organized as follows: Section 2.2 presents the main VRP variants for logistics and their classification applied in the literature review. Section 2.3 classifies the identified algorithms; Section 2.4 presents the research methodology followed for selecting the papers and Section 2.5 the quantitative results. In Section 2.6, some of the most significant publications found in the review process are further discussed concerning the correlation between VRP and applied algorithms to solve distribution cases. Finally, in Section 2.7, the research focusing on routing applications is presented, while in Section 2.8 the concluding remarks are presented.

2.2 The Vehicle Routing Problem and its Variants

In this section, the variants that most logistics and distribution companies face in their daily operations, regarding the routing of vehicles and scheduling of deliveries, and the way the variants are related to real-life cases, are analyzed. Firstly, the capacity of the vehicles constitutes a key factor of the vehicle routing problem, as it is the most studied factor for real-life distribution cases (Kim et al. 2015; Mańdziuk and Nejman 2015) and the one that researchers and practitioners first considered. Capacity forms two variants of the VRP: 1) the Capacitated VRP (CVRP) in which all vehicles are identical and have the same capacity and 2) the heterogeneous fleet VRP (HVRP), in which multiple types of vehicles exist, each of which is defined by a different capacity, fixed and variable costs (Prins 2009). In actual practice, few companies have a fleet of identical vehicles. Instead, in their effort to serve customers cost-effectively, companies use different types of vehicles. Small-sized vehicles mainly serve customers in city centers (last-mile distribution). In contrast, bigger-sized vehicles primarily serve customers and retailers separated by more considerable distances and require larger orders in terms of volume and weight. In both cases, when formulating the problem, it is assumed that vehicles will return to the central depot after the completion of their route. However, it is expected, especially for Third-Party Logistics (3PL) companies that seek to reduce their fixed costs, to cooperate with transportation companies that operate a fleet of vehicles at their disposal. In such cases, partners' vehicles do not necessarily return to the warehouse after completing the route. This case is recognized as the Open VRP (OVRP) (Zachariadis and Kiranoudis 2010) in the research community.

Regardless of whether the fleet of vehicles is heterogeneous or homogeneous, some researchers, in the last decade, consider either two-dimensional (2D-VRP) either three-dimensional (3D-VRP) loading constraints for ensuring that the distributed items can be feasibly loaded into the vehicles (Zachariadis et al. 2016). When considering these constraints, researchers most commonly split the problem into different problems, including packing and routing. In the present thesis, the 2D and 3D loading constraints are considered as a single variant of the problem, the 3D-VRP, focusing only on the algorithms proposed for the routing of vehicles. Furthermore, another variant related to the loading capacity is the Truck and Trailer VRP (TTVRP). Some customers may be serviced by a truck pulling a trailer in the specific case, while others only by a single truck (Lin et al. 2010). It is related mainly to deliveries and collections in city centers and rural areas with accessibility restrictions (Usberti et al. 2013).

In other cases, traffic congestion, access restrictions, and environmental regulations in cities force companies to use vehicles with lower capacities (Quak and de Koster 2009; Sitek 2014; Perboli and Rosano 2019). In this case, the vehicles can visit only a few customers during the trip. However, they can execute more than one trip during the working day, leading to the Multi-Trip VRP (MTVRP) (Brandão and Mercer 1998). To optimize the procedure and execute the maximum number of trips during the drivers' shifts, satellite facilities (VRPSF) are

commonly used to replenish the vehicles. In this context, the Two-Echelon VRP is applied, and the delivery of goods from the depot to customers is managed by routing vehicles through satellite facilities (Grangier et al. 2016). Therefore, there are two routing procedures: vehicles starting from the depot, delivering to satellites, and returning to the depot, and a second one from satellites to customers and back to the satellites (Crainic et al. 2010). When more than two distribution layers are considered, the variant is called Multi-Echelon VRP (MEVRP). Later in this Chapter, VRPSF, 2EVRP, and MEVRP are classified into one category, the MEVRP, as these variations have many common characteristics.

The requirements of the customers determine to a great extent, the setting of the VRP parameters. One of the most common variants of the problem is the VRP with time windows (VRPTW), where each customer determines a time interval in which the order must be delivered. If the vehicle arrives before the start of the time window, it must wait until it opens, while it cannot arrive after the end of the time window (César and Oliveira 2010). In cases such as this, the time windows are characterized as hard. Another case is the time windows to be soft, meaning that the vehicle may arrive after the end of the time window, but at an additional cost. This variant is significant in logistics since most end-customers determine a time window for the goods to be delivered. It has been inextricably linked to the accuracy of deliveries and customer satisfaction. However, good interpersonal relationships can also improve the quality of services provided. In the last decade, a new variant of the VRP has emerged, the consistent (ConVRP), in which trust between distribution companies and customers becomes a priority. Therefore, companies construct consistent routes over a period of time, which reduces the number of drivers each customer has to deal with (Feillet et al. 2014), and consequently, the friction between them.

It is also essential to highlight that logistics operations do not end at the phase of delivering goods, as the phenomenon of customers returning products is common in practice. Both the VRP with backhauls (VRPB) and the VRP with simultaneous pickups and deliveries (VRPSPD) study the case of deliveries and pickups during the execution of the routes. In VRPSPD, the goods start from the central depot and are delivered to customers, while pickups are simultaneously loaded to vehicles before returning to the depot (Montané et al. 2006). In every phase, both the delivery and the pickup loads must be taken into account, as the vehicle's total capacity cannot be exceeded. On the other hand, VRPB, which also involves pickups and deliveries, has an additional limitation, simplifying the problem. All pickup items are collected after the deliveries in every route (Goetschalckx and Jacobs - Blecha 1989). These two variants are considered as a single one, under the name VRPPD (VRP with pickups and deliveries) in the classification of the literature, as both of them involve pickups and deliveries.

Environmental pollution has forced governments to set environmental regulations to reduce noise, traffic, CO_2 emissions and consequently improve the quality of life for citizens. The fact that a big part of CO_2 emissions originates from road freight transportation could not leave transportation companies unaffected. Therefore, along with minimizing transportation costs,

companies need to minimize CO_2 emissions, a problem that is identified in the literature as the Green VRP (GVRP) (Lin et al. 2014) or as the Pollution Routing Problem (PRP). The PRP is somehow more difficult to solve as the emissions depend not only on the vehicle's speed but also on the vehicle's load in every step of the distribution process. Bektaş and Laporte (2011) were the first that studied the specific variant of the VRP.

Additionally, electric vehicles and hybrid vehicles that can operate both electrically and with traditional fuel contribute to minimizing pollution and CO_2 emissions. These versions of the problem are known as Electric VRP (EVRP), and Hybrid VRP (HVRP), respectively and have lately attracted companies' and researchers' attention (Mancini 2017). These four VRP variants (GVRP, PRP, EVRP, and HVRP) are considered as one category, under the name GVRP, in the classification of the literature.

The accuracy of deliveries depends to a great extent on the travel time between delivery points, pick up points, and depots. This problem has been tackled via the use of a function, in which the departure time is the independent variable and is known as the Time-Dependent VRP (TDVRP) (Figliozzi 2012). This attribute is crucial for better predicting traffic congestion and travel time between nodes and checking the feasibility of routes. The initial routing and scheduling, where delivery and pickup orders are known before starting the routes, can become much more reliable through this variant. However, orders and unforeseen events may appear dynamically during the execution of the route (Flatberg et al. 2007). In this case, changes in the scheduling of deliveries are made to satisfy new customer orders and to avoid delays caused by unforeseen events. The dynamic and stochastic version of the problem (known as Dynamic VRP - DVRP and Stochastic VRP - SVRP), must be reinforced by real-time communication between the vehicles and the planners of the transportation companies to face each case effectively (Pillac et al. 2013). Nowadays, the need for dynamic routing also arises from traffic congestion, which usually affects a stochastic way of implementing deliveries (Yu and Yang 2019).

Another variable that is highly dependent on the number of distribution centers that collaborate for the distribution of goods is the Multi-Depot VRP (MDVRP). If a single company manages multiple distribution centers, customers are most commonly assigned to the nearest warehouse, which contains the goods each customer ordered (Renaud et al. 1996). Therefore, the MDVRP can be treated as a series of multiple single depot problems, simplifying the initial one. With this assumption, the problem can be addressed adequately, especially when new customers appear every day, leading to non-fixed routes. However, there are cases that this simplification cannot be made, such as in a collaborative distribution network (Collaborative VRP), where a group of companies cooperates to minimize the operating and distribution costs. The Collaborative VRP is usually considered an extension of the MDVRP, which can improve the load factor of vehicles, reduce crisscross transportation and enhance the efficiency of logistics network operation (Wang et al. 2017). Therefore, the Collaborative and the Multi-Depot VRP are united in a single category in the classification under the name MDVRP.

Finally, two variants of the VRP that are less common among transportation companies are (i) the Split Delivery VRP (SDVRP) and (ii) the Periodic VRP. In the first case, the constrain that each customer must be visited exactly once and by one vehicle is relaxed, and the customers' demand is allowed to be split (Silva et al. 2015) among the available vehicles. The specific relaxation may prove to be advantageous in some distribution environments, such as when the mean customer demand is a little over than the vehicle's capacity (Archetti et al. 2008), offering increased load factors of vehicles and reduced number of delivery routes, but at the expense of increased traveled distance. As for the PVRP, the routes are constructed over a planning horizon (Coene et al. 2010), which may be days or even weeks. This model is essential for logistics companies that manage fixed orders every day, needing an optimal plan for that period.

Indeed, in most cases, both in practice and in theory, multiple constraints and challenging features are considered simultaneously so that the problem addressed reflects most in real-life routing cases. Some representative real-life constraints that are considered in such cases, according to Rabbouch et al. (2019) and Caceres-Cruz et al. (2014), are the capacity and the number of vehicles, ready and due times for serving each customer, the heterogeneous fleet of vehicles, and the different warehouses. These constraints are associated with some of the VRP variants that have already been analyzed, such as the Heterogeneous Fleet, the Multi Depot, the Pickup and Deliveries, the Open, and the Time Windows. More specifically, Penna et al. (2019) manage to tackle all the above VRP variants at the same time, while Belmecheri et al. (2013) and Belloso et al. (2019) address the heterogeneous fleet VRP with backhauls and time windows. In the last few years, the consideration of multiple constraints (and multiple VRP variants) at the same time is defined by researchers as the Rich VRP (Lahyani et al. 2015). The researchers, who study the Rich VRP, use algorithms commonly dealing with the containing variants to simplify the problem. Since the current research studies the correlation between the VRP variants and algorithms, Rich VRP has been incorporated in the search term to include all the related variants discussed in these studies. Still, it is not classified as an individual variant of the problem as it refers to other VRP variants.

Over the years, researchers have studied more and more variations of the VRP to address cases that logistics companies face in their daily operations. The VRP variants analyzed in this section reflect most of the challenges logistics companies face in freight distribution. This is enhanced by the cooperating and software development companies operating in the logistics field that confirm that these variants are the most commonly addressed.

The existence of multiple variables and constraints forces logistics companies to find ways to optimize their operations and minimize their costs. Information Systems can strengthen this approach by incorporating in their functionality the use of algorithms. Multiple algorithms have been developed and proposed to address the VRP. Initially, exact algorithms were proposed due to the high quality of the solutions produced. Still, in large-scale problems (more than 100 customers), the computation time needed for an exact algorithm to find the optimum solution

increases so much that it ends up being impractical. Consequently, heuristic and metaheuristic algorithms were developed, offering a better balance between solution quality and computation time.

2.3 Solution Methods

In the current section, the algorithms which effectively deal with a set of VRP variants are studied. The selected variants are the most common ones faced by logistics companies in distribution, and their characteristics and constraints meet the requirements of freight transportation. Therefore, the taxonomy focuses on two key characteristics, namely the VRP variants, and the algorithms. The literature review deals with the algorithms addressing the selected variants of the VRP and is carried out to determine the research community's trends for both cases.

More specifically, the classification of the algorithms based on the research of Labadie et al. (2016) and Lin et al. (2014) is shown in Figure 2.1. Exact algorithms mainly include Lagrange Relaxation methods and Column Generation but are not further analyzed due to the limited research interest. On the other hand, heuristic algorithms are separated into three main categories, Construction, Two-Phase, and Local Improvement heuristics, each containing specific algorithms, as shown in Figure 2.1. The interest of the research is mainly focused on this level of analysis, even if each category of heuristics contains specific algorithms. Other literature reviews, such as those of Eksioglu et al. (2009), Lahyani et al. (2015), and Braekers et al. (2016), either do not focus on algorithms, either do not present this level of analysis. They only emphasize on the three main categories of algorithms, exact, heuristics, and metaheuristics, while in the current thesis the analysis is extended to a more detailed level. Consequently, the three main categories of algorithms are further analyzed.

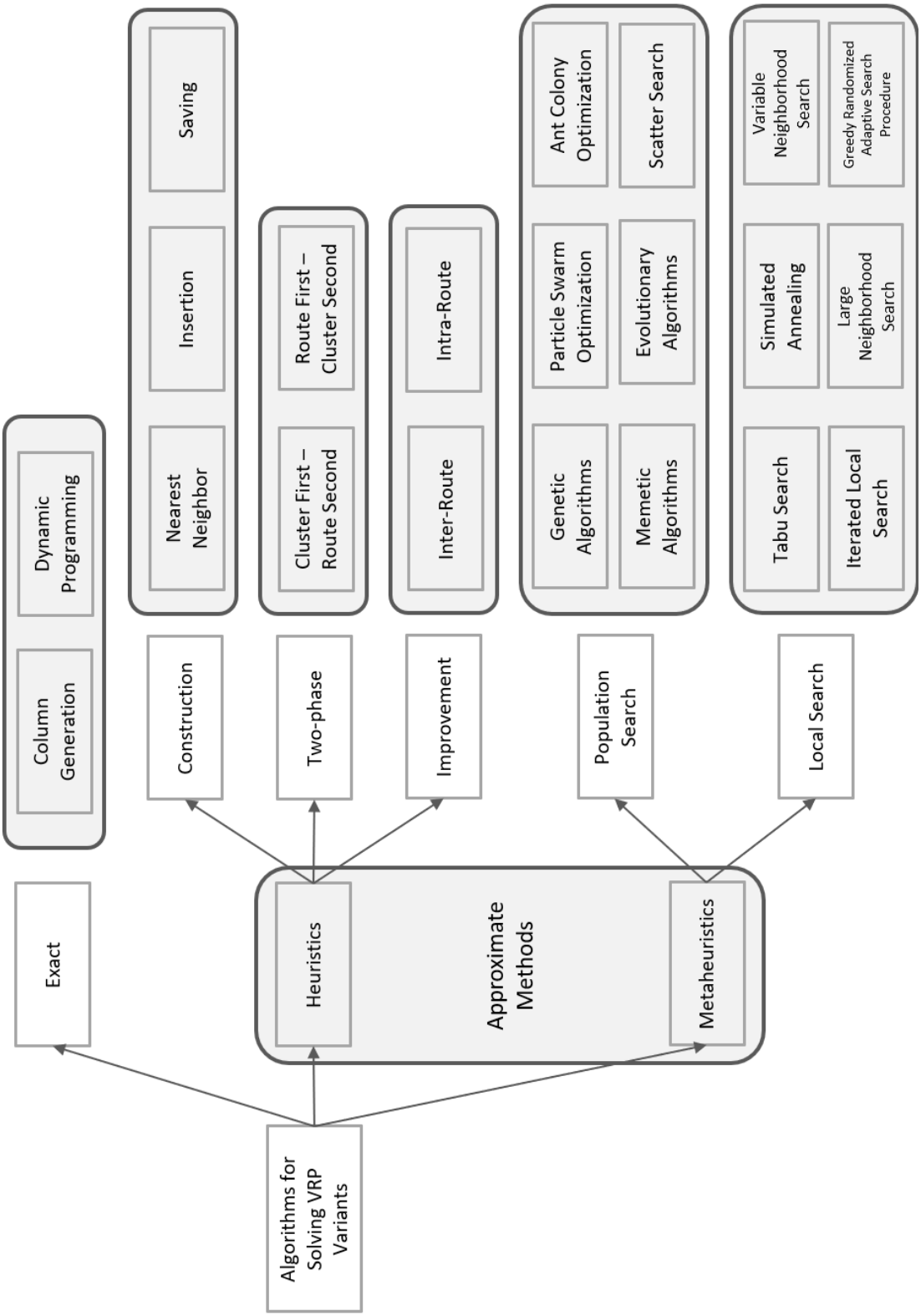


Figure 2.1 Classification of Algorithms for the VRP

2.3.1 Exact Approaches

Exact algorithms compute every possible solution until the best is reached, performing very poorly in computation time, even to fairly small instances (El-Sherbeny 2010). Exact methods are classified into three categories: Lagrange relaxation-based methods, column generation and dynamic programming. The Lagrange relaxation method penalizes violations of inequality constraints using a Lagrange multiplier, implying a cost on violations. Aggarwal et al. (2017) applied this method, while in the work of Fisher (1994), Holland (1975) and Kohl and Madsen (1997), a K-tree approach is followed by Lagrange relaxation. Desrochers et al. (1992) applied to the VRPTW a column generation method to solve up to 100-customer problems. A review of the exact methods up to 2012 is reported in Baldacci et al. (2012 and El-Sherbeny (2010).

2.3.2 Heuristic Algorithms

Distribution companies facing the VRPTW, need methods capable of producing high-quality solutions in limited computation time. Therefore, multiple heuristic algorithms have been proposed, divided into construction and local search heuristics. Route construction heuristics are separated into sequential and parallel methods. In the first method, nodes are selected sequentially until a feasible solution is constructed regarding vehicle capacity and time windows. On the other hand, parallel methods build several solutions simultaneously (Bräysy and Gendreau 2005). In the heuristic of Clarke and Wright (1964), called "Savings", every customer is supplied by a dedicated vehicle. Combining two end customers, i and j, results in a cost-saving of . However, when joining two close customers, in terms of distance, but far apart in time, waiting times are increased.

The time-oriented nearest neighbor method, enriched in the present Chapter, constructs routes sequentially by finding the "closest" customer to the last customer served and ends when no more unrouted customers left (Solomon 1987). Insertion heuristics are also tour building heuristics, which involve the criteria of and for inserting a new customer u, between two customers i and j on the route. The sweep algorithm, introduced by Gillet et al. (1979), divides the problem into two sub-problems, one of assigning locations to routes and one of minimizing the length of each route, and has been enhanced by Solomon (1987) and Potvin and Rousseau (1993).

Local search methods, on the contrary, modify the initial solution for improved neighboring solutions. The improvement of the solution can be within a route, called intra-route, or between different routes, called inter-route. The 2-opt, 3-opt, 2H-opt, and Or-opt are some intra-route procedures, while the λ-interchange (Osman 1993), relocate, exchange (Gendreau et al. 1992), ejection chain are some inter-route procedures. Through these procedures, either the decrease of the number of vehicles or the total traveled distance is attempted, either both.

Finally, heuristic algorithms are mainly applied for constructing the initial population in metaheuristic algorithms and implemented in routing and scheduling software. Their simplicity

and the integration of various variants and parameters have made them popular and capable of solving complex real-life cases.

2.3.3 Metaheuristic Algorithms

In contrast to heuristics, metaheuristics intend to escape from the local optimum as they explore a more significant subset of the solution space. Metaheuristic algorithms are more advanced procedures and contain Population Search and Local Search methods. These two subcategories of metaheuristics are further analyzed in sections 2.3.3.1 and 2.3.3.2.

2.3.3.1 Population Search Metaheuristics

Population-based metaheuristics aim at generating a new solution from a set of solutions, either by combining and pairing existing ones or by making them cooperate through a learning process. Genetic algorithms (GA's) are the most common metaheuristic algorithms and are inspired by biological evolution. Initially, a population (parents) of solutions, known as chromosomes, is selected. Then in each iteration, two parents are selected from the population, according to their fitness function, and are combined through the crossover procedure to provide the offsprings (children). Consequently, the mutation procedure is applied to ensure the diversity of the population. Finally, the best solutions are selected to be used as parents in the next iteration.

Memetic algorithms (MA's) are an "advanced" version of GA's as for each child, a local improvement method is applied. Both algorithms belong in the greater category of Evolutionary Algorithms (EA's) (Labadie et al. 2016). On the contrary, Scatter Search (SS) needs a smaller set of solutions, containing both good and diversified solutions in which local improvement methods are applied. In order to ensure that solutions are different from each other, a distance metric needs to be defined. The SS method was introduced by Glover (1977), and since then, multiple researchers have proposed SS algorithm for the VRP. Belfiore and Fávero (2007) tackles the HVRPTW while Russell and Chiang (2006) the VRPTW solely.

The general category of swarm-based algorithms includes Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). In ACO, ants move in the graph, building paths according to probabilistic rules. The ACO algorithm is based on the behavior of ants using pheromone-based strategies and has been used for solving multiple VRP variants. Gambardella et al. (1999) developed such an algorithm for the VRPTW, producing efficient results, while some of the best ACO algorithms that have been developed for addressing the VRP variants are those of Bullnheimer et al. (1999) and Kalayci and Kaya (2016). Additionally, in PSO a population of "particles" exists, and each one moves from its position to another in the search space. The move of each particle is influenced both by the best position the specific particle has achieved and the best solution of all particles (Eberhart and Kennedy 1995).

2.3.3.2 Local Search Metaheuristics

On the other hand, Local Search metaheuristics aim at exploring the solution space by moving the current solution to another promising one in the neighborhood (Mekamcha et al. 2019). Tabu Search (TS) is one of the most frequently applied algorithms and depends on the principle of continuing the search even if a local optimum has been reached. Therefore, a move is applied in the current solution even if the objective deteriorates. However, it should be noted that a previously visited solution cannot be revisited as the tabu list records the recent searches. The tabu search (TS) algorithm was introduced by Glover (1986), and since then, multiple TS algorithms have been proposed, like those of Badeau et al. (1997), Chiang and Russell (1997), Cordeau et al. (2001), and Taillard et al. (1997).

Another algorithm, the Large Neighborhood Search (LNS), explores a wide search space. In the specific algorithm, the initial solution is partially destroyed according to the destroy operators and finally repaired, according to the reinsertion operators. The Adaptive LNS method is similar, except that the most efficient operators subject to change during the search are applied in each iteration (Hornstra et al. 2020).

Another metaheuristic, the Guided Local Search (GLS), operates by assigning penalties in the objective function so that the search space becomes more expansive and avoid trapping in a local optimum (Kilby et al. 1999; Zhong and Cole 2005; Arnold and Sörensen 2019a). This is because the search will be transported to a new area of the solution space. Simulated Annealing (SA) also belongs in LS metaheuristics and was proposed by Kirkpatrick et al. (1983). This method is inspired by the behavior of atoms and solids when the temperature changes. Concerning its application to the VRP, the algorithm makes repeatedly small local changes in an initial solution until no move can offer improvements in the objective function. In some cases, a move that worsens the objective function may be allowed for escaping from a local optimum (Chiang and Russell 1996; Kuo 2010).

As for the Variable Neighborhood Search (VNS), which Mladenović and Hansen (1997) proposed, it is based on the fact that a systematic change of neighborhood is permitted when a local search algorithm makes no improvement. Moreover, the Greedy Randomized Adaptive Search Procedure (GRASP) is another LS metaheuristics applied in the VRP but less often compared to others. In the specific method, in each iteration, an initial solution needs to be obtained by a greedy randomized heuristic. Consequently, the GRASP algorithm, which is a multi-start metaheuristic, aims to improve the solution by applying local search procedures. The best solution obtained from all iterations constitutes the best-obtained solution. Finally, the Iterated Local Search (ILS) algorithm is based on creating a sequence of a locally optimal solution by making a modification, named perturbation, to the current local optimum, as well as by performing local search procedures after starting from the modified solution (Subramanian et al. 2010; Silva et al. 2014).

Finally, in their effort to optimally solve the VRP, researchers combine algorithms to exploit the advantages of each procedure. Especially in the last few years, hybrid algorithms that combine metaheuristics are developed to achieve better results (Kaboudani et al. 2018; Rajabi-Bahaabadi et al. 2019). This is an attribute studied in the literature review to determine which algorithms are used in solving multiple VRP variants.

2.4 Literature Review Methodology

The VRP is a well-studied field as its variants deal with real-life cases and problems that most distribution and logistics companies face. Researchers have formed multiple variants to cover all cases and improve the routing of vehicles and scheduling of deliveries. The external environment in which logistics companies work, their size, their customers, and their partners determine the problems and variants that must be considered to a great extent. In Section 2.2, the VRP variants related to freight transportation and distribution are thoroughly described to justify how these variants are related to real-life cases. More specifically, selecting the VRP variants presented in this section has resulted mainly from the annotated bibliography (Marinakis and Migdalas 2007). The authors considered and analyzed the Capacitated VRP, the VRP with Time Windows, the VRP with pickup and deliveries, the Multi-Depot VRP, the Stochastic and Dynamic VRP, the Heterogeneous Fleet VRP, the Periodic VRP, and the Open VRP. The Split Delivery, the Time-dependent, the Green, the Truck and Trailer, and the Multi-trip VRP are among the variants that emerged from the articles of Eksioglu et al. (2009), Lahyani et al. (2015), Braekers et al. (2016), and Elshaer and Awad (2020) which are review papers. As for the VRP with 3D loading constraints, the Multi-echelon VRP and the Consistent VRP have emerged independently as they seem to have attracted the interest of researchers the last few years. Through this way, the VRP variants that fit most in the research on freight transportation are included. However, apart from the VRP variants, emphasis must be given to the algorithms developed and proposed for solving those problems. Distribution companies that consider multiple VRP variants simultaneously cannot effectively face all variants without the contribution of algorithms. The integration of an algorithm into the system follows its testing and evaluation process through either benchmark instances presented in the literature or real-life cases.

As there is no previous research correlating VRP variants and algorithms solving them, the need for such research has emerged. Therefore, a literature review process adopted from the research of Eksioglu et al. (2009), Braekers et al. (2016), and Gayialis et al. (2019) is followed. The research protocol is well defined as it aims at an efficient and well-recorded review, including multiple VRP variants. The main goal of the present Chapter is to identify the trends of VRP variants and the applied algorithms offered to tackle them and their correlation. Additionally, papers that are considered significant and pioneers in the research community are presented. The papers having the most citations were considered significant, and they were further discussed in this review.

Scopus database (https://www.scopus.com) was used for an advanced search, as it covers a wide range of peer-reviewed academic publications. Only articles published in journals, in the English language and the last decade, are considered. Initially, only publications containing at least one of the previously defined VRP variants, the term "algorithms", as well as, at least one of the terms "benchmark instances", “tested”, “test”, simulate”, “simulation", “validate” and “validation” in their title, abstract, or keywords are selected. Through this procedure, it is ensured that the resulted publications propose an algorithm and test and validate the algorithm either to benchmark datasets, either to real-life cases, making them more reliable. Also, the fact that the VRP has been intensively studied while this research concentrates on freight transportation lead to the decision not to include specific terms. Such terms are "waste", “cash”, “hazardous”, "concrete", "healthcare", “care”, “blood”, "bus" and "transportation of people", as they cannot be considered commodities and they have unique transportation condition which forms another field of study.

Furthermore, the terms “bike”, “bicycle”, "flying robots" and "UAV” were excluded as they refer to different means of transportation than vans and tracks. The terms "location routing" and "inventory routing" were also excluded from the research as it was decided to exclude any combined problems. Also, this research is limited to the Subject Areas of “Computer Science”, “Engineering”, “Mathematics”, “Decision Sciences” and “Business, Management, and Accounting”. The research term includes only specific journals, selected according to the Scimago Journal Rank (https://www.scimagojr.com) to reduce the number of papers analyzed, focusing on the most prestigious journals. More specifically, only journals that belong in the first quartile (Q1) are considered and contained in the research protocol. The exact search term applied to the Scopus database is presented in Table 2.1.

Table 2.1 Search Term Applied

Literature Review Search Term
TITLE-ABS-KEY ((("Time windows" OR "VRPTW") OR ("Multitrip" OR "Multi-trip" OR "MTVRP") OR (“Satellite Facilities” OR “Echelon”) OR (("Hybrid" AND "VRP") OR ("Green" AND "VRP")) OR ("Time-dependent VRP" OR "Time dependent VRP" OR "TDVRP") OR ("Pollution routing" OR “PRP”) OR (“Electric VRP”) OR ("Capacitated VRP" OR "CVRP") OR ("VRP" AND ("two-dimensional" OR "three-dimensional" OR "3D loading constraints" OR "2D loading constraints")) OR “Truck and Trailer” OR ("Dynamic VRP" OR "DVRP") OR ("Simultaneous Pickups and Deliveries" OR "VRPSPD" OR "Backhauls" OR "VRPB") OR (“Rich VRP”) OR ("Heterogeneous Fleet" OR "Heterogeneous" OR "HFVRP") OR ("Multi depot" OR "Multi-depot" OR "MDVRP") OR ("Open VRP" OR "OVRP") OR (“Split Delivery VRP” OR “SDVRP”) OR ((“Periodic” AND “VRP”) OR “PVRP”) OR ((“Stochastic VRP”) OR “SVRP”) OR (“Collaborative VRP”) OR ((“Consistent” AND “VRP”) OR “ConVRP”) OR (“VRP with workload balance”) OR (“site-dependent” AND “VRP”)) AND ("Vehicle routing" AND

("Benchmark Instances" OR "Benchmark" OR “validate” OR “validation” OR “test” OR “tested” OR “simulate” OR “simulation” OR “case study”) AND ("Algorithm")) AND NOT ("Waste" OR “cash” OR “hazardous” OR "Concrete" OR "Healthcare" OR "Cold" OR "Care" OR "Location-routing" OR "Location routing" OR "UAV" OR "Transportation of people" OR "Flying robots" OR "Bus" OR "Asset localization" OR "Inventory routing" OR "Blood" OR "Bike" OR "Bicycle")) AND (LIMIT-TO (SRCTYPE, "j")) AND (LIMIT-TO (DOCTYPE, "ar")) AND (LIMIT-TO (SUBJAREA, "COMP") OR LIMIT-TO (SUBJAREA, "MATH") OR LIMIT-TO (SUBJAREA, "ENGI") OR LIMIT-TO (SUBJAREA, "DECI") OR LIMIT-TO (SUBJAREA, "BUSI")) AND ((PUBYEAR > 2009)) AND (LIMIT-TO (EXACTSRCTITLE, "Computers And Operations Research") OR LIMIT-TO (EXACTSRCTITLE, "European Journal Of Operational Research") OR LIMIT-TO (EXACTSRCTITLE, "Expert Systems With Applications") OR LIMIT-TO (EXACTSRCTITLE, "Applied Soft Computing Journal") OR LIMIT-TO (EXACTSRCTITLE, "Computers And Industrial Engineering") OR LIMIT-TO(EXACTSRCTITLE, "Information Sciences") OR LIMIT-TO (EXACTSRCTITLE, "International Transactions In Operational Research") OR LIMIT-TO (EXACTSRCTITLE, "Soft Computing") OR LIMIT-TO (EXACTSRCTITLE, "Annals Of Operations Research") OR LIMIT-TO (EXACTSRCTITLE, "Journal Of The Operational Research Society") OR LIMIT-TO (EXACTSRCTITLE, "Transportation Research Part C Emerging Technologies") OR LIMIT-TO (EXACTSRCTITLE, "Transportation Science") OR LIMIT-TO (EXACTSRCTITLE, "Applied Mathematical Modelling") OR LIMIT-TO (EXACTSRCTITLE, "Engineering Optimization") OR LIMIT-TO (EXACTSRCTITLE , "Operations Research") OR LIMIT-TO (EXACTSRCTITLE, "Transportation Research Part B Methodological") OR LIMIT-TO (EXACTSRCTITLE, "Transportation Research Part E Logistics And Transportation Review") OR LIMIT-TO (EXACTSRCTITLE , "Journal Of Advanced Transportation") OR LIMIT-TO (EXACTSRCTITLE, "Swarm And Evolutionary Computation") OR LIMIT-TO (EXACTSRCTITLE, "IEEE Transactions On Automation Science And Engineering") OR LIMIT-TO (EXACTSRCTITLE, "International Journal Of Advanced Manufacturing Technology") OR LIMIT-TO (EXACTSRCTITLE, "International Journal Of Production Economics") OR LIMIT-TO (EXACTSRCTITLE, "Journal Of Intelligent Manufacturing") OR LIMIT-TO (EXACTSRCTITLE, "Optimization Letters") OR LIMIT-TO (EXACTSRCTITLE, "International Journal Of Industrial Engineering Computations") OR LIMIT-TO (EXACTSRCTITLE, "Neurocomputing") OR LIMIT-TO (EXACTSRCTITLE, "Advances In Production Engineering And Management") OR LIMIT-TO (EXACTSRCTITLE, "Flexible Services And Manufacturing Journal") OR LIMIT-TO (EXACTSRCTITLE, "IEEE Access") OR LIMIT-TO (EXACTSRCTITLE, "IEEE Computational Intelligence Magazine") OR LIMIT-TO (EXACTSRCTITLE, "IEEE Systems Journal") OR LIMIT-TO (EXACTSRCTITLE, "IEEE Transactions On Systems

Man And Cybernetics Part C Applications And Reviews") OR LIMIT-TO (EXACTSRCTITLE, "Informs Journal On Computing") OR LIMIT-TO (EXACTSRCTITLE, "International Journal Of Bio Inspired Computation") OR LIMIT-TO (EXACTSRCTITLE, "International Journal Of Geographical Information Science") OR LIMIT-TO (EXACTSRCTITLE, "International Journal Of Shipping And Transport Logistics") OR LIMIT-TO (EXACTSRCTITLE, "International Journal Of Sustainable Transportation"))

This specific research strategy resulted in 334 publications, each of which was studied thoroughly to determine its relevance. The fact that the VRP is well studied in the literature has led to its connection with other research topics. Typical examples are the dial-a-ride and the team orienteering problems that do not refer to freight transportation. Also, the single-vehicle case, which is known as the Travelling Salesman Problem (TSP), was decided to be excluded from the research. In this research study, it is not the intention to review the complete TSP literature or research topics related to the VRP; instead, it focuses on the VRP and the variants described in Section 2.2. Following the described process and as seen in the next section, 263 articles were categorized according to the classification of the proposed algorithms. In this way, an established statistical analysis is presented.

2.5 Research Results Analysis

In this subsection, the numerical results that have emerged from the categorization of the relevant papers are presented. The detailed classification results of the 263 articles are given in Appendix A of this thesis, through a link from which an excel file is downloaded. These papers concluded from the methodological approach presented in Section 2.4. The classification of the articles has been separated into two strands, the VRP variants and the applied algorithms for solving these variants. The present research focuses on published articles from 2010 until the first quarter of 2020 to identify the last decade's trends. The results of the research indicate that there is a constant interest in solving the selected VRP variants, as about 24 papers are published every year. Most publications come from the journals “Computers & Operations Research”, “European Journal of Operational Research”, “Expert Systems with Applications”, “Applied Soft Computing Journal” and “Computers & Industrial Engineering”. Over the last decade, these journals have published at least 20 articles that meet the requirements in the specific research approach.

Table 2.2 presents the number of publications that deal with each one of the selected VRP variants. In their effort to address real-life problems, most papers examine more than one variant of the problem. This is the reason why the cumulative relative percentage is more than 100%. Capacitated VRP is, by far, the most studied variant (82.89%) as it is the simplest variant and, therefore, can be easily combined with others. Papers that do not consider the CVRP deal

instead with the Heterogeneous Fleet VRP (17.11%). Variants that use time windows are significantly studied as they adequately reflect real-life cases and thus appear in almost half of the problems (46.39%). Respectively, the VRPPD is appeared in one-fifth of the reviewed papers due to its high correlation with real-life distribution cases. The GVRP seems to attract the attention of researchers during the last 6 years due to the current need for minimizing CO_2 emissions and fuel consumption and the use of electric vehicles. The industrial partners and software development companies confirmed that these VRP variants are the most commonly addressed variables and constraints in the distribution process.

Since many companies have multiple warehouses that need to be considered for the distribution process, as well as, that more often in today's competitive environment, companies collaborate and cooperate in the distribution process for reducing operating cost, made the MDVRP and the Collaborative VRP among the most studied variants of the VRP. Also, compared to other classification and reviews, like those of Eksioglu et al. (2009) and Braekers et al. (2016), which are among the most analyzed, there are no data relative to the MEVRP. This variant has gained the interest of researchers and practitioners in the last few years due to the need of companies to optimize the routing of vehicles in every stage of the supply chain.

Also, the VRP with two and three-dimensional loading constraints present an increased interest since 2013. This variant has been favored by the evolution of computer science and has given researchers the opportunity to consider simultaneously two complex problems, the routing of vehicles and the packing. Respectively, the DVRP and the TDVRP have taken advantage of new technologies, which offer real-time and big data. Both can significantly enhance the efforts of planners not only to create an initial and reliable plan of routes but also to make better decisions when needed in less time. On the other hand, variants such as the OVRP, the SDVRP, and the MTVRP, seem to have a relatively low but constant interest from researchers over the last decade due to their correlation with real-life logistics cases that do not frequently occur.

Finally, some variants of the VRP seem to receive the least attention from the research community. More specifically, the Periodic VRP and the ConVRP have relative percentages of less than 1.95 % over the years. That is mainly because few companies can plan their routes overtime in today's dynamic environment, as well as concerning the ConVRP. It is a relatively new variant of the VRP. Also, the TTRP is appeared in only 4 articles out of 263 extracted, and though trailers are used in some cases, such as in milk collection, it is a commonly neglected variant of the VRP in the last decade.

Table 2.2 VRP Variants over Years

VRP Variants	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	Total	Relative Percentage
CVRP	18	11	12	15	19	34	24	24	18	31	12	218	82.89%
VRPTW	7	5	7	7	10	16	12	12	13	22	11	122	46.39%
VRPPD	3	1	3	6	4	10	6	5	3	8	4	53	20.15%
HVRP	3	2	1	3	7	4	6	2	4	11	2	45	17.11%
MDVRP	1	1	2	3	4	5	3	1	3	3	3	29	11.03%
GVRP				1	3	2	7	6	1	8	2	30	11.41%
DVRP	1	1	2		3	4	2	1	1	5	1	21	7.98%
3D-VRP				2	2	4	3		3	1	2	17	6.46%
OVRP	4	1		1	1	1	1		1	1	1	12	4.56%
SDVRP		1			2	1	2	1		3		10	3.80%
TDVRP		1	1	1			2	4		1		10	3.80%
MEVRP				2		1	1		1	4		9	3.42%
MTVRP	1	1				1	1	1		3	1	9	3.42%
PVRP		2								2	1	5	1.90%
ConVRP			1			1			1	1		4	1.52%
TTRP	2	1							1			4	1.52%

In Table 2.3, the amount of papers that propose or develop each one of the algorithms of the classification shown in Section 2.3, is also presented. It should be pointed out that hybrid algorithms were presented as a separate category, including algorithms explicitly stated as "hybrid" in the studied papers. Metaheuristic algorithms dominate proposed algorithms with 252 cases, followed by Classical Heuristics (122) and Hybrid algorithms (53). Exact algorithms (32) are developed less often due to their computational complexity and their limitation of applying successfully only to small-scale problems. It is an expected outcome that heuristic and metaheuristic algorithms are used more often than the rest, mainly because the VRP is an NP-Hard Problem. It should be noted that the sum of the number of publications does not equal to 263, which is the number of the studied papers because many of the examined papers either propose hybrid algorithms (53 articles) which by definition include more than one method or combine multiple algorithms.

In a more detailed analysis, construction algorithms seem to appear more frequently as they are applied to solve the problem and create the initial solution used by metaheuristic algorithms. Local improvement heuristics are also widely used due to their ease of implementation and their ability to improve solutions fast. The first metaheuristics proposed for the VRP, clearly surpassing other algorithms in terms of solution quality, were Simulated Annealing and Tabu Search, which researchers still study. However, GA's have attracted the interest of researchers in the last decade, as they are the second most frequently applied metaheuristic. In total, Evolutionary algorithms, which contain GA's and MA's, are developed and applied more often than swarm-based algorithms (ACO and PSO). As for the local search metaheuristics, researchers equally apply and develop these algorithms, except the GLS and the GRASP, which

are less studied. Metaheuristic algorithms can offer good quality solutions or even optimum results when tested on benchmark instances. Simultaneously, their computational speed improves as technological advances. These two factors guarantee the research interest in these algorithms in the future.

As already mentioned above, the VRP variants are rarely studied individually. In most cases, researchers tackle a combination of these variants simultaneously to a better approximate real-life case. The CVRP, the HVRP, and the VRPTW are among the most examined variants as shown in Table 2.1, combined with almost all the other variants. The only exception is the combination of the CVRP with HVRP, as these two variants present different and mutually exclusive initial parameters relating to the capacity of the vehicles used. The CVRP is combined most with other variants because all vehicles are identical, which simplifies the problem significantly. As for the rest variants, multiple combinations between the VRPTW, the VRPPD, the HVRP, the MDVRP, and the GVRP are observed and formed by researchers.

In Table 2.4, the correlation between VRP variants and their algorithms is being presented. Many papers suggest hybrid resolving algorithms for almost all variants of VRP, indicating a trend in the research community towards these algorithms. Construction and local improvement heuristics are involved in the solution of all VRP variants. As for the VRPTW, while TS and SA used to be among the most common applied algorithms, a new direction towards Evolutionary Algorithms (mainly GA's) seems to have formed. In cases that only the CVRP and the VRPTW are considered, the chromosome's structure in GA's contributes to the faster execution of the crossover and mutation operators and the storage of the population. More specifically, the chromosome is an array with length equal to the number of customers and contains all customers-nodes. The sequence of the elements-customers determines the order of visitations. In the last decade, the VRPTW is more frequently faced as multiobjective, and the simultaneous existence and development of multiobjective EA's leads in an increased number of articles solving the VRPTW with EA's.

Table 2.3 Applied Algorithms per Year

	Applied Method	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	Total
	Exact	2	2	1	5	2	2	2	4	4	6	2	32
Heuristics (122)	Construction	8	6	5	4	7	9	9	3	7	10	2	70
	Two-Phase		1				1	1			1		4
	Local Improvements	6	1	4	5	2	12	5	2	2	9		48
Metaheuristics (252)	Local Search (145)												
	Simulated Annealing (SA)	1			2	2	6	3	3	1	1	1	20
	Tabu Search (TS)	4	2	1	1	4	1	1	5	2	3	1	25
	Variable Neighborhood Search (VNS)	2	1	2	1	5	5	3	3	4	10	4	40
	Iterated Local Search (ILS)	2		1	1	2	6	1	1	1	3	2	20
	Adaptive Large Neighborhood Search (ALNS) and Large Neighborhood Search (LNS)	1	1	1		1	5	9	2	1	5	2	28
	Guided Local Search (GLS)	1											1
	Greedy randomized adaptive search procedure (GRASP)						2		1				3
	Other Local Search Algorithms (Other LSA's)	1	1	1			1		2	1		1	8
	Population Search (107)												
	Genetic Algorithms (GA)	3	1	2	3	4	6		3	4	5	2	33
	Memetic Algorithm (MA)	1					1	1		1	2	1	7
	Other Evolutionary Algorithms	2		1	1	1	4	2	1	3	2	1	18
	Ant Colony Optimization (ACO)	1	2	1	1	2	2	4	2	2	2	2	21
	Particle Swarm Optimization (PSO)		1	2	2		2	1	3	2	2		15
	Other Swarm Intelligence Algorithms (Other SIA's)					1	2	1	2	1	3	1	11
	Scatter Search (SS)	1				1							2
	Hybrid	2	1	3	4	4	11	5	4	7	8	4	53

Table 2.4 Correlation between VRP Variants and Algorithms

Variants	Exact	Heuristics			Local Search Metaheuristics								Population Search Metaheuristics						Hybrid
		Construction	Two-Phase	Local Improvement	SA	TS	VNS	ILS	ALNS and LNS	GLS	GRASP	Other LS	GA	MA	Other EA's	ACO	PSO	Other SIA's	SS
CVRP	26	60	3	42	16	20	31	15	23	1	3	8	27	5	17	18	13	9	2	44
VRPTW	15	29	1	15	9	16	13	3	14		1	2	22	5	14	12	7	6	1	32
VRPPD	7	16	2	11	6	7	11	6	3		1	2	5		4	4	4			10
HFVRP	6	10	1	6	4	5	9	5	5				6	2	1	3	2	2		9
MDVRP	3	12		5	2		4	7	1			1	9			2	2	1		11
GVRP	5	6	1	3	2	4	5	2	9				2			1		1		5
DVRP	1	8		3		1	1	1	2		1		3	1	2	4	3	1		1
2D-VRP		4		5	3	1	6	2	2						1			1		2
OVRP	2	2		2	1	1	2	4		1		1			1		1	1		2
SDVRP	3	5		3			1	1	1											
TDVRP		4		1			2		1				2			2	1	1		1
MEVRP	4	3		1			1		2			1	1	2				1		2
MTVRP	2	3		2			2	1	2				1	1						2
PVRP	2				1	1										2				
ConVRP		2		2		2	2		1											
TTRP	1	1	1	1	1		1													1

The HVRP is a significant variant for logistics and transportation companies, as most of them have a mixed fleet of vehicles. However, only 17.11% of articles consider the specific variant. The most common algorithms applied in the HVRP are i) GA's, ii) the VNS, iii) the ALNS and LNS algorithms, and iv) the ILS algorithm. The three local search algorithms are applied in a single solution leading to shorter computation times. Instead, EA's which are frequently applied in this problem, need to storage solutions (which vehicle is used in each route), leading to increased computation times. However, the evolution process can contribute to a better solution in a reasonable time depending on the population size and generations.

The Green, the Electric, the Hybrid VRP, and the Pollution Routing Problem are considered one category in the analysis. These variants have been proposed and appeared for the last fifteen years and are thoroughly studied by researchers. In these first attempts for solutions reducing CO_2 emissions and fuel consumption, ALNS and LNS algorithms are mostly applied due to their advantage of being applied in a single solution and their ease of implementation. More specifically, even simple "destroy" and "reinsertion" operators can be very effective in any class of the VRP and, therefore, to the GVRP and the PRP. Finally, the fact that in this class of VRP variants, the fleet is mixed, leads to the conclusion that algorithms that can be efficient in the HVRP can also be in the Green, the hybrid, and the Electric VRP, as the vehicles' capacities and costs differentiate in most of these variants as well.

Local based metaheuristics prevail population-based in the VRPPD. The fact that transportation companies must consider both delivering and returning quantities of goods leads to increased complexity. According to Avci and Topaloglu (2015), it is the last 15 years that population-based metaheuristics have been applied in the VRPPD. The advances in technology and computer science enhance these algorithms, which need a population set in order to operate.

There are specific algorithms that are implemented in most VRP variants such as SA, TS, VNS, ILS, LNS, EA's, PSO, and ACO. These algorithms produce efficient results and, therefore, are proposed and implemented by researchers. Although metaheuristics have dominated the other methods, local improvement heuristics remain very popular and are used mainly as components of more sophisticated hybrid systems. In contrast, construction algorithms are mainly used for the creation of the initial solution or population, used as input in metaheuristics. Finally, a small part of the reviewed articles employs an exact method of problem-solving exclusively.

2.6 Applied Algorithms in VRP Variants

In this section, the papers that either propose novel algorithms and offer efficient results, either have been recognized by the research community, are presented. More specifically, insights to the algorithms proposed in these papers, in each one on the VRP variants selected are given in Section 2.2.

2.6.1 Capacitated VRP

CVRP is perhaps the simplest variant of the VRP, as the only constraint considered is the vehicles' capacity, which is assumed to be identical in terms of actual capacity space and costs. In most cases, the CVRP is combined with other variants, with the most frequent being the VRPTW, which adds more constraints to the setting of the problem. The only papers that deal exclusively with the CVRP are Altabeeb et al. (2019) that propose a firefly algorithm for solving the CVRP, combined with a local search and a genetic operator, to search a wider space and avoid getting stuck in a local optimum. Also, Lysgaard and (Wøhlk 2014) study the cumulative CVRP, in which the sum of arrival times at the customers aims to be minimized instead of the total distance. They propose a branch and cut and-price algorithm for obtaining the optimum solutions of the problem.

2.6.2 VRP with Time Windows

The VRPTW is one of the most widely studied topics in vehicle routing, and therefore, multiple algorithms have been proposed for its solution. (Ghoseiri and Ghannadpour 2010) treat the problem as multiobjective, where both the total traveled distance, and the number of vehicles must be minimized. They proposed a Genetic Algorithm that applies the Pareto ranking process to find the non-dominated solutions. Through this approach, possible biased solutions, either to the total number of vehicles or the total traveled distance, are prevented.

(Lei et al. 2011), in their research work, they propose an adaptive large neighborhood heuristic. Initially, a construction heuristic is applied for providing an adequate solution. At each iteration, several heuristics remove customers and destroy the current solution before insertion heuristics repair the damaged solutions. The selection of the destroy and repair heuristic in each phase is made probabilistically, while their combination offers a wide exploration of the solution space. (Vidal et al. 2013) addressed a wide range of large scale VRPTW, with route-duration constraints and additional attributes involving requirements for customer assignments to particular vehicle types, depots, or planning periods. For solving this large class of VRPTW, they proposed a new hybrid genetic search with advanced diversity control.

Yassen et al. (2015) proposed a harmony search (HS) metaheuristic with a local search (LS) algorithm with a proper balance between HS exploration and LS exploitation for solving the VRPTW. Nalepa and Blocho (2016) dealt with the unclear tuning of the numerous parameters, one of the main drawbacks of memetic algorithms for the VRPTW, and proposed an adaptive memetic algorithm. Parameters of the algorithm, such as the selection scheme, the population size, and the number of child solutions generated for each pair of parents, are adjusted dynamically during the search. Hu et al. (2018) examined a more realistic variation of VRPTW that involves demand and travel time uncertainty. To tackle large instances, they designed a two-stage method based on a modified variable neighborhood search heuristic. In the first phase, the total number of vehicles is minimized, while the second phase minimizes the total traveled distance.

2.6.3 VRP with Pickups and Deliveries

Transportation companies are not responsible only for delivering goods to customers, but in many cases, for picking goods too. This need arises in reverse logistics and is tackled by multiple researchers, either by considering deliveries and pickups are executed simultaneously (VRPSPD), either by considering the pickups are executed after the end of deliveries (VRPB). (Çatay 2010) proposed an ACO algorithm that uses the nearest-neighbor heuristic (NNH) to construct the initial solution, and subsequently, a new savings-based visibility function is applied. Additionally, (Goksal et al. 2013) developed a hybrid algorithm based on PSO and Variable Neighborhood Descent (VND) for solving the VRPSPD. The initial solution set is formed both with random solutions and with solutions constructed by the NNH. The PSO algorithm uses VND algorithms as a local search procedure for improving these solutions. Finally, (Wang et al. 2015) combined the VRPSPD with the VRPTW and proposed a parallel SA method for solving this problem.

2.6.4 Heterogenous Fleet VRP

The HVRP is a generalization of the classical capacitated VRP by assuming that the fleet of vehicles is composed of different types of vehicles, each characterized by different capacities and costs. HVRP is rarely examined standalone, and it is usually combined with other variants. (Subramanian et al. 2012) proposed a hybrid algorithm consisting of an iterated local search based heuristic and set partitioning (SP) formulation for solving the HVRP. The SP model is solved utilizing a Mixed Integer Programming solver that interactively calls the ILS heuristic during its execution. Also, Koç et al. (2015) tackled the HVRP with time windows and introduced a hybrid evolutionary algorithm that combined two state-of-the-art metaheuristic concepts, adaptive large neighborhood search and population-based search to achieve great results.

Finally, (Belloso et al. 2019) analyzed the HVRP with backhauls and proposed a solution based on a multi-start biased-randomized heuristic. This algorithm uses an iterative method that relies on solving smaller homogeneous instances of the problem and then uses these subsolutions as partial solutions for the original heterogeneous instance.

2.6.5 VRP with Multiple Depots and Collaborative VRP

MDVRP is an extension of the classical VRP problem by adding multiple depots that can serve the customers. Yu and Yang (2011) added a virtual central depot, thus converting MDVRP to VRP with the virtual depot as the original. They proposed an improved ant colony optimization process with coarse-grain parallel strategy, ant-weight strategy, and mutation operation for its solution. Juan et al. (2015) presented a hybrid approach that combined biased randomization, assign customers to depots, and improve routing solutions, with the iterated local search metaheuristic. Their approach is easy to implement and can be parallelized naturally.

The research of genetic algorithms (GA's) for solving the MDVRP by Karakatič and Podgorelec (2015) is also worth mentioning. They claimed that GA's are on par with other algorithms and are very efficient when addressing the MDVRP, with the main advantage of a GA being the linear scaling with the growing problem size. Thus, it is preferred for solving large NP problems over exact and other heuristic algorithms.

Wang et al. (2020) studied the collaborative VRP with time window assignments, where the distribution centers of multiple service providers collaborate to improve the efficiency of their logistics operations and make sure that customers are served on time, while the operations costs are reduced. (Wang et al. 2020) proposed a hybrid heuristic that combines a customer clustering algorithm (K-means clustering), Clark-Wright saving algorithm, and the extended NSGA-II algorithm, which is a multiobjective evolutionary algorithm. The two objectives considered by the researchers are the minimization of the total operating cost and the number of vehicles needed for the deliveries.

2.6.6 Green VRP

From this class of problems, which contains the Green, the Hybrid, the Electric, and the Pollution, the Hybrid VRP seems to attract the most attention from researchers. It consists of a set of customers, a fleet of alternative fuel vehicles (electrical and hybrid), and a set of refueling stations to minimize the total traveled distance while also making use of the refueling stations to restore their autonomy. Schneider et al. (2014) tackled the mixed fleet VRP with time windows in which both electric cars and internal combustion cars are used. They presented a hybrid heuristic that combines a variable neighborhood search algorithm with a tabu search heuristic as a solution method. (Goeke and Schneider 2015)dealt with the same problem and developed an adaptive large neighborhood search, enhanced by a local search for intensification. (Hiermann et al. 2016)

Hiermann et al. (2016) solved the same problem utilizing branch-and-price and proposing a hybrid heuristic, which combines an ALN with an embedded local search and labeling procedure for intensification.

Yu et al. (2017) also focused on the hybrid VRP in which all the vehicles are plugged in hybrid electrical vehicles (PHEV), and they proposed a simulated annealing algorithm with a restart strategy. Last but not least, Andelmin and Bartolini (2019) also dealt with alternative fuel vehicles and their refueling stations and offered a multigraph reformulation of the GVRP in which refueling stations are not explicitly modeled. They exploited this reformulation by tailoring classical local search operators to work directly on it and combining them to develop a multi-start algorithm.

2.6.7 Open VRP

OVRP is another well-known combinatorial optimization problem that addresses the service of a set of customers using a fleet of non-depot returning vehicles. Repoussis et al. (2010)

developed a population-based hybrid metaheuristic consisting of the basic solution framework of evolutionary algorithms and memory-based trajectory local search algorithms, such as tabu search and guided local search to drive the local search process and to explore the solution space. Salari et al. (2010) presented a heuristic improvement procedure based on integer linear programming techniques. Given an initial feasible solution to be possibly improved, the method then destroys and repairs this solution by solving an ILP model to find a new, improved feasible solution. MirHassani and Abolghasemi (2011) proposed a particle swarm optimization algorithm and a particular decoding method for assigning customers to routes, and one point moves for improving the found solutions.

2.6.8 Dynamic and Stochastic VRP

In most VRP's the data is assumed to be known before starting the routes and not being amenable to change. However, real-world routing problems are dynamic, as unforeseen events can constantly occur. Hong (2012) tackled the dynamic VRP with time windows (DVRPTW), which is modeled as a series of static VRPTW. Each phase has an event-trigger mechanism, which is a new request arrival during the operation in the specific case. The author proposed a large neighborhood search (LNS) that utilizes the remove and reinsert procedure. The newly arrived request is considered as a removed node, which is being reinserted into the current solution. Mavrovouniotis and Yang (2015), in their article, proposed an ant colony optimization algorithm with immigrant schemes, in which three different cases are investigated, random, elitism, and memory-based. These three immigrant schemes determine the way to introduce new solutions (called immigrants) and replace a small portion of the current population. Finally, Baradaran et al. (2019) addressed the heterogeneous fleet VRP with multiple hard prioritized time windows and proposed an artificial bee colony algorithm (which belongs in the class of swarm intelligent algorithms) in order to face three different models. The first model considers travel times and transportation costs deterministic, the second considers the fixed and variable costs as stochastic, while the third model considers the travel times and costs as normal random variables (Stochastic VRP).

2.6.9 Multi-Trip VRP

A variant of the classical VRP is the problem that deals with multiple uses of vehicles, that arises when customers have either great demands or when they are close to each other. Azi et al. (2010), tackle the MTVRP along with the VRPTW, through a branch-and-price approach, in which a column generation approach is also integrated. This was the first attempt to solve the MTVRP with an exact algorithm. François et al. (2016) developed and tested two Adaptive Large Neighborhood Search (ALNS) algorithms in the MTVRP, which does not consider duration tour constraints.

2.6.10 Multi-Echelon VRP

It is the last few years that researchers deal with the MEVRP and the VRPSF and the 2EVRP, which belong in this class. More specifically, Baldacci et al. (2013) propose an exact algorithm for the two-echelon capacitated VRP (2E-CVRP), which is transformed into a multi depot CVRP with side constraints, while the routing and the handling costs aim to be minimized. In the paper of Breunig et al. (2019) the electric two-echelon VRP is faced, in which both conventional and electric vehicles are used. They propose two algorithms, an LNS-based metaheuristic and an exact algorithm for optimizing charging visitations and the routing procedures in both levels.

2.6.11 Time-dependent VRP

The travel time between two points-customers depends on traffic congestion, which in turn is affected by multiple factors, such as unforeseen events, the time of the day, and the weather. Balseiro et al. (2011), solve the TDVRPTW where vehicles must deliver goods to a set of customers concerning their time windows, while travel time between two points, depends on the time of departure. They proposed an ACO algorithm hybridized with insertion heuristics. More specifically, the algorithm is based on the Multiple Ant Colony System framework, which coordinates two colonies, one for reducing the number of routes and one for optimizing the feasible solutions found by minimizing the total time. Since ants produce solutions with unrouted customers, insertion heuristics are further developed for incorporating customers into the solution. Figliozzi (2012), faces the same VRP variants and presents an Iterative Route Construction and Improvement (IRCI) algorithm. The construction phase includes a sequential heuristic, in which an auxiliary route building heuristic is reiterated during its execution of the construction algorithm. Since the set of the initial solutions have been generated, the improvement algorithm, based on the ruin and recreate approach presented by (Schrimpf et al. 2000), is implemented in a subset of the routes.

2.6.12 Two Dimensional and Three Dimensional VRP

In the specific category of the VRP, the dimension of vehicles is considered as a loading constrain. Leung et al. (2013) addressed the heterogeneous fleet VRP with two-dimensional loading constraints. Vehicles have different capacities, fixed and variable operating costs, and dimensions (width and length), which determine the loading constraints. For these types of problems, both packing and routing algorithms are needed, and therefore, Leung et al. (2013) proposed a simulated annealing algorithm with local search heuristics for the routing of vehicles and six packing algorithms for the two-dimensional loading constraints.

Zachariadis et al. (2016) also focus their research on the VRP with two-dimensional constraints, paired with simultaneous pickups and deliveries, while vehicles are identical. This problem is highly complicated since delivery goods are unloaded, while pickup goods are loaded to the vehicle. However, the algorithm proposed in this publication, a combination of a

construction heuristic algorithm with local search procedures, manages to offer good quality solutions in reasonable computation times.

2.6.13 Consistent VRP

The consistent VRP has attracted the least attention from researchers. This is due to the fact that most logistics companies, (i) do not have stable deliveries over a period of time, neither can plan their routes for multiple days, (ii) do not focus on creating consistent routes so that frequent customers being delivered by the same driver, approximately at the same time over the planning period, while it is a relatively new variant of the problem, proposed by Groër et al. (2009). However, Tarantilis et al. (2012) addressed the specific problem by proposing savings and insertion based construction heuristics paired with a Tabu Search algorithm, while Xu and Cai (2018) proposed a Variable Neighborhood Search algorithm for addressing the ConVRP.

2.6.14 Split Delivery VRP

In the Split Delivery VRP, the assumption that each customer is served by a single vehicle and only once is relaxed. Archetti et al. (2008) proposed two branch and cut algorithms for solving this variant of the VRP and tested it with up to 100 customers and with a time limit of 2 h. Han and Chu (2016) also studied this variant, coupled with minimum delivery amounts, and proposed a multi-start two-phased variable neighborhood descent heuristic algorithm. More specifically, a construction algorithm generates the initial solution, which is then improved by a VND procedure. The algorithm was tested in 128 cases and managed to find 81 best-known solutions and 43 new optimum solutions.

2.6.15 Periodic VRP

Yu and Yang (2011) state that there is little literature to use heuristics for periodic VRP with time windows. This statement is reinforced by the research, as no heuristic algorithm was proposed for solving the PVRP. More specifically, Yu and Yang (2011) proposed an improved ant colony optimization algorithm with multi-dimension pheromone information for solving the periodic VRP with time windows. On the other hand, Baldacci et al. (2011) and Rothenbächer (2019) proposed exact algorithms for solving the problem.

2.6.16 Truck and Trailer VRP

The Track and Trailer Routing Problem has not attracted the interest of researchers. In these few articles that study this variant of the VRP, Lin et al. (2010) developed and proposed a simulated annealing heuristic-based algorithm, and was tested not only existing benchmark problems but also in newly generated problems. Also, Rothenbächer et al. (2018) proposed an exact algorithm for the TTRP with time windows, and the computational results indicate that it outperforms other approaches facing this problem.

2.7 Applications to Practice

The above research confirms that the VRP is widely addressed by researchers and consists one of the most studied problems in logistics. However, there is a limited number of research papers that lead to the integration of algorithms into routing systems. In order to identify the studies that have led to algorithms integrations and system development another literature review is applied, focusing on advanced systems addressing the vehicle routing. Respectively, the SCOPUS database was exploited for this research. The term is applied in Title, Abstract, Keywords and is ("vehicle routing problem" AND ("information system" OR "system" OR "software") AND ("metaheuristic" OR "heuristic" OR "exact") AND "case study").

The number of papers concluding from the research is limited to 92. However, only 5 of them discuss the development of a routing system or the integration of an algorithm in a real routing application. In addition, the studies focus on different aspects of distribution.

Fanti et al. (2014) focused on the postal delivery operations, while Abbatecola et al. (2016) on both postal delivery and waste collection. Both studies aimed to develop decision support systems (DSS), focusing mainly on the architecture of the systems, the mathematical formulation of the problem, and their real-life testing. Abbatecola et al. (2016), integrated a two-phase heuristic into the system for addressing the delivery procedure, while Fanti et al. (2014) an Augment-insert heuristic one. Both studies, however, do not analytically present the algorithmic approaches as their aim wasn't to propose the integration of a novel method into a system. Additionally, Tarantilis and Kiranoudis (2007) proposed a flexible adaptive memory-based algorithm for addressing the Heterogeneous Fixed Fleet VRP (HFVRP). The metaheuristic algorithm was integrated into a DSS solving two case studies. The first originated from a construction company, and the second one from a dairy company. Finally, Tarantilis et al. (2004a) proposed a DSS employing a metaheuristic algorithm for solving the OVRP. The proposed DSS geocodes and maps customers' locations by connecting to a Geographical Information System (GIS).

All the aforementioned researches focused on a limited number of VRP variants, resulting in limiting the potential real-life cases that can benefit from the use of their systems. The research of Erdoğan (2017) is the only one that comes close to the presented research, as an open-source spreadsheet solver is developed for addressing most of the variants. The researcher connects online maps with Microsoft Excel and extracts significant data related to routing and scheduling. The application is tested in two real-world case studies from the healthcare and tourism sectors and manages to offer decent results. The limitation of this solution is the increased computation time needed as Visual Basic (VB), used by the researcher, is not a very efficient programming language. Other languages such as C++ and Python are more suitable for the optimization and the software development, as they perform way more efficient calculations.

To conclude, the advance in technology is tremendous over the years, and the optimization methods and algorithms proposed for addressing the VRP are numerous. However, only a few studies that propose algorithms lead to real-life applications and systems. One reason is the specialist knowledge needed for developing an operable system. In addition, real-life applications and software packages are, in most cases created by software development companies. In such cases, the algorithms are black-boxes since they constitute the intellectual property of companies.

2.8 Concluding Remarks

This Chapter studies and presents multiple VRP variants that have been studied over the years and are correlated to freight distribution. Additionally, the algorithmic approaches that have been developed and proposed for solving these cases are also categorized and presented. Both the variants and algorithms are the main fields of study and their extensive literature review aims as a first step to identify their correlation.

The literature review has resulted from a well-defined methodological approach and research protocol. An initial group of 334 papers extracted from the Scopus database published between 2010 and the first quarter of 2020. The articles were further shorted to a group of 263 relevant papers after applying deselection criteria. Emphasis was given in the applied algorithms proposed and developed for the VRP variants to distribute goods. Therefore, papers that are irrelevant from the freight transportation perspective were excluded from the analysis.

The articles were classified according to the VRP variants and the applied method proposed for their solution to identify the trends in each case and present their correlation. A trend towards Evolutionary Algorithms and Local Search metaheuristics seems to be formed, as they are applied in many variants of the problem and offer very efficient results. Their ability to be combined with other algorithms, while also offering efficient solutions, led hybrid algorithms being applied more and more frequently in many variants of the VRP. That is because from the algorithms combined, the advantages of each algorithm are exploited.

Also, it should be noted that logistics and distribution companies have to fulfill restrictions of their customers and the external environment dictate for optimizing their operations. In the last few years, environmental regulations have emerged and are studied by researchers. However, only a few algorithms have been applied to the relevant VRP variants. This will inevitably lead to increased interest in the variants of the VRP, that are related to environmental issues, such as the Green, the Hybrid, the Electric VRP, and the PRP. Simultaneously, in a competitive environment such as todays, where customers are increasingly demanding, the overall distribution process needs to be optimized. Therefore, the research interest about the multi-echelon VRP, which contains multiple layers of distribution and routing, may be increased in the following years. These two statements are reinforced by the fact that they are being studied more intensively from 2013 and onwards. Also, the interest in the collaborative VRP may

increase in the future since significant benefits in a network of logistics companies are created. The collaboration between multiple logistics companies is a challenge, but an opportunity as well, for optimizing their deliveries and their operating costs. Finally, a topic for further investigation and research is that a minimum number of articles treat simultaneously a combination of different variants of the VRP (as is usual in the complexity of real-life scheduling problems), with most researchers limiting their study to 3 variants of the VRP. Only Penna et al. (2019) were found to simultaneously tackle a broad range of variants, whose hybrid Variable Neighborhood algorithm manages to treat 6 variants of the problem.

Additionally, the literature review was not limited to the study of the VRP variants, the proposed algorithms, and their correlation, but was also expanded to the research studies that propose optimization algorithms for addressing the VRP and that are also integrated into real-life routing applications. The results led to only a limited number of articles proposing both optimization algorithms and their integration to routing systems. The results also confirm the gap that exist in research community regarding the algorithms that are developed and also integrated into real-life systems.

Specifically, most researchers focus on other research areas such as the development of improved algorithms and mathematical models addressing the VRP, without studying the integration of the algorithms into the system. This is not absurd since the development of a system requires a lot of development and programming knowledge and skills, as well as time and manpower. However, the entire process, the key parts for developing a routing system, as well as the functionalities of a system are crucial and can prove the importance of high-efficient algorithms that fit to the studied problem, and the selection of the right technologies.

To sum up, the literature review led to important conclusions regarding the algorithms that are most commonly implemented in specific VRP variants. Additionally, the review defined the VRP variants that are most commonly studied by researchers, along with their trends, and was the base for selecting the VRP variants that are addressed in the rest of the chapters. Moreover, the literature review managed to clarify the gap that exists in research community regarding the optimization algorithms that are integrated into real-life systems.

References

Abbatecola L, Fanti MP, Mangini AM, Ukovich W (2016) A Decision Support Approach for Postal Delivery and Waste Collection Services. IEEE Trans Autom Sci Eng 13:1458–1470. https://doi.org/10.1109/TASE.2016.2570121

Aggarwal D, Kumar V, Girdhar A (2017) Lagrangian relaxation for the vehicle routing problem with time windows. In: 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). pp 1601–1606

Altabeeb AM, Mohsen AM, Ghallab A (2019) An improved hybrid firefly algorithm for capacitated vehicle routing problem. Appl Soft Comput 84:105728. https://doi.org/https://doi.org/10.1016/j.asoc.2019.105728

Andelmin J, Bartolini E (2019) A multi-start local search heuristic for the Green Vehicle Routing Problem based on a multigraph reformulation. Comput Oper Res 109:43–63. https://doi.org/10.1016/j.cor.2019.04.018

Archetti C, Savelsbergh MWP, Speranza MG (2008) To split or not to split: That is the question. Transp Res Part E Logist Transp Rev 44:114–123. https://doi.org/https://doi.org/10.1016/j.tre.2006.04.003

Arnold F, Sörensen K (2019) Knowledge-guided local search for the vehicle routing problem. Comput Oper Res 105:32–46. https://doi.org/https://doi.org/10.1016/j.cor.2019.01.002

Avci M, Topaloglu S (2015) An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. Comput Ind Eng 83:15–29. https://doi.org/10.1016/j.cie.2015.02.002

Azi N, Gendreau M, Potvin J-Y (2010) An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. Eur J Oper Res 202:756–763. https://doi.org/https://doi.org/10.1016/j.ejor.2009.06.034

Badeau P, Guertin F, Gendreau M, et al (1997) A parallel tabu search heuristic for the vehicle routing problem with time windows. Transp Res Part C Emerg Technol 5:109–122

Baldacci R, Bartolini E, Mingozzi A, Valletta A (2011) An Exact Algorithm for the Period Routing Problem. Oper Res 59:228–241. https://doi.org/10.1287/opre.1100.0875

Baldacci R, Mingozzi A, Roberti R (2012) Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. Eur J Oper Res 218:1–6. https://doi.org/10.1016/j.ejor.2011.07.037

Baldacci R, Mingozzi A, Roberti R, Calvo RW (2013) An Exact Algorithm for the Two-Echelon Capacitated Vehicle Routing Problem. Oper Res 61:298–314. https://doi.org/10.1287/opre.1120.1153

Balseiro SR, Loiseau I, Ramonet J (2011) An Ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows. Comput Oper Res 38:954–966. https://doi.org/10.1016/j.cor.2010.10.011

Baradaran V, Shafaei A, Hosseinian AH (2019) Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach. Comput Ind Eng 131:187–199. https://doi.org/https://doi.org/10.1016/j.cie.2019.03.047

Bektaş T, Laporte G (2011) The Pollution-Routing Problem. Transp Res Part B Methodol 45:1232–1250. https://doi.org/10.1016/j.trb.2011.02.004

Belfiore PP, Fávero LPL (2007) Scatter search for the fleet size and mix vehicle routing problem with time windows. Cent Eur J Oper Res 15:351–368. https://doi.org/10.1007/s10100-007-0036-9

Belloso J, Juan AA, Faulin J (2019) An iterative biased-randomized heuristic for the fleet size and mix vehicle-routing problem with backhauls. Int Trans Oper Res 26:289–301. https://doi.org/10.1111/itor.12379

Belmecheri F, Prins C, Yalaoui F, Amodeo L (2013) Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. J Intell Manuf 24:775–789. https://doi.org/10.1007/s10845-012-0627-8

Braekers K, Ramaekers K, Nieuwenhuyse I Van (2016) The Vehicle Routing Problem: State of the Art Classification and Review. Comput Ind Eng 99:300–313. https://doi.org/10.1016/j.cie.2015.12.007

Brandão JCS, Mercer A (1998) The multi-trip vehicle routing problem. J Oper Res Soc 49:799–805. https://doi.org/10.1057/palgrave.jors.2600595

Bräysy O, Gendreau M (2005) Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transp Sci 39:104–118. https://doi.org/10.1287/trsc.1030.0056

Breunig U, Baldacci R, Hartl RF, Vidal T (2019) The electric two-echelon vehicle routing problem. Comput Oper Res 103:198–210. https://doi.org/https://doi.org/10.1016/j.cor.2018.11.005

Bullnheimer B, Hartl RF, Strauss C (1999) An improved Ant System algorithm for the Vehicle Routing Problem. Ann Oper Res 89:319–328

Caceres-Cruz J, Arias P, Guimarans D, et al (2014) Rich Vehicle Routing Problem: Survey. ACM Comput Surv 47:1–28. https://doi.org/10.1145/2666003

Çatay B (2010) A new saving-based ant algorithm for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. Expert Syst Appl 37:6809–6817. https://doi.org/10.1016/j.eswa.2010.03.045

César H, Oliveira B De (2010) A hybrid search method for the vehicle routing problem with time windows. Ann Oper Res 180:125–144. https://doi.org/10.1007/s10479-008-0487-y

Chiang W-C, Russell RA (1996) Simulated annealing metaheuristics for the vehicle routing problem with time windows. Ann Oper Res 63:3–27. https://doi.org/10.1007/BF02601637

Chiang WC, Russell RA (1997) A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows. INFORMS J Comput 9:417–430. https://doi.org/doi:10.1287/ijoc.9.4.417

Clarke G, Wright JW (1964) Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Oper Res 12:568–581. https://doi.org/10.1287/opre.12.4.568

Coene S, Arnout A, Spieksma FCR (2010) On a periodic vehicle routing problem. J Oper Res Soc 61:1719–1728. https://doi.org/10.1057/jors.2009.154

Cordeau JF, Laporte G, Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. J Oper Res Soc 52:928–936. https://doi.org/10.1057/palgrave.jors.2601163

Crainic TG, Perboli G, Mancini S, Tadei R (2010) Two-Echelon Vehicle Routing Problem: A satellite location analysis. Procedia - Soc Behav Sci 2:5944–5955. https://doi.org/https://doi.org/10.1016/j.sbspro.2010.04.009

Desrochers M, Desrosiers J, Solomon M (1992) A new optimization algorithm for the vehicle routing problem with time windows. Oper Res 40:342–354. https://doi.org/10.1287/opre.40.2.342

Eberhart R, Kennedy J (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. pp 1942–1948

Eksioglu B, Vural AV, Reisman A (2009) The vehicle routing problem: A taxonomic review. Comput Ind Eng 57:1472–1483. https://doi.org/10.1016/j.cie.2009.05.009

El-Sherbeny NA (2010) Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. J King Saud Univ 22:123–131. https://doi.org/10.1016/j.jksus.2010.03.002

Elshaer R, Awad H (2020) A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Comput Ind Eng 140:106242. https://doi.org/https://doi.org/10.1016/j.cie.2019.106242

Erdoğan G (2017) An open source Spreadsheet Solver for Vehicle Routing Problems. Comput Oper Res 84:62–72. https://doi.org/10.1016/j.cor.2017.02.022

Fanti MP, Laraspata R, Iacobellis G, et al (2014) A Decision Support System approach for the postal delivery operations. In: 2014 IEEE International Conference on Automation Science and Engineering (CASE). pp 588–593

Feillet D, Garaix T, Lehuédé F, et al (2014) A new consistent vehicle routing problem for the transportation of people with disabilities. Networks 63:211–224. https://doi.org/10.1002/net.21538

Figliozzi MA (2012) The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. Transp Res Part E Logist Transp Rev 48:616–636. https://doi.org/10.1016/j.tre.2011.11.006

Fisher ML (1994) Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees. Oper Res 42:626–642. https://doi.org/10.2307/171617

Flatberg T, Hasle G, Kloster O, et al (2007) Dynamic And Stochastic Vehicle Routing In Practice. In: Zeimpekis V, Tarantilis CD, Giaglis GM, Minis I (eds) Dynamic Fleet Management: Concepts, Systems, Algorithms & Case Studies. Springer US, Boston, MA, pp 41–63

François V, Arda Y, Crama Y, Laporte G (2016) Large neighborhood search for multi-trip vehicle routing. Eur J Oper Res 255:422–441. https://doi.org/https://doi.org/10.1016/j.ejor.2016.04.065

Gambardella LM, Taillard É, Agazzi G (1999) Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In: New Ideas In Optimization. pp 63–76

Gayialis SP, Konstantakopoulos GD, Tatsiopoulos IP (2019) Vehicle Routing Problem for Urban Freight Transportation: A Review of the Recent Literature. In: Sifaleras A, Petridis K (eds) Operational Research in the Digital Era--ICT Challenges, Springer P. Springer, pp 89–104

Gendreau M, Hertz A, Laporte G (1992) New insertion and postoptimization procedures for the traveling salesman problem. Oper Res 40:1086–1094. https://doi.org/10.1287/opre.40.6.1086

Ghoseiri K, Ghannadpour SF (2010) Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Appl Soft Comput 10:1096–1107. https://doi.org/10.1016/j.asoc.2010.04.001

Gillet BE, Miller LE, Johnson JG (1979) Vehicle dispatching—Sweep algorithm and extensions. In: Disaggregation. Springer, pp 471–483

Glover F (1977) HEURISTICS FOR INTEGER PROGRAMMING USING SURROGATE CONSTRAINTS. Decis Sci 8:156–166. https://doi.org/https://doi.org/10.1111/j.1540-5915.1977.tb01074.x

Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13:533–549. https://doi.org/10.1016/0305-0548(86)90048-1

Goeke D, Schneider M (2015) Routing a mixed fleet of electric and conventional vehicles. Eur J Oper Res 245:81–99. https://doi.org/10.1016/j.ejor.2015.01.049

Goetschalckx M, Jacobs - Blecha C (1989) The vehicle routing problem with backhauls. Eur J Oper Res 42:39–51. https://doi.org/10.1016/0377-2217(89)90057-X

Goksal FP, Karaoglan I, Altiparmak F (2013) A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. Comput Ind Eng 65:39–53. https://doi.org/10.1016/j.cie.2012.01.005

Grangier P, Gendreau M, Lehuédé F, Rousseau L-M (2016) An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. Eur J Oper Res 254:80–91. https://doi.org/https://doi.org/10.1016/j.ejor.2016.03.040

Groër C, Golden B, Wasil E (2009) The Consistent Vehicle Routing Problem. Manuf Serv Oper Manag 11:630–643. https://doi.org/10.1287/msom.1080.0243

Han AF-W, Chu Y-C (2016) A multi-start heuristic approach for the split-delivery vehicle routing problem with minimum delivery amounts. Transp Res Part E Logist Transp Rev 88:11–31. https://doi.org/https://doi.org/10.1016/j.tre.2016.01.014

Hiermann G, Puchinger J, Ropke S, Hartl RF (2016) The Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations. Eur J Oper Res 252:995–1018. https://doi.org/10.1016/j.ejor.2016.01.038

Holland JH (1975) Adaptation in natural and artificial systems Ann Arbor. Univ Michigan Press 1:975

Hong L (2012) An improved LNS algorithm for real-time vehicle routing problem with time windows. Comput Oper Res 39:151–163. https://doi.org/10.1016/j.cor.2011.03.006

Hornstra RP, Silva A, Roodbergen KJ, Coelho LC (2020) The vehicle routing problem with simultaneous pickup and delivery and handling costs. Comput Oper Res 115:104858. https://doi.org/10.1016/j.cor.2019.104858

Hu C, Lu J, Liu X, Zhang G (2018) Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. Comput Oper Res 94:139–153. https://doi.org/https://doi.org/10.1016/j.cor.2018.02.006

Juan, A.A., Adelantado, F., Grasman, S.E., Faulin, J., & Montoya-Torres JR (2009). S the capacitated vehicle routing problem with maximum traveling distance and service time requirements: an approach based on MC simulation. W '09. (2009) Solving the capacitated vehicle routing problem with maximum traveling distance and service time requirements: an approach based on Monte Carlo simulation. In: Proceedings of the 2009 Winter Simulation Conference (WSC '09). pp 2467–2475

Juan AA, Pascual I, Guimarans D, Barrios B (2015) Combining biased randomization with iterated local search for solving the multidepot vehicle routing problem. Int Trans Oper Res 22:647–667. https://doi.org/10.1111/itor.12101

Kaboudani Y, Ghodsypour SH, Kia H, Shahmardan A (2018) Vehicle routing and scheduling in cross docks with forward and reverse logistics. Oper Res Int J. https://doi.org/10.1007/s12351-018-0396-z

Kalayci CB, Kaya C (2016) An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. Expert Syst Appl 66:163–175. https://doi.org/https://doi.org/10.1016/j.eswa.2016.09.017

Karakatič S, Podgorelec V (2015) A survey of genetic algorithms for solving multi depot vehicle routing problem. Appl Soft Comput 27:519–532. https://doi.org/10.1016/j.asoc.2014.11.005

Kilby P, Prosser P, Shaw P (1999) Guided local search for the vehicle routing problem with time windows. In: Meta-heuristics. Springer, pp 473–486

Kim G, Ong Y, Heng CK, et al (2015) City Vehicle Routing Problem (City VRP): A Review. IEEE Trans Intell Transp Syst 16:1654–1666. https://doi.org/10.1109/TITS.2015.2395536

Kirkpatrick S, Gelatt CD, Vecchi MP, P. VM (1983) Optimization by Simulated Annealing. Science (80-) 220:671 LP-680. https://doi.org/10.1126/science.220.4598.671

Koç Ç, Bektaş T, Jabali O, Laporte G (2015) A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. Comput Oper Res 64:11–27. https://doi.org/10.1016/j.cor.2015.05.004

Kohl N, Madsen OBG (1997) An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. Oper Res 45:395–406. https://doi.org/10.1287/opre.45.3.395

Kuo Y (2010) Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. Comput Ind Eng 59:157–165. https://doi.org/https://doi.org/10.1016/j.cie.2010.03.012

Labadie N, Prins C, Prodhon C, Prins C (2016) Metaheuristics for Vehicle Routing Problems. John Wiley and Sons

Lahyani R, Khemakhem M, Semet F (2015) Rich vehicle routing problems: From a taxonomy to a definition. Eur J Oper Res 241:1–14. https://doi.org/10.1016/j.ejor.2014.07.048

Lei H, Laporte G, Guo B (2011) The capacitated vehicle routing problem with stochastic demands and time windows. Comput Oper Res 38:1775–1783. https://doi.org/10.1016/j.cor.2011.02.007

Leung SCH, Zhang Z, Zhang D, et al (2013) A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. Eur J Oper Res 225:199–210. https://doi.org/https://doi.org/10.1016/j.ejor.2012.09.023

Lin C, Choy KL, Ho GTS, et al (2014) Survey of Green Vehicle Routing Problem : Past and future trends. Expert Syst Appl 41:1118–1138

Lin S-W, Yu VF, Chou S-Y (2010) A note on the truck and trailer routing problem. Expert Syst Appl 37:899–903. https://doi.org/https://doi.org/10.1016/j.eswa.2009.06.077

Lysgaard J, Wøhlk S (2014) A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. Eur J Oper Res 236:800–810. https://doi.org/https://doi.org/10.1016/j.ejor.2013.08.032

Mancini S (2017) The Hybrid Vehicle Routing Problem. Transp Res Part C Emerg Technol 78:1–12. https://doi.org/10.1016/j.trc.2017.02.004

Mańdziuk J, Nejman C (2015) UCT-Based Approach to Capacitated Vehicle Routing Problem. In: Rutkowski L, Korytkowski M, Scherer R, et al. (eds) Artificial Intelligence and Soft Computing. Springer International Publishing, Cham, pp 679–690

Marinakis Y, Migdalas A (2007) Annotated bibliography in vehicle routing. Oper Res Int J 7:27–46. https://doi.org/10.1007/BF02941184

Mavrovouniotis M, Yang S (2015) Ant algorithms with immigrants schemes for the dynamic vehicle routing problem. Inf Sci (Ny) 294:456–477. https://doi.org/https://doi.org/10.1016/j.ins.2014.10.002

Mekamcha K, Souier M, Bessenouci HN, Bennekrouf M (2019) Two metaheuristics approaches for solving the traveling salesman problem: an Algerian waste collection case. Oper Res Int J. https://doi.org/10.1007/s12351-019-00529-6

MirHassani SA, Abolghasemi N (2011) A particle swarm optimization algorithm for open vehicle routing problem. Expert Syst Appl 38:11547–11551. https://doi.org/10.1016/j.eswa.2011.03.032

Montané FAT, Galvão RD (2006) A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Comput Oper Res 33:595–619. https://doi.org/10.1016/j.cor.2004.07.009

Nalepa J, Blocho M (2016) Adaptive memetic algorithm for minimizing distance in the vehicle routing problem with time windows. Soft Comput 20:2309–2327. https://doi.org/10.1007/s00500-015-1642-4

Osman IH (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann Oper Res 41:421–451. https://doi.org/10.1007/BF02023004

Penna PHV, Subramanian A, Ochi LS, et al (2019) A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. Ann Oper Res 273:5–74. https://doi.org/10.1007/s10479-017-2642-9

Perboli G, Rosano M (2019) Parcel delivery in urban areas: Opportunities and threats for the mix of traditional and green business models. Transp Res Part C Emerg Technol 99:19–36. https://doi.org/https://doi.org/10.1016/j.trc.2019.01.006

Pillac V, Gendreau M, Guéret C, Medaglia AL (2013) A review of dynamic vehicle routing problems. Eur J Oper Res 225:1–11. https://doi.org/10.1016/j.ejor.2012.08.015

Potvin J-Y, Rousseau J-M (1993) A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. Eur J Oper Res 66:331–340. https://doi.org/10.1016/0377-2217(93)90221-8

Prins C (2009) Two memetic algorithms for heterogeneous fleet vehicle routing problems. Eng Appl Artif Intell 22:916–928. https://doi.org/10.1016/j.engappai.2008.10.006

Quak HJ, de Koster MBM (2009) Delivering Goods in Urban Areas: How to Deal with Urban Policy Restrictions and the Environment. Transp Sci 43:211–227. https://doi.org/10.1287/trsc.1080.0235

Rabbouch B, Saâdaoui F, Mraihi R (2019) Efficient implementation of the genetic algorithm to solve rich vehicle routing problems. Oper Res Int J. https://doi.org/10.1007/s12351-019-00521-0

Rajabi-Bahaabadi M, Shariat-Mohaymany A, Babaei M, Vigo D (2019) Reliable vehicle routing problem in stochastic networks with correlated travel times. Oper Res Int J. https://doi.org/10.1007/s12351-019-00452-w

Renaud J, Laporte G, Boctor FF (1996) A tabu search heuristic for the multi-depot vehicle routing problem. Comput Oper Res 23:229–235. https://doi.org/10.1016/0305-0548(95)O0026-P

Repoussis PP, Tarantilis CD, Bräysy O, Ioannou G (2010) A hybrid evolution strategy for the open vehicle routing problem. Comput Oper Res 37:443–455. https://doi.org/10.1016/j.cor.2008.11.003

Rothenbächer A-K (2019) Branch-and-Price-and-Cut for the Periodic Vehicle Routing Problem with Flexible Schedule Structures. Transp Sci 53:850–866. https://doi.org/10.1287/trsc.2018.0855

Rothenbächer A-K, Drexl M, Irnich S (2018) Branch-and-Price-and-Cut for the Truck-and-Trailer Routing Problem with Time Windows. Transp Sci 52:1174–1190. https://doi.org/10.1287/trsc.2017.0765

Russell RA, Chiang W-C (2006) Scatter search for the vehicle routing problem with time windows. Eur J Oper Res 169:606–622. https://doi.org/https://doi.org/10.1016/j.ejor.2004.08.018

Salari M, Toth P, Tramontani A (2010) An ILP improvement procedure for the Open Vehicle Routing Problem. Comput Oper Res 37:2106–2120. https://doi.org/10.1016/j.cor.2010.02.010

Schneider M, Stenger A, Goeke D (2014) The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. Transp Sci 48:500–520. https://doi.org/10.1287/trsc.2013.0490

Schrimpf G, Schneider J, Stamm-wilbrandt H, Dueck G (2000) Record Breaking Optimization Results Using the Ruin and Recreate Principle. J Comput Phys 159:139–171. https://doi.org/10.1006/jcph.1999.6413

Silva MM, Subramanian A, Ochi LS (2015) An iterated local search heuristic for the split delivery vehicle routing problem. Comput Oper Res 53:234–249. https://doi.org/https://doi.org/10.1016/j.cor.2014.08.005

Silva MM, Subramanian A, Ochi LS, et al (2014) An iterated local search heuristic for the split delivery vehicle routing problem. Eur J Oper Res 53:454–464. https://doi.org/https://doi.org/10.1016/j.cor.2014.08.005

Sitek P (2014) A Hybrid Approach to the Two-Echelon Capacitated Vehicle Routing Problem (2E-CVRP). In: Szewczyk R, Zieliński C, Kaliczyńska M (eds) Recent Advances in Automation, Robotics and Measuring Techniques. Springer, Cham, pp 251–263

Solomon MM (1987) Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Oper Res 35:254–265. https://doi.org/10.1287/opre.35.2.254

Subramanian A, Drummond LMAA, Bentes C, et al (2010) A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. Comput Oper Res 37:1899–1911. https://doi.org/https://doi.org/10.1016/j.cor.2009.10.011

Subramanian A, Penna PHV, Uchoa E, Ochi LS (2012) A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem. Eur J Oper Res 221:285–295. https://doi.org/https://doi.org/10.1016/j.ejor.2012.03.016

Taillard É, Badeau P, Gendreau M, et al (1997) A tabu search heuristic for the vehicle routing problem with soft time windows. Transp Sci 31:170–186. https://doi.org/10.1287/trsc.31.2.170

Tarantilis CD, Diakoulaki D, Kiranoudis CT (2004) Combination of geographical information system and efficient routing algorithms for real life distribution operations. Eur J Oper Res 152:437–453. https://doi.org/https://doi.org/10.1016/S0377-2217(03)00035-3

Tarantilis CD, Kiranoudis CT (2007) A flexible adaptive memory-based algorithm for real-life transportation operations: Two case studies from dairy and construction sector. Eur J Oper Res 179:806–822. https://doi.org/https://doi.org/10.1016/j.ejor.2005.03.059

Tarantilis CD, Stavropoulou F, Repoussis PP (2012) A template-based Tabu Search algorithm for the Consistent Vehicle Routing Problem. Expert Syst Appl 39:4233–4239. https://doi.org/https://doi.org/10.1016/j.eswa.2011.09.111

Usberti FL, França PM, França ALM (2013) GRASP with evolutionary path-relinking for the capacitated arc routing problem. Comput Oper Res 40:3206–3217. https://doi.org/https://doi.org/10.1016/j.cor.2011.10.014

Vidal T, Crainic TG, Gendreau M, Prins C (2013) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput Oper Res 40:475–489. https://doi.org/10.1016/j.cor.2012.07.018

Wang C, Mu D, Zhao F, Sutherland JW (2015) A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. Comput Ind Eng 83:111–122. https://doi.org/10.1016/j.cie.2015.02.005

Wang Y, Ma X, Li Z, et al (2017) Profit distribution in collaborative multiple centers vehicle routing problem. J Clean Prod 144:203–219. https://doi.org/https://doi.org/10.1016/j.jclepro.2017.01.001

Wang Y, Zhang S, Guan X, et al (2020) Collaborative multi-depot logistics network design with time window assignment. Expert Syst Appl 140:112910. https://doi.org/https://doi.org/10.1016/j.eswa.2019.112910

Xu Z, Cai Y (2018) Variable neighborhood search for consistent vehicle routing problem. Expert Syst Appl 113:66–76. https://doi.org/https://doi.org/10.1016/j.eswa.2018.07.007

Yassen ET, Ayob M, Nazri MZA, Sabar NR (2015) Meta-harmony search algorithm for the vehicle routing problem with time windows. Inf Sci (Ny) 325:140–158. https://doi.org/https://doi.org/10.1016/j.ins.2015.07.009

Yu B, Yang ZZ (2011) An ant colony optimization model: The period vehicle routing problem with time windows. Transp Res Part E Logist Transp Rev 47:166–181. https://doi.org/10.1016/j.tre.2010.09.010

Yu G, Yang Y (2019) Dynamic routing with real-time traffic information. Oper Res Int J 19:1033–1058. https://doi.org/10.1007/s12351-017-0314-9

Yu VF, Redi AANP, Yang C-L, et al (2017) Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem. Appl Soft Comput 52:657–672. https://doi.org/https://doi.org/10.1016/j.asoc.2016.10.006

Zachariadis EE, Kiranoudis CT (2010) An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. Comput Oper Res 37:712–723. https://doi.org/10.1016/j.cor.2009.06.021

Zachariadis EE, Tarantilis CD, Kiranoudis CT (2016) The Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries and Two-Dimensional Loading Constraints. Eur J Oper Res 251:369–386. https://doi.org/https://doi.org/10.1016/j.ejor.2015.11.018

Zhong Y, Cole MH (2005) A vehicle routing problem with backhauls and time windows: a guided local search solution. Transp Res Part E Logist Transp Rev 41:131–144. https://doi.org/https://doi.org/10.1016/j.tre.2003.12.003

Chapter 3: The Vehicle Routing Problem with Time Windows

3.1 Introduction

The Vehicle Routing Problem with Time Windows (VRPTW) is a well-known variant of the VRP, which has received considerable attention in recent years and reflects many real-world scenarios in distribution operations. The VRPTW consists of geographically scattered customers that need to be served within a predetermined time interval (time window), only once and by a single vehicle. The total quantity of goods distributed in each route cannot exceed the vehicle's capacity, while each vehicle starts and ends its route at the depot. A vehicle may arrive at a customer before the opening of the time window and wait until the agreed service time, but is not allowed to deliver goods if it arrives after the time window closes (Nagata et al. 2010; Ait Haddadene et al. 2019; Juárez Pérez et al. 2019). Instead, many research papers (Koskosidis et al. 1992; Taillard et al. 1997a; Hashimoto et al. 2006; Iqbal and Rahman 2012; Taş et al. 2014; Wu et al. 2019) consider that some or all customers have soft time windows and may be served before the opening of the time window or after the time window closes, paying a penalty parameter each time the time window is violated. Time windows become even more challenging in urban transportation where uncertain traffic conditions exist, meaning that traveling times change dynamically. In the current Chapter, as well as in a greater attempt to cover the requirements of most distribution cases, the time windows are considered hard and cannot be violated as this is a more realistic scenario for urban freight transportation, where a possible delay may have severe consequences.

Moreover, solving the VRPTW is of great interest both for the research community and logistics and transportation companies, as it is crucial for delivering goods cost-effectively and facing customer requirements. If not all, most customers have the need to define a time slot in which they must be served, making the VRPTW the most common variant both in research and in industry. The connection between the two fields is a two-way one, where companies and their associates and customers determine the constraints that are transformed into mathematical formulas and modeled by researchers. Researchers, aided by advanced computer science, propose and develop algorithms that solve the VRPTW. However, not all algorithms are easy to implement to solve real-life problems, mainly due to their computation time. Starting with exact algorithms, they compute every possible solution until the best one is reached, while performing very poorly in computation time, even when dealing with fairly small instances (El-Sherbeny 2010). Despite this limitation, multiple exact algorithms have been proposed by researchers (Kallehauge 2008; Qureshi et al. 2009; Azi et al. 2010; Baldacci et al. 2012) for tackling the VRPTW, all of which solve distribution cases with up to 100 customer orders. Consequently, since real-life cases include, in most cases, more than 100 customer orders, researchers aim to propose and develop algorithms that can handle bigger size cases.

According to Arnold and Sörensen (2019b), heuristic algorithms provide the best trade-off between solution quality and computation time. This theory is reinforced by some paradigms of the successful implementation of heuristic algorithms solving the VRPTW. More specifically, the work of Solomon (1987) is considered pioneer as multiple algorithms where proposed, including the Time-oriented Nearest Neighbor (TONN), which constructs routes sequentially, by finding the "closest" customer to the last one served and ends when no more unrouted customers left. Additionally, Pisinger and Ropke (2007) proposed a unified heuristic that can solve a class of VRP variants, including time windows. Taillard et al. (1997a) developed a TS heuristic considered the time windows as soft.

Even though heuristics offer good quality solutions, the latest advances in technology have enabled metaheuristic algorithms to search a wider solution space in limited CPU time, while offering efficient results (Kalayci and Kaya 2016). Metaheuristics categorized in Population Search and in Local Search are developed quite often for addressing the VRPTW. In most cases, the VRPTW is faced as a single objective as the individual objective functions are composed of a single one. This approach may significantly simplify the problem; however, it is not very reliable as a small variation in the weights may lead to different solutions (Konak et al. 2006). Tan et al. (2006a) and Ombuki et al. (2006) are some of the first that considered the VRPTW as multiobjective while also applying the Pareto optimality concept. In both articles, evolutionary algorithms were developed and proposed, and then tested in Solomon's benchmark instances, offering new Pareto optimal solutions. Ever since, multiple researchers have studied and considered the multiobjective nature of the problem, as advances in technology and increasing computing capabilities have emerged. In many cases, multiobjective evolutionary algorithms (MOEA) are proposed or combined with other methods due to their search capabilities and their efficiency. Some very efficient multiobjective algorithms, according to their solutions in Solomon's database, were proposed by Chiang and Hsu (2014), Garcia-Najera and Bullinaria (2011), and Baños et al. (2013).

Concerning the contribution of this Chapter, it is noted in encountering the problem as multiobjective and developing a Multiobjective Large Neighborhood Search (MOLNS) algorithm that exploits the solution obtained from the construction heuristic algorithm. The proposed MOLNS algorithm is sufficient in minimizing both the number of vehicles and the total traveled distance by applying destroy and repair operators while also maintaining the concept of Pareto optimality (Yun et al. 2016). To the best of my knowledge, this is the first time a Large Neighborhood Search algorithm is applied in the multiobjective VRPTW. The algorithm is being tested in Solomon's 56 benchmark instances with 100 customers. The results of the algorithm are compared to the best-published results of the literature, indicating a very efficient algorithm. The proposed algorithm considers the problem as multiobjective and can offer more competitive solutions with a viable trade-off between the quality of the solution and the computation time.

In the remainder of the Chapter, the VRPTW is defined and formulated in Section 3.2, while in Section 3.3, the proposed MOLNS algorithm aiming to solve real-life VRPTW cases, is thoroughly presented. The computation results of the proposed algorithm in Solomon's benchmark instances are presented in Section 3.4. In the same section, the expected application of the algorithm in routing and scheduling software is discussed. Finally, Section 3.5 contains the conclusions of the Chapter.

3.2 Problem Description and Mathematical Formulation

The VRPTW is defined by a set of identical vehicles denoted by K, and by a direct network *G (N, A)*, where N is the set of nodes and $A = (i, j): i \neq j, i, j \in N$ is the set of arcs. Node *0* represents the central depot, while $N^* = \{N/0\}$ represents customers. Every arc *(i, j)*, which is a path from node i to node j is characterized by a distance which is indicated as d_{ij}. Respectively t_{ij} stands for the travel time from node *i* to node *j*, and has the same value with d_{ij}, as the assumption that each distance unit corresponds to one specific time unit is made. Each customer i is characterized by the demand q_i, the service time needed s_i, and the time window (e_i, l_i), where e_i is the opening time and l_i the closing time. Customers must be served by exactly one vehicle that may arrive any time within the time window. The arrival time is given by a_i. Additionally, if the vehicle arrives before the beginning of the time window (e_i), it must wait for w_i^k time, until service is possible, while no vehicle may arrive after the end of the time interval (l_i). Additionally, all the essential variables are gathered and presented below, before the mathematical model.

d_{ij}	Distance between node i and node j
t_{ij}	Transit time between node i and node j
q_i	Demand of goods from node i
e_i	Node's i open time
l_i	Node's i close time
s_i	Service time needed at node i
a_i	Time of arrival of vehicle to node i
w_i^k	Waiting time of vehicle k at customer i

The decision variable x_{ij}^k is equal to 1 if vehicle k drives from node *i* to node *j*, and *0* otherwise. The objective of the VRPTW is to service all customers, minimizing the number of vehicles and the total traveled distance while simultaneously ensuring that all the constraints are satisfied. The mathematical model of the VRPTW is presented below:

$$Objective\ 1: \sum_{k \in K} \sum_{j \in N^*} x_{0j}^k \tag{3.1}$$

$$Objective\ 2: \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^k \tag{3.2}$$

$$\sum_{i\in N} q_i \sum_{j\in N} x_{ij}^k \le q_{max}^k, \forall k \in K \tag{3.3}$$

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1, \forall i \in N \tag{3.4}$$

$$\sum_{j\in N^*} x_{0j}^k = 1, \forall k \in K \tag{3.5}$$

$$\sum_{i\in N} x_{il}^k - \sum_{j\in N} x_{lj}^k = 0, \forall l \in N, \forall k \in K \tag{3.6}$$

$$\sum_{j\in N^*} x_{j0}^k = 1, \forall k \in K \tag{3.7}$$

$$a_0 = w_0^k = s_0 = 0 \tag{3.8}$$

$$\sum_{j\in N} x_{ij}^k (a_i + w_i^k + s_i + t_{ij}) \le l_j, \forall i \in N, \forall k \in K \tag{3.9}$$

$$e_i \le a_i + w_i^k \le l_i, \forall i \in N, \forall k \in K \tag{3.10}$$

$$x_{ij}^k \in \{0,1\}, \quad \forall (i,j) \in A, \forall k \in K \tag{3.11}$$

The objective function (3.1) states that the total number of vehicles and the total traveled distance should both be minimized based on the objective function (3.2). The constraint (3.3) states that a vehicle's capacity cannot be exceeded and set (3.4) that each customer must be served exactly once and by one vehicle. The set of constraints (3.5), (3.6) and (3.7) make sure that each vehicle starts from the depot *{0}*, visits and serves a certain number of customers, and finally returns to the depot *{0}*. Constraints (3.8), (3.9) and (3.10) indicate that for a trip from node *i* to node *j*, no vehicle may arrive at customer *j* after the end of the time window, (l_i). Simultaneously, the time of arrival to a customer, depends on the time of arrival, the waiting time and the service time, to the previous served customer. An example of the VRPTW is illustrated in Figure 3.1.

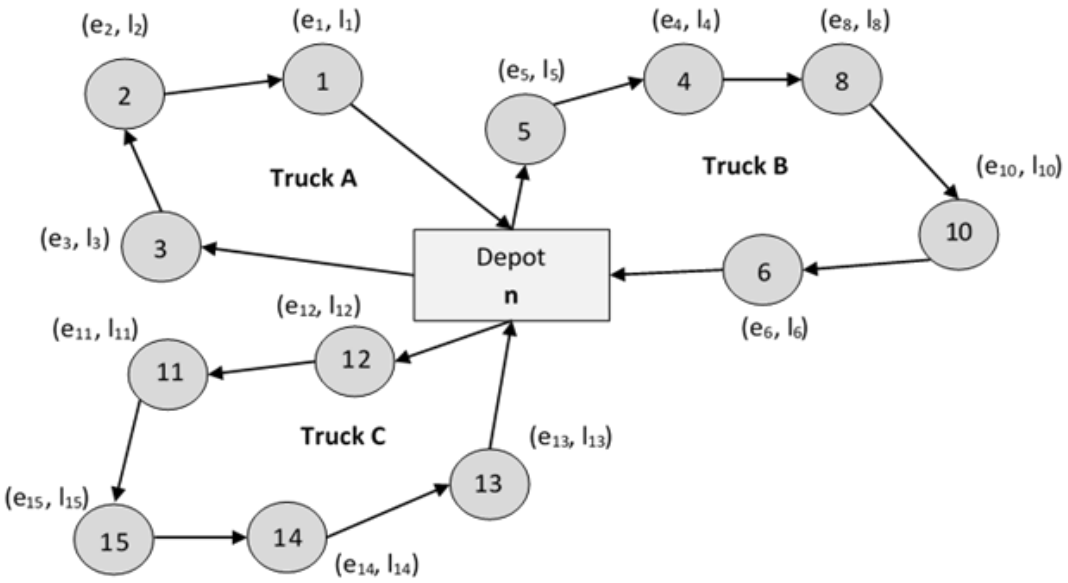


Figure 3.1 Illustration of the VRPTW

3.3 Problem Solution

The proposed MOLNS algorithm, as a first step, obtains the initial solution by applying the Time-oriented nearest neighbor algorithm, which is a construction heuristic one. Consequently, in each iteration, the destroy and repair operators are applied for improving the solution. If the newly obtained solution is better than the current best one, then it replaces it, and it is used as input in the next iteration. To evaluate whether a solution is better than another, Pareto optimality concept is applied, and it is thoroughly analyzed in Section 3.3.1. The construction heuristic algorithm is described in Section 3.3.2. In Section 3.3.3 and 3.3.4, the destroy and repair operators are presented respectively. Finally, the general framework of the MOLNS algorithm is presented in Section 3.3.5. The steps mentioned above are given in Figure 3.2.

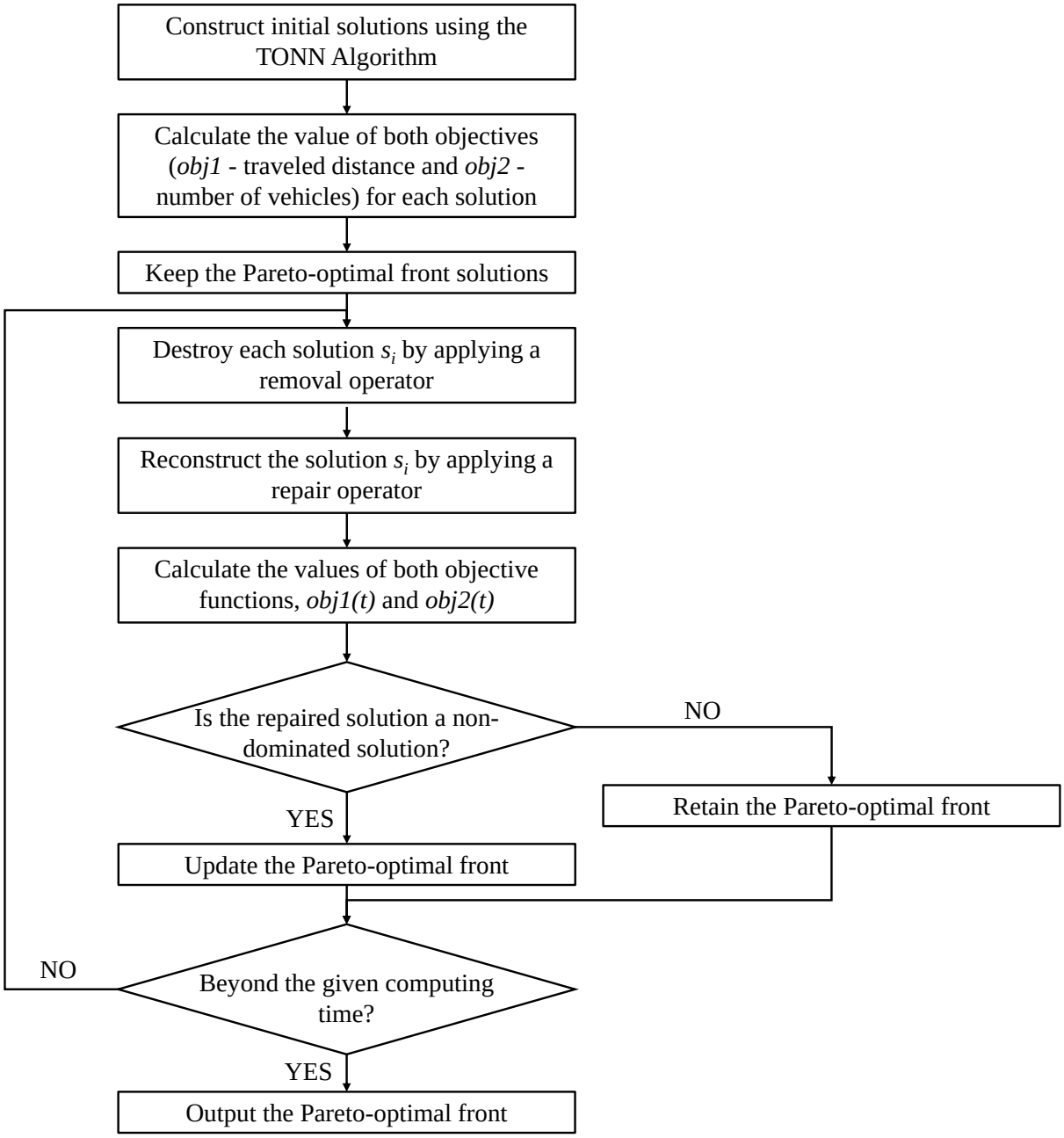


Figure 3.2 Steps of the MOLNS Algorithm

3.3.1 Pareto Optimal

Deliveries to end-customers and consumers constitute a key part of supply chain and logistics operations. Multiple stakeholders are involved, such as third-party logistics (3PL) companies, transportation companies or even production companies that have their own fleet of vehicles. Depending on the case, either the number of vehicles or the total traveled distance is more important or contributes more to the cost. Therefore, it is important to obtain every possible non-dominated solution, and each stakeholder makes the decision and selects the solution that is most profitable (Ehrgott 2005). More specifically, as shown in Figure 3.2 (a), when moving from one Pareto optimal solution to another, there is always a decrease in one objective and a

simultaneous increase in the second objective. More specifically, when comparing solutions X_2 and X_3 of a minimization problem, it is observed that the solution X_2 has a lower value in objective 2 and a higher value in objective 1. Therefore, it cannot decide which solution is more profitable with no further information as it depends on the case.

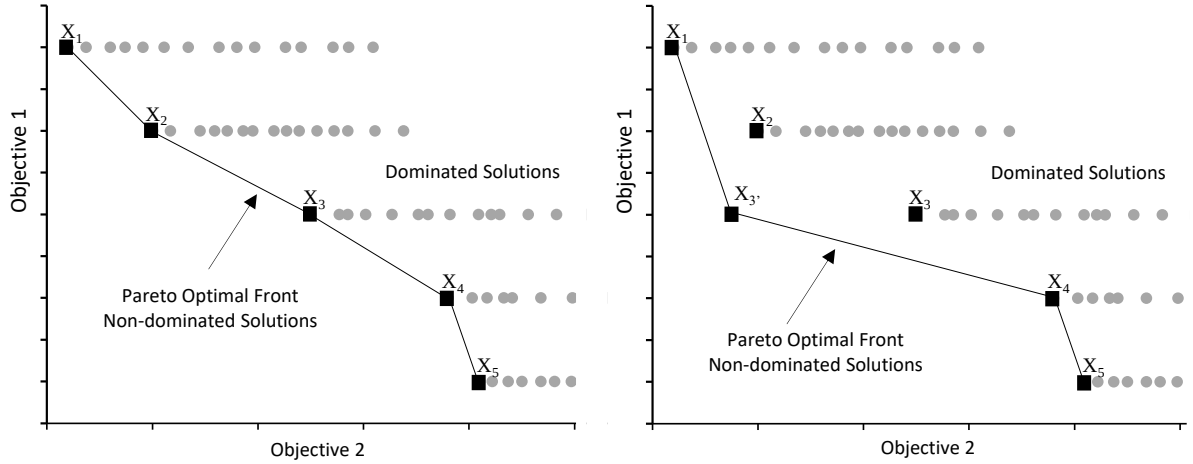


Figure 3.3 Pareto Optimal Front

Both in the phase of construction and in the phase of the LNS algorithm, every obtained solution is checked for belonging either to the dominated or the non-dominated solutions as shown in Figure 3.3 (a). If a solution obtained from the algorithm belongs to the non-dominated solutions, then the Pareto optimal front is updated. More specifically, in each iteration of the MOLNS, the set of optimal solutions (X_1, X_2, X_3, X_4, X_5), that belong in the initial Pareto optimal front are selected, as presented in Figure 3.2 (a). In case solution $X_{3'}$ is obtained after implementing the destroy and repair operators of the proposed MOLNS algorithm, which improves the value of objective 2 compared to solution X_3. Both values of objective 1 and objective 2 compared to solution X_2, the Pareto optimal front must be updated as shown in Figure 3.3 (b). Therefore, every solution obtained from the proposed algorithm is compared to the non-dominated solutions to ensure optimal solutions are maintained and continuously updated.

3.3.2 Time-Oriented Nearest Neighbor Algorithm

Several heuristic algorithms have been developed for obtaining solutions for the VRP. In the present Chapter, the Time-oriented Nearest Neighbor (TONN) algorithm is selected for obtaining the initial solutions due to its simplicity and speed. The algorithm works by building routes that start from the depot (node 0), and consequently, the "closest" to the last visited node unrouted customer is added. In searching the "closest" customer, time window constraints and capacity constraints must be respected; otherwise, no customer can be added. A new route is started if no customer is found, unless no more customers are left to add, meaning that all routes have been constructed and the process is completed. Solomon (Solomon 1987) proposed this method in an attempt to solve the VRPTW optimally.

The "closeness" of each customer is measured through metric c_{ij}. This metric measures the direct distance between the two customers d_{ij}, the time difference between the completion of service at *i* and the beginning of service at *j*, T_{ij}, and the urgency of delivery to customer *j*, u_{ij}.

$$T_{ij} = (a_j + w_j^k) - (a_i + w_i^k + s_i)$$
$$u_{ij} = l_j - (a_i + w_i^k + s_i + t_{ij})$$
$$c_{ij} = \beta_1 \cdot d_{ij} + \beta_2 \cdot T_{ij} + \beta_3 \cdot u_{ij}, \qquad \beta_1 + \beta_2 + \beta_3 = 1$$

3.3.3 Multiobjective Large Neighborhood Search

As described in Section 3.3.2, customers are added in each route, based on their "closeness" to the last added customer. However, time windows affect to a great extent the initial solution, which in many cases, deviates from the optimum. Therefore, after the initial solutions are constructed, the MOLNS algorithm that partially destroys and then repairs the solution is applied to minimize both the number of vehicles and the total distance traveled. The general framework of the LNS algorithm was designed by Ropke and Pisinger (Ropke and Pisinger 2006). The destroy and repair moves are thoroughly analyzed in Sections 3.3.4.1 and Section 3.3.4.2.

3.3.3.1 Destroy Operators

In this section, five different destroy operators are described for selecting the customers who will be removed from their routes. Most operators are adapted by Ropke and Pisinger (2006) and Demir et al. (2012), while a new one (Distant and Waiting-time destroy) is presented. From all destroy operators, *s* customers are selected and removed from their routes.

3.3.3.1.1 Route Destroy

The specific operator removes an entire route from the solution. In each iteration, the route is randomly selected and "destroyed" from the solution. In a later phase, and since all customers must be served, the customers of the "destroyed" route are reinserted according to the insertion operator.

3.3.3.1.2 Random Customers Destroy

In this operator, a specific number of customers (s) removed from the existing solution is initially determined. Consequently, an empty list of size *s* is created and filled with customers who are randomly selected. The random nature of this operator increases the diversity, as well as the searching space. As shown in Figure 3.3, three customers are randomly selected (*s = 3*) and removed from the route that each one belongs.

3.3.3.1.3 Distant Customers Destroy

In this operator, as with the case of random customers destroy, the number of customers (*s*) removed from the solution is determined. Consequently, for each customer k the metric $c_k = d_{ik} + d_{kj}$ is calculated, where *i* is the previous delivered customer and *j* the next delivered

customer, according to *k*, while d indicates the distance between two customers. Then, all customers are sorted according to the metric ck and select the s "worst" customers (highest c_k).

3.3.3.1.4 Waiting-time Destroy

For each customer to be served, the metric wt calculates the difference between the start of the time window and arrival time. More specifically, the metric wt for customer *i*, is calculated by $wt_i = \max\{0, e_i - t_i\}$, where t_i is the arrival time at customer *i*. Consequently, customers are sorted according to metric wt_k and *s* number of customers that have the highest waiting times (wt_i) are selected.

3.3.3.1.5 Distant and Waiting-time Destroy

This operator combines in a single metric both the distant customers (3.3.3.1.3) and the waiting-time (3.3.3.1.4) operators, by assigning weights to both metrics. Respectively to the previous operators, a specific number of customers (***s***), with the worst metric are selected.

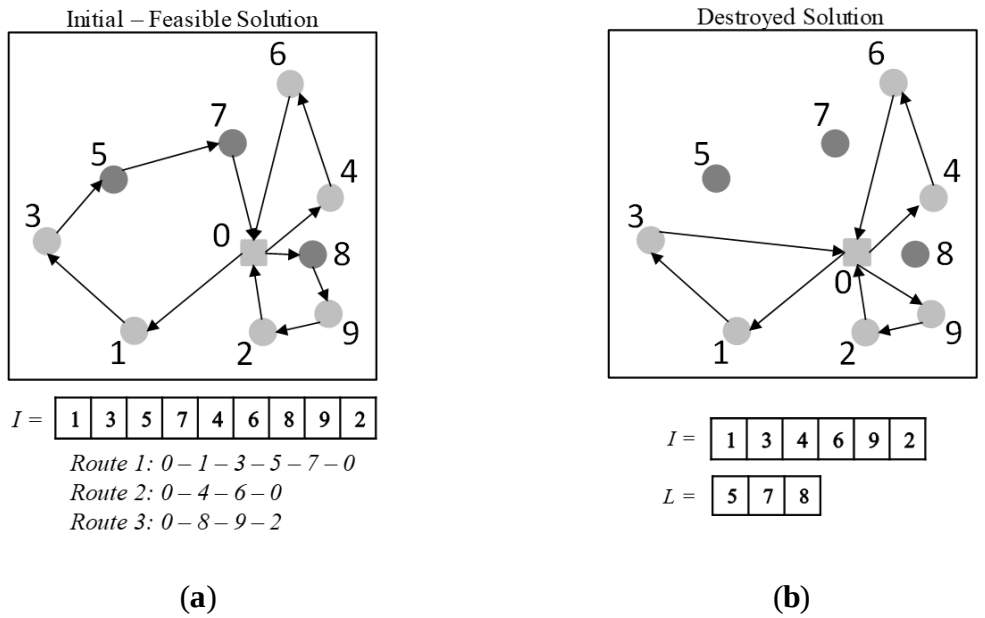


Figure 3.4 Destroy Operator

Regardless of the destroy operator, the initial solution is presented as an *I* list containing all customers. The sequence of the elements of I determines the order of customer visitations. When a constraint is violated (capacity or time window), then a new route begins, as shown in Figure 3.4 (a). Also, after the destroy operator is applied, the customers that no longer belong to the route are saved in another list *L*, as shown in Figure 3.4 (b).

3.3.3.2 Repair Operators

After a destroy operator is applied, a repair operator must be implemented so that all customers are reinserted in the solution. In the VRPTW there is a necessary precondition to serve all

customers' needs. In the current Chapter, three repair operators are applied for improving the initial constructed solution and are analyzed.

3.3.3.2.1 Closer Customer Repair

This operator cooperates with all removal operators. The selected customers of list L, are reinserted in the routes and, more specifically, in the sequence of deliveries where the total distance traveled is minimized. The specific repair procedure is separated into two different operators. The first one searches for the minimum distance while the number of vehicles must remain the same as before the removal operator was applied. In addition, the second operator searches for the minimum distance regardless of the number of vehicles.

3.3.3.2.2 Repair for Minimizing the Number of Vehicles

The specific repair operator is combined only with the route removal operator to decrease the number of vehicles. If all customers of the selected route can be inserted in the rest of the routes, then the number of vehicles is reduced by one, as shown in Figure 3.5. Simultaneously, the algorithm seeks to find the best sequence and schedule of deliveries to manage the distance traveled. However, the fact that VRPTW is a multi-objective optimization problem does not guarantee us that both the number of vehicles and the total distance traveled are simultaneously minimized. Instead, decreasing the number of vehicles by reinserting "destroyed" customers in the best-fitted sequence of deliveries, may cause increased total distance traveled. Figure 3.5 describes how the route removal operator is combined with the specific repair operator. More specifically, among route 1: $\{0 \rightarrow i^- \rightarrow i \rightarrow i^+ \rightarrow 0\}$ route 2: $\{0 \rightarrow n \rightarrow m \rightarrow 0\}$ and route 3: $\{0 \rightarrow j^- \rightarrow j \rightarrow j^+ \rightarrow 0\}$, the second one (route 2) is randomly selected and "destroyed". In the search for inserting customers n and m in the rest of the routes, route 1': $\{0 \rightarrow i^- \rightarrow m \rightarrow i \rightarrow i^+ \rightarrow 0\}$ and route 3': $\{0 \rightarrow j^- \rightarrow n \rightarrow j \rightarrow j^+ \rightarrow 0\}$ are formed.

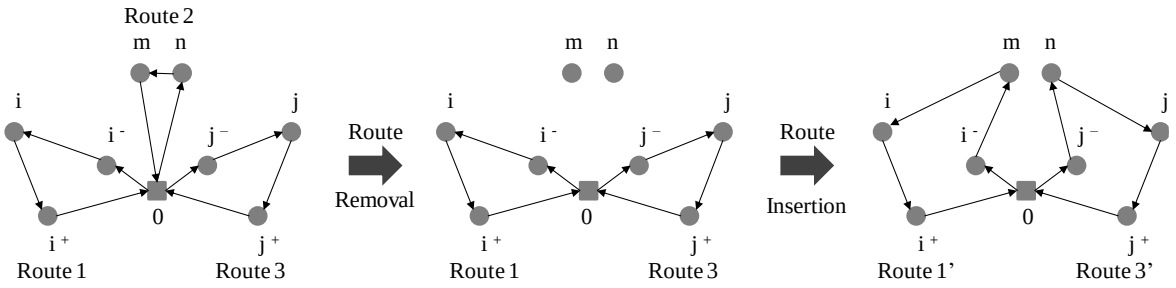


Figure 3.5 Route Destroy and Repair

When implementing the destroy and repair operators, it is essential to ensure the diversity of searching and the global search capabilities. Therefore, after saving the removed customers in list L, as described in Section 3.3.1, the sequence of insertion must be considered. Repeating the same sequence of customers insertion may lead to getting stuck in a local optimum. In the presented repair operators, a random order for inserting customers is produced, for increasing the search space of the algorithm.

3.3.4 General Framework

Initially, the Pareto optimal solutions constructed from the TONN algorithm, are utilized as an input to the MOLNS algorithm, described in Figure 3.6. In each iteration that the MOLNS algorithm is implemented, the destroy and repair operators are applied in each one of the Pareto optimal solutions. Consequently, for each solution, a new one - X_{new} is obtained. If X_{new} solution constitutes a Pareto optimal solution, as it belongs to the non-dominated solutions, then the Pareto optimal front is renewed; otherwise, no changes are made.

```
1.  Construct solutions with the Time-oriented Nearest Neighbor heuristic
2.  Save Pareto optimal solutions in P
3.  While stopping criteria not met do
4.      For each Pareto optimal solution – X_p do
5.          Apply Removal and Insertion operators in X_p
6.          New solution is produced – X_new
7.          If X_new is a Pareto optimal solution then
8.              Add X_new in Pareto optimal solutions (P)
9.              Update the Pareto optimal front
10.         Else
11.             Pareto optimal solutions (P) do not change
12.         End If
13.     Next
14. End while
```

Figure 3.6 The General Framework of the MOLNS

3.4 Computational Study

This section describes computational experiments carried out to evaluate the performance of the proposed algorithm. The algorithm was coded in Python and ran on a personal computer with the following specifications: Intel Core i7 - 8550U 1.8GHz with 16GB memory. The proposed algorithm was tested in Solomon's VRPTW benchmark instances with 100 customers, available at Solomon (1987). In the specific dataset, C1 and C2 problems have geographically clustered customers. In R1 and R2, geographical data are randomly generated, while in problem sets RC1 and RC2, a mix of random and clustered structures are included. Problem sets R1, C1 and RC1 have a short scheduling horizon, narrow time windows, and low vehicle capacities, while sets R2, C2, and RC2 have a long scheduling horizon, wide time windows, and high vehicle capacities, defining the number of customers serviced by the same vehicle.

Table 3.2 and Table 3.3 present a summary of the results obtained from the proposed MOLNS algorithm, as well as the best-published solutions in the literature. The column labeled BP gives

the best-published results, while column MOLNS, the results of the Large Neighborhood Search algorithm. Each solution is characterized by the number of vehicles (NV) needed and the total distance traveled (TD). The published work in which the optimum solution is proposed is presented in column "Reference", the computation time needed to obtain each solution is given in column “Time” in seconds, while the percentage deviation in the total distance traveled between the BP and the LNS is presented in column "Deviation".

Each instance runs three times, while the computation time limit is set to be 15 minutes, which is a reasonable time for logistics companies to obtain their plan of deliveries. This statement is enhanced both by other studies on Solomon’s instances, as shown in Table 3.1, as well as by the computational study of chapter 6. More specifically, as Table 3.1 shows, most studies run the algorithm multiple times to obtain the optimal results, while the computation time ranges from half a second to an hour. Moreover, in chapter 6 multiple real-life instances are solved by another similar LNS algorithm, and even instances with up to 750 customers can be efficiently solved in 15 minutes. Consequently, the time limit of 15 minutes seems to be reasonable when compared to other studies as well as to the computational results of the current thesis. Finally, in most cases, the best-obtained results needed less time than the time limit to obtain the optimal solution. More specifically, the average time needed for obtaining each optimal solution is approximately 1 minute.

Table 3.1 Average Computation Time, Computing Environment and Runs

Approach	Average computation time (sec)					CPU (Language)
	R1	R2	RC1	RC2	#runs	
Ombuki et al. (2006)	not found in the original paper				10	1.6 GHz (Matlab)
Lim and Zhang (2007)	1576.8				1	Pentium IV 2.8 GHz (Java)
Alvarenga et al. (2007)	3600				3	not found in the original paper
Labadi et al. (2008)	151.8	210.9	152.3	199.6	1	3 GHz (Delphi)
Nagata et al. (2010)	300				5	Opteron 2.4 GHz (C++)
Ghoseiri and Ghannadpour (2010)	>500	>900	>500	>1300	10	1.6 GHz (Matlab)
Garcia-Najera and Bullinaria (2011)	not found in the original paper				30	2218 2.6 GHz nodes
Chiang and Hsu (2014)	12	23	10.5	19.1	10	Intel i7-3770 3.4 GHz (C++)

Moreover, since the VRPTW is faced as multiobjective, in some instances, more than one solution was obtained. In order for the BP solution and the solution obtained from the MOLNS to be comparable, the number of vehicles must be the same. Therefore, the deviation is measured in the total distance traveled between the BP solution and the solution of the MOLNS. It can be easily observed that instances with geographically clustered customers (C1 and C2) have more efficient results than problems with random geographical data, as the mean deviation, in that case, is 0.30%. In randomly generated geographical data (R), the mean deviation is higher and reaches 3.37%, while in mixed geographical customers' data (RC) is 3.47%. Finally, the mean deviation for all instances is less than 2.85%, which can be considered very efficient since the optimal results have emerged from multiple researchers over time. In total, 32.47% of the comparable solutions have a deviation of less than 1%, most of them in C instances. Simultaneously, 98.70% of the results deviate less than 10% from the optimum. Finally, three new Pareto-optimal solutions are proposed in Solomon's benchmark instances, in RC202, RC207, and R201. In RC202 problem, a decrease in the total distance traveled is accomplished, but at the expense of an increased number of vehicles. In R201 and RC202 instances, the algorithm managed to improve the solutions by decreasing the total distance traveled, while the number of vehicles remains steady. The exact routes and the schedule of deliveries in each of the three new optimal solutions that are proposed in the current research are given in Appendix B.

Additionally, when comparing results obtained from algorithms, it is crucial to consider the computation time needed. Chiang and Hsu (2014) manage to produce high quality solutions within a half-minute and thoroughly present data related to the computation time for other algorithms addressing this problem. Ghoseiri and Ghannadpour (2010) and Ombuki et al. (2006) that also consider the VRPTW as multiobjective, do not present the computation time needed, while the proposed algorithm manages to produce high-efficiency results in less than a minute (52 seconds on average). Of course, the CPU and the programming language are also important factors affecting the computation time needed.

Table 3.2 Results of the MOLNS in Solomon Instances with Short Scheduling Horizon

Problem Type	MOLNS NV	MOLNS TD	BPNV	BPTD	Deviation (%)
C101	10	828.94	10	827.3	0.20
C102	10	828.94	10	827.3	0.20
C103	10	828.94	10	826.3	0.32
C104	10	828.94	10	822.9	0.73
C105	10	828.94	10	827.3	0.20
C106	10	828.94	10	827.3	0.20
C107	10	828.94	10	827.3	0.20
C108	10	828.94	10	827.3	0.20
C109	10	828.94	10	827.3	0.20

R101	19	1654.93	18	1607.7	-
R102	18	1475.33	17	1434	-
R103	14	1240.44	13	1175.67	-
R104	10	1010.72	10	974.2	3.75
R105	15	1389.85	15	1346.12	3.25
R106	13	1269.14	13	1234.6	2.80
R107	11	1102.72	11	1051.84	4.84
R108	10	991.57	10	942.9	5.16
R109	12	1177.76	12	1101.99	6.88
R110	12	1129.60	12	1068	5.77
R111	12	1108.70	12	1048.7	5.72
R112	10	964.15	10	953.63	1.10
RC101	15	1662.56	15	1619.8	2.64
RC102	14	1486.35	14	1461.33	1.71
RC103	12	1291.95	12	1196.12	8.01
RC104	10	1162.53	10	1135.48	2.38
RC105	15	1604.53	15	1519.29	5.61
	16	1575.31	16	1518.6	3.73
RC106	13	1400.09	13	1371.69	2.07
RC107	12	1259.55	12	1212.83	3.85
RC108	11	1205.13	11	1117.53	7.84

Table 3.3 Results of the MOLNS in Solomon Instances with Long Scheduling Horizon

Problem Type	**MOLNS NV**	**MOLNS TD**	**BPNV**	**BPTD**	**Deviation (%)**
C201	3	591.56	3	589.1	0.42
C202	3	591.56	3	589.1	0.42
C203	3	591.56	3	591.17	0.07
C204	3	590.6	3	590.6	0.00
C205	3	588.88	3	586.4	0.42
C206	3	588.49	3	586	0.42
C207	3	588.29	3	585.8	0.43
C208	3	588.32	3	585.8	0.43
R201	4	1305.25	4	1252.37	4.22
	5	1208.55	5	1193.29	1.28
	6	1174.98	6	1171.2	0.32
	7	1156.73	7	1173.75	-1.45
R202	4	1093.67	4	1079.39	1.32
	5	1065.73	5	1041.1	2.37

R203	4	915.43	4	901.2	1.58
	5	901.72	5	890.50	1.26
R204	3	775.99	3	749.42	3.55
	4	750.32	4	743.23	0.95
R205	3	1075.1	3	994.43	8.11
	4	975.21	4	959.74	1.61
	5	964.23	5	954.1	1.06
R206	3	979.21	3	906.14	8.06
	4	909.83	4	889.39	2.30
	5	907.35	5	879.89	3.12
R207	3	851.89	3	812.76	4.81
R208	2	754.99	2	725.75	4.03
	3	731.84	3	706.86	3.53
R209	4	898.23	4	864.15	3.94
R210	4	941.58	4	924.79	1.82
R211	3	838.14	3	767.82	9.16
	4	782.75	4	755.82	3.56
RC201	4	1497.89	4	1406.91	6.47
	5	1329.59	5	1279.65	3.90
	6	1296.83	-	-	-
	7	1284.48	7	1273.51	0.86
	8	1281.81	8	1272.28	0.75
RC202	4	1199.53	4	1162.54	3.18
	5	1140.2	5	1118.66	1.93
	7	1109.21	-	-	-
	8	1104.94	8	1099.54	0.49
RC203	4	985.54	4	945.08	1.42
	5	938.04	5	926.82	1.21
RC204	3	805.46	3	798.41	0.88
RC205	5	1340.38	5	1236.78	8.38
	6	1223.50	6	1187.98	2.99
	7	1162.43	7	1161.81	0.05
RC206	3	1316.42	3	1146.32	14.84
	4	1121.83	4	1081.83	3.70
	5	1097.07	5	1068.77	2.65
RC207	4	1031.62	4	1001.85	2.97
	5	970.78	5	982.58	-1.20
RC208	3	859.13	3	828.14	3.74
	4	810.99	4	783.035	3.57

3.5 Concluding Remarks

The results obtained from the MOLNS algorithm were compared to the best-published solutions in Solomon's dataset, providing us significant insights about this method. Based on the conducted research, no algorithm can obtain all Pareto optimal solutions in each instance in Solomon's dataset as it becomes clear. Instead, the results have been conjunctly obtained from multiple algorithms since the VRPTW firstly appeared. The results of the proposed MOLNS algorithm are near-optimal, with only minor deviations from the optimal, and even include three new non-dominated solutions. The computational experiments indicate that the results are efficient and are obtained in a very realistic time. As both efficiency and speed are critical factors when implementing an algorithm in software solutions, the proposed MOLNS algorithm can be examined for implementation.

Additionally, the fact that the algorithm is a multiobjective one gives the opportunity to select the solution that best fits the needs. That is significant in cases that the best trade-off between two objectives needs to be found. The algorithm's benefits seem to be significant and that is why each algorithm constitutes a library that includes functions for addressing the problem. The algorithm libraries are the core of the system and are stored into the server of the software development company, so that it can be accessed by the presented in Chapter 6 system through the appropriate request. In a later phase, that potential purchasers will define if two objectives can be useful, the system could easily offer users the ability to select which algorithm to be applied.

Furthermore, in order to define the efficiency of the algorithm in terms of running time another multiobjective metaheuristic algorithm that can solve the VRPTW is developed. The algorithm is an Evolutionary Algorithm (EA) that handles simultaneously time windows, as well as simultaneous pickups and deliveries. The EA is presented below in Chapter 4.

References

Ait Haddadene S, Labadie N, Prodhon C (2019) Bicriteria Vehicle Routing Problem with Preferences and Timing Constraints in Home Health Care Services. Algorithms 12:152. https://doi.org/doi.org/10.3390/a12080152

Alvarenga GB, Mateus GR, de Tomi G (2007) A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. Comput Oper Res 34:1561–1584. https://doi.org/10.1016/j.cor.2005.07.025

Arnold F, Sörensen K (2019) What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. Comput Oper Res 106:280–288. https://doi.org/10.1016/j.cor.2018.02.007

Azi N, Gendreau M, Potvin J-Y (2010) An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. Eur J Oper Res 202:756–763. https://doi.org/https://doi.org/10.1016/j.ejor.2009.06.034

Baldacci R, Mingozzi A, Roberti R (2012) Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. Eur J Oper Res 218:1–6. https://doi.org/10.1016/j.ejor.2011.07.037

Baños R, Ortega J, Gil C, et al (2013) A hybrid meta-heuristic for multi-objective Vehicle Routing Problems with Time Windows. Comput Ind Eng 65:286–296. https://doi.org/10.1016/j.cie.2013.01.007

Chiang TC, Hsu WH (2014) A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows. Comput Oper Res 45:25–37. https://doi.org/10.1016/j.cor.2013.11.014

Demir E, Bektaş T, Laporte G (2012) An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. Eur J Oper Res 223:346–359. https://doi.org/https://doi.org/10.1016/j.ejor.2012.06.044

Ehrgott M (2005) Multicriteria optimization: Second edition

El-Sherbeny NA (2010) Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. J King Saud Univ 22:123–131. https://doi.org/10.1016/j.jksus.2010.03.002

Garcia-Najera A, Bullinaria JA (2011) An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Comput Oper Res 38:287–300. https://doi.org/10.1016/j.cor.2010.05.004

Ghoseiri K, Ghannadpour SF (2010) Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Appl Soft Comput 10:1096–1107. https://doi.org/10.1016/j.asoc.2010.04.001

Hashimoto H, Ibaraki T, Imahori S, Yagiura M (2006) The vehicle routing problem with flexible time windows and traveling times. Discret Appl Math 154:2271–2290. https://doi.org/10.1016/j.dam.2006.04.009

Iqbal S, Rahman MS (2012) Vehicle routing problems with soft time windows. In: 2012 7th International Conference on Electrical and Computer Engineering. pp 634–638

Juárez Pérez M., Pérez Loaiza RE, Quintero Flores PM, et al (2019) A Heuristic Algorithm for the Routing and Scheduling Problem with Time Windows : A Case Study of the Automotive Industry in Mexico. Algorithms 12:111. https://doi.org/10.3390/a12050111

Kalayci CB, Kaya C (2016) An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. Expert Syst Appl 66:163–175. https://doi.org/https://doi.org/10.1016/j.eswa.2016.09.017

Kallehauge B (2008) Formulations and exact algorithms for the vehicle routing problem with time windows. Comput Oper Res 35:2307–2330. https://doi.org/https://doi.org/10.1016/j.cor.2006.11.006

Konak A, Coit DW, Smith AE (2006) Multi-objective optimization using genetic algorithms: A tutorial. Reliab Eng Syst Saf 91:992–1007. https://doi.org/10.1016/j.ress.2005.11.018

Koskosidis YA, Powell WB, Solomon MM (1992) An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints. Transp Sci 26:69–85. https://doi.org/10.1287/trsc.26.2.69

Labadi N, Prins C, Reghioui M (2008) A memetic algorithm for the vehicle routing problem with time windows. RAIRO-Operations Res 42:415–431. https://doi.org/10.1051/ro:2008021

Lim A, Zhang X (2007) A Two-Stage Heuristic with Ejection Pools and Generalized Ejection Chains for the Vehicle Routing Problem with Time Windows. INFORMS J Comput 19:443–457. https://doi.org/10.1287/ijoc.1060.0186

Nagata Y, Bräysy O, Dullaert W (2010) A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Comput Oper Res 37:724–737. https://doi.org/https://doi.org/10.1016/j.cor.2009.06.022

Ombuki B, Ross BJ, Hanshar F (2006) Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl Intell 24:17–30. https://doi.org/10.1007/s10489-006-6926-z

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34:2403–2435. https://doi.org/10.1016/j.cor.2005.09.012

Qureshi AG, Taniguchi E, Yamada T (2009) An exact solution approach for vehicle routing and scheduling problems with soft time windows. Transp Res Part E Logist Transp Rev 45:960–977. https://doi.org/10.1016/j.tre.2009.04.007

Ropke S, Pisinger D (2006) An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transp Sci 40:455–472. https://doi.org/10.1287/trsc.1050.0135

Solomon MM (1987) Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Oper Res 35:254–265. https://doi.org/10.1287/opre.35.2.254

Taillard E, Badeau P, Gendreau M, et al (1997) A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. Transp Sci 31:170–186. https://doi.org/10.1287/trsc.31.2.170

Tan KC, Chew YH, Lee LH (2006) A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem. Comput Optim Appl 34:115–151. https://doi.org/10.1007/s10589-005-3070-3

Taş D, Jabali O, Van Woensel T (2014) A vehicle routing problem with flexible time windows. Comput Oper Res 52:39–54. https://doi.org/10.1016/j.cor.2014.07.005

Wu L, He Z, Chen Y, et al (2019) Brainstorming-Based Ant Colony Optimization for Vehicle Routing With Soft Time Windows. IEEE Access 7:19643–19652. https://doi.org/10.1109/ACCESS.2019.2894681

Yun Y, Nakayama H, Yoon M (2016) Generation of Pareto optimal solutions using generalized DEA and PSO. J Glob Optim 64:49–61. https://doi.org/10.1007/s10898-015-0314-3

Chapter 4: The Vehicle Routing Problem with Simultaneous Pickups and Deliveries and Time Windows

4.1 Introduction

Supply chain networks include multiple operations such as distributing goods to retailers and consumers, warehousing products, and supplying raw materials to manufacturers. Additionally, they also include collecting products from end customers to be recycled, remanufactured, or repaired (Govindan et al. 2015). Given the range of supply chain operations, the routing of vehicles and the scheduling of deliveries constitute a small but essential part. Therefore, executing efficient routing of vehicles and scheduling of deliveries ensures that the transportation costs will be reduced and the provided services will be improved.

Besides the VRPTW, which is the most commonly encountered VRP variant after the CVRP, researchers also focus on the VRP with Simultaneous Pickups and Deliveries (VRPSPD). In VRPSPD, the main variation from the initial problem is that each customer can receive and return goods (Nagy and Salhi 2005). This variation creates an extra constraint as at any stage of the route, where goods are both delivered and collected, the vehicle's capacity cannot be violated. By combining these two variants, this complex problem comes closer to real-life distribution cases that most logistics companies face daily.

Moreover, the conversion of real distribution cases to variants of the VRP would have no value if not making the most of advances in technology and computer science. Both sectors have allowed researchers to develop more efficient algorithms to address the variants of the VRP. As in the case of solely addressing time windows, so in VRPSPD, exact algorithms are not developed that often. Their time complexity and the limited number of customers that exact algorithms can handle make the specific category less popular among others (Giaglis et al. 2004) for addressing the VRPSPD. Only a few studies have proposed exact algorithms for this variant (Subramanian et al. 2011, 2013), most of which with up to 100 customer orders.

As for heuristic algorithms that offer good quality solutions within a short time-space while also having the ability to solve problems of any size are developed more often than exact algorithms. Some of the most known and studied heuristics that applied for multiple variants of the VRP, including the VRPSPD, are (i) the Time-Oriented Nearest Neighbor (TONN) (Solomon 1987), (ii) the Sweep (Gillett and Miller 1974) and (iii) the Savings (Clarke and Wright 1964) algorithm. Despite being more than 30 years old, these algorithms remain the basis of other more sophisticated and improved algorithms and are still used in commercial routing software (Gayialis and Tatsiopoulos 2004).

However, the center of attention has shifted to metaheuristic algorithms as they combine advantages from both exact and heuristic algorithms. Metaheuristics are separated into two main categories, population search and local search methods (Labadie et al. 2016). Population-based approaches maintain reasonable solutions in memory, combined and cooperate through

a learning procedure for generating new solutions. On the other hand, local search approaches aim at exploring the solution space by moving the current solution to another promising one in the neighborhood. Multiple algorithms of both categories have been developed for addressing both the VRPTW and the VRPSPD efficiently. According to the current research, among the most frequently developed algorithm for these variants of the VRP are evolutionary and genetic algorithms (Berger et al. 2003), particle swarm optimization (Ai and Kachitvichyanukul 2009; Gong et al. 2012), ant colony optimization (Gambardella et al. 1999), simulated annealing (Czech and Czarnas 2002; Woch and Łebkowski 2009), large neighborhood search (Konstantakopoulos et al. 2020b) and tabu search (Chiang and Russell 1997; Yousefikhoshbakht et al. 2014)

In most cases, the algorithms that address the VRPTW and the VRPSPD produce biased solutions either to the number of vehicles or the total distance traveled, that are the two objectives. However, in recent years, some researchers consider the VRP as multiobjective which means that more than one solution can be considered optimal according to the values of the two objective functions. More specifically, Ombuki et al. (2006) and Tan et al. (2006b) were the first to propose a multiobjective genetic and evolutionary algorithm, respectively, for the VRPTW, offering new non-dominated solutions. Ghoseiri and Ghannadpour (2010) used goal programming for the formulation and a genetic algorithm for solving the multiobjective formulation of the VRPTW, while Baños et al. (2013) proposed a hybrid algorithm that combined evolutionary strategies with simulated annealing. Both Garcia-Najera and Bullinaria (2011) and Chiang and Hsu (2014) proposed MOEA's which updated multiple non-dominated solutions in Solomon's instances. Instead, researchers have not studied the multiobjective nature of the VRPSPD to the same extent. Gong et al. (2018) proposed an efficient bee evolutionary algorithm for this problem and was verified by simulating the reverse logistics of engineering machinery remanufacturing company. A characteristic feature of all these algorithms is that they maintain the Pareto-optimality concept, which means that a set of optimal solutions may result in each case. The solutions belonging to this set are called non-dominated solutions, as no solution can be considered better than the others without further information, while solutions that do not belong in that set are known as dominated solutions.

In the current Chapter, a MOEA that uses the non-dominated sorting procedure proposed by Deb et al. (2002), for solving the vehicle routing problem with simultaneous pickups and deliveries and with time windows (VRPSPDTW), while both the number of vehicles and the total distance traveled must be minimized, is proposed. The initial population of solutions utilized by the MOEA are produced by an improved variation of the TONN heuristic algorithm. Additionally, the proposed MOEA incorporates a new crossover operator and the swap node operator (mutation). The MOEA is tested both in Solomon's benchmark instances for the VRPTW and Salhi and Nagy's for the VRPSPD since there is no dataset in the literature to combine both problems. The computational results indicate that the algorithm is very efficient, offering competitive results compared with all the best-published of literature. However, since

most studies do not focus on the multiobjective nature of the VRPSPDTW, neither do to the multiple optimal solutions (non-dominated solutions) that have been proposed in each benchmark instance over the years. This results in not evaluating the efficiency of algorithms correctly since, in many cases, the obtained solutions are not compared with all best-published solutions in each instance. Therefore, the contribution of this Chapter is not limited to addressing the VRPSPDTW as multiobjective in the variation of the TONN algorithm and in the crossover operator that is integrated into the algorithm. The presented research extends to the complete presentation of all non-dominated solutions in Solomon's and Salhi and Nagy's datasets.

The algorithm also integrates and addresses the need of vehicles to refuel their tank or recharge their batteries. This attribute is rarely addressed is real-life cases as petrol and diesel vehicles have high autonomy, while the time needed for refueling is minimum. The driving range and the refueling time have great impact on electric vehicles that have limited driving range and increased refueling times. In most mathematical models the driving range is considered unlimited. However, over the years electric vehicles are used more often, making recharging a necessary procedure that needs to be considered during distribution. Therefore, the mathematical formulation has been created so that to consider the autonomy of each vehicle type. Finally, the algorithm is applied in real-life data that are adapted in the needs and requirements of the study to estimate the environmental and economic impact for each vehicle type (petrol, diesel, electric).

The next sections of the Chapter include the formulation of the problem, the presentation of the proposed algorithm, its validation results and, finally, the conclusions of the research. More specifically, the mathematical formulation of the VRPSPDTW is presented in Section 4.2. Section 4.3 analyzes the structure of the proposed algorithm, the improved variation of the TONN algorithm, the new proposed crossover operator, and the mutation operator. The computational results are presented and compared with the best-published solutions in benchmark instances in Section 4.4. In Section 4.5 the economic and environmental benefits by the use of petrol, diesel and electric vehicles is discussed. Finally, the conclusions are discussed in Section 4.6.

4.2 Problem Description and Mathematical Formulation

The VRPSPDTW is a variant of VRP that combines time windows and simultaneous pickups and deliveries set by customers. More specifically, VRPSPDTW is defined by a set of identical in terms of capacity (Q) vehicles denoted by K and by a direct network $G = \{F \cup N, A\}$, where vertex $N = \{0, 1, \dots n\}$ is the set of customers, vertex $F = \{n+1, n+2, \dots n+s\}$ is the set of charging and gas stations (CGS), and $A = (i,j): i \neq j, i,j \in F \cup N$ denotes the set of arcs. Node 0 represents the central depot, while $N^* = N/\{0\}$ represents the customers. Every arc (i,j), which is a path from node i to node j, is characterized by a distance which is indicated

as d_{ij} and by a travel time t_{ij}. Each client $i \in N$ is characterized by a delivery quantity q_i, a pickup quantity r_i, a service time s_i for unloading and a time window (e_i, l_i), where e_i is the opening and l_i the closing time. The depot and the CGSs are also characterized by a service time s_i and a time window (e_i, l_i) that corresponds to the working hours. The start of service of vehicle k at node j, is denoted by a_j^k and depends on the time of arrival as vehicles arriving before the beginning of the time window must wait until service is possible e_i. If the vehicle arrives between e_i and l_i, the service starts immediately, while no vehicle may arrive after l_i. Additionally, each vehicle k has a specific driving range dr^k and a distance consumption rate f^k. The decision variable x_{ij}^k is equal to 1 if vehicle k executes the route from node i to node j. Otherwise x_{ij}^k is equal to 0. The second decision variable, y_i^k indicates the quantity of items on board of vehicle k when departing from node i. The third decision variable is g_i^k, which indicates the remaining charge or gas level when arriving at customer *i*, according to the vehicle type. The objective of the VRPSPDTW is to serve all customers in N^*, while minimizing both the number of vehicles and the total distance traveled. Additionally, all the essential variables are gathered and presented below before the mathematical model of the VRPSPDTW.

d_{ij}	Distance between node i and node j
t_{ij}	Transit time between node i and node j
q_i	Delivery demand of node i
r_i	Pickup demand from node i
e_i	Node's i open time
l_i	Node's i close time
s_i	Service time at node i
y_{ij}^k	Quantity of items on board of vehicle k when departing from node i
dr^k	Driving range of vehicle k
a_i^k	Start of service of vehicle k at customer j
g_i^k	Remaining gas or power of vehicle k when departing from node i
f^k	Distance consumption rate of vehicle k

$$Objective\ 1: \sum_{k \in K} \sum_{j \in N^*} x_{0j}^k \tag{4.1}$$

$$Objective\ 2: \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^k \tag{4.2}$$

$$\sum_{i \in N} q_i \sum_{j \in N} x_{ij}^k \leq Q, \forall k \in K \tag{4.3}$$

$$\sum_{i \in N} r_i \sum_{j \in N} x_{ij}^k \leq Q, \forall k \in K \tag{4.4}$$

$$\sum_{(i,j)\in A} x_{ij}^k (y_i^k + r_j - q_j) \le Q, \forall k \in K \quad (4.5)$$

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1, \forall i \in N \quad (4.6)$$

$$\sum_{j\in N^*} x_{0j}^k = 1, \forall k \in K \quad (4.7)$$

$$\sum_{i\in N} x_{il}^k - \sum_{j\in N} x_{lj}^k = 0, \forall l \in N^*, \forall k \in K \quad (4.8)$$

$$\sum_{j\in N^*} x_{j0}^k = 1, \forall k \in K \quad (4.9)$$

$$a_i^k + s_i + \sum_{j\in N} x_{ij}^k t_{ij} \le l_j, \forall i \in N, \forall k \in K \quad (4.10)$$

$$a_j^k = max\left\{\sum_{j\in N} x_{ij}^k \left(a_i^k + s_i + t_{ij}\right), e_j\right\}, \forall i \in N, \forall k \in K \quad (4.11)$$

$$0 \le g_j^k \le dr^k, \forall j \in A, \forall k \in K \quad (4.12)$$

$$g_i^k - f^k d_{ij} x_{ij}^k \ge 0, \forall (i,j) \in A, \forall k \in K \quad (4.13)$$

$$x_{ij}^k \in \{0,1\}, \forall (i,j) \in A, \forall k \in K \quad (4.14)$$

$$y_i^k \ge 0, \forall i \in N, \forall k \in K \quad (4.15)$$

In the present Chapter, the VRPTW is encountered as a multiobjective problem and the two objectives, needed to be minimized, are the number of vehicles and the total distance traveled and are described by function (4.1) and (4.2), respectively. Constraints (4.3) and (4.4) state that the total quantity of products delivered and collected in total cannot exceed the capacity of the vehicle, while constraint (4.5) states that, at any stage of the distribution process, the total quantity of products in the vehicles cannot exceed its capacity. Equation (4.6) states that each customer must be served exactly once and by one vehicle since x_{ij}^k can obtain only 1, or 0 as values. The set of constraints (4.7), (4.8) and (4.9) make sure that each vehicle starts from the depot and visits and serves the customers that belong in the route before it finally returns to the depot. Constraint (4.10) indicates that for a trip from node i to node j, no vehicle may arrive at customer j after the end of the time window, l_j, while constraint (4.11) that the earliest time for starting serving customer j is related to the time of arrival, depending on whether the time of arrival is before or after the start of the time window e_i. Constraints (4.12) and (4.13) ensure that fuel/battery feasibility of each vehicle. An example of the VRPSPD is illustrated in Figure 4.1. It is clear that the quantity of goods on board cannot exceed the vehicle's capacity at any stage of the distribution (in every customer).

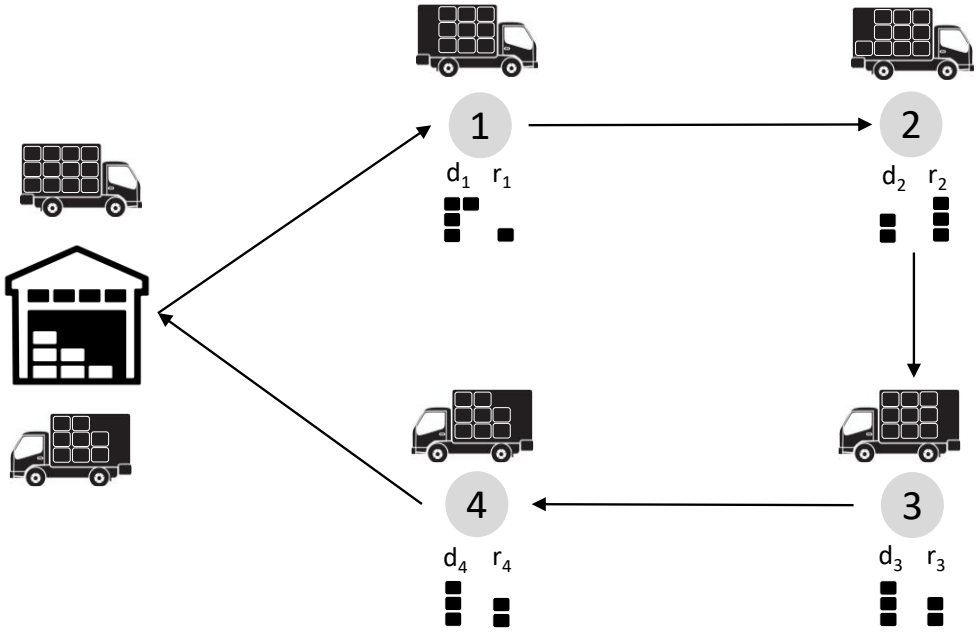


Figure 4.1 Illustration of the VRPSPD

4.3 Solution Method – Evolutionary Algorithm

In order to address the VRPSPDTW that is presented above, an efficient multiobjective evolutionary algorithm was developed. More specifically, Section 4.3.1 discusses the importance of multiobjective optimization and the Pareto-optimality concept. Section 4.3.2 presents the improved version of the TONN algorithm used to create the initial population of solutions. Section 4.3.3 describes the structure of the chromosome applied in the MOEA. In Section 4.3.4, the methodological procedure of the MOEA, along with the pseudocode, are thoroughly described. Finally, the new proposed crossover operator and the mutation operator are described in Section 4.3.5. The steps that are mentioned above are given in Figure 4.2.

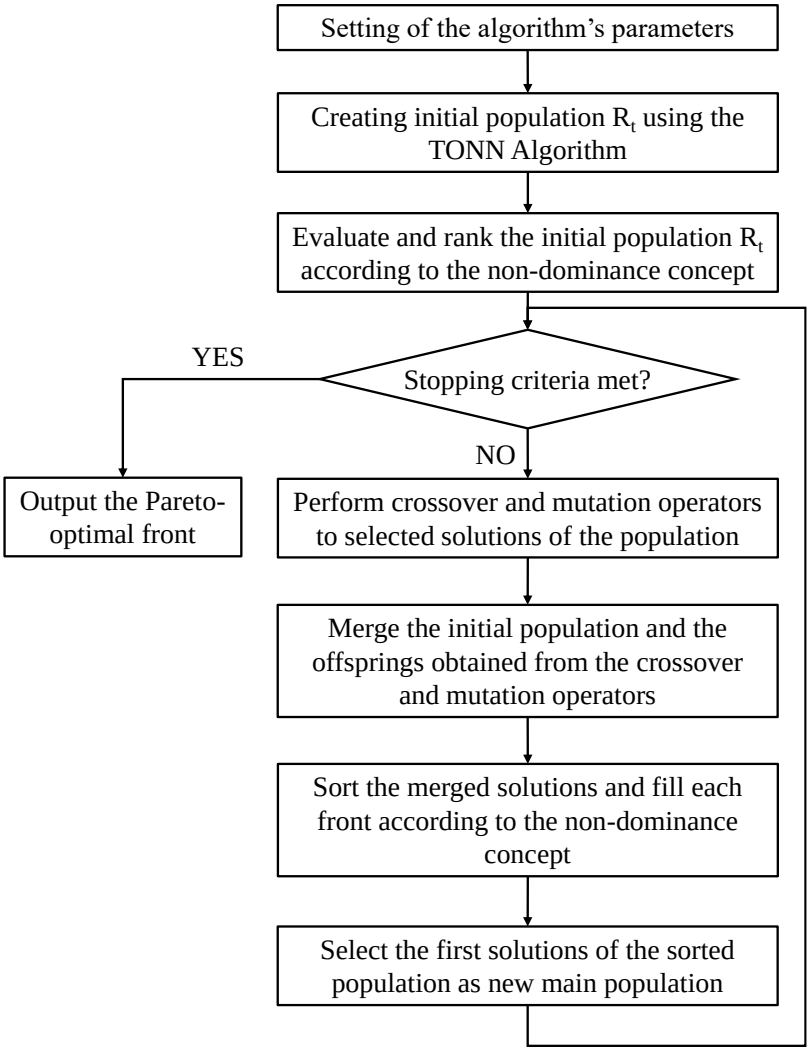


Figure 4.2 Steps of the MOEA Algorithm

4.3.1 Multiobjective Optimization

A set of optimal solutions can be obtained in multiobjective optimization problems, instead of a single solution. The solutions belonging to this set are non-dominated with respect to each other. The main characteristic of the non-dominated solutions is that while moving from one solution to another, there is always a certain amount of sacrifice in one objective to achieve a certain amount of gain in another objective, as shown in Figure 4.3. More specifically, no solution between the non-dominated solutions $X_1$and X_2 can be considered better since the mathematical relation $F_1(X_1) > F_1(X_2)$ and $F_2(X_1) < F_2(X_2)$ is valid. This means that by moving from solution X_1 to X_2, a decrease in F_1 objective with a simultaneous increase in F_2 objective is observed. Therefore, without further information, no solution can be considered better than the other and dominate. The set of all the non-dominated solutions is called Pareto-optimal front as shown in Figure 4.3

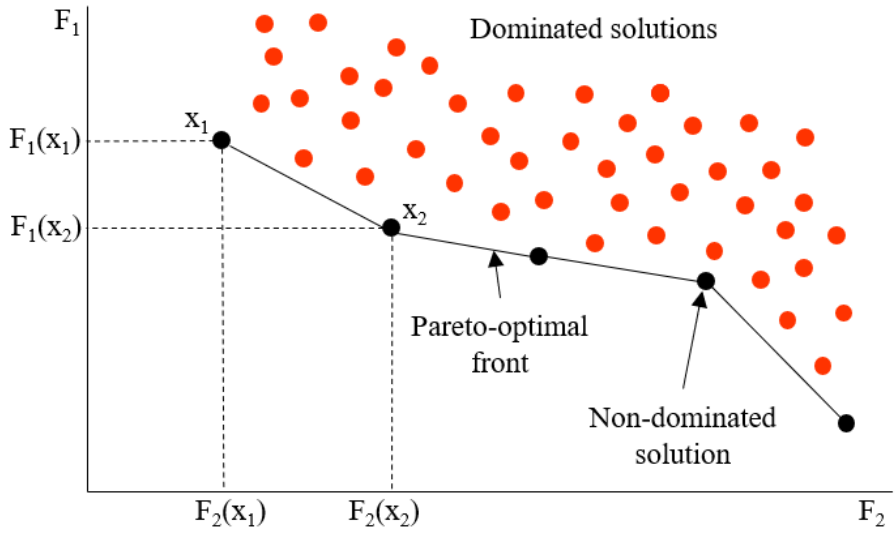


Figure 4.3 Dominated, Non-dominated, and Pareto-optimal Front

In the case of the VRP, the different obtained solutions are translated into different plans of routes and deliveries. These different plans serve the interests of multiple stakeholders that choose the most profitable one. The choice is based on further information and on the nature of each company. For example, different plans may be selected when the fleet of vehicles is company-owned and when the distribution of goods is assigned to third party companies. Therefore, it is crucial to consider the VRPSPDTW as multiobjective, where the two objectives are the number of vehicles, and the total distance traveled to cover the needs of multiple stakeholders. Through this approach and in search of a cost-effective plan, more than one solution may result. Therefore, it is at the discretion of the decision-maker to select the most profitable plan.

4.3.2 Construction Method

The initial population of solutions contributes significantly to the efficiency of the proposed MOEA, where it is used as an input. Therefore, an efficient algorithm, in terms of time and solution's quality, is needed. The fact that the TONN construction algorithm meets these requirements is the main reason it is selected for the construction procedure. Solomon (1987) first proposed the TONN algorithm and belongs in the sequential methods that construct one route at a time. More specifically, every route starts from the depot (node 0) and is built by adding the "closest" customer that has not been included in any other route to the last visited node. If no other customer can be added to the route, a new route starts due to capacity or time window constraints. The routes construction procedure ends when no more customers are left to add.

The fact that the TONN algorithm was first proposed for the VRPTW, which is only one of the variants considered in the current Chapter, leads to adaptations in the closeness function to fit in the VRPSPDTW. The closeness is now given by the metric $c_{ij} = w_1 d_{ij} + w_2 v_{ij} + w_3 u_{ij}$, which involves the distance d_{ij} between customers i and j, the capacity left to the vehicle k

after serving customer j, $v_{ij} = Q - y_i^k - q_j$ and the urgency of delivery to customer j, $u_{ij} = l_j - (a_i + s_i + t_{ij})$. In this case, customer i is the previously served customer, while j the one for whom the metric c_{ij} is minimized.

The second variation in the TONN is based on the fact that time, distance, and capacity are not on the same scale, affecting the constructed solutions. Therefore, a min-max normalization procedure [30], is applied in the dataset $(t_{ij}, d_{ij}, e_i, l_i, s_i, q_i)$. This normalization technique provides linear transformation and fits data in a pre-defined boundary, which is [0, 1]. The formula used for the normalization procedure is:

$$X' = [(X_0 - X_{min})/(X_{max} - X_{min})](D - C) + C$$

with X' being the new value for variable X, [C, D] the pre-defined boundary, X_0 the current value for variable X, and X_{min} minimum and X_{max} the maximum value in dataset X.

This pre-processing data procedure improves the accuracy and achieves the best performance in the tested dataset. That is why, in the case of c_{ij} metric, in which the weights w_1, w_2 and w_3 are assigned to d_{ij}, v_{ij} and u_{ij} respectively and sum up to 1, the normalization of data is essential and can offer improved results. Therefore, the dataset used in the construction algorithm involves normalized data.

4.3.3 Chromosome Structure

The chromosome structure significantly affects the algorithm in terms of efficiency and time complexity since it is involved in the crossover and mutation operators. Therefore, similar to most studies that propose evolutionary algorithms for the VRP, it was decided to represent the chromosome as an array with length equal to the total number of customers. Each object of the chromosome is a gene and is represented by an integer which refers to the customer's node (id). The sequence of the genes determines the order of customers' visitation. When a sequence cannot be continued due to the violation of capacity or time window constraints, a new route begins. An example of a chromosome with 9 customers is presented in Figure 4.4 and the order of visitations in each route.

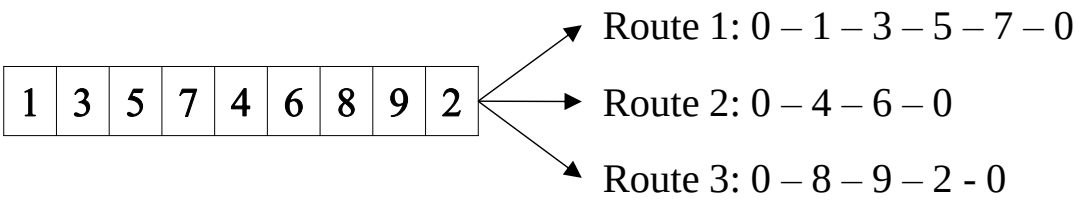


Figure 4.4 Chromosome Structure

However, in the case of vehicles, that have limited driving range, the algorithm also considers the need either of recharging before the battery is discharged (electric vehicles), either of refueling the tank (conventional vehicles). Therefore, between two consecutive genes, a charging or a gas station (CGS) may intervene, but simultaneously the station's ID is not

contained in the chromosome. Additionally, both the time and the distance required for the vehicle to reach the CGS are taken into account. The possibility of a visit to a station (CGS) is greater in electric vehicles due to their limited driving range.

4.3.4 Proposed Evolutionary Algorithm

After the TONN algorithm generates the initial population (R_0) of size N, the MOEA is applied for evolving solutions and obtain the Pareto-optimal front. More specifically, in each generation t, taking into account the parents' population R_t set, the algorithm implements the crossover operator in all N parents and generates N offsprings saved in Q_t (offsprings' population). The proposed crossover operator is thoroughly described in Section 4.3.5. Consequently, the chromosomes belonging to the union set $R_t \cup Q_t$ (entire population) are sorted according to the non-dominated sorting and create Pareto fronts (sets) $F_1, F_2, \dots F_K$. This procedure is used for sorting the solutions in sets $(F_1, F_2, \dots F_K)$ according to the Pareto dominance principle. This means that each set F_i contains solutions, with F_1 being the best non-dominated set that is represented as Pareto-optimal front in Figure 4.4. Accordingly, F_2 set contains solutions that are dominated only by those in set F_1. The set R_{t+1}, which contains the parents of the next generation starts being filled with solutions of sets F_1, F_2, and so on, until N chromosomes fill the R_{t+1} set. Through this procedure, the best solutions in each iteration are maintained and used for evolving the solutions. The described procedure is followed for a specific number of iterations, which is the maximum number of generations. The pseudocode of the MOEA is shown in Figure 4.5.

```
1.  Set maximum number of generations (max_gen), population
    size (N), mutation and crossover probabilities, t = 0
2.  Normalize data
3.  Construct the initial population R_t of size N by applying the
    proposed TONN algorithm
4.  While t ≤ max_gen do
5.      For i = 1 to N with step = 2 do
6.          Select parents P_i and P_{i+1} from R_t
7.          Execute crossover of P_i and P_{i+1} and generate
            offsprings O_i and O_{i+1}
8.          Mutate offsprings O_i and O_{i+1}
9.          Insert offsprings O_i and O_{i+1} in set Q_t
10.     End For
11.     Create a union set R_t ∪ Q_t
12.     Apply the non-dominated sorting to the union set for
        building Pareto fronts F
13.     i = 0
14.     Set F_i as the current front
15.     t = t + 1
16.     While length of population R_t ≤ N do
17.         For each solution in the front F_i do
18.             If length of population R_t ≤ N then
19.                 Add solution in population R_t
20.             End If
21.         End For
22.     i = i + 1
23.     End While
24. End While
```

Figure 4.5 Pseudocode of the MOEA

4.3.5 Crossover and Mutation Operators

Multiple crossover operators, such as those presented in Solomon (1987), have been developed over the years for the VRP. The most efficient crossover operator implemented in most genetic and evolutionary algorithms is the Best Cost Route Crossover (BCRC) that is thoroughly described in Ombuki et al. (2006). More specifically, from the P set, two parents (P_i, P_{i+1}) are selected, and from each one, a single route is also selected. The customers that belong to each route are then removed from the opposite parent. More specifically, customers selected from parent P_i are removed from parent P_{i+1}, and reversely. Since all customers must be served, they must also be contained in the chromosome. The insertion of a customer is made in the position of the chromosome that minimizes the cost. After the insertion procedure, two new chromosomes, known as offsprings, are generated. Through this procedure, the set Q_t is filled with N new solutions, before the non-dominated sorting procedure is applied and consequently set P_{t+1} arises.

The proposed and applied crossover operator works similarly to the BCRC and is called Sequential Crossover (SC). The differences between the two crossover operators are noted in two points.

- In selecting customers, as in SC, a random number of consecutive genes are selected from each chromosome instead of a single route as in BCRC. In this way, the diversity and, consequently the search space increases.
- In the reinsertion procedure, the customers can only be inserted in specific positions. More specifically, they can be inserted in the position of the chromosome for which the total distance traveled is minimized. However, they can also be inserted in the position of the chromosome for which the total distance traveled is minimized while simultaneously the number of vehicles is not increased (compared to the parent). This procedure also increases the diversity and the search capabilities of the algorithm.

The mutation operator followed is a swap node operator. In this case, two customers (genes) of the chromosome are randomly selected and swapped. The main contribution of this procedure is the differentiation of chromosomes, which may help in escaping from local optimum. Finally, both the mutation and the crossover operator are not applied in each generation and in each set of parents (P_i, P_{i+1}). Instead, each operator has a specific probability of being applied, which is defined at the beginning, as shown in Figure 4.5.

4.4 Computational Study

The proposed algorithm cannot be simultaneously tested in time windows and pickups and deliveries due to the lack of a dataset containing both attributes. Therefore, the algorithm is tested in two different datasets, Solomon's, for the VRPTW and Salhi and Nagy's, for the VRPSPD. In Solomon's dataset, the instances are separated into those with geographically clustered customers' data (C1 and C2), with randomly generated geographical customers data (R1 and R2) and with mixed geographical customers data (RC1 and RC2). Also, problem sets C1, R1 and RC1 have short scheduling horizon and narrow time windows, while set C2, R2, and RC2 have long scheduling horizon and wide time windows. The proposed algorithm was coded in Python and ran in a computer with the following specifications: Intel Core i7 - 8550U 1.8GHz with 16GB memory.

Additionally, the dataset of Salhi and Nagy (1999), is based on the dataset of Christofides et al. (1979). More specifically, all data remain the same except the demand that is separated in a delivery and in a pickup quantity. Finally, the evolutionary algorithm parameters used in the proposed algorithm are the same with those presented in Ombuki et al. (2006) so that the results be representative and comparable.

- Population size = 300
- Number of generations = 350

- Crossover probability = 0.80
- Mutation probability = 0.10

Another parameter that is defined is the maximum computation time. In the current research, the time limit is set at 15 minutes as in section 3.4, which is considered a reasonable computation time for 100 customers. Additionally, the best-obtained results need less time than the time limit to obtain the optimal solution. More specifically, the average time needed for obtaining each optimal solution is approximately 13 minutes. Finally, each instance is solved three times and the best-obtained solution is given in Table 4.1 and Table 4.2.

4.4.1 Results in the VRPTW

The VRPTW has been widely studied and, therefore, for most of Solomon's instances, a set of non-dominated solutions (Pareto-optimal front) has arisen over the years from multiple researchers, leading to the best-published results in these instances. The computational results obtained from the proposed MOEA are compared, in Tables 4.1 and 4.2, with the best-published ones in the literature to evaluate their efficiency. More specifically, Table 4.1 contains instances with short scheduling horizon, narrow time windows, and small vehicles' capacities, while Table 4.2 contains instances with long scheduling horizon, wide time windows and large vehicles' capacities. In each table and for each instance of the dataset, the set of non-dominated solutions characterized by the total distance traveled (TD), and the number of vehicles (NV) are presented. Column "MOEA" presents the results obtained by the proposed algorithms; column "Best-Published" showcases the best-published results. Finally, the column "Deviation (%)" presents the percentage difference between best-published and best-obtained results.

The results are considered competitive compared to the best published in the literature. The deviation presented in both tables refers to the total traveled distance when the number of vehicles, both in the best-published results and those of the MOEA, are equal; otherwise, it is not calculated. The mean deviation is 2.64%, a criterion that proves the effectiveness of the algorithm. However, the MOEA seems to have better efficiency with geographically clustered customers' data (C datasets), irrespective of the scheduling horizon and the range of the time windows, as the deviation of the obtained results with the best-published range from 0.00% to 0.73%.

Table 4.1 Results of the MOEA in Solomon's Instances with Short Scheduling Horizon

Problem Instance	MOEA NV	TD	Best-Published NV	TD	Deviation (%)	Problem Instance	MOEA (%)	TD	Best-Published NV	TD	Deviation (%)
C101	10	828.94	10	827.3	0.20	**R110**	-	-	11	1054.23	-
C102	10	828.94	10	827.3	0.20		12	1122.48	12	1068	5.10
C103	10	828.94	10	826.3	0.32	**R111**	-	-	10	1096.72	-
C104	10	828.94	10	822.9	0.73		-	-	11	1084.76	-
C105	10	828.94	10	827.3	0.20		12	1099.70	12	1048.7	4.86
C106	10	828.94	10	827.3	0.20	**R112**	-	-	9	982.14	-
C107	10	828.94	10	827.3	0.20		10	965.12	10	953.63	1.20
C108	10	828.94	10	827.3	0.20	**RC101**	-	-	14	1650.14	-
C109	10	828.94	10	827.3	0.20		15	1649.50	15	1619.8	1.83
R101	20	1667.33	18	1607.7	-	**RC102**	-	-	12	1554.75	-
R102	18	1499.67	17	1434	-		-	-	13	1470.26	-
R103	14	1237.50	13	1175.67	-		14	1504.56	14	1461.33	2.96
R104	-	-	9	1007.24	-	**RC103**	-	-	11	1261.67	-
	10	1006.48	10	974.2	3.31		12	1292.51	12	1196.12	8.06
R105	-	-	14	1377.11	-	**RC104**	10	1165.00	10	1135.48	2.60
	15	1391.86	15	1346.12	3.40	**RC105**	-	-	13	1629.44	-
R106	-	-	12	1251.98	-		14	1575.46	14	1542.29	2.15
	13	1287.07	13	1234.6	4.25		15	1558.06	15	1519.29	2.55
R107	-	-	10	1104.66	-		-	-	16	1518.6	-
	11	1084.15	11	1051.84	3.07	**RC106**	-	-	11	1424.73	-
R108	-	-	9	958.66	-		-	-	12	1394.43	-
	10	994.89	10	942.9	5.51		13	1417.86	13	1371.69	3.37
R109	-	-	11	1194.73	-	**RC107**	-	-	11	1222.16	-
	12	1177.76	12	1101.99	6.88		12	1248.62	12	1212.83	2.95
	13	1167.19	-	-	-	**RC108**	-	-	10	1139.82	-
R110	-	-	10	1118.59	-		11	1209.42	11	1117.53	8.22

As for geographically random customer data (R and RC), multiple non-dominated solutions at each instance exist as the search space is more extensive, especially in cases with long scheduling horizon and wide time windows (C2 and RC2). The MOEA, which integrates the proposed crossover operator, can search a vast solution space to find non-dominated solutions. As a result, two new non-dominated solutions are proposed in Solomon's R202 and R210 instances. In the R202 instance, a decrease in the total distance traveled is accomplished, while in R210 instance, two non-dominated solutions are updated, as shown in Figure 4.6. More specifically, in R210 instance, the proposed solution with objective values (NV=4, TD=920.13) improves both the solution with objective values (NV=4, TD=924.79), since the distance traveled is decreased, as well as the solution with objective values (NV=5, TD=922.30), since both the number of vehicles and the total distance traveled are decreased.

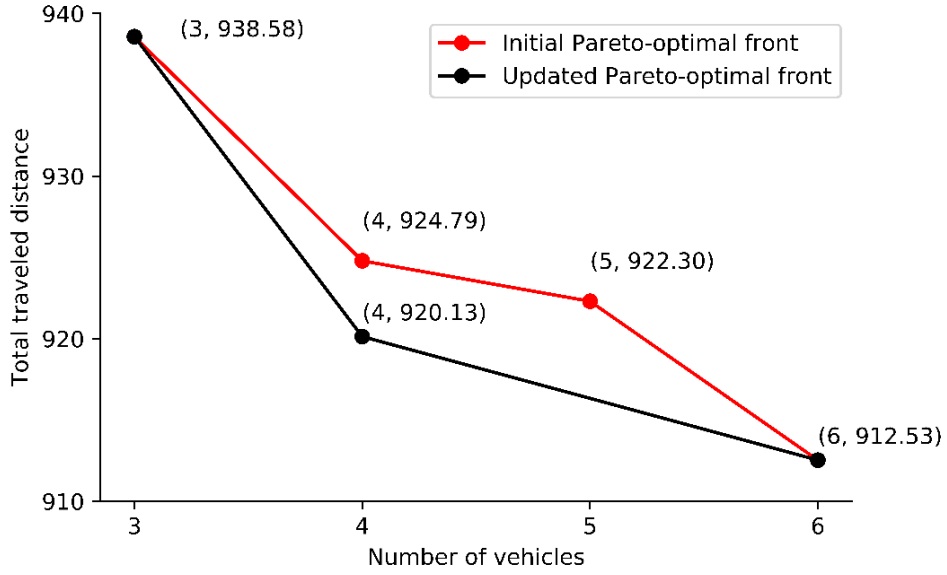


Figure 4.6 Updated Pareto-optimal front on R210 Problem Instance

Table 4.2 Results of the MOEA in Solomon's Instances with Long Scheduling Horizon

Problem	MOEA		Best-Published		Deviation	Problem	MOEA		Best-Published		Deviation
Instance	**NV**	**TD**	**NV**	**TD**	**(%)**	**Instance**	**NV**	**TD**	**NV**	**TD**	**(%)**
C201	3	591.56	3	589.1	0.42	**R209**	3	956.00	3	909.16	5.15
C202	3	591.56	3	589.1	0.42		4	881.64	4	864.15	2.02
C203	3	591.56	3	591.17	0.07		5	875.63	5	859.39	1.89
C204	3	590.6	3	590.6	0.00	**R210**	-	-	3	938.58	-
C205	3	588.88	3	586.4	0.42		4	920.13	4	924.79	-0.50
C206	3	588.5	3	586	0.43		-	-	5	922.30	-
C207	3	588.5	3	585.8	0.46		-	-	6	912.53	-
C208	3	588.32	3	585.8	0.43	**R211**	-	-	2	885.71	-
R201	4	1290.75	4	1252.37	3.06		3	838.85	3	767.82	9.25
	5	1202.7	5	1193.29	0.79		4	770.54	4	755.82	1.95
	-	-	6	1171.2	-	**RC201**	4	1473.09	4	1406.91	4.70
	-	-	7	1173.75	-		5	1329.59	5	1279.65	3.90
	-	-	8	1150.92	-		7	1284.48	7	1273.51	0.86
	-	-	9	1148.48	-		8	1282.64	8	1272.28	0.81
R202	-	-	3	1191.7	-	**RC202**	-	-	3	1365.65	-
	4	1085.64	4	1079.39	0.58		4	1200.07	4	1162.54	3.23
	5	1014.96	5	1041.1	-2.51		5	1129.46	5	1118.66	0.97
R203	3	1026.22	3	939.5	9.23		6	1125.22	8	1099.54	-
	4	902.25	4	901.2	0.12	**RC203**	-	-	3	1049.62	-
	5	891.03	5	890.5	0.06		4	958.55	4	945.08	1.43
	-	-	6	874.87	-		5	938.04	5	926.82	1.21
R204	-	-	2	825.52	-	**RC204**	3	801.18	3	798.41	0.35
	3	751.79	3	749.42	0.32		-	-	4	788.66	-
	4	746.01	4	743.23	0.37	**RC205**	4	1340.53	4	1297.19	3.34
	-	-	5	735.86	-		5	1304.35	5	1236.78	5.46

R205	3	1048.74	3	994.43	5.46		6	1210.00	6	1187.98	1.85
	4	1021.15	4	959.74	6.40		-	-	7	1161.81	-
	5	1004.27	5	954.16	5.25	**RC206**	3	1301.87	3	1146.32	13.57
R206	3	970.22	3	906.14	7.07		4	1121.53	4	1081.83	3.67
	4	897.25	4	889.39	0.88		5	1094.2	5	1068.77	2.38
	5	892.18	5	879.89	1.40		6	1087.27	7	1054.61	-
R207	-	-	2	890.61	-	**RC207**	-	-	3	1061.14	-
	3	854.95	3	812.76	5.19		4	1023.02	4	1001.85	2.11
	-	-	4	797.99	-		-	-	5	982.58	-
R208	-	-	2	725.75	-		-	-	6	966.37	-
	3	731.84	3	706.86	3.53	**RC208**	3	859.52	3	828.14	3.79
							4	805.66	4	783.035	2.89

The solutions in Solomon's instances that improve the results and offer new non-dominated solutions are presented in Appendix C.

4.4.2 Results in the VRPSPD

The VRPSPD is studied less frequently than the VRPTW, despite being the most common problem distribution and logistics companies face in their daily operations. The proposed MOEA, which is able to address the VRPSPDTW, is tested in the dataset of Salhi and Nagy, and the results obtained from the proposed MOEA algorithm are compared to the best-published. As in the case of time windows, the algorithm manages to produce new non-dominated solutions in instances CMT2X, CMT6X, CMT7X, CMT8X, CMT9X, CMT10X. In each of these 6 cases, the number of vehicles is reduced at the expense of increased total distance traveled, as shown in Table 4.3. In addition, the mean TD deviation between the best-published results and the best-obtained ones from the MOEA was found to be 8.50%. As shown in Table 4.3, the TD deviation for each problem is not calculated for all solutions. Instead, the TD deviation is calculated only when the MOEA and the best-published results have equal NV values.

Table 4.3 Results of the MOEA in Salhi and Nagy Dataset

Problem	**MOEA**		**Best-published**		**Deviation**
CMT	**NV**	**TD**	**NV**	**TD**	**(%)**
1X	3	481.13	3	467	3.03
2X	6	771.01	-	-	-
	7	730.21	7	676	8.02
3X	5	753.83	5	703	7.23
4X	7	897.65	7	847	5.98
5X	10	1130.28	10	1025	10.27
6X	5	606.33	6	557	-
7X	9	1040.67	11	919	-
8X	8	1036.52			-
	9	955.21	9	896	6.61
9X	12	1744.19	-	-	-
	13	1741.36	-	-	-

	14	1698.21	-	-	-
	15	1354.51	15	1215	11.48
10X	17	1937.02	19	1571	-
11X	5	904.52	4	868	-
12X	6	712.21	5	667	-
13X	11	1757.99	11	1560	12.69
14X	10	918.25	10	826	11.17

The solutions in Salhi and Nagy's instances that improve the results and offer new non-dominated solutions are given in Appendix C.

4.5 Conventional vs Electric Vehicles

While the distribution of goods has attracted the interest of researchers for more than 60 years, it is only the last few years that the GHG emissions are considered in the VRP, due to the ecological awareness of transport sustainability, leading to green logistics. In the case of Electric VRP (EVRP), the vehicles that are used for the distribution process, have electric batteries. This implies a limited driving range and visitations to Charging Stations (CS) to recharge the batteries, when needed, for executing the routes (Erdelić et al. 2019). However, the EVRP can be much more challenging and complex if the battery consumption and CO_2 emissions are related to the load and the speed of the vehicle (Lin et al. 2016). Erdelić and Carić (2019) present a survey on the EVRP, focusing on consumption models and additional emerged EVRP variants, such as hybrid vehicles, charging stations, charging functions and dynamic traffic conditions. In the current research, it is assumed that the consumption and CO_2 emissions are fixed, irrespectively the vehicle's load and speed.

Moreover, the distribution cases that are studied in the present subsection, are faced by a logistics company in the city of Athens, Greece. However, the data and other characteristics adapt to the requirements of the studied case. Therefore, the vehicles are considered identical in all distribution cases, while in practice, the company owns different types of vehicles. As for the customers' data (57 in total), they contain their addresses, their demand and return quantities (Kg), and their time windows. More specifically, 5 different cases are tested in order to extract more reliable conclusions. Starting from the first case (Case 1), and until the last one (Case 5), the quantities of goods that each customer requests, either to be delivered or to be returned, increase in size and constitute a bigger part of the vehicle's capacity.

Additionally, considering the identical vehicles, three cases are discriminated according to the fuel type, (i) Petrol, (ii) Diesel, and (iii) Electric vehicles. However, to extract reliable conclusions from the research, the vehicles are of the same brand (Nissan) and the same model (NV200). Consequently, the distribution cases are tested for the three different vehicle types. In each case, the fleet of vehicles is composed only of the same type of vehicles. In addition, the fixed costs were calculated, taking into account depreciation time (10 years) and annual

vehicle use (250 working days per year). The increased fixed cost of electric vehicles is at least expected since the purchase price is higher than the other typeσ of vehicles.

On the other hand, the variable cost is inversely proportional to fixed costs, since it depends on fuel value. Furthermore, the driving range in electric vehicles is significantly lower, leading to the need for battery recharge, which is also time-consuming (45 minutes to recharge the empty battery fully). Batteries also cause another problem, which is the reduced capacity. However, all the above disadvantages are balanced by zero CO_2 emissions. The above-mentioned characteristics and specifications of vehicles are presented in Table 4.4.

Table 4.4 Features and Specifications of Vehicles

Vehicle Type	Fixed Cost (€)	Variable Cost (€/Km)	Driving Range (Km)	Capacity (Kg)	CO_2 Emissions (gr/km)
Petrol	8.00 €	0.0972	750	795	166
Diesel	8.82 €	0.0555	1100	728	131
Electric	11.70 €	0.0196	275	630	0

The computational results of the research, containing the number of vehicles needed (number of routes) to execute the plan of routes, the total distance, CO_2 emissions and the costs (variable, fixed, and total), are calculated for each vehicle type (Petrol, Diesel, and Electric), as well as for each distribution case. Table 4.5, Table 4.6, and Table 4.7 contain the results of Petrol, Diesel, and Electric vehicles, respectively.

Table 4.5 Economic and Environmental Impact of Petrol Vehicles

Case	Number of Routes	Total Distance (Km)	CO_2 Emissions (Kg)	Fixed Cost (€)	Variable Cost (€)	Total Cost (€)
1	4	378.82	62.88	32.00	36.82	68.82
2	5	392.93	65.23	40.00	38.19	78.19
3	5	427.78	71.01	40.00	41.58	81.58
4	8	527.09	87.50	64.00	51.23	115.23
5	9	561.64	93.23	72.00	54.59	126.59
Total	31	2288.26	379.85	248.00	222.41	470.41

Table 4.6 Economic and Environmental Impact of Diesel Vehicles

Case	Number of Routes	Total Distance (Km)	CO_2 Emissions (Kg)	Fixed Cost (€)	Variable Cost (€)	Total Cost (€)
1	4	377.25	49.50	35.28	20.93	56.21
2	5	413.29	54.14	44.1	22.94	67.04
3	6	421.71	55.24	52.92	23.41	76.33
4	9	563.91	73.87	79.38	31.30	110.68
5	9	630.35	82.58	79.38	34.99	114.37

Total	33	2406.51	315.33	291.06	133.56	424.62

It can be easily observed that petrol vehicles have the highest CO_2 emissions. Diesel vehicles have also increased emissions, but they gain significant interest due to their low distribution cost. As for electric vehicles, zero emissions are accomplished, but at the expense of high distribution cost compared to other cases of vehicles. The main factor affecting deliveries with electric vehicles, leading to increased cost, is that more vehicles are needed to execute the deliveries compared to the other types of vehicles in each distribution case. That is due to the limited capacity of electric vehicles, as batteries have increased weight and volume, which prevent them from having the same capacity as other types of vehicles.

Table 4.7 Economic and Environmental Impact of Electric Vehicles

Case	Number of Routes	Total Distance (Km)	CO_2 Emissions (Kg)	Fixed Cost (€)	Variable Cost (€)	Total Cost (€)
1	5	394.64	0	58.5	7.72	66.22
2	6	432.91	0	70.2	8.48	78.68
3	7	469.17	0	81.9	9.19	91.09
4	10	617.21	0	117.00	12.09	129.09
5	11	640.09	0	128.70	12.55	141.25
Total	39	2554.02	0	456.3	50.02	506.32

Moreover, in Case 1, where each customer has small demand, in terms of weight, multiple customers can be integrated into the same route. As a result, the number of electric vehicles that are utilized in deliveries are not much more than conventional vehicles, and therefore fixed costs are on the same level. Additionally, the variable costs remain very low in the case of electric vehicles, unlike to conventional vehicles. On the other hand, when the items are of bigger size (Case 5), the total distribution cost of electric vehicles is much higher, due to the increased number of vehicles needed. In conclusion, electric vehicles could be very efficient, both in terms of cost and CO2 emissions, in distribution of small size items such as parcels and mail.

Surely, Greek companies should invest at least to diesel vehicles that will offer both decreased CO2 emissions and distribution costs, if not to electric. The emissions reduction and the cost increment concerning the case of electric vehicles can be misleading. That is because the production of electricity in Greece is based, in a great extent, on the use of poorer quality fuel such as lignite and less to renewable sources. It is only the last few years that renewable sources are utilized. Consequently, electric vehicles may have zero emissions, however the production of electricity has increased emissions that can be minimized only by using renewable energies. Additionally, it is clear that the acquisition cost of electric vehicles is higher compared to petrol and diesel, as occurred with all new technologies. However, over the years, the acquisition cost is expected to be reduced and to be more cost effective for the distribution process.

4.6 Concluding Remarks

To sum up, in this Chapter, the vehicle routing problem with simultaneous pickups and deliveries and time windows was examined, aiming to find an effective solution to the problem by proposing an evolutionary algorithm that solves the multiobjective VRPSPDTW. The first step to solving this problem was applying an improved TONN construction algorithm for generating the initial population of solutions. Consequently, these solutions were integrated into the proposed MOEA in order to find the best routes plan that minimizes the two objectives: the number of vehicles and the total distance traveled. These two objectives may be contradictory and this is why some instances of the problem may have multiple possible solutions. The Pareto-optimality concept was used to balance the results for the different objectives and find the best overall solution to the problem. Besides addressing the VRPSPDTW as a multiobjective problem, another contribution of the Chapter was proven to be the newly developed crossover operator (Sequential Crossover), which significantly affects the efficiency of the results obtained from the MOEA.

The algorithm's efficiency was validated using the dataset of Solomon for the VRPTW, and the dataset of Salhi and Nagy (1999) for the VRPSPD, and the outcomes obtained were compared to the best-published results, indicating the high efficiency of the algorithm in terms of solution's quality.

The specific algorithm offers slightly better results compared to the MOLNS algorithm proposed in Chapter 3. Moreover, the EA algorithm after the appropriate modifications has the ability to address multiple variants of the VRP, as well as distribution cases that involve electric vehicles that have a limited driving range. That is why the algorithm was tested in various cases that the fleet is composed of different types (petrol, diesel and electric) so that to estimate the economic and environmental impact. Among the findings of the research is that in distribution cases with low-volume goods the limitations of electric vehicles (driving range) do not greatly affect costs. However, as the volume of goods increases, more electric vehicles compared to other vehicle types are needed, leading to an increased distribution cost. This is due to the decreased capacity of electric vehicles. Irrespective of the cost, CO_2 emissions are eliminated by the use of electric vehicles.

Additionally, despite the advantages by using electric vehicles, in Greece the technologies concerning the vehicles are still far behind. This is obvious by the age and the type of vehicles used in the distribution. Moreover, the use of electric vehicles is not sufficiently supported by the infrastructure as only a limited number of recharging stations exist even in the capital of Greece, Athens. Considering this, the transition to more advanced vehicles, and more specifically to electric vehicles, may be difficult for logistics and distribution companies in Greece. However, even the replacement of petrol vehicles with diesel would offer significant reduction both to distribution costs and to CO_2 emissions.

Concluding, since the algorithm has the ability to efficiently handle multiple attributes, it also constitutes a library that includes all the necessary functions, stored in the software development company's server, and can be accessible by the presented in Chapter 6 system. Finally, the EA is most likely to be applied in the system and be offered as a choice to the users that need to optimize simultaneously two contradictory objectives simultaneously.

The algorithm's benefits seem to be significant and that is why each algorithm constitutes a library that includes functions for addressing the problem. The algorithm libraries are the core of the system and are stored into the server of the software development company, so that it can be accessed by the presented in Chapter 6 system through the appropriate request.

References

Ai TJ, Kachitvichyanukul V (2009) A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Comput Oper Res 36:1693–1702. https://doi.org/https://doi.org/10.1016/j.cor.2008.04.003

Baños R, Ortega J, Gil C, et al (2013) A hybrid meta-heuristic for multi-objective Vehicle Routing Problems with Time Windows. Comput Ind Eng 65:286–296. https://doi.org/10.1016/j.cie.2013.01.007

Berger J, Barkaoui M, Bräysy O (2003) A Route-Directed Hybrid Genetic Approach For The Vehicle Routing Problem With Time Windows. INFOR Inf Syst Oper Res 41:179–194. https://doi.org/10.1080/03155986.2003.11732675

Chiang TC, Hsu WH (2014) A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows. Comput Oper Res 45:25–37. https://doi.org/10.1016/j.cor.2013.11.014

Chiang WC, Russell RA (1997) A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows. INFORMS J Comput 9:417–430. https://doi.org/doi:10.1287/ijoc.9.4.417

Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. Wiley, Chichester

Clarke G, Wright JW (1964) Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Oper Res 12:568–581. https://doi.org/10.1287/opre.12.4.568

Czech ZJ, Czarnas P (2002) Parallel simulated annealing for the vehicle routing problem with time windows. In: 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing. pp 376–383

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197. https://doi.org/10.1109/4235.996017

Erdelić T, Carić T (2019) A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches. J Adv Transp 2019:5075671. https://doi.org/10.1155/2019/5075671

Erdelić T, Carić T, Erdelić M, Tišljarić L (2019) Electric vehicle routing problem with single or multiple recharges. Transp Res Procedia 40:217–224. https://doi.org/https://doi.org/10.1016/j.trpro.2019.07.033

European Automobile Manufacturers Association (2019) Vehicles in use - Europe 2019

Gambardella LM, Taillard É, Agazzi G (1999) Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In: New Ideas In Optimization. pp 63–76

Garcia-Najera A, Bullinaria JA (2011) An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Comput Oper Res 38:287–300. https://doi.org/10.1016/j.cor.2010.05.004

Gayialis SP, Tatsiopoulos IP (2004) Design of an IT-driven decision support system for vehicle routing and scheduling. Eur J Oper Res 152:382–398. https://doi.org/10.1016/S0377-2217(03)00031-6

Ghoseiri K, Ghannadpour SF (2010) Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Appl Soft Comput 10:1096–1107. https://doi.org/10.1016/j.asoc.2010.04.001

Giaglis MG, Minis I, Tatarakis A, Zeimpekis V (2004) Minimizing logistics risk through real-time vehicle routing and mobile technologies. Int J Phys Distrib Logist Manag 34:749–764. https://doi.org/10.1108/09600030410567504

Gillett BE, Miller LR (1974) A Heuristic Algorithm for the Vehicle-Dispatch Problem. Oper Res 22:340–349. https://doi.org/10.1287/opre.22.2.340

Gong G, Deng Q, Gong X, et al (2018) A Bee Evolutionary Algorithm for Multiobjective Vehicle Routing Problem with Simultaneous Pickup and Delivery. Math Probl Eng 2018:2571380. https://doi.org/10.1155/2018/2571380

Gong YJ, Zhang J, Liu O, Hung CHS (2012) Optimizing the Vehicle Routing Problem With Time Windows: A Discrete Particle Swarm Optimization Approach. IEEE Trans Syst Man, Cybern Part C Appl Rev 42:224–267. https://doi.org/10.1109/TSMCC.2011.2148712

Govindan K, Soleimani H, Kannan D (2015) Reverse logistics and closed-loop supply chain: A comprehensive review to explore the future. Eur J Oper Res 240:603–626. https://doi.org/10.1016/j.ejor.2014.07.012

Hiermann G, Puchinger J, Ropke S, Hartl RF (2016) The Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations. Eur J Oper Res 252:995–1018. https://doi.org/10.1016/j.ejor.2016.01.038

Konstantakopoulos GD, Gayialis SP, Kechagias EP, et al (2020) A Multiobjective Large Neighborhood Search Metaheuristic for the Vehicle Routing Problem with Time Windows. Algorithms 13:. https://doi.org/10.3390/a13100243

Labadie N, Prins C, Prodhon C, Prins C (2016) Metaheuristics for Vehicle Routing Problems. John Wiley and Sons

Lin J, Zhou W, Wolfson O (2016) Electric Vehicle Routing Problem. Transp Res Procedia 12:508–521. https://doi.org/https://doi.org/10.1016/j.trpro.2016.02.007

Nagy G, Salhi S (2005) Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. Eur J Oper Res 162:126–141. https://doi.org/https://doi.org/10.1016/j.ejor.2002.11.003

Ombuki B, Ross BJ, Hanshar F (2006) Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl Intell 24:17–30. https://doi.org/10.1007/s10489-006-6926-z

Salhi S, Nagy G (1999) A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. J Oper Res Soc 50:1034–1042. https://doi.org/10.1057/palgrave.jors.2600808

Solomon MM (1987) Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Oper Res 35:254–265. https://doi.org/10.1287/opre.35.2.254

Subramanian A, Uchoa E, Pessoa AA, Ochi LS (2011) Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. Oper Res Lett 39:338–341. https://doi.org/https://doi.org/10.1016/j.orl.2011.06.012

Subramanian A, Uchoa E, Pessoa AA, Ochi LS (2013) Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. Optim Lett 7:1569–1581. https://doi.org/10.1007/s11590-012-0570-9

Tan KC, Chew YH, Lee LH (2006) A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem. Comput Optim Appl 34:115–151. https://doi.org/10.1007/s10589-005-3070-3

Woch M, Łebkowski P (2009) Sequential Simulated Annealing for the Vehicle Routing Problem with Time Windows. Decis Mak Manuf Serv 3:87–100

Yousefikhoshbakht M, Didehvar F, Rahmati F (2014) A combination of modified tabu search and elite ant system to solve the vehicle routing problem with simultaneous pickup and delivery. J Ind Prod Eng 31:65–75. https://doi.org/10.1080/21681015.2014.893928

Chapter 5: The Heterogeneous Fleet Vehicle Routing Problem with Simultaneous Pickups and Deliveries and Time Windows

5.1 Introduction

In order to overcome the challenge of vehicle routing, logistics and distribution companies need to minimize their distribution cost by defining the fleet composition and the sequence of customers' deliveries of each route (Qureshi et al. 2009). This is not always an easy task for logistics companies since routes planning and deliveries scheduling are affected by multiple parameters and constraints, depending on the distribution case. In addition, the urbanization and the rise of e-commerce over the last few years has led to an increase in the daily orders that companies need to serve (Özarık et al. 2021). All these factors exacerbate the complexity of the problem and consequently complicate its resolution.

Additionally, regardless of industry, every company must have the ability to recognize and follow the needs of the external environment and utilize the latest technological advances. This can offer companies efficient solutions in terms of product and service quality and valuable time savings. The same stands for logistics companies whose main activity lies in their distribution operations. Therefore, to optimize their operations, logistics companies first need to consider the multiple parameters and constraints involved in routing vehicles and scheduling deliveries and, consequently, implement advanced routing and scheduling systems that can facilitate their distribution operations (Bräysy and Hasle 2014).

In the current Chapter, an Adaptive Large Neighborhood Search (ALNS) algorithm is developed for enhancing the efforts of logistics companies that operate in the field of freight distribution and thus considers the VRP variants that are most commonly addressed in this field. Among the most frequently encountered variants of the VRP in freight distribution, according to the current research, as well as to the research of Elshaer and Awad (2020), are (i) the VRP with Time Windows (VRPTW), (ii) , the Heterogeneous Fleet VRP (HVRP) and (iii) the VRP with Simultaneous Pickups and Deliveries (VRPSPD). The VRPTW and the VRPSPD have been thoroughly analyzed in Chapter 3 and Chapter 4, respectively. As for the HVRP, which is another VRP variant, is frequently studied by researchers and practitioners, but not as commonly as the VRPTW and VRPSPD. The Capacitated VRP (CVRP) significantly simplifies the problem, as the vehicles are identical in terms of cost and capacity (Máximo and Nascimento 2021), making the HVRP appear and be studied less commonly by researchers. However, this is contradictory to real-life cases, as the fleet of vehicles in most logistics companies differentiate, as multiple types of vehicles exist (Liu 2013; Irnich et al. 2014). Moreover, each type of vehicle has different fixed and variable costs, capacity, latest returning times, and a limited number of available vehicles that all need to be considered when planning the delivery schedules. As a result, and to cover the needs of most real-life logistics

companies, regardless of the types of vehicles each company owes, it was decided to deal with the heterogeneity of vehicles and, consequently, the HVRP.

Additionally, drivers' shifts constitute one additional significant parameter both in the current Chapter and in real-life cases, since the drivers have specific working hours. That means that vehicles can start the distribution process from the depot after the start of the drivers' shift and cannot return back after the end of their shift. This limitation has gained ground over the last few years as new stricter policies have pushed companies to comply with the labor legislation while also aiming to reduce road accidents due to overworking (Alcaraz et al. 2019). However, despite the importance of the variant for logistics companies, it has not yet attracted the appropriate attention from researchers studying the VRP, probably due to its complexity. In the current research, and to simplify this restriction, each shift has been transformed into the earliest starting and latest returning time of the vehicle.

The aforementioned parameters and constraints, when simultaneously considered, cover the needs of most logistics companies operating in the field of freight distribution. However, the complexity of the problem dramatically increases when multiple variants of the VRP are simultaneously considered, and that is the main reason for the reduced number of methods proposed by researchers compared to those with single VRP variants. In addition, most of these algorithms are not integrated into routing systems but instead are limited to research studies. Examples of such studies include Penna et al. (2019), who proposed a hybrid heuristic algorithm for a class of VRP's with a heterogeneous fleet, and Jiang et al. (2014), who studied the Heterogeneous Fleet VRP with Time Windows and proposed a tabu search algorithm belong in this category. Both algorithms manage to solve numerous datasets of the literature, proving their efficiency, but have not been implemented in software solutions and tested in real-life cases. The research that comes closest to the presented on is given by Erdoğan (2017), who developed an open-source spreadsheet solver for addressing most of the variants that are also considered in the current thesis. The researcher manages to connect online maps with Microsoft Excel and extract significant data related to routing and scheduling. However, the limitation of this solution is the increased computation time needed as Visual Basic (VB), which the researcher uses, is not as efficient as C++ and Python that are most often used for optimization and software development.

Moreover, the ALNS algorithm, besides being applied in all the above-mentioned VRP variants, is also applied in a Collaborative case, where three Third-Party Logistics (3PL) locating and distributing goods in the same area, cooperate in the distribution process. This approach may seem difficult to implement in Greece due to the ground that is not fertile yet for such synergies. However, this theoretical approach's economic and environmental benefits are enough to convince companies to move toward this direction in the future.

To sum up, the research conducted in this Chapter comes to support existing studies and present a metaheuristic algorithm that is used as a library, with all the necessary functions, and stored

in the software development company's server, so that to be accessible by the cloud routing system, presented in the next Chapter, that addresses the above VRP variants simultaneously. The current algorithm library constitutes the third that is developed for supporting the system. The algorithm developed in Python programming language can solve large-scale distribution cases, but at this stage is tested in datasets of the literature, including up to 100 customers. In order to enhance this effort and provide the desired outcomes, the variables and constraints involved in freight distribution and the mathematical formulation are subsequently defined in Section 5.2. In Section 5.3, the algorithmic approach, which includes the construction algorithm and the ALNS metaheuristic, is presented. The computational results obtained from the algorithm, when tested in datasets of the literature for the HVRPSPDTW, are presented in Section 5.4. Section 5.5 describes the Collaborative VRP and how the ALNS algorithm's implementation can contribute financially and environmentally to the cooperating companies. Finally, the concluding remarks of the Chapter are presented in Section 5.6.

5.2 Problem Description and Mathematical Formulation

All the characteristics and requirements described in Section 5.1 are set as variables and constraints in the problem formulation that follows. More specifically, a direct network G (N, A), where N is the set of nodes and $A = (i,j): i \neq j, i,j \in N$ is the set of arcs, is first defined. Node 0 represents the central depot, while $N^* = \{N/0\}$ represents the customers. Every arc (i,j), which is a path from node i to node j is characterized by its distance which is indicated as d_{ij}, and by its travel time t_{ij}. Each customer is characterized by the quantities of goods d_i and r_i that must be delivered and collected respectively, and by the time s_i^k that the vehicle needs for serving each customer. All goods that must be delivered, are transported from the central depot to customers, while the collected items follow the opposite direction. Also, each customer, as well as the depot, has a time window $[e_i, l_i]$, that represents the earliest and the latest times to start. Regarding to the depot, its time window, $[e_0, l_0]$ = [E, L], determines the earliest departure time from the depot and the latest returning time to the depot.

Additionally, the fleet of the heterogeneous vehicles is represented by set K, while each vehicle is indexed by $k \in K$. Each vehicle k corresponds to a unique vehicle type $c \in C$. Therefore, in order to correlate these two sets, S_c that represents the set of vehicles of type c is formed. Each vehicle type is characterized by its capacity Q_c, its fixed cost fc, its variable cost v_c, as well as by the maximum number of available vehicles, n_c. Finally, a driver is assigned in a one-to-one relationship to each vehicle, and is characterized by the starting time of the shift, p_k, as well as the ending time of the shift, m_k of the driver.

Finally, the studied problem requires some further variables so that the formulation be consistent. Starting with the decision variable x_{ij}^k that is equal to 1 if vehicle k drives from node i to node j, and 0 otherwise, and continuing with variable y_{ij}^k that refers to the total quantity on board, on arc (i, j). Moreover, the time variables include i) a_i^k which refers to the arrival time

of vehicle k at node i, ii) w_i^k that refers to the waiting time of vehicle k at node i, and iii) b_i^k which refers to the departure time of vehicle k from node i. Concluding, all the essential variables are gathered and presented below, before the mathematical model.

q_i	Delivery demand of node i
r_i	Pickup demand from node i
e_i	Node's i open time
li	Node's i close time
s_i^k	Service time of vehicle k at node i
Q_c	Capacity of type c vehicles
f_c	Fixed cost of type c vehicles
v_c	Variable cost of type c vehicles
n_c	Number of available vehicles for each type
p_k	Earliest starting time of vehicle
m_k	Latest returning time of vehicle
y_{ij}^k	Total pickup items on board of the vehicle k, on arc (i, j)
a_i^k	Arrival time of vehicle k at node i
w_i^k	Waiting time of vehicle k at node i
b_i^k	Departure time of vehicle k from node i

The studied problem can be formulated as follows:

$$\sum_{c\in C} f_c \sum_{k\in S_c}\sum_{j\in N^*} x_{0j}^k + \sum_{c\in C} v_c \sum_{k\in S_c}\sum_{(i,j)\in A} d_{ij}x_{ij}^k \quad (5.1)$$

$$\sum_{k\in S_c}\sum_{j\in N^*} x_{0j}^k \le n_c, \forall c \in C \quad (5.2)$$

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k \le 1, \forall i \in N \quad (5.3)$$

$$\sum_{j\in N} x_{0j}^k \le 1, \forall k \in K \quad (5.4)$$

$$\sum_{i\in N} x_{i0}^k \le 1, \forall k \in K \quad (5.5)$$

$$\sum_{i\in N} x_{il}^k - \sum_{j\in N} x_{lj}^k = 0, \forall l \in N^*, k \in K \quad (5.6)$$

$$\sum_{i\in N} q_i \sum_{j\in N} x_{ij}^k \le \mathrm{Q}_c, \forall k \in S_c, c \in C \quad (5.7)$$

$$\sum_{i\in N} r_i \sum_{j\in N} x_{ij}^k \le \mathrm{Q}_c, \forall k \in S_c, c \in C \quad (5.8)$$

$$\left(y_{ij}^k - q_j + r_j\right)x_{ij}^k \le \mathrm{Q}_c, \forall i \in N, j \in N, k \in S_c, c \in C \quad (5.9)$$

$$a_0^k \leq \mathrm{m}_k \forall k \in K \quad (5.10)$$

$$b_0^k \geq p_k, \forall k \in K \quad (5.11)$$

$$e_i \leq a_i^k + w_i^k \leq l_i \quad (5.12)$$

$$x_{ij}^k \in \{0,1\}, (i,j) \in A, \forall k \in K \quad (5.13)$$

$$y_{ij}^k \geq 0, \forall (i,j) \in A, k \in K \quad (5.14)$$

The objective function (5.1) aims to minimize the total cost derived from the composition of the variable and the fixed cost of each vehicle used in the distribution process. If a vehicle is not used for the distribution process, then the fixed and variable costs are considered zero. Constraint (5.2) ensures that the available number of vehicles for each type, is not exceeded, while constraint (5.3) that each customer is visited by only one vehicle. The set of constraints (5.4), (5.5) and (5.6) ensure that each vehicle serving a customer surely departs from this customer as well, and that each vehicle starts from the depot and returns to it after the completion of the route. Constraint (5.7) states that the total quantity of items delivered from each vehicle will not exceed the vehicle's capacity. Respectively, constrain (5.8) is applied in the case of goods returned from each customer to the depot. Constraint (5.9) ensures that the quantity of goods on board of each vehicle does not exceed its capacity at any stage of the route execution. Constraint (5.10) and (5.11) state that each vehicle may depart from the depot when the driver's shift opens and return to the central depot before the end of the driver's shift. Constraint (5.12) ensures that no vehicle may arrive at a customer after the end of the time window and that the serving of each customer starts after the opening of the time window. Finally, the decision variable x_{ij}^k is equal to 1 if vehicle k drives from node i to node j, and 0 otherwise, while variable y_{ij}^k is always a non-negative number.

5.3 Solution Method – Adaptive Large Neighborhood Search

The proposed ALNS algorithm integrated into the routing and scheduling system belongs in the category of local search metaheuristics. The algorithm is applied in a solution that is exploited by a construction heuristic algorithm that is analyzed in Section 5.3.1. Consequently, the individual pieces of the ALNS algorithm are presented in Section 5.3.2.

5.3.1 Construction Procedure

The construction algorithm applied in the current research is based on the Time-oriented Nearest Neighbor (TONN) heuristic, which was first proposed by Solomon (1987), and has been enhanced and adjusted to the requirements of the VRP variants that are considered in the current research. However, before applying the variation of the TONN algorithm, it is essential to construct and model the form of solutions firstly. Therefore, an array with all the available

vehicles is firstly constructed, so that each element of the array includes the unique id and the type of the vehicle. The type of the vehicle contains all the necessary characteristics and specification of the vehicles (capacity, variable and fixed costs etc.). In contrast, the unique id of each vehicle is correlated to a specific driver and thus to a specific earliest starting and latest returning time. Consequently, all the necessary data of vehicles to address the problem have been included in the vehicles' array. In addition, another array that includes the routes of vehicles is constructed. Since each vehicle corresponds to a single route, each row on the vehicles' array corresponds to a single row in the array of routes, as shown in Figure 5.1.

Moreover, the TONN algorithm belongs in sequential heuristics, which means that one route is built until all customers have been integrated into the routes. A new route starts only when no other customer can be added. However, the fact that each vehicle is characterized by different capacities, costs, shifts, and customer requirements (time windows, simultaneous pickups and deliveries) greatly affects the customers inserted into the routes. Therefore, a metric that calculates the "closest" customer to the last visited one is used in this procedure, and through this way, the array of routes is filled. This means that all customers have been assigned to vehicles and routes, and consequently, a complete solution has been extracted that can now be fed to the ALNS algorithm.

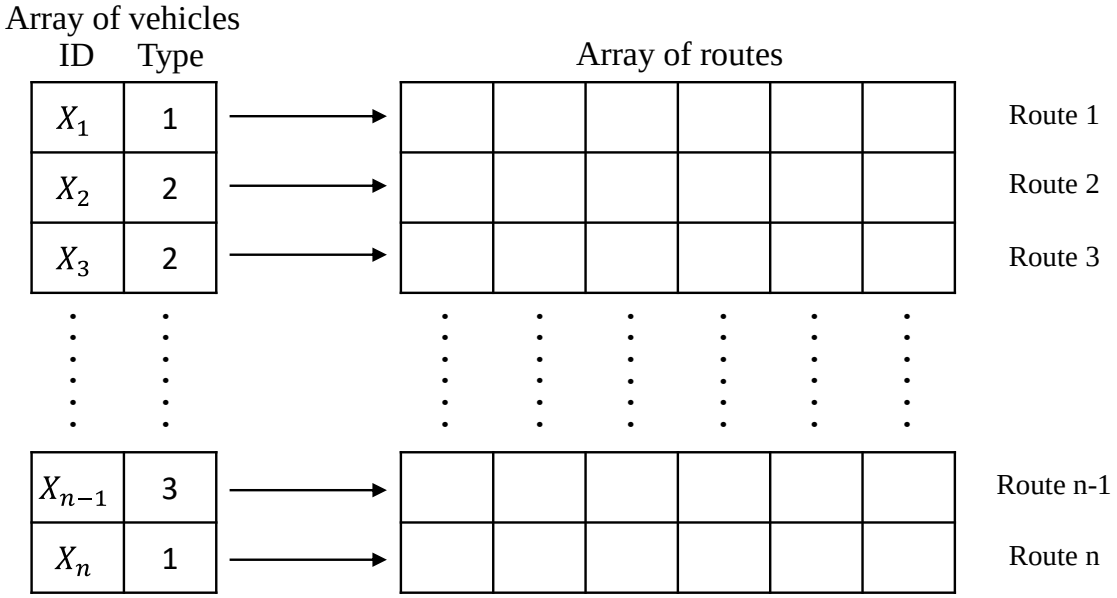


Figure 5.1 Array of Vehicles and Routes

5.3.2 Adaptive Large Neighborhood Search Algorithm

5.3.2.1 General Framework

The large neighborhood search (LNS) framework, which explores a large neighborhood by destroying and repairing solutions, was proposed by Shaw (1998). The Adaptive Large Neighborhood Search (ALNS) algorithm is an extension of the LNS procedure that also operates with various destroy and repair operators. Both procedures need an initial solution,

such as the one produced by the TONN algorithm, to be applied and operated. However, their main difference is observed in the probability of each operator to be applied. In LNS, the probability of each destroy operator (π^d), and of each repair operator (π^r) to be applied is steady and is defined at the start of the procedure. On the contrary, in the ALNS algorithm, the probabilities of each destroy and repair operator (π^d and π^r respectively) to be chosen increase the more the operator has contributed to the improvement of the solution. In each iteration of the algorithm, a new solution is constructed (X') after the destroy and repair operator, that is compared to the optimal solution so far (X^*). If the constructed solution X' is better than the optimal X^*, then the optimal solution is renewed. That is the general framework of the algorithm, which is also given in Figure 5.2.

Algorithm Adaptive Large Neighborhood Search

1: Construct an initial feasible solution X
2: $X^* \leftarrow X$
3: initialize probability of each destroy (π^d) and repair (π^r) operator to be applied
4: **while** the stopping criterion is not met **do**
5: probabilistically select a destroy and a repair operator, based on their weights π^d and π^r
6: apply the selected destroy and repair operator to X^* solution
7: $X' \leftarrow$ the new solution that has resulted from the selected destroy and repair operators
8: **if** X' satisfies the acceptance criterion **then**
9: **if** $c(X') \leq c(X^*)$ **then**
10: $X^* \leftarrow X'$
11: readjust weights π^d and π^r
12: return X^*

Figure 5.2 Framework of the ALNS Algorithm

5.3.2.2 Adaptive Mechanism

In each iteration of the ALNS algorithm, a destroy and a repair operator is applied. For adjusting the probability of each operator to be applied, the search is divided into time segments of t computation time. The probability of each operator is calculated by taking into account the performance of the operator and the improvement that has offered into the solution, during each segment. More specifically, the probability of each destroy operator for the next segment (t+1) is calculated by $\pi_{t+1}^d = \pi_t^d(1 - w^d) + w^d * ({s^d}/{i^d})$, where π_t^d is the probability of each operator to be applied, in the previous segment (t), s^d the number of times the operator applied and offered an improved solution, i^d the number of times any operator was applied and offered improved results, and finally w^d, the weight of the successful improvements in the specific

time segment. The same equation is adapted in the case of repair operators and is given by, $\pi_{t+1}^{r} = \pi_{t}^{r}(1 - w^{r}) + w^{r} * ({s^{r}}/{i^{r}})$.

5.3.2.3 Destroy Operators

In the current paper, six destroy and three repair operators are applied for improving the initially produced solution. Through this approach, and since the probability of each operator is calculated according to its success, the produced solution is greatly improved while stacking on local optimal is avoided. In addition, before selecting the destroy operators, multiple operators of the literature were tested. However, those that offer the best balance between computation speed and results were selected. More specifically, the destroy operators implemented in the current research are the i) random, ii) single route, iii) two routes, iv) distant, v) longer waiting time, and vi) least beneficial. Through these operators, customers are removed from the existing routes. The operators are analytically given below, for clarifying the way customers are selected and removed.

Random Removal. In this operator, a specific number of customers (q) that will be removed from the existing solution is initially determined. Specifically, the customers are selected randomly in the random destroy operator and are then removed from the routes. The number q of the selected customers ranges and depends on the size of the problem, meaning the number of orders.

Route Removal. As its name indicates an entire route is randomly selected in the single route destroy operator. Specifically, a route is destroyed and its customers aim to be inserted either to the existing routes or to a new lower cost vehicle.

Distant Removal. The distant destroy operator aims to find the more distant customers from the previously served and the upcoming customer, and remove them from routes. Thus, for each customer *j* the total distance $d_{ij} + d_{jk}$ is calculated, where *i* is the previously served customer and *k* the upcoming customer, as depicted in Figure 5.3.

Waiting-time Removal. In the waiting-time removal, for each customer *j* the waiting time $w_j = e_j - a_j$ is calculated. As it has already defined in section 3, e_j is the opening time of customer's j time window, and a_j the arrival time at customer j. In both the distant and waiting-time destroy operator, the number q of the selected customers ranges and depends on the total number of orders. Consequently, the q worst customers, based on the two operators mentioned above, are selected and removed from routes.

Two-routes removal. Additionally, in the two routes destroy operator, all customers that belong in the two routes are selected. All routes are evaluated according to the cost per load ($dc_z/(q_0 + r_0)$, where $dc_z = f_z + v_z \sum_{(i,j)\in A} d_{ij} x_{ij}^{z}$ is the distribution cost of route *x*, and q_0 the quantity on board when starting the route and r_0 the quantity of goods on board when returning to the depot. That means that from a pool of routes with increased cost per load factor,

two routes are selected randomly, and their customers are removed. The two routes are selected randomly from the pool to ensure that the solution will not stack in a local optimum since different routes may be selected in each iteration.

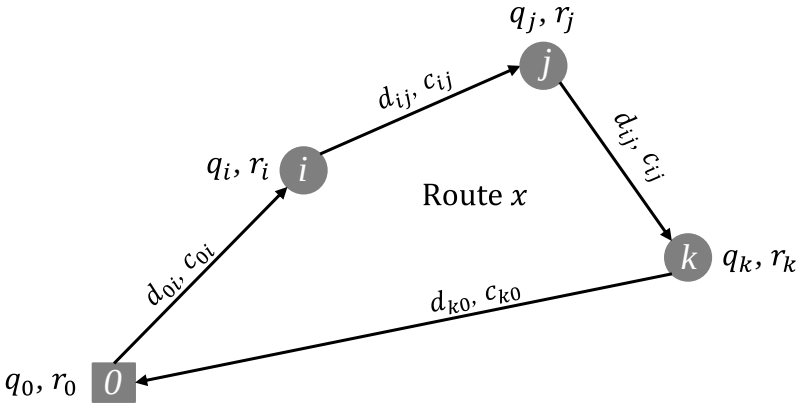


Figure 5.3 Route Example and Sequence of Deliveries

Uneconomical Removal. Finally, in the uneconomical destroy operator, customers are sorted according to factor u. For each customer j the factor u_j is calculated by $u_j = c_{ij}/q_j + c_{jk}/r_j$, where i is the previously served customer and k the upcoming customer, as depicted in Figure 5.3. Moreover, c_{ij} indicates the cost for moving from customer i to j, c_{jk} the cost for moving from customer j to k, q_j the delivery demand of customer j and r_j the returned quantity. That means that for each customer the cost per unit, both for the delivered items (c_{ij}/q_j) and the collected (c_{jk}/r_j), is calculated, constituting the criterion for sorting customers. Additionally, in this operator, as in the rest destroy operators, the number q of the customers to be selected ranges and depends on the number of orders. Therefore, according to indicator u, the q "worst" customers, namely those with increased cost per unit, are selected and removed from routes.

The "two routes" and the "least beneficial" destroy operators have been developed and adapted for the algorithm, which is its integration into the system. The specific operators may not be very efficient in datasets of the literature where not all these variables are considered but can be very efficient in real-life cases. However, for the algorithm to be efficient in any circumstance, multiple operators are developed and applied. Moreover, the ALNS can evaluate the operators according to their success and therefore apply the more productive, more often.

5.3.2.4 Repair Operators

After implementing a destroy operator, a repair method must be applied to reconstruct the solution. In the current research, three repair operators are implemented to find the best plan of routes and the schedule of deliveries, in terms of total distribution cost, by inserting the removed customers either into existing routes or into new routes. The repair operators that are applied in the current research are given below.

Cost insertion. The current repair operator works by inserting each customer into the position of the routes that the total distribution cost is minimized. The customer can even be positioned in a new route in the total cost is minimized.

Cost insertion without creating new routes. The second operator also uses the same criterion and restricts the search into routes that do not increase the fixed cost of all routes. That means that no new route can be created for inserting customers. This can be violated only in cases that there is no alternative and consequently a removed customer cannot be inserted into an existing route. In this case, the current operator is transformed into the cost insertion.

Traveled distance insertion. The third repair operator aims to place customers in the route and sequence, that the total traveled distance is minimized.

Finally, as in the case of destroy operators, multiple repair operators exist for avoiding stacking on local optimum, with the probabilities of the operators to be adapted according to their success to improve the solution. A repairing case with a single unrouted customer is depicted in Figure 5.4. The repair operator affects in a great extent in which route and sequence the customer will be placed.

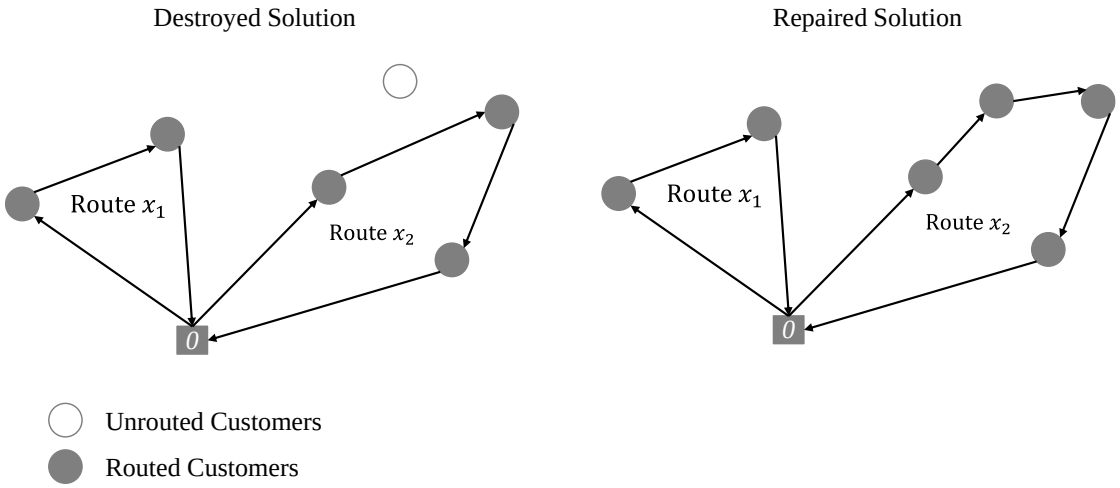


Figure 5.4 Repair of a Destroyed Solution

5.4 Computational Study – Case Studies

5.4.1 Datasets of the Literature

Before integrating the metaheuristic algorithm into the system, the algorithm was tested in datasets of the literature. The fact that there is no dataset considering all the VRP variants that are studied in the current research simultaneously, led us to address multiple datasets. Through this approach, the algorithm's efficiency can be better estimated as the algorithm is tested in multiple well-known datasets, and the results are compared to the best-published in the literature.

The study and validation of the algorithm starts with the HVRP that is divided into two main categories, (i) the fleet size and mix vehicle routing problem (FSMVRP) and (ii) the heterogeneous fixed fleet vehicle routing problem (HFVRP), or as it is otherwise known, the vehicle routing problem with a heterogeneous fleet of vehicles (VRPHE). The FSMVRP was introduced by Golden et al. (1984) and works with an unlimited number of available vehicles of each type. On the other hand, the HFVRP was introduced by Taillard (1999) and works with a limited number of available vehicles of each type. In both cases, the objective is to minimize the total distribution cost that is formed by the fixed (F) and variable (V) vehicle costs.

Moreover, the FSMVRP is further categorized according to the vehicle costs that are considered. More specifically, three different cases have been created, (i) the FSMVRP-F Golden et al. (1984), that only fixed costs are considered (while the variable cost is the same for each type of vehicles), (ii) the FSMVRP-V Taillard (1999), that only variable costs are considered, and (iii) the FSMVRP-FV (Ferland and Michelon 1988) that both fixed and variable vehicle costs are considered. In each of the three cases, all types of vehicles have different capacities and an unlimited number of available vehicles.

Over the years, only a few researchers have tried to solve the VRP that simultaneously considers heterogeneous vehicles and time windows. Liu and Shen (1999) were the first to tackle the FSMVRP with time windows (FSMVRPTW), by reclaiming Solomon's VRPTW benchmark instances. However, this variant only considers different capacities and fixed costs for each vehicle type, while no variable costs and no limited number of available vehicles are considered, greatly simplifying the problem. With this opportunity, multiple researchers addressed the HFVRP with time windows (HFVRPTW), as it considers characteristics that did not previously faced. Most papers proposing algorithms for addressing the HFVRPTW, focus on real-life applications, while Jiang et al. (2014) were the first that reclaimed Solomon's VRPTW benchmark instances for creating a dataset for the HFVRPTW. In the current study, the algorithm is tested in all of these datasets and in real-life distribution cases to obtain a more comprehensive view of the algorithm. In the subsections that follow, the results of each of the dataset testing are presented.

It should be noted that the datasets mentioned above only cover the variants of vehicle heterogeneity and time windows without considering simultaneous pickups and deliveries. Therefore, to validate the algorithm in this variant, the dataset of Salhi and Nagy (1999) that includes 14 problem instances was exploited. This way, the algorithm's efficiency was tested in every variant considered in the research.

5.4.1.1 FSMVRP

Starting with the FSMVRP, three cases were solved, FSMVRP-F, FSMVRP-FV and FSMVRP-V. Each problem instance runs ten times, and from each, we exclude the average value and the best-obtained. Moreover, regardless of the case, the algorithm presents similar results as the mean deviation between the best-published and the best-obtained results in

FSMVRP-F is 2.02%, in FSMVRP-FV is 3.18%, and finally in FSMVRP-V it is 1.90%. Additionally, the results obtained from the algorithm in each case instance are thoroughly presented in Tables 5.1, 5.2 & 5.3 and are also compared to the best-published ones from the literature. Finally, the deviation of any of the problem instances does not exceed 5.95%, which is an essential factor that confirms the efficiency of the algorithm in this specific dataset.

Table 5.1 FSMVRP-F Dataset Results

Problem	Best-published	References	Best-obtained	Avg. Value	Deviation
13	2406.36	(Choi and Tcha 2007)	2480.15	2505.46	3.07%
14	9119.03	(Choi and Tcha 2007)	9153.82	9199.63	0.29%
15	2586.37	(Choi and Tcha 2007)	2600.68	2606.80	0.55%
16	2720.43	(Choi and Tcha 2007)	2801.26	2817.26	2.97%
17	1744.83	(Choi and Tcha 2007)	1789.9	1823.9	2.58%
18	2371.49	(Choi and Tcha 2007)	2512.63	2512.63	5.95%
19	8661.81	(Taillard 1999)	8880.12	8908.03	2.52%
20	4039.49	(Choi and Tcha 2007)	4175.62	4215.58	3.37%
Avg.	4212.04	-	4299.935	4331.14	2.02%

Table 5.2 FSMVRP-V Dataset Results

Problem	Best-Published	References	Best-obtained	Avg. Value	Deviation
13	1491.86	(Gendreau et al. 1999)	1568.21	1574.25	5.12%
14	603.21	(Taillard 1999)	630.25	634.86	4.48%
15	999.82	(Gendreau et al. 1999)	1025.63	1028.51	2.58%
16	1131	(Wassan and Osman 2002)	1149.25	1172.52	1.61%
17	1031	(Gendreau et al. 1999)	1059.6	1065.35	2.77%
18	1801.4	(Gendreau et al. 1999)	1848.52	1854.52	2.62%
19	1100.56	(Wassan and Osman 2002)	1153.25	1158.33	4.79%
20	1530.16	(Wassan and Osman 2002)	1575.84	1585.74	2.99%
Avg.	1212.72	-	1251.31875	1259.26	3.18%

Table 5.3 FSMVRP-FV Dataset Results

Problem	Best-Published	References	Best-obtained	Avg. Value	Deviation
13	2964.65	(Choi and Tcha 2007)	3099.25	3156.51	4.54%
14	9126.9	(Choi and Tcha 2007)	9153.82	9166.31	0.29%
15	2634.96	(Choi and Tcha 2007)	2674.32	2683.25	1.49%
16	3168.92	(Choi and Tcha 2007)	3193.19	3205.45	0.96%
17	2004.48	(Tarantilis et al. 2004b)	2068.74	2087.56	3.21%
18	3147.99	(Choi and Tcha 2007)	3201.56	3241.32	1.70%
19	8664.29	(Choi and Tcha 2007)	8869.32	8917.41	2.37%
20	4154.49	(Choi and Tcha 2007)	4289.37	4312.52	3.25%
Avg.	4484.09	-	4450.69	4609.95375	1.90%

5.4.1.2 FSMVRPTW

As it has already been mentioned, Liu and Shen (1999) deployed and combined the dataset of Golden et al. (1984) for the FSMVRP, and the dataset of Solomon for the VRPTW with 100 customers. The dataset proposed by Solomon includes sets:

- C1 and C2, with geographically clustered customers
- RC1 and RC2, in which a mix of random and clustered structures is included
- R1 and R2 in which geographical data are randomly generated

As observed in column "Instance Set" next to each set letter (C, RC and R) there is an index (1 or 2). Index 1 in C, RC and R sets indicates a short scheduling horizon, narrow time windows, and low vehicle capacities. On the contrary, index 2 indicates that the sets have a long scheduling horizon, wide time windows, and high vehicle capacities. Liu and Shen (1999) also added in this column three cost structures A, B, and C that are characterized by different fixed vehicle costs. A structure includes the highest fixed cost, while C the minimum. Finally, the variable cost in each problem instance, and regardless of the vehicle type, is equal to 1.

Table 5.4 presents the summarized results of the proposed ALNS algorithm when tested in the FSMVRPTW dataset of Liu and Shen (1999). The best-obtained results that have emerged from ten runs of the algorithm are compared to the best-obtained from the HEA (Koç et al. 2015), MDA (Bräysy et al. 2008), BPDRT (Bräysy et al. 2009), UHGS (Vidal et al. 2014) and MSDA (Jiang et al. 2014), as these studies propose the best-published results from the literature. The first column indicates the instance set, as well as the number of cases tested. The columns TC show the average total cost of each method (including the proposed ALNS) for each instance set and column Dev. the percent deviation of the values exploited by each method compared to the proposed ALNS method. Finally, in the column (BKS <) the number of instances that the ALNS algorithm performed better results compared to all the methods were defined.

Table 5.4 Summarized Results on the FSMVRPTW

Instance	MDA		BPDRT		UHGS		HEA		MSDA		ALNS	BKS
Set	TC	Dev.	TC	Dev.	TC	Dev.	TC	Dev.	TC	Dev.	TC	<
R1A (12)	4068.59	7.32%	4060.96	7.52%	4031.28	8.31%	4041.46	8.04%	4064.25	7.43%	4366.33	0
R1B (12)	1854.60	8.27%	-	-	1841.43	9.05%	1839.40	9.17%	1854.55	8.28%	2008.06	0
R1C (12)	1539.48	8.79%	1539.90	8.76%	1530.25	9.45%	1525.56	9.78%	1539.48	8.79%	1674.82	0
C1A (9)	7085.56	8.83%	7085.91	8.83%	7082.98	8.87%	7082.98	8.87%	7085.56	8.83%	7711.30	0
C1B (9)	2335.11	6.17%	-	-	2332.90	6.27%	2332.90	6.27%	2335.11	6.17%	2479.28	0
C1C (9)	1615.75	6.20%	1615.40	6.23%	1615.49	6.22%	1615.39	6.23%	1615.75	6.20%	1715.97	0
RC1A (8)	4944.48	6.35%	4935.52	6.55%	4891.25	7.51%	4916.41	6.96%	4944.48	6.35%	5258.60	0
RC1B (8)	2121.62	-3.01%	-	-	2107.08	-2.34%	2103.21	-2.16%	2121.62	-3.01%	2057.74	6
RC1C (8)	1741.78	7.34%	1749.66	6.86%	1734.36	7.80%	1725.44	8.36%	1741.78	7.34%	1869.61	0
R2A (11)	3193.41	3.80%	3180.59	4.22%	3151.95	5.17%	3150.30	5.22%	3193.39	3.80%	3314.89	0
R2B (11)	1392.92	6.11%	-	-	1351.91	9.32%	1351.51	9.36%	1392.92	6.11%	1477.96	1
R2C (11)	1149.65	6.22%	1149.11	6.27%	1128.71	8.19%	1126.42	8.41%	1149.65	6.22%	1221.16	0
C2A (8)	5690.87	5.19%	5689.40	5.21%	5686.75	5.26%	5686.75	5.26%	5690.87	5.19%	5985.96	0
C2B (8)	1698.51	6.93%	-	-	1686.75	7.67%	1686.75	7.67%	1698.51	6.93%	1816.19	0
C2C (8)	1186.03	6.76%	1185.70	6.79%	1185.19	6.84%	1185.19	6.84%	1186.03	6.76%	1266.20	0
RC2A(8)	4241.33	4.63%	4231.25	4.88%	4210.10	5.41%	4202.61	5.60%	4241.33	4.63%	4437.89	0
RC2B(8)	1704.13	8.17%	-	-	1686.63	9.29%	1686.47	9.30%	1703.38	8.22%	1843.32	0
RC2C(8)	1374.55	8.61%	1385.32	7.77%	1358.24	9.92%	1359.33	9.83%	1374.55	8.61%	1492.97	0

The developed ALNS algorithm seems highly competitive given the results, as the deviation in all instance sets does not exceed 8.83%. Additionally, in set RC1B it managed to offer better solutions in 6 instances, while in set R2B in 1 of them. Finally, irrespective of the instance set, the clustering of customers or the time windows width, the ALNS algorithm performs equally well judging from the similar deviation in each instance set.

5.4.2 HFVRP

5.4.2.1 HFVRP

The dataset of Taillard (1999) came a few years later than the one of Golden et al. (1984) and has attracted the interest of various researchers. Many consider that the limited number of available vehicles of each type seems to be a more realistic scenario for logistics operations. Therefore, the ALNS algorithm was also tested in this dataset, as shown in Table 5.5, and managed to produce a new best-published result in Problem Instance 16, as it reduced the total cost slightly by 0.06%. In the rest Problem Instances, the algorithm indicates good quality results, with the deviation ranging from 1.34% up to 7.85%, as shown in Table 5.5. Specifically, each problem runs 10 times, and from these runs emerge both the best-obtained results and the average value.

Table 5.5 HFVRP Dataset Results

Problem	Best-Published	References	Best-obtained	Avg. Value	Deviation
13	1517.84	(Li et al. 2007)	1538.12	1547.92	1.34%
14	607.53	(Li et al. 2007)	620.38	622.44	2.12%
15	1015.29	(Tarantilis et al. 2004b)	1044.24	1050.34	2.85%
16	1144.94	(Li et al. 2007)	1144.23	1150.25	**-0.06%**
17	1061.96	(Li et al. 2007)	1100.13	1106.40	3.59%
18	1823.58	(Li et al. 2007)	1891.25	1920.95	3.71%
19	1117.51	(Taillard 1999)	1205.23	1223.73	7.85%
20	1534.17	(Li et al. 2007)	1602.42	1613.14	4.45%
Avg.	1228.21		1268.25	1297.84	3.23%

5.4.2.2 HFVRPTW

The last dataset that the ALNS algorithm was tested is Jiang et al. (2014). The authors derived Solomon's dataset and created different vehicle types characterized by their capacity, fixed and variable cost, number of available vehicles, and the latest returning time to the depot. The ALNS algorithm can manage all the above characteristics, including the latest returning time that can be considered as the end of the drivers' shifts. While seeming to be the closest one to real-life distribution cases, this dataset has yet to attract the interest of researchers it deserves. The proposed ALNS algorithm performs highly efficiently as, in 9 out of the 56 instances that exist, it proposes new best solutions, as seen in Table 5.6. The best-obtained result of each problem case results from 10 runs of the algorithm. Furthermore, all these instances range from -5.64% up to 7.57, with the mean deviation being 2.19%. Considering the heterogeneity of the

data, the algorithm performs with high efficiency even in cases where multiple variables and constraints are simultaneously considered.

Table 5.6 HFVRPTW Dataset Results

Problem Case	Best-Published Total Cost	Best-Obtained Total Cost	Dev. (%)	Problem Case	Best-Published Total Cost	Best-Obtained Total Cost	Dev. (%)
HC101	1885.33	1908.24	1.22	HR112	4252.71	4441.68	4.44
HC102	1890.66	1848.43	-2.23	HR201	1765.74	1666.2	-5.64
HC103	1908.04	1852.11	-2.93	HR202	1536.81	1517.77	-1.24
HC104	1809.78	1841.47	1.75	HR203	1337.39	1374.59	2.78
HC105	1854.73	1865.86	0.60	HR204	1114.94	1156.57	3.73
HC106	1880.64	1929.07	2.58	HR205	1263.91	1314.19	3.98
HC107	1839.52	1847.95	0.46	HR206	1180.44	1221.39	3.47
HC108	1826.49	1830.6	0.23	HR207	1102.06	1159.41	5.20
HC109	1799.22	1878.54	4.41	HR208	1007	1052.89	4.56
HC201	1313.28	1341.41	2.14	HR209	1119.04	1190.51	6.39
HC202	1283.58	1273.89	-0.75	HR210	1307.53	1293.28	-1.09
HC203	1259.97	1287.54	2.19	HR211	1010.22	1086.68	7.57
HC204	1256.09	1254.25	-0.15	HRC101	5703.97	5864.28	2.81
HC205	1325.84	1349.77	1.80	HRC102	5556.02	5739.09	3.29
HC206	1263.63	1286.34	1.80	HRC103	5438.89	5560.29	2.23
HC207	1307.35	1326.54	1.47	HRC104	5331.41	5485.83	2.90
HC208	1190.81	1238.6	4.01	HRC105	5705.79	5923.23	3.81
HR101	5125.252	5190.83	1.28	HRC106	5528.42	5763.65	4.25
HR102	4982.39	4976.31	-0.12	HRC107	5451.31	5559.39	1.98
HR103	4661.22	4764.39	2.21	HRC108	5322.31	5447.13	2.35
HR104	4530.55	4640.73	2.43	HRC201	4501.65	4607.61	2.35
HR105	4570.39	4802.39	5.08	HRC202	4408.53	4483	1.69
HR106	4431.31	4621.01	4.28	HRC203	4321.87	4305.16	-0.39
HR107	4391.14	4519.26	2.92	HRC204	4306.65	4284.38	-0.52
HR108	4280.05	4508.36	5.33	HRC205	4452.88	4578.74	2.83
HR109	4339.86	4581.27	5.56	HRC206	4419.09	4439.73	0.47
HR110	4272.25	4474.73	4.74	HRC207	4343.55	4423.39	1.84
HR111	4366.32	4533.23	3.82	HRC208	4276.15	4286.65	0.25

5.4.3 VRPSPD

The testing of the algorithm in VRPSPD constitutes an equally important stage of the algorithm's validation. The dataset of Salhi and Nagy (1999) is the only one of the studied datasets that address the VRPSPD. Consequently, this specific dataset is addressed individually, and the testing results are presented in Table 5.7. In this table, for each problem instance, the results obtained by the ALNS algorithm after ten runs are given in the

homonymous column, along with the best-published results. Once again, the results validate the algorithm's efficiency since the mean deviation between the obtained and best-published solutions is just 3.72%. Finally, in all problem instances, the deviation (column "Deviation (%)") between the solution obtained by the ALNS algorithm and the best-published ones does not exceed 6.18%.

Table 5.7 Comparing Routing Methods in Real-life Distribution Cases

Problem	ALNS		Best-published		Deviation (%)
Instance	NV	TD	NV	TD	
CMT1X	3	467.85	3	467	0.18%
CMT2X	7	700.35	7	676	3.60%
CMT3X	5	733.8269	5	703	4.39%
CMT4X	7	869.23	7	847	2.62%
CMT5X	11	1087.28	10	1025	6.08%
CMT6X	5	588.33	6	557	5.62%
CMT7X	9	973.3	11	919	5.91%
CMT8X	9	898.25	9	896	0.25%
CMT9X	15	1234.96	15	1215	1.64%
CMT10X	19	1613.2	19	1571	2.69%
CMT11X	4	904.52	4	868	4.21%
CMT12X	5	708.21	5	667	6.18%
CMT13X	11	1603.5	11	1560	2.79%
CMT14X	10	874.52	10	826	5.87%

5.5 The Case of Collaborative Logistics

5.5.1 Benefits by Applying the ALNS to the Collaborative VRP

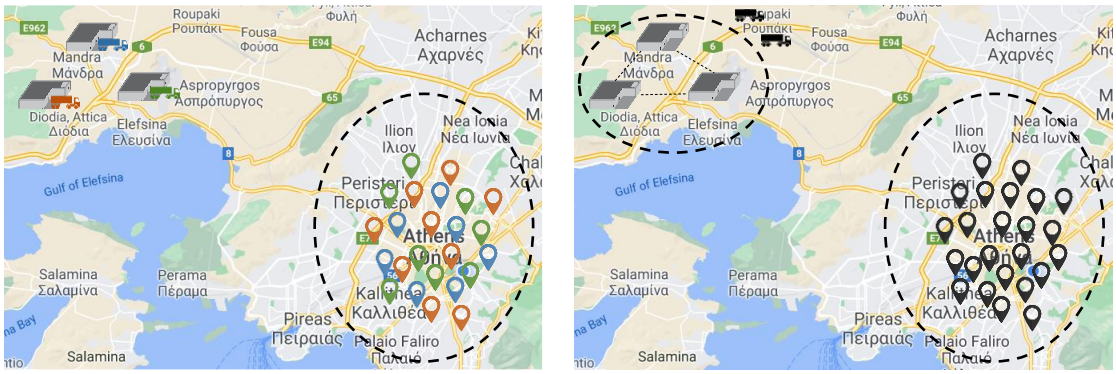
As stated in Chapter 2, the Multi-depot VRP and the Collaborative VRP are among the most commonly addressed variants. However, through the cooperation with logistics and software development companies, it was decided to avoid integrating any of these variants to the system, as it would get out of scope. That is mainly because the MDVRP is addressed in most cases as multiple single-depot cases, in which customers are assigned into a specific warehouse. As for the Collaborative VRP, the ground is not fertile for such actions yet. However, since new approaches need to be made for reducing the distribution cost and emissions, the case of collaboration is studied.

Let us just consider that a few years ago, logistics companies, city inhabitants and consumers would have been satisfied with just accurate (on-time) and low-cost deliveries, as they constitute the main goals in distribution for these parties. However, in the current state, where the reduction of traffic congestion and air pollution in city centers have gained increased priority for all stakeholders (Xiao and Konak 2016), new technologies, applications (Kechagias

et al. 2020a) and strategies (Kechagias et al. 2020b) are needed to be implemented, for satisfying the needs of all parties involved. In this direction, companies could share their resources, such as their vehicles, to reduce logistics costs and minimize the environmental impact of the last-mile distribution process. This strategy is either known as logistics pooling or collaborative logistics (Wang et al. 2018) and has been implemented in multiple cases worldwide (Rodrigues et al. 2018).

In Athens, Greece, collaborative logistics was by no means a common strategy, preferred by logistics and distribution companies in the past. The logistics companies were not keen on sharing information, software tools, and platforms, which becomes essential when sharing their vehicles. This stance stems from the lack of openness and spirit of cooperation and has been the main reason for pushing away the idea of synergies. However, the stance against the collaborative routing of vehicles and scheduling of deliveries seems to change, as companies have started to recognize its benefits. These benefits are mainly related to the reduction of distribution costs and greenhouse gas emissions and the increase of competitiveness.

In the present thesis, the prevailing conditions of the distribution of goods from 3PL companies to customers located in city centers, are presented, along with the concept of collaboration between these companies. Figure 5.5 presents both cases and describes that companies located in the same area organize and execute routes separately in the existing situation. On the other hand, according to the proposed concept, shared vehicles that receive goods from the warehouses of multiple companies execute routes that serve all customers. While the main benefit seems to be the decrease of needed vehicles at first sight, this cannot be thought as de-facto without conducting further analysis. Therefore, to enhance the research, and extract more accurate and precise results, real-life distribution cases of three 3PL companies need to be analyzed.



(a) (b)

Figure 5.5 (a) Current State Prevailing on most 3PL Companies in Greece. (b) Proposed Collaborative Strategy

More specifically, the benefits from the transition from the current state to collaborative logistics are estimated and calculated through the metaheuristic algorithm presented in the previous sections of Chapter 5. In the AS-IS scenario, the developed algorithm solves for each company separately, multiple daily distribution cases. On the contrary, in the collaborative

routing and scheduling scenario, the daily distribution cases of all companies are united, forming a larger daily distribution case. In this case, the vehicles involved in the distribution originate from all 3PL companies and may need to load goods from all warehouses before executing the routes and the schedule of deliveries.

The 3PL companies, when operating separately, combine and consider specific variants which are (i) the Heterogeneous Fleet VRP (HFVRP), in which vehicles are different in terms of capacity, fixed and variable costs (Subramanian et al. 2012), (ii) the VRP with Time Windows (VRPTW), that customers indicate time slots in which vehicles may arrive and start serving (de Oliveira and Vasconcelos 2010; Konstantakopoulos et al. 2020b), and (iii) the VRP with Simultaneous Pickups and Deliveries (VRPSPD). These are the variants that are considered in the proposed ALNS algorithm. In the synergy distribution cases, the Collaborative VRP (CVRP) (Gansterer and Hartl 2018) is also considered, along with the previous variants. This means that all the above characteristics and requirements exist, along with the fact that the resources, vehicles and deliveries of multiple companies are combined. The aim in both cases is to minimize the total distribution cost while all constraints are respected.

5.5.2 Collaborative Logistics and VRP Variants

The highly competitive industry of logistics and distribution forces companies to find efficient solutions in order to either maintain their position in business or expand it. These solutions may range from advanced information systems, algorithms and new technologies to collaborations between companies, or even to the combination of those. In addition, while technological solutions are most often preferred by companies due to their efficiency, their cost is sometimes prohibitive, leading companies to other solutions. On this premise, and due to the increased demands and orders in urban areas, logistics companies plan specific logistics processes jointly.

Route planning constitutes an operation that belongs in the specific category of supply chain collaborations. However, a prerequisite in order logistics companies to collaborate in route planning is to perform similar logistics processes and functions and operate at the same level in a supply chain (Verdonck et al. 2013; Ferrell et al. 2020). Researchers identify this type of collaboration as horizontal (Pan et al. 2019; Santos et al. 2021b). On the other hand, the case in which multiple stakeholders (manufacturers, suppliers, distribution centers, carriers, customers etc.), that operate in different levels in a supply chain collaborate, is identified as vertical (Cleophas et al. 2019; Santos et al. 2021a). Both types of collaboration can be beneficial for companies and improve operational planning, but they focus on different aspects of a supply chain.

In the current chapter of the thesis, the horizontal collaboration between three 3PL companies in the routing and distribution process is studied. This means that the customer requests of all companies that represent a shipment either from the distribution center of the company to the customer or the opposite are shared, along with their resources (vehicles). These variables define to a great extent the joint vehicle routing problem that can also be mathematically

formulated. Soysal et al. (2018) and Montoya-Torres et al. (2016) proposed such models and concluded that joint route planning offers significant reduction in the distribution cost and emissions. However, multiple factors may influence this reduction, such as the size of orders, the number of collaborating companies, as well as the variables and constraints considered in the routing procedure (Cruijssen and Salomon 2004). That is why defining the variants of the VRP that are involved in the routing process is very important.

More specifically, in the Collaborative VRP researchers focus on the multiple depots that exist, since each logistics company has its central depot. Due to this fact, many researchers consider the CVRP an extension of the Multi-depot VRP (MDVRP) (Krajewska et al. 2008; Karakatič and Podgorelec 2015; Montoya-Torres et al. 2015), where additional resources and orders are shared. In the current research, the Collaborative VRP is formulated as a multi-depot routing case. Additionally, the variants of the VRP presented in the previous sections of Chapter 5, are also the variables and constraints that the studied 3PL companies consider in their operations. The main difference compared to the variables considered in the previous sections of Chapter 5 is that in each vehicle, the CO_2 emissions are considered, while all drivers operate in the same shift.

In conclusion, all the above variables and constraints derived from the 3PL companies are considered in this study and included in the mathematical formulation of the next section. More specifically, the customer requests and the resources of logistics companies are gathered in a central pool (Verdonck et al. 2013). The collaboration strategy is formulated as a multi-depot vehicle routing problem with a heterogeneous fleet of vehicles, simultaneous pickups and deliveries, and time windows. As in most VRP variants, this problem aims to minimize the total distribution cost by defining the fleet composition and the specific schedule of deliveries in each route (Laporte 2007). This effort is enhanced by the ALNS algorithm that handles all these variables and constraints, aiming to offer efficient and sustainable routes.

5.5.3 Problem Description

In this class of VRP variants that are studied in the current Chapter, the set of depots (warehouses) D, as well as the set of customers N need to first be defined. The set $G = D \cup N$ defines the entire set, while $A = \{(i,j): i \neq j \in G\}$, the set of arcs. Each arc (i, j), which is a path from i to j, is characterized by the distance d_{ij}, and by the travel time t_{ij}. Each customer i is associated with a demand for deliveries r_i, a demand for pickups p_i, a service time s_i, and a time window $[e_i, l_i]$. The depots also have time windows which are emerged by the working hours of the logistics center. In each case, a vehicle may arrive before the start of the time window e_i, and may have to wait until it opens, but never after the end of the time window, l_i.

In addition, the fleet of vehicles is represented by set K, while each vehicle $k \in K$ is characterized by its capacity Q_k, its variable cost v_k and its fixed costs f_k. In each route, and consequently in each vehicle, the goods on board can in no case, as well as in no stage of the

distribution process, to exceed the vehicle's capacity. Therefore, in each customer, both the delivered and the collected quantity of goods must be considered. All these variables, along with the limitations, make the studied problem complex and hard to solve.

In the current state, each 3PL company has a single depot, and manages the data of customers and vehicles itself. On the other hand, in the case of Collaborative VRP, the depots come from the 3PL companies that cooperate, while the data of vehicles and customers of the cooperating companies are being united, forming problem of bigger size. However, the mathematical formulation remains the same, irrespective of the case, and is given below.

$$\sum_{k\in K}\sum_{i\in D}\sum_{j\in N} x_{ij}^{k}\cdot f_k + \sum_{k\in K}\sum_{(i,j)\in A} d_{ij}\cdot x_{ij}^{k}\cdot v_k \quad (5.15)$$

$$\sum_{i\in D}\sum_{j\in N} x_{ij}^{k} \le 1, \forall k \in K \quad (5.16)$$

$$\sum_{i\in G} x_{il}^{k} - \sum_{j\in G} x_{lj}^{k} = 0, \forall l \in N, k \in K \quad (5.17)$$

$$\sum_{j\in N} x_{ij}^{k} = 1, \forall i \in G, k \in K \quad (5.18)$$

$$\sum_{j\in G} (u_{ij}^{k} - r_j + p_j)\cdot x_{ij}^{k} \le \mathrm{Q}_k, \forall i \in G, k \in K \quad (5.19)$$

$$\sum_{j\in N} u_{ij}^{k} \le \mathrm{Q}_k, \forall i \in D, k \in K \quad (5.20)$$

$$\sum_{i\in N} u_{ij}^{k} \le \mathrm{Q}_k, \forall j \in D, k \in K \quad (5.21)$$

$$e_i \le a_i^{k} + w_i^{k} \le l_i, \forall i \in G, k \in K \quad (5.22)$$

$$x_{ij}^{k} \in \{0,1\}, (i,j) \in A, \forall k \in K \quad (5.23)$$

$$u_{ij}^{k} \ge 0, \forall (i,j) \in A, k \in K \quad (5.24)$$

The objective function (5.15) seeks to minimize the total distribution cost, that consists of the fixed and variable costs of all routes. Inequality (5.16) ensures that each vehicle starts from one of the available depots (if multiple exist, such as in the Collaborative VRP), while equations (5.17) and (5.18) that each customer is served only once and by one vehicle. Constraints (5.19), (5.20) and (5.21) ensure that the load of goods in every stage of the distribution process, including the departure from the depot and the return to the depot, cannot exceed the vehicle's capacity. Constrain (5.22) ensures that each vehicle cannot start serving any customer before the time window starts or after its close. Finally, the decision variable x_{ij}^{k} is equal to 1, if the

route from i to j is executed, u_{ij}^k is the load on vehicle k and on arc (i, j), while α_i^k is the arrival time of vehicle k at customer i, and w_i^k the waiting time of vehicle k at customer *i*.

5.5.4 Case Studies

The need of companies to reduce both their distribution costs and their environmental footprint has led researchers and practitioners to search for new strategies to achieve these goals. The collaborative VRP studied in the current research can manage both factors and enhance this effort. The data that are essential for addressing the problem is exploited from three medium-size 3PL companies. All companies are located in Mandra, a suburb in the western area of Athens, Greece, and distribute goods in the city center. Therefore, their cases can offer significant results concerning collaborations between companies in their distribution operations.

Initially, the distribution cost and the CO_2 emissions are calculated for each company, and for numerous daily distribution cases that each company addresses, separately. However, since the three companies distribute goods in Athens, the daily distribution cases that each company faces, are also combined producing a single daily case, with multiple depots, shared vehicles, and customer data resulting from all companies, in order to form the case of collaborative logistics. The distribution cost and the CO_2 emissions are additionally calculated, and the results are compared to those exploited from the first case to estimate the benefits of this strategy.

In order to create the schedule of deliveries, as well as to calculate the distribution cost and the total emissions, multiple data are needed. These data originate and are related to both the customers and the vehicles and are used either as variables or as restrictions of the vehicle routing and scheduling problem.

Concerning the customers, each one separately is characterized by his ID number, address, the delivery and pickup quantity, and time window, as shown in Table 5.8. The addresses are essential for extracting all the necessary distances and travel times between customers (including the depots). For creating the distance and travel time matrices, online services are exploited. More specifically, these services, including geocoding and distance and travel time calculation, are offered by HERE Technologies (www.here.com). All these data, along with the delivery and pickup quantities, as well as with the time windows, are further exploited by the heuristic algorithm as variables and constraints of the problem for optimizing the plan of routes and schedule of deliveries. More specifically, the delivery and pickup quantities are considered in every stage of the distribution process so that the goods on board will not exceed the vehicle's capacity, while the time windows are considered to avoid delays.

Table 5.8 Representative Data of Customers

ID	Address	Delivery quantity (kg)	Pickup quantity (kg)	Start of time window	End of time window
1001	12 Arrianou, Athens, Attica 116 35, Greece	24.3	8.1	07:30	08:30
1002	1 Angelon, Peristeri, Attica 121 36, Greece	105.2	12	09:15	10:15
1003	12 Ithakis, Athens, Attica 112 51, Greece	31.8	2.5	08:00	09:00
1004	3 Troias, Athens, Attica 113 62, Greece	11.2	33.2	13:00	15:00
xxxx	6 Ippokratous, Athens, Attica 114 72, Greece	78.4	38.6	12:30	14:00

As for the vehicles, the variable and fixed costs have been calculated based on the data provided by each one of the 3PL companies. More specifically, the fixed cost of each vehicle is estimated by its annual and capital cost, which include depreciation, maintenance and repairs. According to the annual use of each vehicle, which is the working days per year, and by considering that each vehicle executes a single route each day, the fixed cost is then estimated in a daily basis. Moreover, the variable cost is based on the fuel consumption of each vehicle (liters/ 100 km) and the cost of fuel (€/ liter). Representative data of vehicles used in the distribution processes of all 3PL companies are given in Table 5.9.

Table 5.9 Data of Vehicles

Type	Gross weight (tn)	Load Capacity (tn)	Fixed cost (€)	Variable cost (€/km)	CO_2 emissions (gr/tkm)
Petrol small van	1.975	0.795	14	0.101	143.4
Diesel small van	1.928	0.728	19	0.055	109.2
Large van	2.8	1.2	22	0.129	137.4
Semi-light trucks	5.5	3.0	32	0.189	107.6
Light trucks	12.5	7.5	42	0.252	86.0

As for the CO_2 emissions due to the use of vehicles, most manufacturers offer relevant information, such as the grammars of CO_2 emitted for every tone transported over a distance of one kilometer, in urban environment. Consequently, the emissions depend on the factor c^k (gr/tkm), the tare weight of the vehicle w^k, the load of goods on board on every arc (i, j), u_{ij}^k, as well as on the distance of the arc, d_{ij}. The mathematical expression that calculates CO_2 emissions is $\sum_{(i,j)\in A}\left[\left(u_{ij}^k + w^k\right)\cdot c^k \cdot d_{ij} \cdot x_{ij}^k\right]$, and is valid for every vehicle $k \in K$. This approach offers a good estimation of CO_2 emissions, even if the speed, age, and other factors affect the emissions (Koç et al. 2014).

All the data presented in Table 5.8 and Table 5.9 are essential and significant for distribution companies that address the HFVRP, the VRPSPD and the VRPTW. The specific VRP variants correspond to real-life problems and cases that most of the Greek companies operating in logistics face. Whether companies operate individually and execute single routes, or collaborate and share their resources, the needed data remains the same for optimizing the routing and scheduling problem.

5.5.5 Computational Results

Initially, multiple daily distribution cases that each 3PL company faces have been addressed and solved by the proposed heuristic algorithm. The daily schedules and results produced from each one of the cases are analyzed, offering the average cost, emissions, number of vehicles, vehicle load etc. Therefore, the results and the mean values that refer to each 3PL company individually correspond to a daily base and are indicated as 3PL C1, 3PL C2, and 3PL C3, respectively, as shown in Table 5.10. Moreover, the aggregated results of the three 3PL companies are presented in the row "Cumulative results" of Table 5.10.

In addition, the daily distribution cases of all companies are united and form more complex ones that are addressed by the algorithm as the collaborative distribution cases. The results of the multiple collaborative logistics cases, in which the depots and the customers of all companies are simultaneously considered, also offer mean values (in cost, emissions, vehicles, load factor, etc.) given in the "Collaborative case" row. Finally, the results exploited both from the collaborative case and the cumulative case are compared, while their deviations are also calculated in multiple factors.

Table 5.10 Collaborative Logistics vs Current Situation Results

Case	# of customers	Total distance	Total cost	# of vehicles	Emissions (kg of CO_2)	Load Factor
3PL C1	137	1722	500.37 €	16	443.48	90.88 %
3PL C2	385	4251	1299.86 €	33	1244.73	84.57 %
3PL C3	226	2918	812.73 €	21	793.60	88.47 %
Cumulative	748	8891	2612.96 €	70	2481.81	87.16 %
Collaborative	748	7992	2527.18 €	66	2438.36	83.38 %
Deviation	-	-10.11%	-3.28%	-5.71%	-1.75%	-4.35 %

As shown in Table 5.10, the first company (3PL C1) serves on average 137 customers, the second one (3PL C2) 385, and the third one (3PL C3) 226, in a daily base. From the results of the three cases, it is easy to say that as the number of customers increases, the distribution cost, the total traveled distance, the number of vehicles needed, and the emissions also increase. Instead, the mean load factors in all 3PL companies remain at the same level, about 75-80%. On the other hand, in the collaborative case, the average number of customers served is 748, which is equal to those in the cumulative case, while the depots from which vehicles load the goods are three.

The comparison of results obtained from the cumulative and collaborative cases indicates that the collaboration between 3PL companies can offer improvements in multiple levels. The first aspect that is greatly improved, and constitutes the main objective that 3PL companies need to minimize, is the distribution cost, in which a reduction of 3.28% is observed. In addition, the emissions are also decreased (-1.75%), reducing them by 42.45 kg daily. Even though the reduction seems small, and considering that the companies operate at least 300 days per year, annually 12.735 tons of CO2 can be avoided. This would be hard to happen without reducing the total traveled distance (-10.11%) and in the number of vehicles needed (-5.71%). Finally, the only factor that is worsening through collaboration is the load factor, and more specifically, from 87.16%, the load factor decreased to 83.38%, which means that there is more unused space on vehicles.

Additionally, there is a significant decrease in the total distribution costs, as the three companies save cumulatively 85.78 € every day, which correspond to 25,725.00€ yearly. However, the sharing of each company in the total distribution cost and, consequently, the savings differs. Multiple factors may affect the sharing of each company, such as the number of customers and the vehicles belonging to each company and involvement in the collaborative routing and scheduling procedures. Therefore, an agreement between the collaborative companies that determines the sharing of each company to the total cost is essential.

5.5.6 Discussion

As found in this study, the competition between logistics companies in Greece, as well as globally is severe. This has led companies to explore the processes that can be optimized and the technologies that need to be applied, in order to survive and operate more efficiently. The routing of vehicles is such an operation that, in most cases, is addressed by advanced systems. However, many companies have concluded that for maximizing cost savings, they need to explore new methods for more efficient operations. Collaborative routing is such an approach that, while initially was not accepted by practitioners, has now been recognized, and companies have a positive attitude towards it (Rodrigues et al. 2018).

Most collaborative routing studies focus on developing complex mathematical models and advanced optimization methods in order to minimize the distribution cost and the traveled distance. Zhang et al. (2020) propose a collaborative model similar to the one of this chapter, without considering the time windows and the environmental impact, as well as an extended variable neighborhood search algorithm. The experimental results showed that the algorithm addressing the specific collaborative case offers cost reduction that ranges from 3.19% up to 37.18% depending on the case. Similar results are obtained in Dahl and Derigs (2011) research, where the data of 50 express carriers were simulated, indicating that the cost can reduce up to 14% by sharing customer requests. Other studies also conclude that the distribution cost and the total traveled distance can be reduced, offering a higher profit margin to the companies that plan routing jointly (Pradenas et al. 2013). However, all the aforementioned studies have not

been tested in real-life cases. Moreover, over the last few years, the need for reducing the emissions has also emerged due to the global warming effect. This has led researchers to focus on this attribute and define how collaborations can enhance sustainable distribution and logistics (Pradenas et al. 2013; Perez-Bernabeu et al. 2015). In this content, Danloup et al. (2015) studied the effects of collaborative distribution on the environment. Their research focuses on food distribution, and their mathematical formulation aims to minimize the CO_2 emissions instead of the cost. The computational results indicate that the average emissions and cost reduction is 9.0 and 12.0 per cent respectively.

The presented research gives equivalent results to any of the aforementioned studies but at the same time provides a novel that collaborative routing can have great impact both on cost (25,725.00€ every year) and emissions (12.735 tons of CO2). Additionally, besides the variables and constraints that are considered in each case, as well as the sector, the results can greatly be affected by the method of calculating the cost and emissions. In the current research, it is considered that both methods are pretty accurate since all the factors affecting the cost and emissions are considered and integrated into the formulation. As for the implications to practice, while the interest in collaborative logistics increases and researchers offer novel mathematical models and algorithms, software solutions and platforms to support this concept have not followed the same path.

5.6 Concluding Remarks

In the current Chapter, multiple variants of the VRP, which have emerged through the research of Chapter 2, as well as through the experience of cooperating logistics and software development companies are considered. The aim is to consider the variants that are most frequently addressed in freight distribution. Therefore, the VRPTW, the VRPSPD, and the HVRP are selected and addressed simultaneously. In addition to these variants, the different shifts of drivers are also considered to follow real-life limitations and labor legislation that can reduce road accidents due to overworking.

Consequently, the mathematical formulation of the problem was developed to clearly define the variables and constraints of the problem in total. The mathematical formulation is exploited for developing the proposed ALNS algorithm, which offers high-efficiency results. In order to enhance this claim, the algorithm is tested in multiple datasets of the literature which focus on the studied variants. Moreover, new optimal solutions are obtained from the algorithm, while the deviation between the obtained from the algorithm and the best-published results are minimum. Concluding, the efficiency of the algorithm is sufficient for being integrated into a routing and scheduling system.

However, the algorithm is not tested only to the variants of the VRP that are most frequently addressed in freight distribution. The algorithm is also implemented, after specific modifications, in one more variant, which is the Collaborative VRP. In this case, the previous

VRP variants (VRPTW, VRPSPD, HVRP) are combined with the characteristics (variables and constraints) of the Collaborative VRP. The aim, in this case, is to define both the economic and environmental impact by the collaboration of companies that locate and operate in the same areas, in the distribution process. The implementation results are encouraging, giving great space to companies to reduce both distribution costs and emissions.

Finally, the proposed ALNS algorithm can handle large amounts of data and multiple variables and constraints. The benefits by its implementation in real-life cases can be significant, considering the efficient results obtained, compared to the best-published. Consequently, its integration into a real-life cloud system could offer great benefits to logistics and distribution companies.

References

Alcaraz, J. J., Caballero-Arnaldos, L., & Vales-Alonso, J. (2019). Rich vehicle routing problem with last-mile outsourcing decisions. Transportation Research Part E: Logistics and Transportation Review, 129, 263–286. https://doi.org/https://doi.org/10.1016/j.tre.2019.08.004

Bräysy, O., Dullaert, W., Hasle, G., Mester, D., & Gendreau, M. (2008). An Effective Multirestart Deterministic Annealing Metaheuristic for the Fleet Size and Mix Vehicle-Routing Problem with Time Windows. Transportation Science, 42(3), 371–386. https://doi.org/10.1287/trsc.1070.0217

Bräysy, O., & Hasle, G. (2014). Chapter 12: Software Tools and Emerging Technologies for Vehicle Routing and Intermodal Transportation. In P. Toth & D. Vigo (Eds.), Vehicle Routing (pp. 351–380). https://doi.org/10.1137/1.9781611973594.ch12

Bräysy, O., Porkka, P. P., Dullaert, W., Repoussis, P. P., & Tarantilis, C. D. (2009). A well-scalable metaheuristic for the fleet size and mix vehicle routing problem with time windows. Expert Systems with Applications, 36(4), 8460–8475. https://doi.org/10.1016/j.eswa.2008.10.040

Choi, E., & Tcha, D.-W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. Computers & Operations Research, 34(7), 2080–2095. https://doi.org/10.1016/j.cor.2005.08.002

Cleophas, C., Cottrill, C., Ehmke, J. F., & Tierney, K. (2019). Collaborative urban transportation: Recent advances in theory and practice. European Journal of Operational Research, 273(3), 801–816. https://doi.org/https://doi.org/10.1016/j.ejor.2018.04.037

Cruijssen, F., & Salomon, M. (2004). Empirical study: Order sharing between transportation companies may result in cost reductions between 5 to 15 percent. CentER Discussion Paper.

Dahl, S., & Derigs, U. (2011). Cooperative planning in express carrier networks — An empirical study on the effectiveness of a real-time Decision Support System. Decision Support Systems, 51(3), 620–626. https://doi.org/https://doi.org/10.1016/j.dss.2011.02.018

Danloup, N., Mirzabeiki, V., Allaoui, H., Goncalves, G., Julien, D., & Mena, C. (2015). Reducing transportation greenhouse gas emissions with collaborative distribution: A case study. Management Research Review, 38(10), 1049–1067. https://doi.org/10.1108/MRR-11-2014-0262

de Oliveira, H. C., & Vasconcelos, G. C. (2010). A hybrid search method for the vehicle routing problem with time windows. Annals of Operations Research, 180(1), 125–144. https://doi.org/10.1007/s10479-008-0487-y

Elshaer, R., & Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Computers & Industrial Engineering, 140, 106242. https://doi.org/https://doi.org/10.1016/j.cie.2019.106242

Erdoğan, G. (2017). An open source Spreadsheet Solver for Vehicle Routing Problems. Computers & Operations Research, 84, 62–72. https://doi.org/10.1016/j.cor.2017.02.022

Ferland, J. A., & Michelon, P. (1988). The Vehicle Scheduling Problem with Multiple Vehicle Types. Journal of the Operational Research Society, 39(6), 577–583. https://doi.org/10.1057/jors.1988.97

Ferrell, W., Ellis, K., Kaminsky, P., & Rainwater, C. (2020). Horizontal collaboration: opportunities for improved logistics planning. International Journal of Production Research, 58(14), 4267–4284. https://doi.org/10.1080/00207543.2019.1651457

Gansterer, M., & Hartl, R. F. (2018). Collaborative vehicle routing: A survey. European Journal of Operational Research, 268(1), 1–12. https://doi.org/https://doi.org/10.1016/j.ejor.2017.10.023

Gendreau, M., Laporte, G., Musaraganyi, C., & Taillard, É. D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. Computers & Operations Research, 26(12), 1153–1173. https://doi.org/10.1016/S0305-0548(98)00100-2

Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. Computers & Operations Research, 11(1), 49–66. https://doi.org/10.1016/0305-0548(84)90007-8

Irnich, S., Schneider, M., & Vigo, D. (2014). Chapter 9: Four Variants of the Vehicle Routing Problem. In Vehicle Routing (pp. 241–271). https://doi.org/10.1137/1.9781611973594.ch9

Jiang, J., Ng, K. M., Poh, K. L., & Teo, K. M. (2014). Vehicle routing problem with a heterogeneous fleet and time windows. Expert Systems with Applications, 41(8), 3748–3760. https://doi.org/10.1016/j.eswa.2013.11.029

Karakatič, S., & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. Applied Soft Computing, 27, 519–532. https://doi.org/10.1016/j.asoc.2014.11.005

Kechagias, E. P., Gayialis, S. P., Konstantakopoulos, G. D., & Papadopoulos, G. A. (2020a). An Application of a Multi-Criteria Approach for the Development of a Process Reference Model for Supply Chain Operations. Sustainability, 12(14), 5791. Retrieved from https://doi.org/10.3390/su12145791

Kechagias, E. P., Gayialis, S. P., Konstantakopoulos, G. D., & Papadopoulos, G. A. (2020b). An Application of an Urban Freight Transportation System for Reduced Environmental Emissions. Systems, 8(4). https://doi.org/10.3390/systems8040049

Koç, Ç., Bektaş, T., Jabali, O., & Laporte, G. (2014). The fleet size and mix pollution-routing problem. Transportation Research Part B: Methodological, 70, 239–254. https://doi.org/https://doi.org/10.1016/j.trb.2014.09.008

Koç, Ç., Bektaş, T., Jabali, O., & Laporte, G. (2015). A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. Computers & Operations Research, 64, 11–27. https://doi.org/10.1016/j.cor.2015.05.004

Konstantakopoulos, G. D., Gayialis, S. P., Kechagias, E. P., Papadopoulos, G. A., & Tatsiopoulos, I. P. (2020). A Multiobjective Large Neighborhood Search Metaheuristic for the Vehicle Routing Problem with Time Windows. Algorithms, 13(10). https://doi.org/10.3390/a13100243

Krajewska, M. A., Kopfer, H., Laporte, G., Ropke, S., & Zaccour, G. (2008). Horizontal cooperation among freight carriers: request allocation and profit sharing. Journal of the Operational Research Society, 59(11), 1483–1491. https://doi.org/10.1057/palgrave.jors.2602489

Laporte, G. (2007). What you should know about the vehicle routing problem. Naval Research Logistics (NRL), 54(8), 811–819. https://doi.org/https://doi.org/10.1002/nav.20261

Li, F., Golden, B., & Wasil, E. (2007). A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. Computers & Operations Research, 34(9), 2734–2742. https://doi.org/10.1016/j.cor.2005.10.015

Liu, F.-H., & Shen, S.-Y. (1999). The Fleet Size and Mix Vehicle Routing Problem with Time Windows. The Journal of the Operational Research Society, 50(7), 721–732. https://doi.org/10.2307/3010326

Liu, S. (2013). A hybrid population heuristic for the heterogeneous vehicle routing problems. Transportation Research Part E: Logistics and Transportation Review, 54, 67–78. https://doi.org/10.1016/j.tre.2013.03.010

Máximo, V. R., & Nascimento, M. C. V. (2021). A hybrid adaptive iterated local search with diversification control to the capacitated vehicle routing problem. European Journal of Operational Research. https://doi.org/10.1016/j.ejor.2021.02.024

Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., & Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering, 79, 115–129. https://doi.org/10.1016/j.cie.2014.10.029

Montoya-Torres, J. R., Muñoz-Villamizar, A., & Vega-Mejía, C. A. (2016). On the impact of collaborative strategies for goods delivery in city logistics. Production Planning & Control, 27(6), 443–455. https://doi.org/10.1080/09537287.2016.1147092

Özarık, S. S., Veelenturf, L. P., Woensel, T. Van, & Laporte, G. (2021). Optimizing e-commerce last-mile vehicle routing and scheduling under uncertain customer presence. Transportation Research Part E: Logistics and Transportation Review, 148, 102263. https://doi.org/10.1016/j.tre.2021.102263

Pan, S., Trentesaux, D., Ballot, E., & Huang, G. Q. (2019). Horizontal collaborative transport: survey of solutions and practical implementation issues. International Journal of Production Research, 57(15–16), 5340–5361. https://doi.org/10.1080/00207543.2019.1574040

Penna, P. H. V., Subramanian, A., Ochi, L. S., Vidal, T., & Prins, C. (2019). A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. Annals of Operations Research, 273(1), 5–74. https://doi.org/10.1007/s10479-017-2642-9

Perez-Bernabeu, E., Juan, A. A., Faulin, J., & Barrios, B. B. (2015). Horizontal cooperation in road transportation : a case illustrating savings in distances and greenhouse gas emissions. International Transactions in Operational Research, 22, 585–606. https://doi.org/10.1111/itor.12130

Pradenas, L., Oportus, B., & Parada, V. (2013). Mitigation of greenhouse gas emissions in vehicle routing problems with backhauling. Expert Systems with Applications, 40(8), 2985–2991. https://doi.org/https://doi.org/10.1016/j.eswa.2012.12.014

Qureshi, A. G., Taniguchi, E., & Yamada, T. (2009). An exact solution approach for vehicle routing and scheduling problems with soft time windows. Transportation Research Part E: Logistics and Transportation Review, 45(6), 960–977. https://doi.org/10.1016/j.tre.2009.04.007

Rodrigues, M., Zampou, E., Zeimpekis, V., Stathacopoulos, A., Teoh, T., & Ayfantopoulou, G. (2018). Cooperative Models for Addressing Urban Freight Challenges: The NOVELOG and U-TURN Approaches. In E. Taniguchi & E. Thompson (Eds.), City Logistics 3: Towards Sustainable and Liveable Cities (pp. 215–234). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781119425472.ch12

Salhi, S., & Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. Journal of the Operational Research Society, 50(10), 1034–1042. https://doi.org/10.1057/palgrave.jors.2600808

Santos, M. J., Curcio, E., Amorim, P., Carvalho, M., & Marques, A. (2021). A bilevel approach for the collaborative transportation planning problem. International Journal of Production Economics, 233, 108004. https://doi.org/https://doi.org/10.1016/j.ijpe.2020.108004

Santos, M. J., Martins, S., Amorim, P., & Almada-Lobo, B. (2021). A green lateral collaborative problem under different transportation strategies and profit allocation methods. Journal of Cleaner Production, 288, 125678. https://doi.org/https://doi.org/10.1016/j.jclepro.2020.125678

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In M. Maher & J.-F. Puget (Eds.), Principles and Practice of Constraint Programming - CP98 (pp. 417–431). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-49481-2_30

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research, 35(2), 254–265. https://doi.org/10.1287/opre.35.2.254

Soysal, M., Bloemhof-Ruwaard, J. M., Haijema, R., & van der Vorst, J. G. A. J. (2018). Modeling a green inventory routing problem for perishable products with horizontal collaboration. Computers & Operations Research, 89, 168–182. https://doi.org/https://doi.org/10.1016/j.cor.2016.02.003

Subramanian, A., Penna, P. H. V., Uchoa, E., & Ochi, L. S. (2012). A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem. European Journal of Operational Research, 221(2), 285–295. https://doi.org/https://doi.org/10.1016/j.ejor.2012.03.016

Taillard, É. D. (1999). A heuristic column generation method for the heterogeneous fleet VRP. RAIRO-Operations Research-Recherche Opérationnelle, 33(1), 1–14. https://doi.org/10.1051/ro:1999101

Tarantilis, C. D., Kiranoudis, C. T., & Vassiliadis, V. S. (2004). A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research, 152(1), 148–158. https://doi.org/10.1016/S0377-2217(02)00669-0

Verdonck, L., Caris, A. N., Ramaekers, K., & Janssens, G. K. (2013). Collaborative Logistics from the Perspective of Road Transportation Companies. Transport Reviews, 33(6), 700–719. https://doi.org/10.1080/01441647.2013.853706

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research, 234(3), 658–673. https://doi.org/10.1016/j.ejor.2013.09.045

Wang, Y., Zhang, J., Assogba, K., Liu, Y., Xu, M., & Wang, Y. (2018). Collaboration and transportation resource sharing in multiple centers vehicle routing optimization with delivery and pickup. Knowledge-Based Systems, 160, 296–310. https://doi.org/https://doi.org/10.1016/j.knosys.2018.07.024

Wassan, N. A., & Osman, I. H. (2002). Tabu search variants for the mix fleet vehicle routing problem. Journal of the Operational Research Society, 53(7), 768–782. https://doi.org/10.1057/palgrave.jors.2601344

Xiao, Y., & Konak, A. (2016). The heterogeneous green vehicle routing and scheduling problem with time-varying traffic congestion. Transportation Research Part E: Logistics and Transportation Review, 88, 146–166. https://doi.org/https://doi.org/10.1016/j.tre.2016.01.011

Zhang, W., Chen, Z., Zhang, S., Wang, W., Yang, S., & Cai, Y. (2020). Composite multi-objective optimization on a new collaborative vehicle routing problem with shared carriers and depots. Journal of Cleaner Production, 274, 122593. https://doi.org/https://doi.org/10.1016/j.jclepro.2020.122593

Chapter 6: Practical Application of the Developed Algorithms into the Cloud Routing System

6.1 Introduction

Since the VRP was first introduced, researchers and practitioners focused on its solution. As was expected, due to the limited methods and the technological limitations in the 1960s, the solutions focused on non-computational methods, and more precisely, on the designing of routes on maps.

However, over the years, the advances in technology and optimization methods helped researchers handle multiple variants of the VRP and develop advanced algorithms to address various distribution cases. Although the optimization methods proposed by researchers are tested in datasets of the literature, indicating high efficiency, they are rarely integrated into information systems or software. This is because researchers focus on developing new algorithms and mathematical formulation that can enhance the theoretical part of the study and a limited number of VRP variants. In addition, software companies have the expertise and the means to develop advanced algorithms that, after testing and adjustments, can cover the needs of each logistics company separately and consequently can be integrated in routing software solutions.

While the optimization algorithm addressing the routing operations constitutes the core of the cloud routing system, it cannot ensure the functioning of the system. This is ensured by following specific steps during its development. Firstly, the requirements of the potential users i.e. logistics and distribution companies are defined and are consequently transformed into variables and constraints. Consequently, the technologies that can be used for developing a cloud system (such as the platform, the database, programming languages for system development and the Front-End interface), as well as those that can be exploited for extracting reliable data for the routing, are choose according to their offerings, potential and acquisition costs. Finally, after the combination of all the above attributes of the system, the testing of the system and of its functionalities is a step that is executed continuously until the final version of the system is created.

As a result, the development of a system is a complex procedure that has multiple steps. However, since the core of the system is the developed optimization algorithm addressing the VRP, software companies treat them as black-boxes. Therefore, algorithms that have been successfully implemented in real-life systems are rarely published in research papers. However, this tendency was overcome in this thesis, as the algorithms proposed in Chapter 3, Chapter 4 and Chapter 5 are stored in the software development company's server and can be accessed anytime. However, in its current form, the cloud routing system uses the algorithm library of the ALNS algorithm that is presented in Chapter 5, and consequently receives its outputs. Moreover, since the MOLNS and MOEA algorithms are also stored in the software

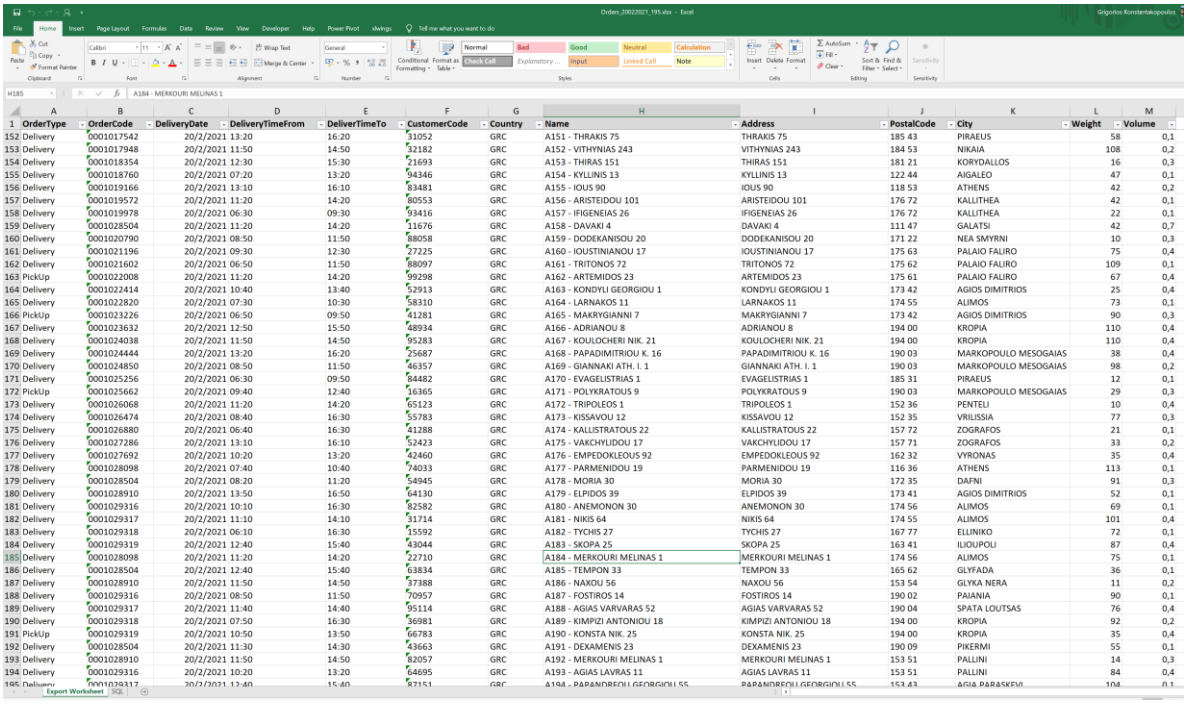
development company's server as libraries, both algorithms can be accessed any time by the cloud system. In the future the user is expected to be able to choose which of the algorithms he wants to use. The decision to connect in this phase only the ALNS algorithm is based on the fact that handles more VRP variants compared to the MOEA and MOLNS algorithms.

Moreover, the cloud routing system is provided as a Software as a Service (SaaS), which means that companies can avoid high upfront purchase costs and ongoing maintenance and upgrades costs since it is available to logistics companies over the internet (Arunachalam et al. 2018). The system also utilizes real-time traffic data and geolocation that enhance efficiency and accuracy (Seongmoon et al. 2005). Finally, since the ALNS algorithm has been tested in datasets of the literature in Chapter 5, its testing continues on real-life distribution cases of logistics companies operating in Athens, Greece.

This thesis combines the parts that both research and industry focus on and presents both the developed algorithms and their integration into the cloud system. In the remainder of the Chapter, the system is presented. More specifically, the master data needed to be fed to the system in order to be functional, are presented in section 6.2. The technologies that are exploited by the system and the system's functionalities are presented in section 6.3. Additionally, the initial routing procedure along with the necessary data are presented in section 6.4, while in section 6.5 the rerouting procedure is described. Finally, the system's evaluation and benefits are presented in section 6.6, while the benefits are given in section 6.7.

6.2 Master Data

The system, in order to be efficient, needs first to be fed with all the necessary data. Consequently, the algorithm will be exploited for creating the plan of routes. Initially, an excel file needs to be uploaded, including the customers' data. The excel file needs to have a specific structure, as shown in Figure 6.1. First of all, in column "A" the order type needs to be defined, which is either a delivery or a pickup. An order code is given in column "B", while the delivery date, as well as the start and the end of the time windows, are given in columns "C', "D" and "E" respectively. Column "F" includes the customer's code, while columns "G", "H", "I", "J" and "K" include data related to the address of each customer, that is very important for executing the geocoding procedure and extracting travel times and distances between order points. Finally, columns "L" and "M" include the weight (Kg) and the volume (m^3) of each order.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	OrderType	OrderCode	DeliveryDate	DeliveryTimeFrom	DeliverTimeTo	CustomerCode	Country	Name	Address	PostalCode	City	Weight	Volume
152	Delivery	0001017542	20/2/2021	13:20	16:20	31052	GRC	A151 - THRAKIS 75	THRAKIS 75	185 43	PIRAEUS	58	0,1
153	Delivery	0001017948	20/2/2021	11:50	14:50	32182	GRC	A152 - VITHYNIAS 243	VITHYNIAS 243	184 53	NIKAIA	108	0,2
154	Delivery	0001018354	20/2/2021	12:30	15:30	21693	GRC	A153 - THIRAS 151	THIRAS 151	181 21	KORYDALLOS	16	0,3
155	Delivery	0001018760	20/2/2021	07:20	13:20	94346	GRC	A154 - KYLLINIS 13	KYLLINIS 13	122 44	AIGALEO	47	0,1
156	Delivery	0001019166	20/2/2021	13:10	16:10	83481	GRC	A155 - IOUS 90	IOUS 90	118 53	ATHENS	42	0,2
157	Delivery	0001019572	20/2/2021	11:20	14:20	80553	GRC	A156 - ARISTEIDOU 101	ARISTEIDOU 101	176 72	KALLITHEA	42	0,1
158	Delivery	0001019978	20/2/2021	06:30	09:30	93416	GRC	A157 - IFIGENEIAS 26	IFIGENEIAS 26	176 72	KALLITHEA	22	0,1
159	Delivery	0001028504	20/2/2021	11:20	14:20	11676	GRC	A158 - DAVAKI 4	DAVAKI 4	111 47	GALATSI	42	0,7
160	Delivery	0001020790	20/2/2021	08:50	11:50	88058	GRC	A159 - DODEKANISOU 20	DODEKANISOU 20	171 22	NEA SMYRNI	10	0,3
161	Delivery	0001021196	20/2/2021	09:30	12:30	27225	GRC	A160 - IOUSTINIANOU 17	IOUSTINIANOU 17	175 63	PALAIO FALIRO	75	0,4
162	Delivery	0001021602	20/2/2021	06:50	11:50	88097	GRC	A161 - TRITONOS 72	TRITONOS 72	175 62	PALAIO FALIRO	109	0,1
163	PickUp	0001022008	20/2/2021	11:20	14:20	99298	GRC	A162 - ARTEMIDOS 23	ARTEMIDOS 23	175 61	PALAIO FALIRO	67	0,4
164	Delivery	0001022414	20/2/2021	10:40	13:40	52913	GRC	A163 - KONDYLI GEORGIOU 1	KONDYLI GEORGIOU 1	173 42	AGIOS DIMITRIOS	25	0,4
165	Delivery	0001022820	20/2/2021	07:30	10:30	58310	GRC	A164 - LARNAKOS 11	LARNAKOS 11	174 55	ALIMOS	73	0,1
166	PickUp	0001023226	20/2/2021	06:50	09:50	41281	GRC	A165 - MAKRYGIANNI 7	MAKRYGIANNI 7	173 42	AGIOS DIMITRIOS	90	0,3
167	Delivery	0001023632	20/2/2021	12:50	15:50	48934	GRC	A166 - ADRIANOU 8	ADRIANOU 8	194 00	KROPIA	110	0,4
168	Delivery	0001024038	20/2/2021	11:50	14:50	95283	GRC	A167 - KOULOCHERI NIK. 21	KOULOCHERI NIK. 21	194 00	KROPIA	110	0,4
169	Delivery	0001024444	20/2/2021	13:20	16:20	25687	GRC	A168 - PAPADIMITRIOU K. 16	PAPADIMITRIOU K. 16	190 03	MARKOPOULO MESOGAIAS	38	0,4
170	Delivery	0001024850	20/2/2021	08:50	11:50	46357	GRC	A169 - GIANNAKI ATH. I. 1	GIANNAKI ATH. I. 1	190 03	MARKOPOULO MESOGAIAS	98	0,2
171	Delivery	0001025256	20/2/2021	06:30	09:50	84482	GRC	A170 - EVAGELISTRIAS 1	EVAGELISTRIAS 1	185 31	PIRAEUS	12	0,1
172	PickUp	0001025662	20/2/2021	09:40	12:40	16365	GRC	A171 - POLYKRATOUS 9	POLYKRATOUS 9	190 03	MARKOPOULO MESOGAIAS	29	0,3
173	Delivery	0001026068	20/2/2021	11:20	14:20	65123	GRC	A172 - TRIPOLEOS 1	TRIPOLEOS 1	152 36	PENTELI	10	0,4
174	Delivery	0001026474	20/2/2021	08:40	16:30	55783	GRC	A173 - KISSAVOU 12	KISSAVOU 12	152 35	VRILISSIA	77	0,3
175	Delivery	0001026880	20/2/2021	06:40	16:30	41288	GRC	A174 - KALLISTRATOUS 22	KALLISTRATOUS 22	157 72	ZOGRAFOS	21	0,1
176	Delivery	0001027286	20/2/2021	13:10	16:10	52423	GRC	A175 - VAKCHYLIDOU 17	VAKCHYLIDOU 17	157 71	ZOGRAFOS	33	0,2
177	Delivery	0001027692	20/2/2021	10:20	13:20	42460	GRC	A176 - EMPEDOKLEOUS 92	EMPEDOKLEOUS 92	162 32	VYRONAS	35	0,4
178	Delivery	0001028098	20/2/2021	07:40	10:40	74033	GRC	A177 - PARMENIDOU 19	PARMENIDOU 19	116 36	ATHENS	113	0,1
179	Delivery	0001028504	20/2/2021	08:20	11:20	54945	GRC	A178 - MORIA 30	MORIA 30	172 35	DAFNI	91	0,3
180	Delivery	0001028910	20/2/2021	13:50	16:50	64130	GRC	A179 - ELPIDOS 39	ELPIDOS 39	173 41	AGIOS DIMITRIOS	52	0,1
181	Delivery	0001029316	20/2/2021	10:10	16:30	82582	GRC	A180 - ANEMONON 30	ANEMONON 30	174 56	ALIMOS	69	0,1
182	Delivery	0001029317	20/2/2021	11:10	14:10	31714	GRC	A181 - NIKIS 64	NIKIS 64	174 55	ALIMOS	101	0,4
183	Delivery	0001029318	20/2/2021	06:10	16:30	15592	GRC	A182 - TYCHIS 27	TYCHIS 27	167 77	ELLINIKO	72	0,1
184	Delivery	0001029319	20/2/2021	12:40	15:40	43044	GRC	A183 - SKOPA 25	SKOPA 25	163 41	ILIOUPOLI	87	0,4
185	Delivery	0001028098	20/2/2021	11:20	14:20	22710	GRC	A184 - MERKOURI MELINAS 1	MERKOURI MELINAS 1	174 56	ALIMOS	75	0,1
186	Delivery	0001028504	20/2/2021	12:40	15:40	63834	GRC	A185 - TEMPON 33	TEMPON 33	165 62	GLYFADA	36	0,1
187	Delivery	0001028910	20/2/2021	11:50	14:50	37388	GRC	A186 - NAXOU 56	NAXOU 56	153 54	GLYKA NERA	11	0,2
188	Delivery	0001029316	20/2/2021	08:50	11:50	70957	GRC	A187 - FOSTIROS 14	FOSTIROS 14	190 02	PAIANIA	90	0,1
189	Delivery	0001029317	20/2/2021	11:40	14:40	95114	GRC	A188 - AGIAS VARVARAS 52	AGIAS VARVARAS 52	190 04	SPATA LOUTSAS	76	0,4
190	Delivery	0001029318	20/2/2021	07:50	16:30	36981	GRC	A189 - KIMPIZI ANTONIOU 18	KIMPIZI ANTONIOU 18	194 00	KROPIA	92	0,2
191	PickUp	0001029319	20/2/2021	10:50	13:50	66783	GRC	A190 - KONSTA NIK. 25	KONSTA NIK. 25	194 00	KROPIA	35	0,4
192	Delivery	0001028504	20/2/2021	11:30	14:30	43663	GRC	A191 - DEXAMENIS 23	DEXAMENIS 23	190 09	PIKERMI	55	0,1
193	Delivery	0001028910	20/2/2021	11:50	14:50	82057	GRC	A192 - MERKOURI MELINAS 1	MERKOURI MELINAS 1	153 51	PALLINI	14	0,3
194	Delivery	0001029316	20/2/2021	10:20	13:20	64695	GRC	A193 - AGIAS LAVRAS 11	AGIAS LAVRAS 11	153 51	PALLINI	84	0,4

Figure 6.1 Structure of Orders Data

In the starting page of the cloud system, the structure and the specific order that the data need to follow are well defined, as shown in Figure 6.2. Additionally, each user can browse for the right file and upload it, as shown in Figure 6.2.

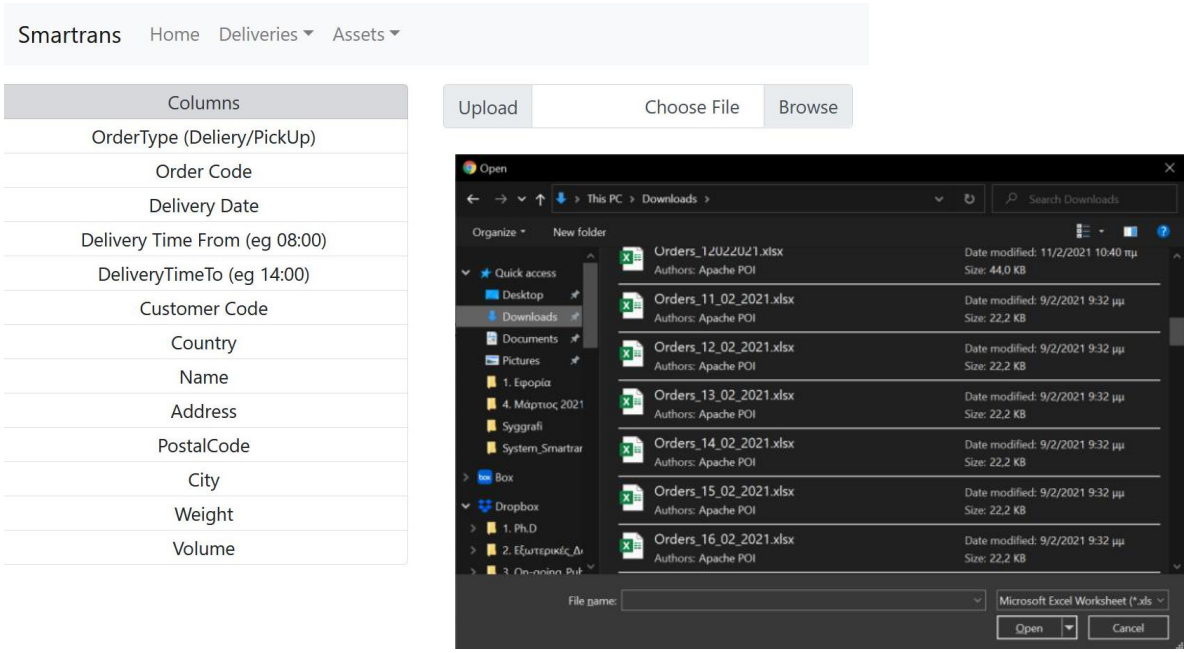


Figure 6.2 Uploading of Data

Additionally, logistics and distribution companies need to define the data related to their assets. More specifically, the address of the central depot, the available vehicles that execute the routes, their capacities (both in kg and m^3), fixed and variable costs, the available drivers and their related information (email and name), and the shifts of drivers. Most of these data

constitute either variables or constraints of the problem and need to be respected. As shown in Figure 6.3, the user needs to define the available types of vehicles, including all the necessary data and every single vehicle, by assigning it to a single vehicle type.

The describes information and data are necessary since they are used as parameters of the problem and for extracting other data that are essential for the routing procedure. As a result, the algorithm cannot operate and offer a plan of routes without these data. In section 6.3, the way customer orders are exploited for extracting the distances and transit times between customers is thoroughly analyzed, and the necessity of all data.

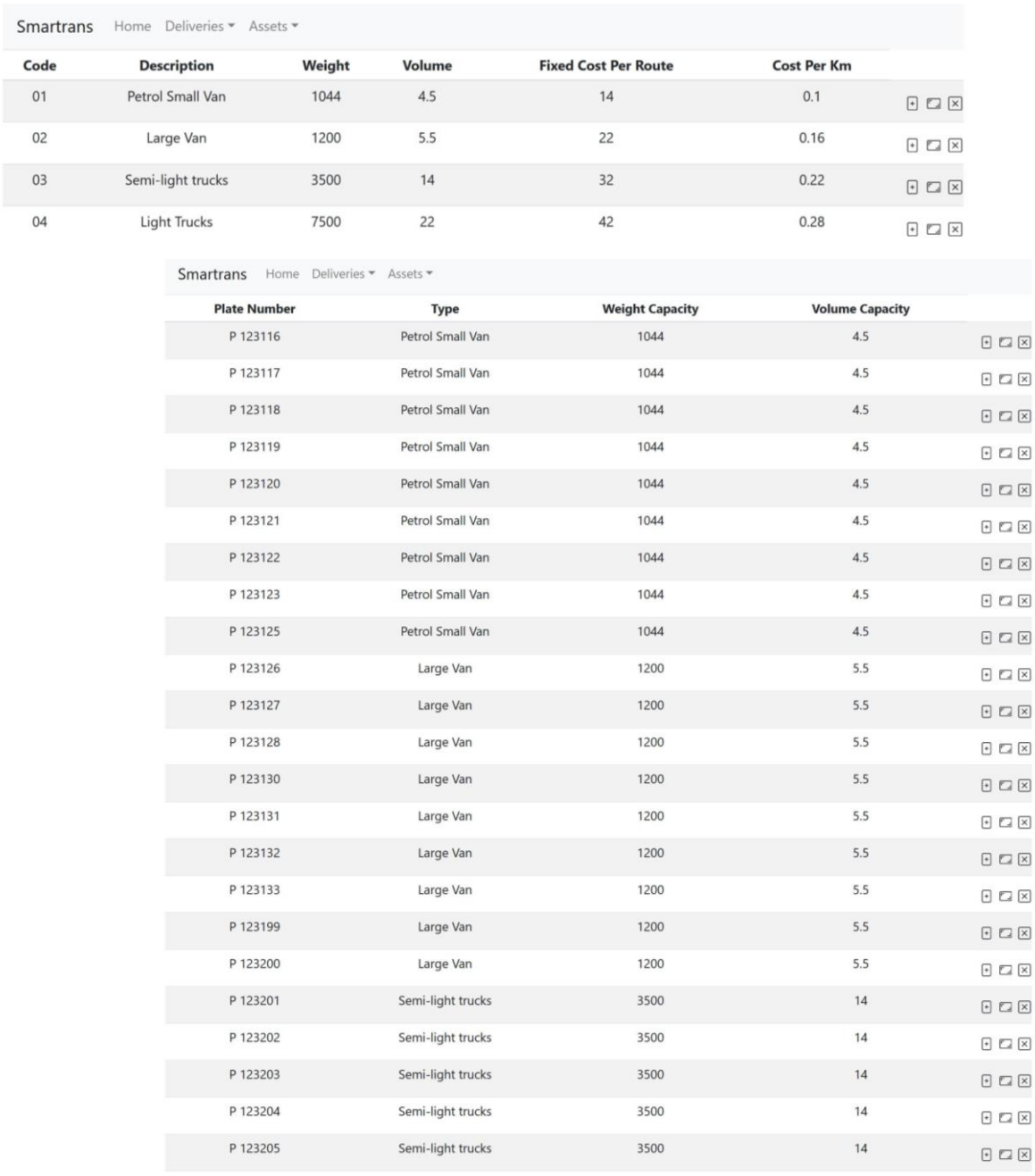


Figure 6.3 Vehicles and Vehicle Types Data

6.3 System's Technologies and Functionalities

The research made a major step further as the developed ALNS algorithm is implemented into a cloud routing system, ready to be used by logistics companies to facilitate their distribution operations. While operational research is highly related to logistics, and many methods have been proposed for addressing the vehicle routing problem, only a few are integrated into real-life systems. This is because the development of a routing system is inextricably linked with software development and related technologies such as cloud computing, database management, Application Programming Interfaces (API's), and webservices. In addition, after finding efficient results for the optimization method in datasets of the literature, it was decided that the algorithm should also implement in a cloud routing system for it to be applicable in real-life distribution cases, where a large amount of data is dynamically managed.

In order to develop a fully operatable system, multiple technologies and services need to be combined and work simultaneously. Firstly, since the provider's created software solution can be given to logistics companies/customers, using the Software as a Service (SaaS) distribution model, it needs to be hosted in a cloud computing platform. In the presented case, the clouding platform is Microsoft Azure, as it can build, run, manage and support large and complex applications that solve today's challenges. Moreover, Azure implemented many industry-required security standards and regulatory compliance requirements that made it prevail to other cloud platforms. Consequently, companies ensure their security and eliminate the cost related to the maintenance of the application's infrastructure, like servers and databases.

Moreover, companies that gain access to the system's services need to upload the master data firstly. The master data in the specific case are the warehouse locations, the orders of the customers (delivery points, delivery times, time windows, delivery quantities etc.), and the available fleet data (drivers, vehicles, capacity), that can be entered by uploading a datasheet, with a particular structure, or manually using the system. These data need to be stored, structured, and managed by the proper database. Oracle Database is selected in the cloud system due to the additional data management capabilities such as deep monitoring, data modeling tools, data visualization, and, more importantly, low code development environments. These additional features, along with the ability of Azure to access Oracle database efficiently, made us select the specific database.

Additionally, the system uses the webservices offered by HERE Global B. V. company through the Azure Representation State Transfer Application Programming Interface (REST API). The REST API conforms to the constraints of REST architectural style and allows the interaction with RESTful services, such as the ones offered by HERE. The system calls through Postman the Azure REST APIs to request and obtain data from HERE services. Specifically, HTTP requests include the necessary information such as the URL, the id, the request method, and the API key. The response from HERE includes a JSON file with the data that are requested.

Such requests are made to HERE Geocoding & Search service to extract customers' coordinates through their addresses.

Furthermore, the coordinates are also used as input for requesting the transit times and distances between delivery points. Therefore, the HTTP requests are significant for the efficient operation of the system. Consequently, some of the data that have been inserted by the users, along with those retrieved by the HTTP requests, are given as input in the ALNS algorithm. The algorithm that is also stored in the cloud has been developed in Python programming language and is accessed once again through an HTTP request. Finally, the results of the routing procedure, which are specific routes with a specific order sequence, are depicted in an online map, provided once again by HERE.

All the above-mentioned technologies and their interactions are presented in Figure 6.4. More specifically, the cloud computing platform can send and receive HTTP requests and responses to and from HERE company, Oracle cloud database and user's front-end interface. The front-end interface gives access to logistics and distribution companies, as well as truck drivers. The back-end is managed by the software provider who provides the cloud computing platform as a SaaS.

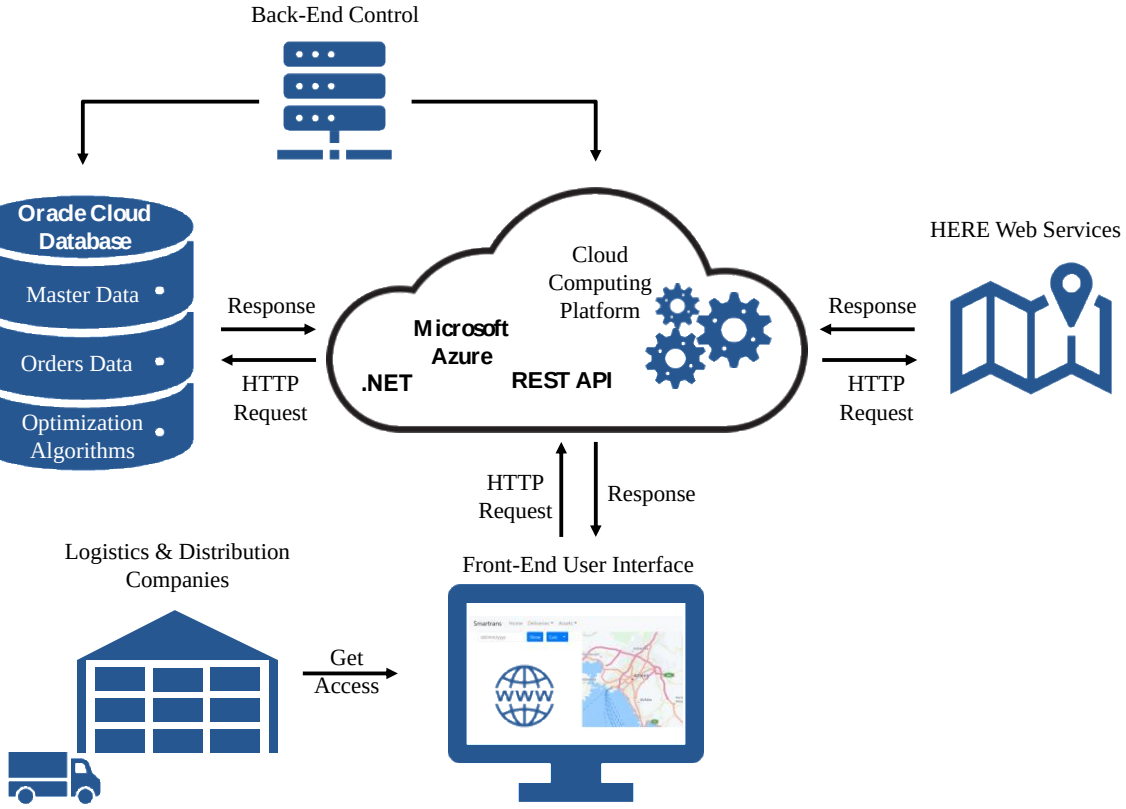


Figure 6.4 System's Technologies

A more analytical description of the data given as input in the HTTP requests and its response is given in the following sections. In Section 6.4, the technologies that are exploited for the initial routing (static) and the functionalities of the system are presented. On the other hand,

the technologies and functionalities of the system during the rerouting and dynamic routing procedures are presented in Section 6.5.

6.4 Initial Routing and Scheduling

For the initial routing, in which data related to customers are known before the distribution starts, transit times and distances between order points and warehouses need to be obtained. Therefore, an HTTP request that follows the RESTful API architectural style and includes the addresses of customers is sent to HERE Geocoding & Search service. The response from HERE includes a JSON file with the coordinates of customers. In cases where the address is not clearly stated and more than one set of coordinates matches the request, HERE gives the ability to manually select the on-map address that better fits the customer, as shown in Figure 6.5.

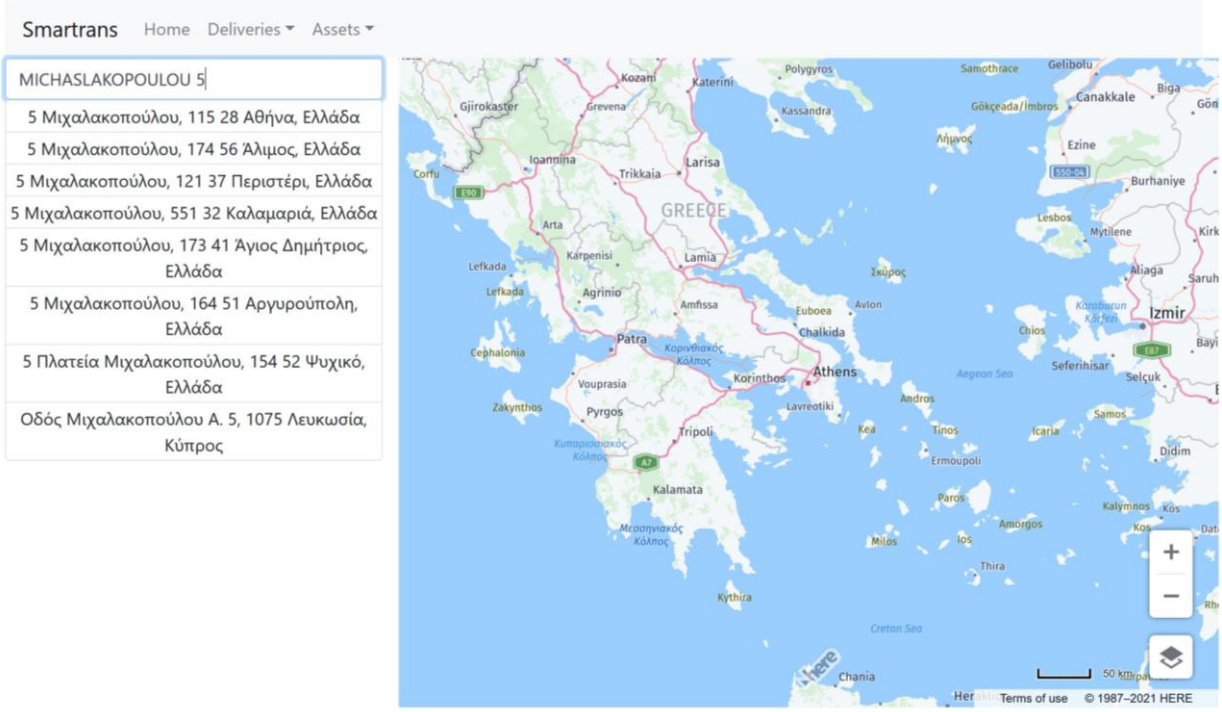


Figure 6.5 Autocomplete Address Functionality

After completing the first phase, which includes the data input and geocoding procedures, the system can visualize the delivery points on the map, as shown in Figure 6.6. Additionally, the system can then execute the routing and scheduling procedure. This requires, firstly, an estimation of the travel times between delivery points that are significant for accurate deliveries and high-quality services and the distances between delivery points used for calculating the distribution cost. These data are exploited once again through HERE Technologies and more accurately through an HTTP request (HERE Matrix Routing API v8), including the coordinates of delivery points. The JSON file that returns as a response to the request includes the distance (M x M) and the travel time (M x M) matrices, where M is the number of delivery points and depots.

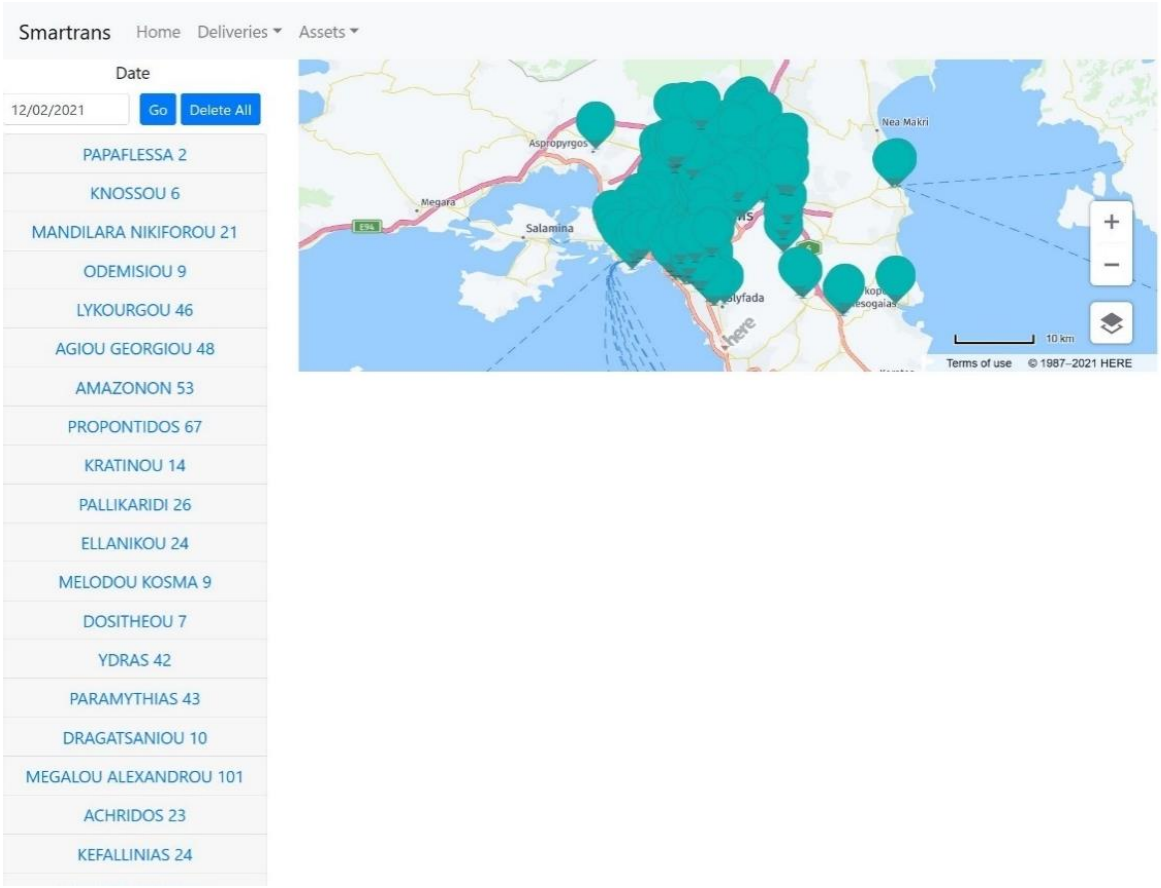


Figure 6.6 Order Point On-map

Moreover, these matrices, along with data such as the delivery and pickup quantities (in kg and in m^3), time windows, vehicle capacities and costs etc., are fed to the routing and scheduling algorithm. Another request, including the aforementioned data, is made to get the plan of routes and schedule of deliveries through the ALNS algorithm stored in the software development company's server. The developed ALNS algorithm is given in Appendix E. The only parameter that the user needs to define is the calculation time of the routing procedure, which affects the accuracy of the system's results as they are strongly time-dependent. The field which the user needs to fill for defining the computation time is shown in Figure 6.7.

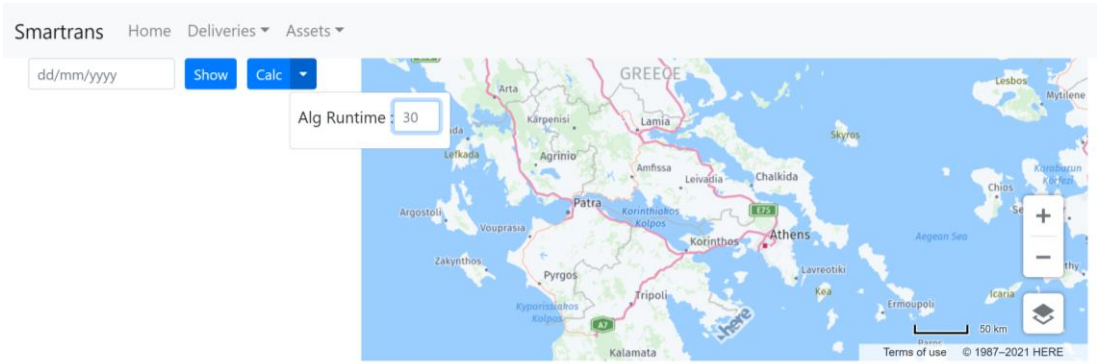


Figure 6.7 Defining the Computation Time

Defining the computation time is important since each company faces distribution cases of different sizes (ranging in most cases from 100 customers to more than 2000 per day). For example, in cases with around 2000 delivery points, a computation time of 30 minutes would be sufficient for the algorithm to produce an efficient route plan. Shorter computation times, i.e., less than 10 minutes, would lead to less efficient plans with higher delivery costs and lower vehicle fleet utilization. The above references are reinforced by Figures 6.8, 6.9, 6.10 and Tables 6.1, 6.2., 6.3 which validate the efficiency of the algorithm in terms of quality and computation time.

Figure 6.8 and Table 6.1 present the algorithm results when tested in a store replenishment distribution case, including 141 orders. The case is solved four times (4 runs) in which the time limit is set to 3600 seconds (1 hour). The ALNS algorithm is based on the destroy and repair operators that in turn base on randomness. That is why the final results of each run in Table 6.1 are different. Besides that, they indicate that 600 to 900 seconds would be enough time to produce an efficient delivery schedule while also maintaining the computation time at low levels.

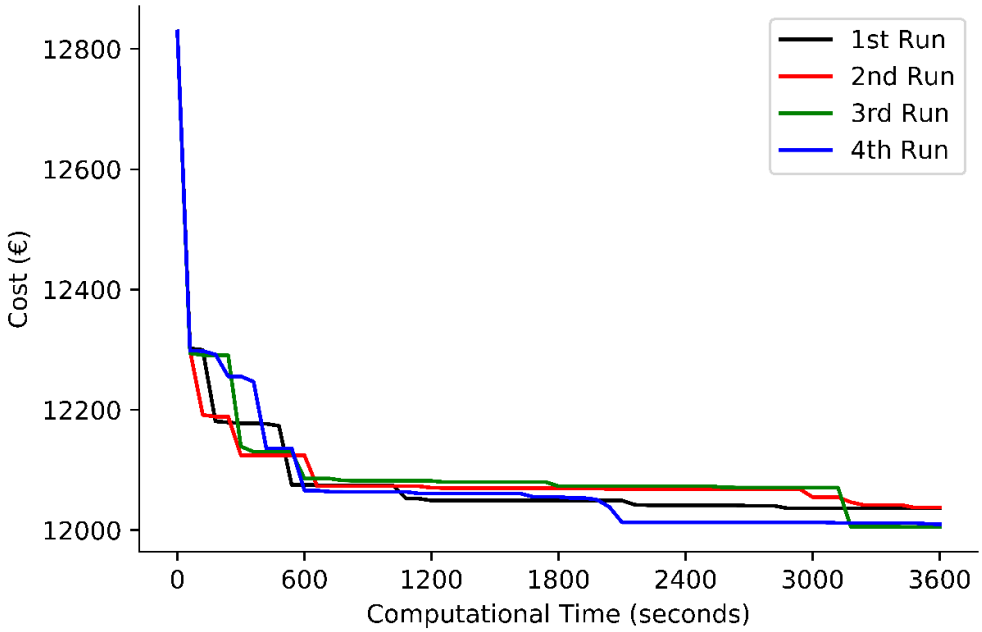


Figure 6.8 Distribution Cost per Running Time in a Store Replenishment Case

Table 6.1 Distribution Cost per Running Time in a Store Replenishment Case

Phase of the Algorithm	Time (sec)	Cost – 1st Run	Cost – 2nd Run	Cost – 3rd Run	Cost – 4th Run
Construction	1	12829.18	12829.18	12829.18	12829.18
ALNS	60	12302.14	12297.97	12294.06	12298.28
	180	12180.88	12188.67	12290.79	12291.99
	300	12177.47	12124.36	12139.09	12255.62
	600	12074.91	12124.36	12085.76	12065.87
	900	12073.71	12073.05	12082.3	12063.66
	1200	12049.19	12070.43	12081.8	12060.9
	1500	12049.19	12069.61	12079.86	12060.9
	1800	12049.19	12069.61	12073.08	12055.3
	2400	12041.46	12068.08	12073.08	12012.89
	3000	12036.38	12054.85	12070.72	12012.89
	3600	12036.16	12037.56	12005.24	12009.84

Additionally, the algorithm is tested in more distribution cases (with more orders), as shown in Figure 6.9 and Figure 6.10, accompanied by Table 6.2 and Table 6.3, respectively. The results presented in Figure 6.9 and Table 6.2 come from a door-to-door delivery case, including 195 orders. The results indicate that 900 seconds (15 minutes) are enough for the planner to extract an efficient plan. Respectively, Figure 6.10 and Table 6.3 present the results from an e-commerce case including 748 orders. With an increased number of orders, the computation time needed in such a case is also higher, and therefore 1800 seconds (30 minutes) seem necessary for producing a low-cost schedule.

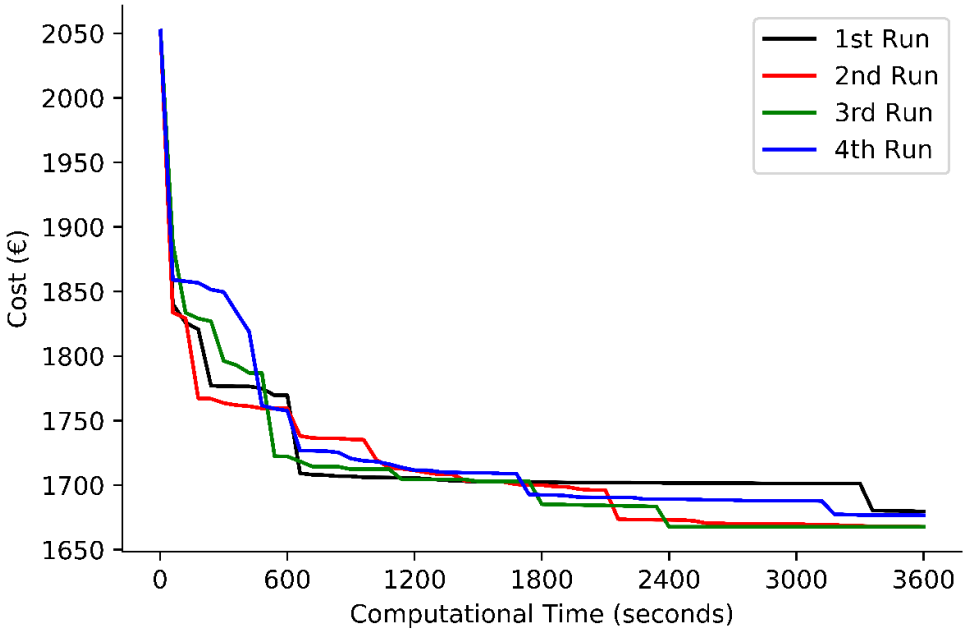


Figure 6.9 Distribution Cost per Running Time in a Door-to-Door Delivery Case

Table 6.2 Distribution Cost per Running Time in a Door-to-Door Delivery Case

Phase of the Algorithm	Time (sec)	Cost – 1st Run	Cost – 2nd Run	Cost – 3rd Run	Cost – 4th Run
Construction	2	2051.96	2051.96	2051.96	2051.96
ALNS	60	1840.26	1833.81	1887.31	1859.06
	180	1820.69	1767.25	1829.07	1856.71
	300	1776.57	1763.59	1796.23	1849.61
	600	1769.54	1759.45	1722.26	1757.62
	900	1706.89	1735.49	1712.30	1720.64
	1200	1705.62	1711.42	1704.36	1711.58
	1500	1702.91	1703.28	1702.87	1709.61
	1800	1702.10	1700.08	1685.25	1692.43
	2400	1701.63	1673.16	1667.70	1689.15
	3000	1701.31	1669.98	1667.70	1687.91
	3600	1679.85	1667.83	1667.70	1676.71

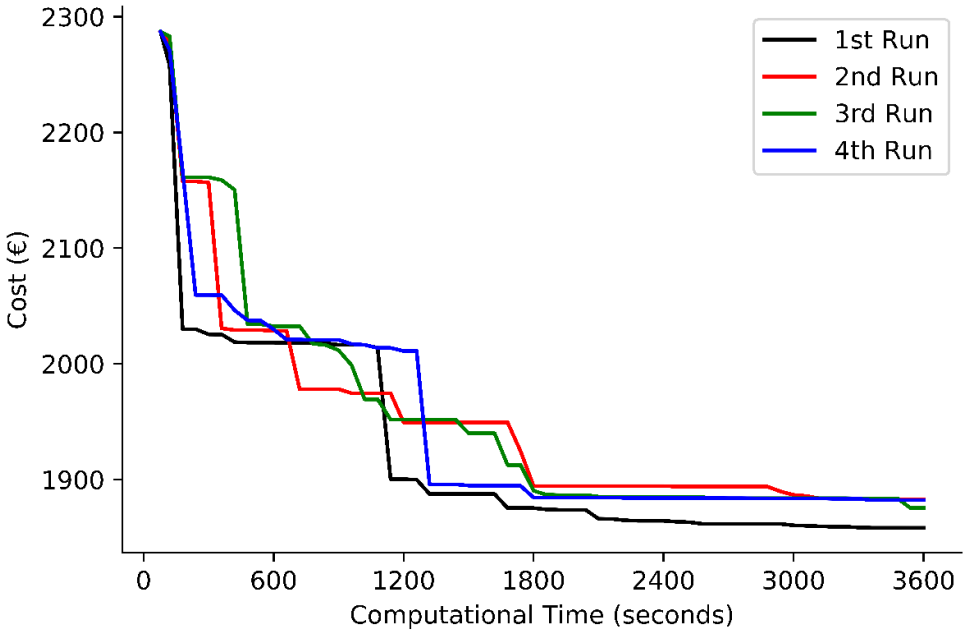


Figure 6.10 Distribution Cost per Running Time in an E-commerce Case

Table 6.3 Distribution Cost per Running Time in an E-commerce Case

Phase of the Algorithm	Time (sec)	Cost – 1st Run	Cost – 2nd Run	Cost – 3rd Run	Cost – 4th Run
Construction	78	2287.02	2287.02	2287.02	2287.02
ALNS	120	2257.88	2278.78	2282.94	2271.01
	180	2029.97	2157.65	2161.21	2167.08
	300	2025.46	2156.62	2161.18	2059.53
	600	2018.37	2028.4	2032.55	2029.79
	900	2016.37	1978.09	2011.7	2020.69
	1200	1900.25	1949.37	1951.84	2011.07
	1500	1887.4	1949.37	1939.91	1894.81
	1800	1875.31	1894.4	1890.3	1884.42
	2400	1864.34	1894.4	1884.76	1884.15
	3000	1860.39	1886.71	1883.8	1883.64
	3600	1858.23	1882.8	1875.3	1882.13

As a result of these computations, the system can now display the plan of routes and schedule of deliveries that has been produced, as shown in Figure 6.11.

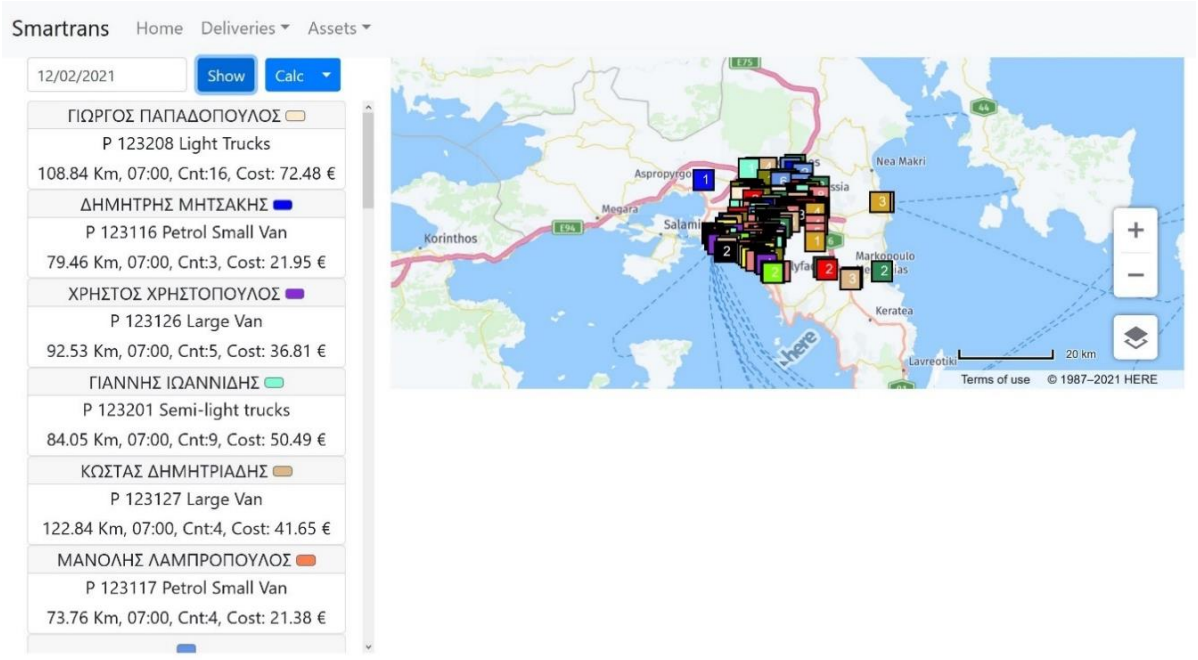


Figure 6.11 Created Plan of Routes

Each route includes specific order points, and their sequence defines the total cost and the traveled distance. The total distribution cost comes from the fixed and variable costs of each route. The variable cost in the presented case is based on the traveled distance of each vehicle (in km) and the cost of fuel per km (€/ km). Consequently, the sequence of deliveries affects the variable cost, since the distance of the route changes. By selecting each route, the user can

isolate the route and analytically view the various data related to this route. More specifically, the estimated time of arrival in each customer/ delivery point, the quantity delivered or picked, its time window, as well as the status of the order can be observed in Figure 6.12.

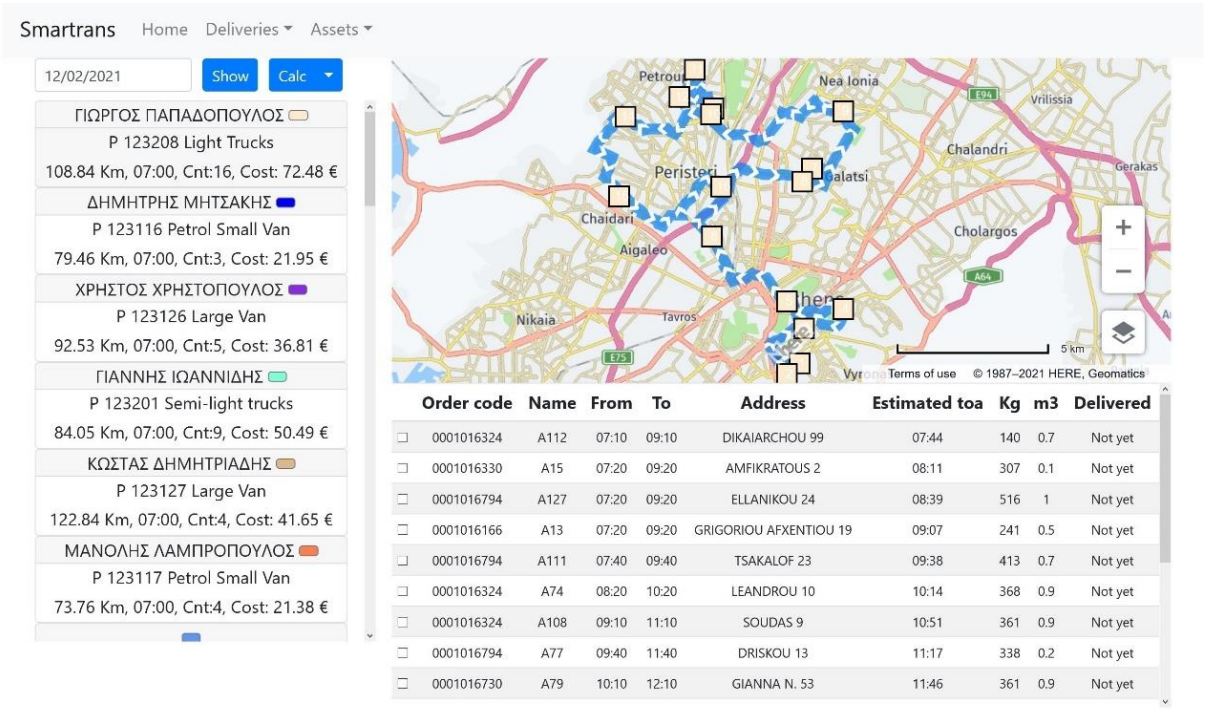


Figure 6.12 Route Representation

Additionally, the system connects to a Proof of Delivery (PoD) system, located on the drivers' vehicles, and enables the detection of the status of the orders in every step of the distribution process. As a result, the driver has the ability to take certain actions upon completion of serving an order through the SMARTRANS PoD application. The PoD is also accessible anywhere, as it can be used by any device connected to the internet, as long as the driver enters the link https://aberon-pod.azurewebsites.net into an internet browser. The driver also needs to enter his official email in the PoD, as shown in Figure 6.13, to see the schedule assigned to, as shown in Figure 6.14.

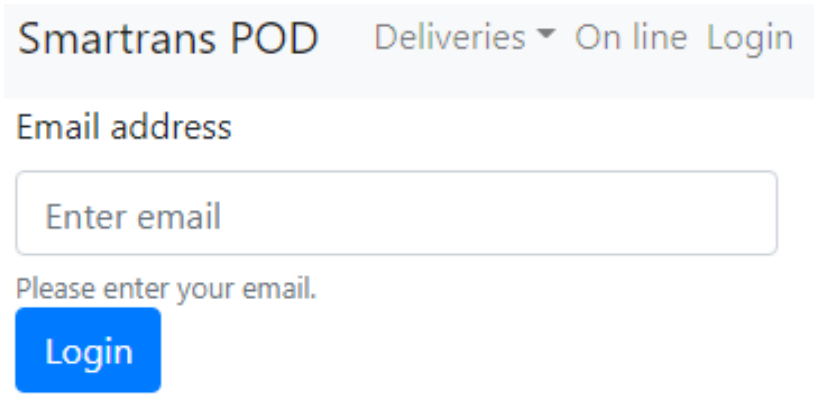


Figure 6.13 Entering the PoD System

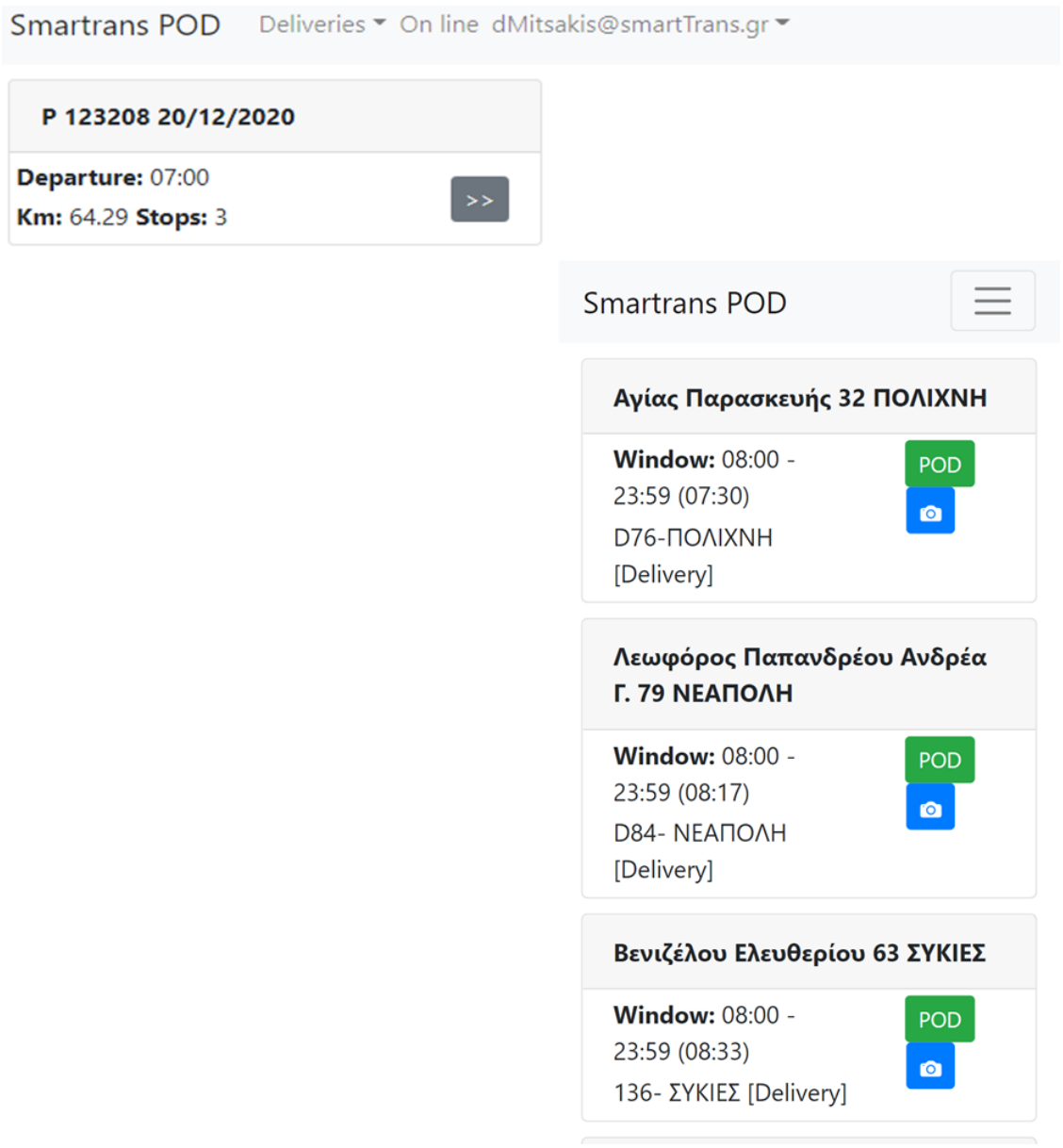


Figure 6.14 Driver's Route

Additionally, the PoD sytem enables and enhances the updating of order status during the distribution process. Moreover, after the completion of an order, the driver can update the status of the order which is also updated automatically in the routing system. In Figure 6.15 are presented both the PoD and the routing system, as well as how by selecting the PoD button, the status of the order is also updated in the routing system.

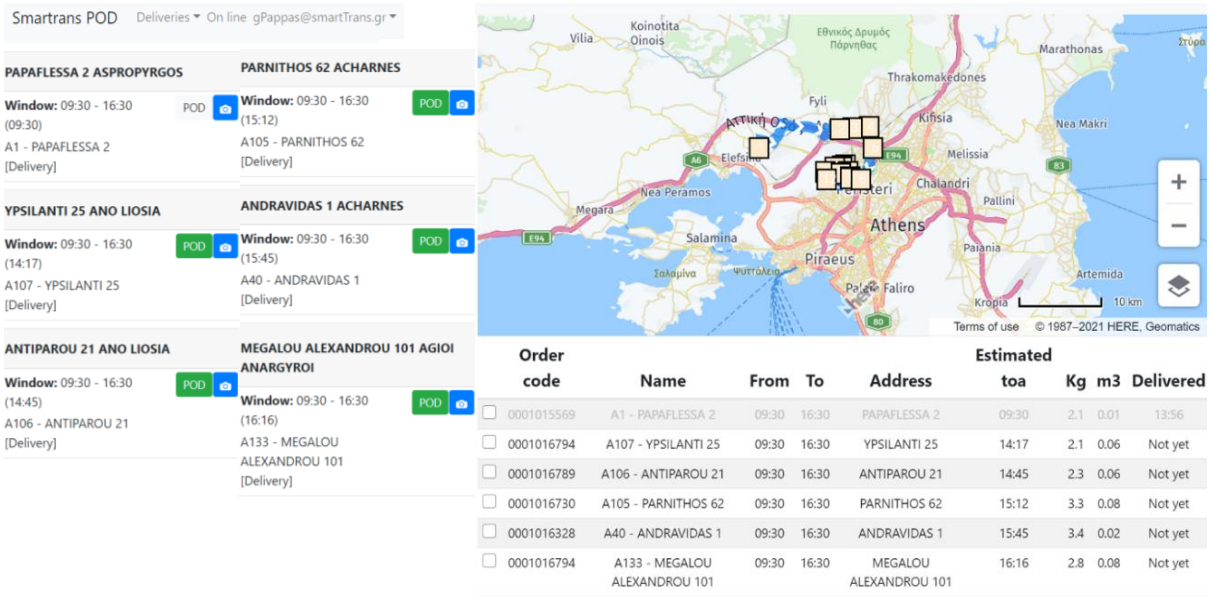
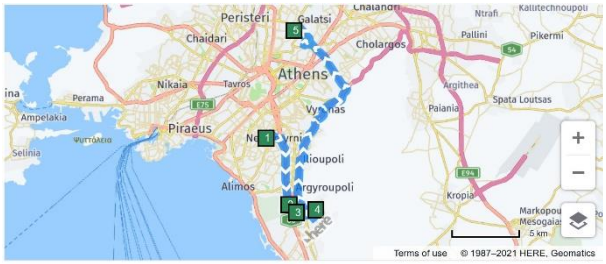


Figure 6.15 Monitoring and Updating the Status and of Orders through the PoD and Routing System

Concluding, in the initial (static) routing, a plan of routes and schedule of deliveries is produced and given to drivers through the PoD system. The static routing can surely enhance planners' efforts since high accurate data from HERE Technologies are exploited for this procedure. However, in some cases, the experience of planners may lead to modifications in the initial plan. Furthermore, unexpected events during the distribution process may lead to delays in the scheduled deliveries. Therefore, the system offers some other functionalities, presented in Section 6.5, to tackle the dynamic nature of routing.

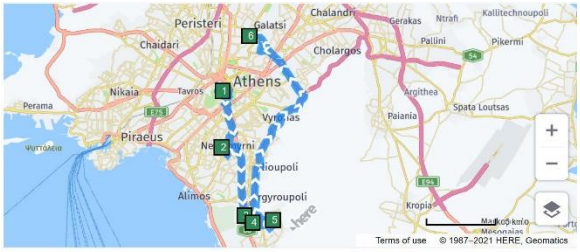
6.5 Rerouting Procedure

As it has already been mentioned, planners may need to modify even if the initial plan of routes has already been extracted. Consequently, the system gives the user the ability to make such modifications on the routes. This action is often executed by logistics and distribution companies, as their requirements may change after the initial plan. In order to permit the addition of an order to an existing route, the system firstly checks that the time windows are respected and that the vehicle's capacity is not overflowed. The algorithm of the drag and drop procedure that execute is given in Appendix E. If this change is allowed and no constraint is violated, then the order is placed in the route, and the estimated time of arrival is recalculated for each order. Such a case is presented in Figure 6.16, where the order with code 0001025256 that was initially located in another route is moved to the green-colored route.



	Order code	Name	From	To	Address	Estimated toa	Kg	m3	Delivere
☐	0001016324	A118 - MANDILARA NIKIFOROU 21	11:40	14:40	MANDILARA NIKIFOROU 21	11:40	87	0.4	Not yet
☐	0001016158	A3 - SOFOKLEOUS 1	09:50	13:30	SOFOKLEOUS 1	12:16	56	0.3	Not yet
☐	0001016730	A60 - RODOPOLIS 39	10:10	13:10	RODOPOLIS 39	12:44	106	0.4	Not yet
☐	0001016730	A123 - PROPONTIDOS 67	10:20	13:20	PROPONTIDOS 67	13:12	116	0.2	Not yet
☐	0001028504	A158 - DAVAKI 4	11:20	14:20	DAVAKI 4	14:09	42	0.7	Not yet

(a)



	Order code	Name	From	To	Address	Estimated toa	Kg	m3	Delivere
☐	0001025256	A170 - EVAGELISTRIAS 1	06:30	09:50	EVAGELISTRIAS 1	07:37	12	0.1	Not yet
☐	0001016324	A118 - MANDILARA NIKIFOROU 21	11:40	14:40	MANDILARA NIKIFOROU 21	11:40	87	0.4	Not yet
☐	0001016158	A3 - SOFOKLEOUS 1	09:50	13:30	SOFOKLEOUS 1	12:16	56	0.3	Not yet
☐	0001016730	A60 - RODOPOLIS 39	10:10	13:10	RODOPOLIS 39	12:44	106	0.4	Not yet
☐	0001016730	A123 - PROPONTIDOS 67	10:20	13:20	PROPONTIDOS 67	13:12	116	0.2	Not yet
☐	0001028504	A158 - DAVAKI 4	11:20	14:20	DAVAKI 4	14:09	42	0.7	Not yet

(b)

Figure 6.16 (a) Initial Route (b) Final Route after Manually Integrating another Order

Moreover, during the execution of deliveries, the logistics companies may need to modify the route of vehicles. These modifications may happen due to cancellations of orders, changes in time windows, as well as unexpected out-of-corporate events such as extreme traffic congestion or weather conditions that may occur during the execution of deliveries. In such cases, when necessary, the system implements rerouting tasks that use the updated data of customers and the real-time traffic data (updated travel times between customers) that prevail at the time by executing an HTTP request to HERE Technologies. Through this approach, the sequence of deliveries may change so that the delays can be minimized.

An example of the rerouting procedure due to order delays is given in Figures 6.17 and 6.18, as when the first order is served, a rerouting task is executed. As shown in Figure 6.17, the first order has a 9.30 to 16.30-time window. However, the order was served after the estimated arrival time and more specifically, at 13:56, as shown in Figure 6.18. This difference between the estimated toa and the actual toa causes great delays in the rest of the plan. More specifically, only 6 of the 15 orders of the route are not affected by this delay, while others have been rescheduled to minimize the route delays. According to the renewed plan, the orders estimated to delay are presented in Figure 6.18 with a red "00:00" in the estimated toa column.

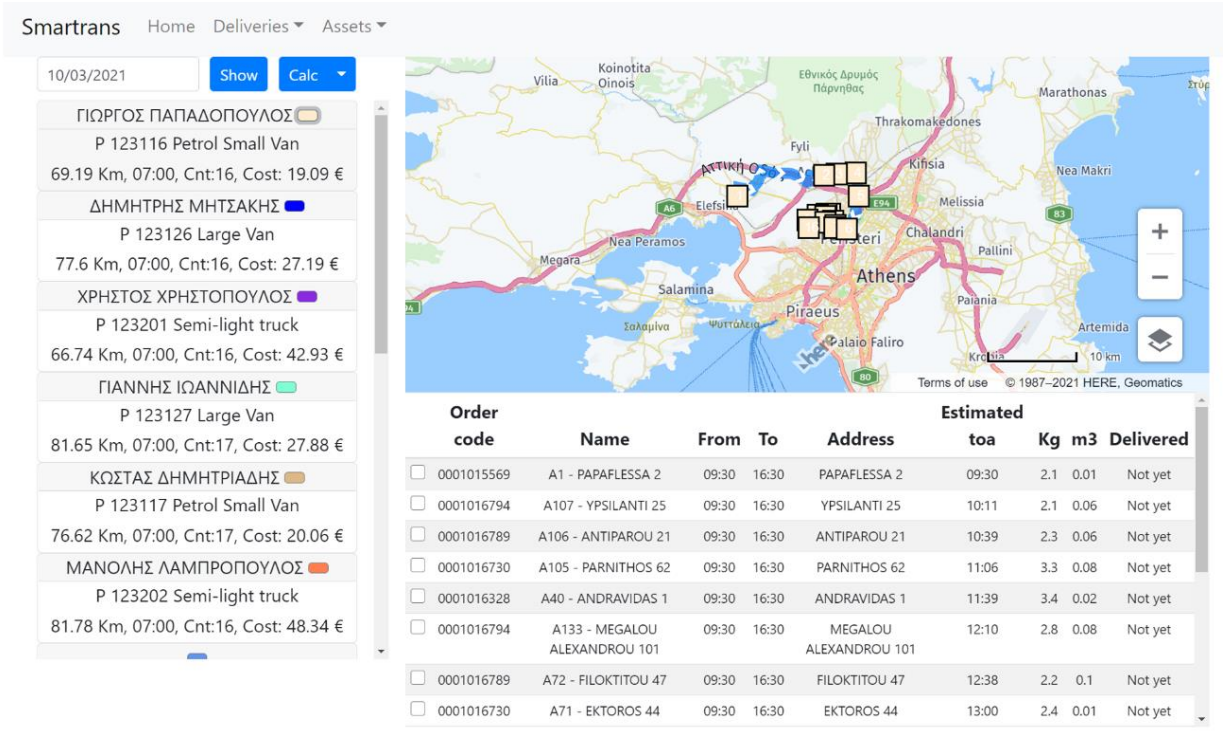


Figure 6.17 Initial Schedule of Deliveries

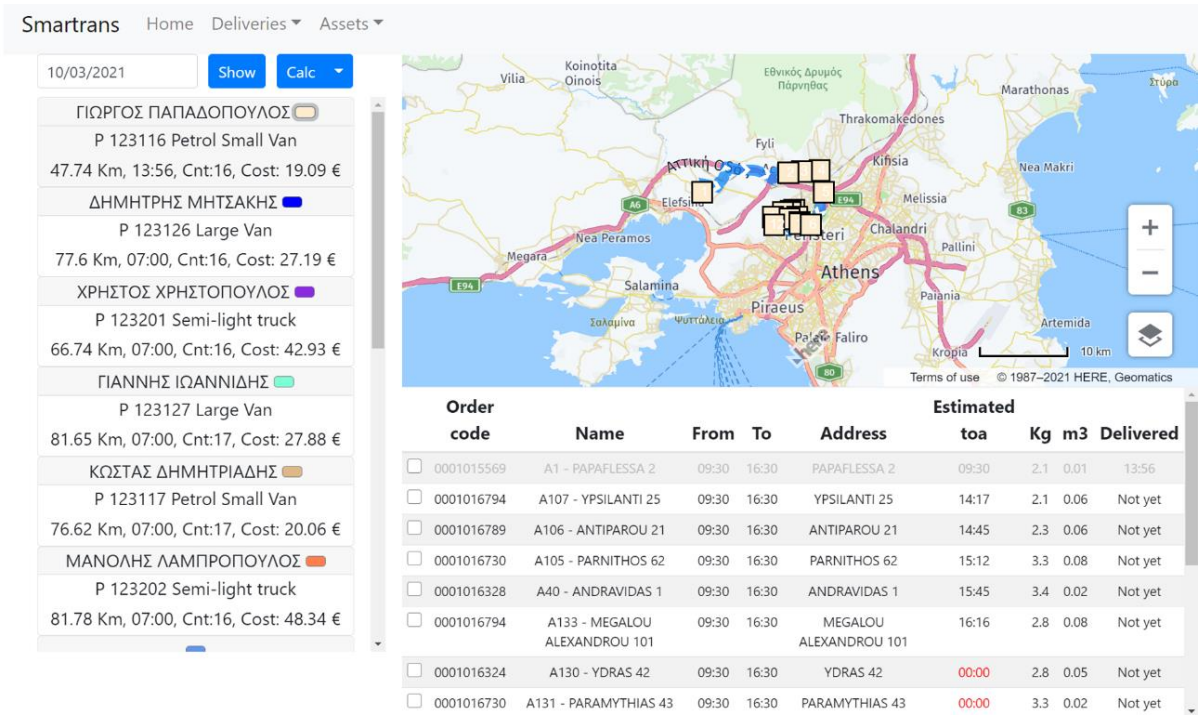


Figure 6.18 Revised Schedule of Deliveries

As it becomes clear, the routing system provided through the cloud can efficiently address the complex problem of routing. The system's functionalities cover, to a great extent, the needs and requirements of companies that operate in freight distribution. In addition, in contrast to most systems where large amount of data is needed that complicate the user's experience, the presented system automatically performs these tasks as the geographical and traffic data are

provided by calling HERE services. Furthermore, these calls are performed through fast HTTP requests, offering reliable real-time results. Finally, as it becomes evident, the system can greatly aid logistics companies to make the right decisions as it constitutes a valuable tool for their distribution operations.

Finally, Figure 6.19 describes the flow of information and data that need to be provided to the routing system in order to be fully operable and produce efficient schedules. Additionally, the inputs and the outputs of the HTTP request sent to the service provider (Here Technologies) are also described. This information and data may change depending on the functionality of the system, as the requirements differ. The system's functionalities are separated into three main categories, the initial (static routing), the route modification and the dynamic routing. Other functionalities described in this section belong in one of the three categories.

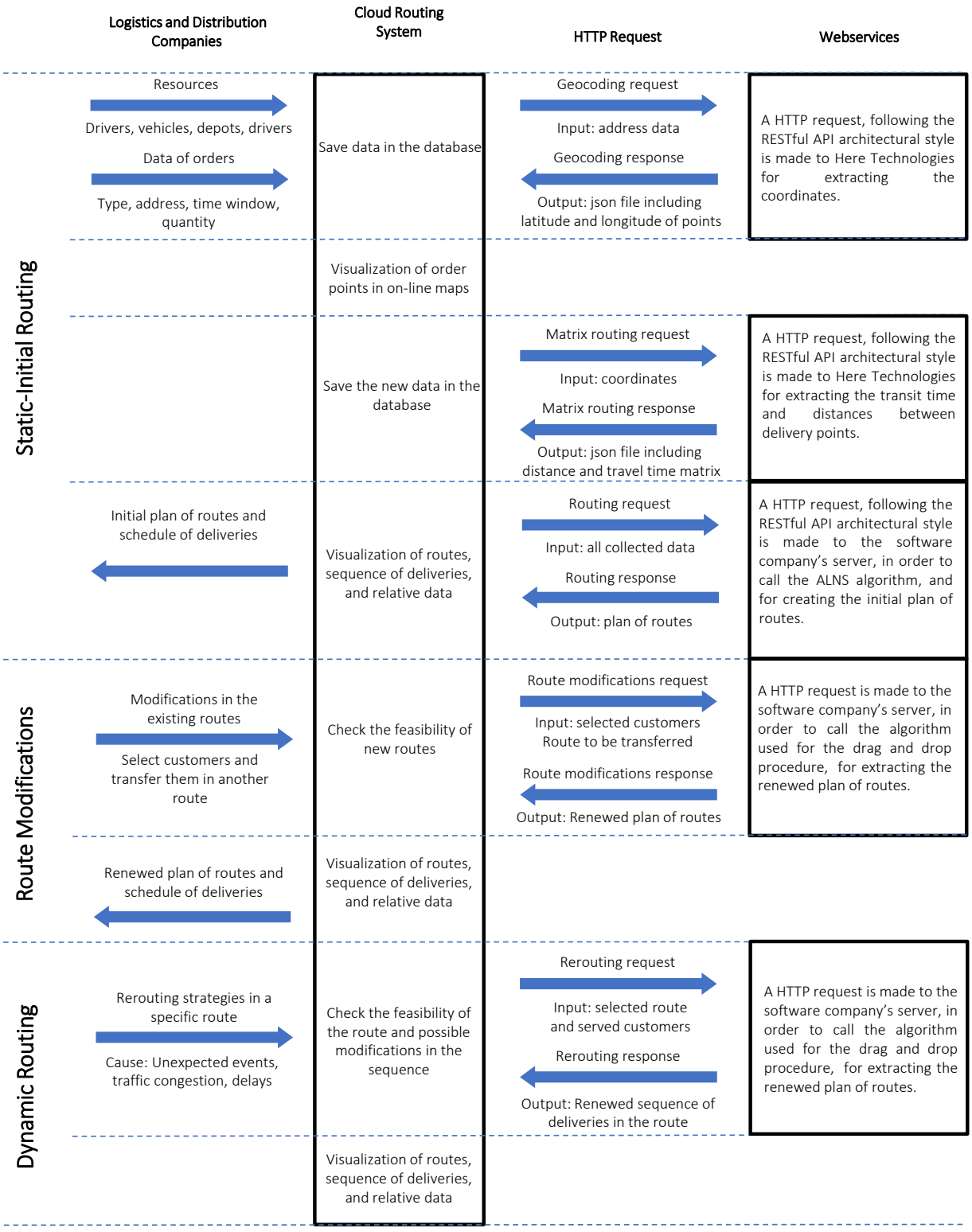


Figure 6.19 Information Flow between Companies, Cloud System and Webservices

6.6 Evaluation and Benefits

As it has already been stated, the algorithms presented in Chapter 3, Chapter 4, and Chapter 5 are integrated into the presented system, while only the ALNS algorithm communicates, at the time, with the system. Since the system is ready to be used by logistics companies to facilitate their distribution operations, it was decided to be further tested in real-life distribution cases, where a large amount of data is dynamically managed. This step, in which a system is evaluated, is avoided in most research studies as only a few algorithms are integrated into real-life systems. This is because the development of a routing system is inextricably linked with software development and related technologies such as cloud computing, database management, Application Programming Interfaces (API's), and webservices that are presented in section 6.3.

In order to evaluate the efficiency of the algorithm and the system in total, three different real-life distribution cases are considered. The first distribution case concerns the store replenishment of a supermarket chain and includes 141 big-size orders. The second case concerns door-to-door deliveries of appliances to consumers that includes 195 orders. Finally, the third case concerns the distribution of goods in an e-commerce channel that includes 748 orders. In each case the plan of routes that emerges from the existing approaches that the companies use, was compared with the one that emerges from the cloud system. The fact that the distribution cases belong in different fields (Store Replenishment, Door-to-door deliveries, E-commerce) and the number and the size of orders vary enhance the research since a wider range of logistics operations is studied.

Moreover, the comparison between the two approaches focuses on the number of routes created and the distribution cost. At this point, it is very important to clarify the significance of defining with a very detailed analysis the fixed and variables costs that form the total distribution cost of logistics companies which itself affects in a great extent the plan of routes and schedule of deliveries to be created. In the three studied companies, the fixed cost is composed of the driver's wage, depreciation and insurance costs, while the variable is composed of the fuel and maintenance cost (Ehmke et al., 2018). In Table 6.4, the average daily numerical data and results of the three companies are presented analytically. More specifically, the results between the existing approach (Ex. Ap.) of the company and of the system (System) in each distribution case are compared, while their difference concerning the cost and the created routes is also calculated (Dif.).

Table 6.4 Comparing Routing Methods in Real-life Distribution Cases

	Store Replenishment			**Door-to-door deliveries**			**E-commerce**		
	Ex. Ap.	**System**	**Dif.**	**Ex. Ap.**	**System**	**Dif.**	**Ex. Ap.**	**System**	**Dif.**
Orders in the specific distribution case	141	141	0	195	195	0	748	748	0
Number of routes created	63	63	0	12	12	0	19	18	-1
Orders per Route	2.24	2.24	0	16.25	16.25	0	39.37	41.55	+2.18
Total distribution cost	11,777	11,771	-6	1711.23	1698.67	-12.56	2198.43	2096.35	-102.08
Cost per Route (€)	186.94	186.84	-0.10	142.60	141.56	1.04	115.71	116.46	+0.75
Cost per Order (€)	83.53	83.48	-0.05	8.78	8.71	-0.07	2.93	2.80	-0.13
Tones Distributed	601.122	600.122	0	10.514	10.514	0	6.922	6.922	0
Distribution Cost (€) per tone	19.62	19.61	-0.01	162.76	161.56	1.20	317.60	302.85	-14.75
Distributed m^3	1289	1289	0	85.5	85.5	0	33.67	33.67	0
Distribution Cost (€) per m^3	9.14	9.13	-0.01	20.14	20.01	-0.13	65.29	62.26	-3.03

As it can easily be observed in Table 6.4, for each company, the use of the system can be profitable for companies. However, the profit in the case of store replenishment is slight as it is reduced by only 6 € in a daily phase on average, while the number of routes created is the same. That is mainly because planners have optimized the procedure and base the plan of routes on their experience and the stability of orders in a daily phase, which allows them to make a few modifications before forming the final plan. In the case of door-to-door deliveries of appliances, the distribution cost is reduced by 12.56 € by using the system, while the routes remain the same. Finally, the e-commerce case presents the most interesting results as both the number of routes and the distribution cost are significantly reduced. More specifically, the number of routes is reduced by 1, while the cost by 102.08 € in a daily phase corresponds to 35,728 € yearly (350 working days). The financial benefits seem important, especially in the e-commerce chain where multiple orders are served daily. However, to decide if the use of the system is profitable for a company, the cost of use needs to be defined. In the current state of the system, where only the service model has been defined (SaaS model), the precise financial profit of companies cannot be clearly defined.

However, the benefits that companies receive by using the routing system are not limited to the decreased distribution cost and the number of vehicles but also expand to the visualization of the routes and the ability to modify them dynamically even during the distribution operations. For example, figure 6.20 presents the plan of routes as it emerges from the system in the door-to-door deliveries case and a detailed plan for one of the routes of the case. By selecting each route, the user can isolate the route and analytically view the various related data. More specifically, the estimated time of arrival to each customer/ delivery point, the delivered or picked quantity, its time window, as well as the status of the order can be observed.

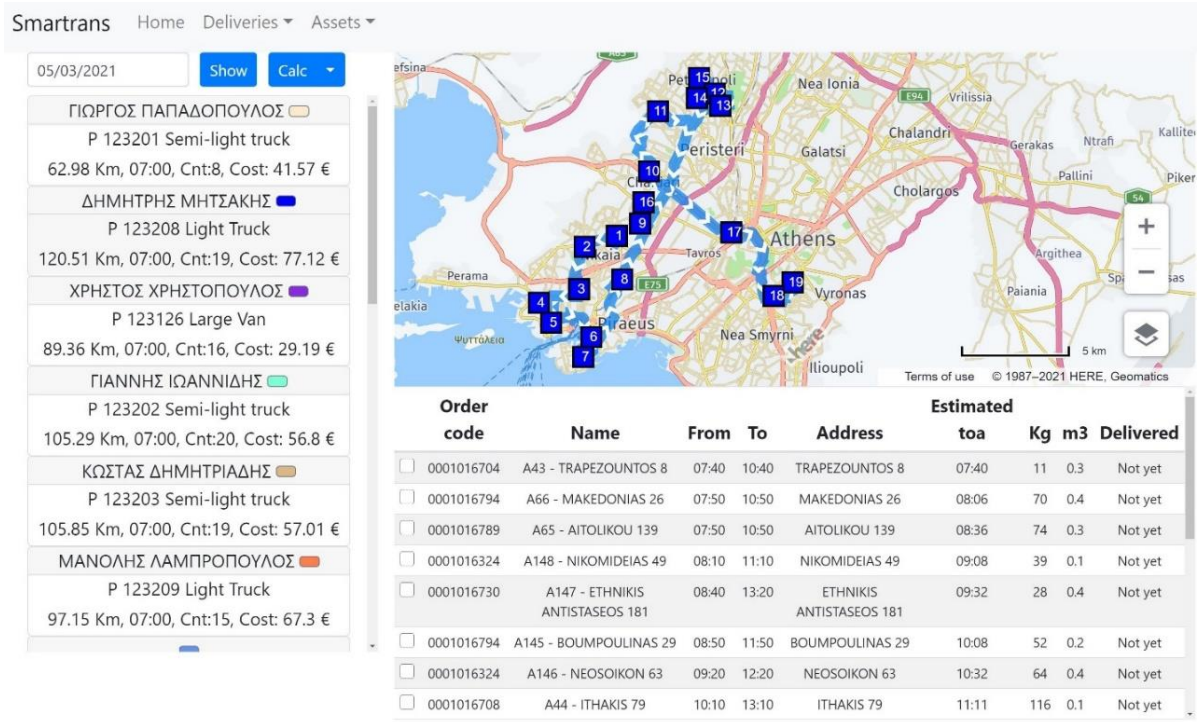


Figure 6.20 Door-to-Door Deliveries through the Cloud System

In practice, static routing is significant for logistics companies and the main functionality of a routing system. However, the need for more intelligent systems in the competitive market o logistics exists. That is why the system addresses the dynamic nature of routing, offers drag and drop procedures to users, and connects with a PoD system. The dynamic routing is executed in a single route when the user considers it appropriate. Unexpected events and traffic congestion are the main reasons.

On the other hand, the drag and drop procedure is executed when minor modifications need to be made between routes. Even in this procedure, the feasibility check is made automatically in order to ensure that no constraint is violated. These system functionalities are considered equally important since they enhance planners' efforts and offer better plans. Combining the experience of experts and the efficiency of an optimization algorithm and an advanced system is the key. Finally, the developed PoD system supports planers during the distribution, allowing them to track deliveries and know the status of each order.

In conclusion, the system offers reduced costs and multiple functionalities to enhance planners' efforts and offer efficient plans. Additionally, it can support logistics companies throughout the entire distribution process, starting from the initial routing and expanding to the real-time updating of order status.

6.7 Concluding Remarks

Routing and scheduling systems are necessary for today's challenging distribution and logistics operations. The complex set of activities, functions, variables and restrictions that logistics companies face and need to consider for creating an efficient distribution plan cannot be handled without advanced routing and scheduling systems. This is even further complicated in city centers where daily deliveries and traffic congestion dramatically increase.

The algorithm and the cloud routing system that was developed aimed to solve this exact inadequacy and difficulty, managing to cover the needs and requirements of most logistics companies operating in urban freight distribution. Besides creating efficient routes and delivery schedules, the system also has multiple functionalities to support planners and enhance the routing process. Such functionalities are mainly the visualization of routes on the map and the presentation of related to the orders and routes information and the modifications and adjustments that can be made to the plan of routes, based on the planners' experience.

In order to support all these functions and create such an operable system, multiple technologies have been exploited and combined, such as cloud computing, database management, APIs, webservices etc., while simultaneously the core of the system is the ALNS algorithm that has been developed. The algorithm has been tested both in datasets of the literature and in real-life distribution cases and was proven to be efficient both in terms of results and computation time.

References

Arunachalam, D., Kumar, N., & Kawalek, J. P. (2018). Understanding big data analytics capabilities in supply chain management: Unravelling the issues, challenges and implications for practice. Transportation Research Part E: Logistics and Transportation Review, 114, 416–436. https://doi.org/10.1016/j.tre.2017.04.001

Seongmoon, K., Lewis, M. E., & White, C. C. (2005). Optimal vehicle routing with real-time traffic information. IEEE Transactions on Intelligent Transportation Systems, 6(2), 178–188. https://doi.org/10.1109/TITS.2005.848362

Chapter 7: Research Conclusions and Future Works

7.1 Synopsis of Research Contributions

The VRP, which constitutes one of the most frequently encountered problems in logistics, was first proposed more than 60 years ago. Over the years, multiple variants have been proposed, each focusing on specific aspects and variables of the problem, leading to specific variations to the generic mathematical formulation. In most variants of the problem, the aim is to find the most cost-effective routes that serve the demand. Consequently, the objective function in most cases is set, which is the distribution cost minimization. However, in recent years, some researchers focus on the minimization of greenhouse gas emissions due to the increased need.

In the current thesis, a class of vehicle routing problems are presented, studied, and addressed. The variants that are considered have emerged through a research study and the cooperation with companies of the logistics industry (distribution and software development companies). Moreover, the VRP is vital for the logistics industry since it affects the profit margins of production and logistics companies. This fact has made them among the most popular and widely studied problems in Supply Chain, while it has also attracted the interest of other fields, such as Mathematics, Operational Research, Information Management and Decision Sciences.

The fact that so many different fields are interested in the VRP and its variants has led to multiple optimization algorithms. Each algorithm may focus on different variants and aspects of the problem. In most cases, the VRPTW is addressed due to its connection with real-life cases. While this variant is widely studied and addressed by researchers and practitioners, the developed MOLNS and MOEA algorithms manage to be highly efficient and offer multiple new-optimal solutions so far. That is not an easy task, primarily when so many researchers have centered around the subject.

The same stands for the VRPSPD and the HVRP, which are also highly related to real-life distribution cases. Initially, collecting goods that can be remanufactured, repaired, or recycled, from retailers, end customers, and stores, is significant and may lead to financial and environmental benefits both for consumers and logistics companies. That is why the VRPSPD is important for both research and industry. As for the vehicles' types, some limited customers have identical vehicles. In most cases companies have different types to fulfil the requirements of distribution. That is the main reason for selecting the HVRP to be a main variant of study. Moreover, addressing the HVRP overlaps the cases in which logistics companies own identical vehicles, and consequently can cover their needs. The ALNS and MOEA algorithms propose new-optimal solutions in the datasets of the literature that are tested, indicating their efficiency. Moreover, in addition to the variants of the VRP that are taken into account and addressed, some variables and constraints are added to the mathematical model in order to bring the problem even closer to reality. Such variables and constraints are drivers' shifts, as well as vehicle autonomy. Both can affect the length of routes and consequently the number of routes

and the distribution cost. However, the benefits can be huge as no working regulation is violated and the schedule can be very precise.

Moreover, the ALNS algorithm besides offering multiple new optimal solutions in various datasets of the literature, considers and addresses multiple variables and constraints, making the proposed algorithm the proper method to be integrated into the cloud routing system. As it has already been mentioned, the three algorithms are all stored in the server that is managed by the software development company. However, the system in its current form communicates and receives the plan of routes produced by the ALNS algorithm. The ALNS was selected in this initial form of the cloud system as it can address more variants of the VRP compared to the other two algorithms. Furthermore, the MOEA and the MOLNS focus on the number of vehicles and the total traveled distance, while the ALNS to the total distribution cost that in most cases is more important for logistics and distribution companies.

However, in cases that an algorithm that is the core of a system, its efficiency is not the only aspect that needs to be studied and considered. The computation time and the number of orders that the system can handle are also important. That is why the specific algorithm is integrated into the cloud system. The ALNS has been tested in multiple datasets with the number of orders ranging from 100 to 748. In all instances, including datasets of the literature and real-life distribution cases, the algorithm can create a solution very fast (less than 80 seconds) while the more it runs, the more the solution is improved (reducing the distribution cost).

Moreover, in addition to the efficiency of the developed algorithms and the results offered, a major contribution of the research is the utilization of real data of distances and transit times that make the routing and the schedule of deliveries even more reliable in terms of estimation of arrival times and cost estimation. Transit times data during static routing is a statistical estimation of cartographic data providers. On the contrary, during the dynamic routing the transit times data results from the traffic that prevails at the specific moment in the road network and therefore any routing process is done with great precision.

In addition to the given transit times and distances utilized by the appropriate providers, in the present dissertation maps are utilized, in which the routes are visualized either cumulatively or individually. In addition, the order/sequence of visits and deliveries is graphically defined, a function that can significantly help drivers as these data can be obtained by the Proof of Delivery system.

7.2 Future Research

The objectives that were initially set, which are (i) the study of a class of VRP variants, (ii) their mathematical formulation, (iii) the development of metaheuristics algorithms addressing the studied VRP variants and (iv) the practical integration of the algorithms into a cloud routing system have been achieved during the doctoral studies. Nevertheless, these objectives and the

individual elements of the study can constitute a starting point for future research activities. In the remainder of the 7.2 section, several directions regarding the future studies are given.

Firstly, over the last few years, researchers focus on hybrid algorithms. These algorithms combine the full potential of multiple metaheuristic algorithms. Each particular algorithm focuses on specific aspects containing advantages and disadvantages. On the other hand, the combination of multiple metaheuristic algorithms aims to eliminate the weak elements of each method while also strengthen their essential features. In view of the benefits, the proposed algorithms of Chapter 3, 4 and 5 will be the base of new proposed hybrid metaheuristics. However, additional metaheuristics will be studied in order to select the most suitable for each case.

Another research direction concerns the aspect of CO_2 emissions which will be considered in all of the three proposed algorithms. In the case of the two multiobjective algorithms, i.e. the MOEA and the MOLNS, the CO_2 emissions will be involved as the one of the two objective functions. More specifically, the minimization of CO_2 emissions will be the one objective function, while the second one will be the distribution cost. This can help companies that in today's challenging market need to follow the national restrictions concerning the greenhouse gas emissions. As for the ALNS algorithm, this attribute should be included as an estimation. Through this way each company will be able to calculate with high accuracy the emissions caused by the distribution process.

Furthermore, each of the proposed optimization methods can be further improved in order to obtain even better results. As for the MOLNS and the ALNS algorithm that have the same base, i.e. the LNS algorithm, new and efficient destroy and repair operators will be examined and studied. Concerning the MOEA, the next steps of the research approach will be focused on the crossover and mutation operators that significantly affect solutions.

Despite the next steps of the research that concern the algorithms, there are several actions that can be made for transforming the cloud routing system to a commercial product. Initially, for the system to become a commercial product that fully supports logistics companies operating in freight distribution, more VRP variants should also be considered by the proposed algorithms and more specifically by the ALNS algorithm that is applied in the current form of the system. More specifically, after testing the algorithm in real-life distribution cases, it was found that a limited number of routes, mainly due to the vehicle capacity constraint, last less than half of the driver's shift. This means that the same driver can execute another route before his shift ends. This variable is recognized as the Multi-trip VRP variant. Improving the algorithm by taking into account this variant will lead to more excellent coverage of the needs of logistics companies, regardless of their size.

Additionally, since the ALNS was tested in the Collaborative VRP, the system may in the future incorporate the specific variant. Consequently, a further research study should focus on the profit-sharing mechanism that the collaborative companies need to define before

cooperating. In each collaboration case, the profit-sharing approach needs to be fair for all parties while also profitable. However, the fact that profit-sharing constitutes another research topic and multiple approaches has already been proposed led us to not consider it as part of the research study. Finally, the system must handle data from multiple logistics providers in compliance with the General Data Protection Regulation (GDPR). This is essential since customers from different companies need to be uploaded in a shared platform for addressing the Collaborative VRP. This further research can provide logistics companies with a valuable tool for more efficient last-mile distribution.

Moreover, while the three proposed algorithms are stored in the software development company's server as libraries, as described in section 6.3, only the ALNS algorithm connects with the cloud routing system through the appropriate HTTP requests, in order to extract the plan of routes. Given that the ALNS algorithm is single-objective and does not cover the needs of companies that need to simultaneously optimize two objectives, a future step of the research is to incorporate the existing algorithms with the cloud routing system. Specifically, a future aim of this study is to give users the ability to select the algorithm that will be applied and extract the plan of routes. In order to succeed this step efficiently, the objectives which concern logistics and distribution companies the most need to be defined, and also integrated into the multiobjective algorithms. In their current form, the MOEA and the MOLNS algorithm do not address as many variables are considered and addressed by the ALNS algorithm.

Concluding, it becomes clear that the developed ALNS algorithm, which is implemented by the cloud routing system, can offer significant benefits to logistics companies. The efficient plan of routes produced by the ALNS algorithm is enhanced by other system functionalities, offering a complete routing solution that strongly supports planners. However, this does not mean that the system cannot be further improved. As already mentioned, integrating more variables and constraints into the algorithm, and subsequently to the cloud routing system, will significantly increase its usability and needs coverage since more requirements and users' needs will be covered.

7.3 Research Outputs

During the Ph.D dissertation, multiple variants of the VRP were studied, and multiple solution methods were developed for addressing them. Therefore, the publications that emerged are presented below.

Scientific Journals

- Kechagias, E. P., Gayialis, S. P., **Konstantakopoulos, G. D.**, & Papadopoulos, G. A. (2019). Traffic flow forecasting for city logistics: a literature review and evaluation. International Journal of Decision Support Systems, 4(2), 159-176.

- Kechagias, E. P., Gayialis, S. P., **Konstantakopoulos, G. D.**, & Papadopoulos, G. A. (2020). An Application of a Multi-Criteria Approach for the Development of a Process Reference Model for Supply Chain Operations. Sustainability, 12(14), 5791. (H Index = 85, IF = 3.251)
- **Konstantakopoulos, G. D.**, Gayialis, S. P., & Kechagias, E. P. (2020). Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. Operational Research, 1-30. (H Index = 22, IF = 2.410)
- **Konstantakopoulos, G.D.**; Gayialis, S.P.; Kechagias, E.P.; Papadopoulos, G.A.; Tatsiopoulos, I.P. A Multiobjective Large Neighborhood Search Metaheuristic for the Vehicle Routing Problem with Time Windows. Algorithms 2020, 13, 243. (H Index = 34, Cite Score = 2.9)
- Kechagias, E. P., Gayialis, S. P., **Konstantakopoulos, G. D.**, & Papadopoulos, G. A. (2020). An Application of an Urban Freight Transportation System for Reduced Environmental Emissions. Systems, 8(4), 49.
- **Konstantakopoulos, G.D.**, Gayialis, S.P., Kechagias, E.P., Papadopoulos, G.A., Tatsiopoulos, I.P. Delivering and Picking Goods under Time Window Restrictions: An Effective Evolutionary Algorithm. Journal of Intelligent & Fuzzy Systems 2020. (H Index = 57, IF = 1.851)
- **Konstantakopoulos, G. D.**, Gayialis, S. P., Kechagias, E. P., Papadopoulos, G. A., & Tatsiopoulos, I. P. (2021). An algorithmic approach for sustainable and collaborative logistics: A case study in Greece. International Journal of Information Management Data Insights, 1(1), 100010.
- Gayialis, S.P., Kechagias, E.P., **Konstantakopoulos, G.D.** A City Logistics System for Freight Transportation: Operational Research and Information Technology in Practice. Operational Research. Operational Research, 1-30. (H Index = 22, IF = 2.410)
- **Konstantakopoulos, G.D.**, Kechagias, E.P., Gayialis, S.P., Tatsiopoulos, I.P. Development of an Optimization Algorithm for a Vehicle Routing and Scheduling System. Engineering Computations. (Under Review)

Book Chapters

- Gayialis, S. P., **Konstantakopoulos, G. D.**, & Tatsiopoulos, I. P. (2019). Vehicle routing problem for urban freight transportation: A review of the recent literature. In Operational Research in the Digital Era–ICT Challenges (pp. 89-104). Springer, Cham.
- **Konstantakopoulos, G.D.**; Kechagias, E.P.; Gayialis, S.P.; Tatsiopoulos, I.P. Green Freight Distribution: A Case Study in Greece. Business and Economics. (Under Review)

Conferences

- Gayialis, S. P., **Konstantakopoulos, G. D.**, Papadopoulos, G. A., Kechagias, E., & Ponis, S. T. (2018). Developing an Advanced Cloud-Based Vehicle Routing and Scheduling System for Urban Freight Transportation. In IFIP International Conference on Advances in Production Management Systems (pp. 190-197). Springer, Cham.
- Gayialis, S. P., Kechagias, E., Papadopoulos, G. A., & **Konstantakopoulos, G. D.** (2019). Design of a Blockchain-Driven System for Product Counterfeiting Restraint in the Supply Chain. In IFIP International Conference on Advances in Production Management Systems (pp. 474-481). Springer, Cham.
- Kechagias, E., Gayialis, S. P., **Konstantakopoulos, G. D.**, Papadopoulos, G. A. (2019) An Advanced Routing and Scheduling System for Dangerous Goods Transportation. In 8th International Symposium and 30th National Conference on Operational Research (pp. 126-130).
- Gayialis, S. P., **Konstantakopoulos, G. D.**, Papadopoulos, G. A., Kechagias, E. P. (2020) An Advanced Transportation System Based on Internet of Things. Proceedings of the International Conference on Industrial Engineering and Operations Management, Dubai, UAE.
- Kechagias, E.P., Gayialis, S.P., **Konstantakopoulos, G.D.**, Papadopoulos, G.A., (2020). Evaluating the Appropriateness of Traffic Forecasting Algorithms for Use in a Freight Transportation System. In Proceedings of XIV Balkan Conference on Operational Research (BALCOR 2020), 84-88.
- **Konstantakopoulos, G. D.**, Gayialis, S. P., Kechagias, E. P. & Papadopoulos, G. A. (2020). Conventional and Electric Vehicles for Freight Distribution: A Case Study in Greece. In Proceedings of XIV Balkan Conference on Operational Research (BALCOR 2020), 292-296.

Appendix A Supplement to Chapter 2

The link below includes the categorization of the 263 articles, which were exploited in the research of Chapter 2.

https://static-content.springer.com/esm/art%3A10.1007%2Fs12351-020-00600-7/MediaObjects/12351_2020_600_MOESM1_ESM.xlsx

Appendix B Supplement to Chapter 3

Solutions in Solomon's instances which improve the results and offer new Pareto-optimal solutions are given below:

Table B.1 New Optimal Solution in RC202 Instance

Number of Route	Route Sequence
1	0-65-83-64-23-21-48-18-19-49-22-20-24-25-77-75-58-52-82-0
2	0-69-68-88-53-99-57-86-87-9-10-97-59-74-13-17-60-100-70-0
3	0-14-47-16-15-11-12-78-73-79-7-6-8-46-4-2-55-0
4	0-71-67-62-76-51-84-56-66-
5	0-92-95-85-63-33-34-31-29-27-26-28-30-32-89-50-93-96-94-91-80-0
6	0-45-5-3-1-42-39-37-36-44-41-38-40-43-35-72-54-68-0
7	0-81-61-90-0
Total Distance	1117.41

Table B.2 New Optimal Solution in RC207 Instance

Number of Route	Route Sequence
1	0-92-95-67-31-29-28-30-63-85-76-18-21-23-19-49-22-51-84-62-50-34-27-26-32-33-89-56-91-80-0
2	0-69-98-88-78-73-79-7-6-2-8-5-3-1-45-4-46-60-55-100-70-68-0
3	0-61-72-71-93-94-81-42-44-40-36-35-37-38-39-43-41-54-96-0
4	0-64-83-99-52-86-75-59-87-74-57-65-90-0
5	0-82-53-12-14-47-17-16-15-11-10-9-13-97-58-77-25-48-20-24-66-0
Total Distance	970.78

Table B.3 New Optimal Solution in R201 Instance

Number of Route	Route Sequence
1	0-5-83-45-82-47-36-19-11-64-49-46-48-0
2	0-95-59-92-42-15-14-98-61-16-44-38-86-85-99-6-94-53-26-0
3	0-28-12-29-76-21-73-40-87-57-43-37-97-96-13-58-0
4	0-33-65-71-30-51-9-81-79-78-34-50-3-68-54-0
5	0-2-72-39-67-23-75-22-41-56-74-4-55-25-24-80-77-0
6	0-52-69-31-88-7-18-8-84-17-91-100-93-60-89-0
7	0-27-62-63-90-10-20-66-35-32-70-1-0
Total Distance	1156.73

Besides the solutions in instances of well known datasets of the literature, Appendix C also includes the MOEA algorithm that is presented in Chapter 4, and is given in the Python programming language.

```
import random
import numpy as np
import pandas as pd
import timeit

start = timeit.default_timer()

filename = "Data_File.xlsx"
dataset = pd.read_excel(filename)
n=len(dataset)
tt = pd.read_excel(travel_times_matrix.xlsx)
dm = pd.read_excel(distance_matrix.xlsx)

#---- seperating the dataset in arrays and matrices --------
idcust = dataset.iloc[:,0].values
cx = dataset.iloc[:,1].values
cy = dataset.iloc[:,2].values
capacity = int(dataset.iloc[0,7])
demand  = dataset.iloc[:,3].values
tws = dataset.iloc[:, 4].values
twe = dataset.iloc[:, 5].values
st = dataset.iloc[:, 6].values
n1 = n - 1

sumroutes = idcust.sum()
paretosize = 15
pareto = np.zeros(shape = (paretosize, n1))
f1 = np.zeros(shape = (paretosize, 1))
f2 = np.zeros(shape = (paretosize, 1))

def quantity(routes):
    s = np.count_nonzero(routes)
    q = 0
    for i in range (1, s+1):
        q = q + demand[routes[i]]
    return q

def converter(route):
    oprt = np.zeros(shape=(n,n))
    k1=0
    t=0
    q=0
    j=0
    s=1
    nv = 1
    k2 = int(route[j])
    oprt[nv, s] = int(k1)
    while j in range(n-1):
```

```
            k2 = int(route[j])
            q = q + demand[k2]
            if q > capacity or t + tt[k1,k2] > twe[k2]:
                k1=0
                q=0
                t=0
                s = 1
                k2 = int(route[j])
                nv = nv + 1
                oprt[nv-1, s] = int(k2)
            else:
                oprt[nv-1, s] = int(k2)
                s = s+1
                if t + tt[k1,k2] < tws[k2]:
                    t = tws[k2] + st[k2]
                else:
                    t = t + tt[k1, k2] + st[k2]
                k1 = int(route[j])
                j = j + 1
        return oprt

    #First function to optimize - Total Number of Vehicles
    def function1(route):
        cch = converter(route)
        chrom_nv = 0
        for i in range(len(cch)):
            if cch[i, 1] != 0:
                chrom_nv = i+1
        return chrom_nv

    #Second function to optimize - Total Travelled Distance
    def function2(route):
        nov1 = function1(route)
        cch = converter(route)
        td = 0
        for i in range(nov1):
            for j in range(n-1):
                td = td + dm[int(cch[i, j]), int(cch[i, j+1])]
        return td

    def paretocheck(routeimp, f1, f2, pareto):
        ff1 = function1(routeimp)
        ff2 = function2(routeimp)
        s1 = np.count_nonzero(f1)
        k = 0
        i1 = 0
        while i1 in range(s1):
            if ff1 == f1[i1] and ff2 < f2[i1]:
                f2[i1] = ff2
                pareto[i1, :] = routeimp.transpose()
                k = 0
                break
            if ff2 < f2[i1] or (ff1 < f1[i1] and ff1 not in f1):
                k = k + 1
            i1 = i1 + 1
        if k == s1:
```

```
105.            f2[s1] = ff2
106.            f1[s1] = ff1
107.            pareto[s1, :] = routeimp.transpose()
108.        return  f1, f2, pareto
109.
110.    def par(f1, f2, pareto):
111.        s1 = np.count_nonzero(f1)
112.        x1 = np.zeros(shape = (s1,1))
113.        paretonew = np.zeros(shape=(paretosize, n1))
114.        f1new = np.zeros(shape=(paretosize, 1))
115.        f2new =  np.zeros(shape=(paretosize, 1))
116.        for i in range(s1):
117.            k = 0
118.            for j in range(s1):
119.                if f2[i] < f2[j] or f1[i] < f1[j] and i != j:
120.                    k = k + 1
121.            if k < s1-1:
122.                x1[i] = 1
123.        k1 = 0
124.        for i in range(s1):
125.            if x1[i] == 0:
126.                paretonew[k1, :] = pareto[i, :]
127.                f1new[k1] = f1[i]
128.                f2new[k1] = f2[i]
129.                k1 = k1 + 1
130.        return f1new, f2new, paretonew
131.
132.    # part of the normalization procedure
133.    minv = int(demand.sum()//capacity)
134.    partstw = (twe.max() + tws.min())/2
135.    partsdm = (dm.max() + dm.min())/2
136.    categ = dm[:,:] / partsdm
137.    norm = partstw/partsdm
138.
139.    #------
        check which customers can not be combined in the same route based on their distance/time and their
         time windows-------
140.    xron = np.zeros(shape=(n, n))
141.    nopos = 0
142.    for i in range(n):
143.        for j in range(n):
144.            if tws[i] + tt[i,j] + st[i] > twe[j] or i == j:
145.                xron[i,j] = 101
146.                nopos = nopos + 1
147.
148.    step1 = 0.07
149.    chrom = np.zeros(shape=(int((1//step1)+2), n-1))
150.    routes = np.zeros(shape=(n, n+1))
151.    pos  = np.zeros(shape=(int((1//step1)+2), 1))
152.    s123 = np.zeros(shape=(int((1//step1)+2), 1))
153.    k = 0
154.
155.    num_chrom = 0
156.    routeimp = np.zeros(shape=(1,n))
157.    #------- starting to explore solutions ------
158.    for w1 in np.arange(0.0, 1.00001, step1):
```

```
159.        w2 = 1 - w1
160.        v_chrom = 0
161.        s_chrom = 0
162.        r = 0
163.        q = True
164.        s = 0
165.        i = 0
166.        xron1 = np.copy(xron)
167.        routes  = np.zeros(shape=(n, n+1))
168.        while q:
169.          if i ==0:
170.            t = 0
171.            sumq = 0
172.            k1 = 0
173.            routes[r, k1] = 0
174.          mindt = 1000000
175.          dok = 0
176.          minf = mindt
177.          for j in range(1, n):
178.            if xron1[i,j] != 101 and sumq + demand[j] <= capacity and t + tt[i,j] <= twe[j]:
179.              minf = w1*dm[i,j]*norm + w2*(-t-tt[i,j] + twe[j])
180.              if minf < mindt:
181.                v_chrom = j
182.                s = j
183.                mindt = minf
184.                dok = dok + 1
185.          if dok == 0 and j == n-1 :
186.            s = 0
187.            r = r + 1
188.          else:
189.            chrom[num_chrom,  s_chrom] = v_chrom
190.            s_chrom = s_chrom + 1
191.          k1 = k1 + 1
192.          routes[r, k1] = s
193.          xron1[:, s] = 101
194.          xron1[i, s] = 101
195.          xron1[s, i] = 101
196.          sumq = sumq + demand[s]
197.          if t + tt[i,s] < tws[s]:
198.            t = tws[s] + st[s]
199.          else:
200.            t = t + tt[i, s] + st[s]
201.          i = s
202.          if xron1.sum() == 101*n*n:
203.            break
204.        if num_chrom == 1:
205.          routeimp = np.copy(chrom[num_chrom,  :])
206.        else:
207.          routeimp = np.copy(chrom[num_chrom,  :])
208.          f1, f2, pareto = paretocheck(routeimp,f1,  f2, pareto)
209.        if num_chrom%6==0:
210.          f1, f2, pareto = par(f1, f2, pareto)
211.        num_chrom = num_chrom + 1
212.      print("Constructed - Initial Solution", np.trim_zeros(f1), np.trim_zeros(f2))
213.      novinitial = function2(routeimp)
214.
```

```
215.    def insert1(noc, routes2, cust, nvinitial):
216.        sss=0
217.        while cust.sum() != 0:
218.            customerchosen = int(cust[sss])
219.            cust[sss] = 0
220.            initialcost = 99999999
221.            sk = np.count_nonzero(routes2)
222.            routes2n = np.copy(routes2)
223.            new = 0
224.            www = False
225.            while new in range(0, sk):
226.                routes2n = np.copy(routes2)
227.                for j in range(0, new):
228.                    routes2n[j] = routes2[j]
229.                routes2n[new] = customerchosen
230.                for j in range(new, n1-1):
231.                    routes2n[j+1] = routes2[j]
232.                if function2(routes2n) <= initialcost and function1(routes2n) <= nvinitial:
233.                    routesopt = np.copy(routes2n)
234.                    initialcost = function2(routes2n)
235.                    www = True
236.                new = new + 1
237.            sss=sss+1
238.            if www == True:
239.                routes2 = np.copy(routesopt)
240.        return routes2
241.
242.    def insert2(noc, routes2, cust, nvinitial):
243.        sss=0
244.        while cust.sum() != 0:
245.            customerchosen = int(cust[sss])
246.            cust[sss] = 0
247.            initialcost = 99999999
248.            sk = np.count_nonzero(routes2)
249.            routes2n = np.copy(routes2)
250.            new = 0
251.            www = False
252.            while new in range(0, sk):
253.                routes2n = np.copy(routes2)
254.                for j in range(0, new):
255.                    routes2n[j] = routes2[j]
256.                routes2n[new] = customerchosen
257.                for j in range(new, n1-1):
258.                    routes2n[j+1] = routes2[j]
259.                if function2(routes2n) <= initialcost:
260.                    routesopt = np.copy(routes2n)
261.                    initialcost = function2(routes2n)
262.                    www = True
263.                new = new + 1
264.            sss=sss+1
265.            if www == True:
266.                routes2 = np.copy(routesopt)
267.        return routes2
268.
269.    def insert3(noc, routes2, cust, nvinitial):
270.        sss=0
```

```
271.        while cust.sum() != 0:
272.            customerchosen = int(cust[sss])
273.            cust[sss] = 0
274.            initialcost = 99999999
275.            sk = np.count_nonzero(routes2)
276.            routes2n = np.copy(routes2)
277.            new = 0
278.            www = False
279.            while new in range(0, sk):
280.                routes2n = np.copy(routes2)
281.                for j in range(0, new):
282.                    routes2n[j] = routes2[j]
283.                routes2n[new] = customerchosen
284.                for j in range(new, n1-1):
285.                    routes2n[j+1] = routes2[j]
286.                if function2(routes2n) <= initialcost and function1(routes2n) <= nvinitial - 1:
287.                    routesopt = np.copy(routes2n)
288.                    initialcost = function2(routes2n)
289.                    www = True
290.                new = new + 1
291.            sss = sss+1
292.            if www == True:
293.                routes2 = np.copy(routesopt)
294.        return routes2
295.
296.    dl = 3
297.    ul = 9
298.
299.    def impnv1(routes):
300.        nvinitial = function1(routes)
301.        routes1 = np.copy(routes)
302.        routesc = converter(routes)
303.        kc = random.randint(0, len(routesc)-1)
304.        while routesc[kc, 1] == 0:
305.            kc = random.randint(0, len(routesc)-1)
306.        customers = routesc[kc, :]
307.        cust = np.trim_zeros(customers)
308.        noc = np.count_nonzero(cust)
309.        routes2 = np.zeros(shape=(n1, 1))
310.        k1 = 0
311.        for i in range(n1):
312.            if routes1[i] not in cust:
313.                routes2[k1] = routes1[i]
314.                k1 = k1 + 1
315.        routes2 = insert3(noc, routes2, cust, nvinitial)
316.        if (function1(routes2) < nvinitial or (function1(routes2) <= nvinitial and function2(routes2) < functio
        n2(routes))) and routes2.sum() == sumroutes:
317.            routes = np.copy(routes2)
318.        return routes
319.
320.    def impnv2(routes):
321.        nvinitial = function1(routes)
322.        routes1 = np.copy(routes)
323.        routesc = converter(routes)
324.        kc = random.randint(0, len(routesc)-1)
325.        while routesc[kc, 1] == 0:
```

```
326.            kc = random.randint(0,  len(routesc)-1)
327.        customers = routesc[kc, :]
328.        cust = np.trim_zeros(customers)
329.        noc = np.count_nonzero(cust)
330.        routes2 = np.zeros(shape=(n1, 1))
331.        k1 = 0
332.        for i in range(n1):
333.            if routes1[i] not in cust:
334.                routes2[k1] = routes1[i]
335.                k1 = k1 + 1
336.        p = random.random()
337.        if p<0.5:
338.            routes2 = insert1(noc, routes2, cust, nvinitial)
339.        else:
340.            routes2 = insert2(noc, routes2, cust, nvinitial)
341.        if (function1(routes2) < nvinitial or (function1(routes2) <= nvinitial and function2(routes2) < functio
        n2(routes)))  and routes2.sum() == sumroutes:
342.            routes = np.copy(routes2)
343.        return routes
344.
345.    #steady number of customer selected according to the waiting time
346.    def impwaiting(routeinitial):
347.        nvinitial = function1(routeinitial)
348.        routes1 = np.copy(routeinitial)
349.        routes = converter(routeinitial)
350.        customers5 = np.zeros(shape = (n1,1))
351.        waiting = np.zeros(shape = (n1,1))
352.        toa = np.zeros(shape = (len(routes), n1))
353.        tstart = np.zeros(shape = (len(routes), n1))
354.        tod = np.zeros(shape = (len(routes), n1))
355.        k = 0
356.        for i in range(0, len(routes)):
357.            t = 0
358.            smax = np.count_nonzero(routes[i,:])
359.            for j in range(0, smax+1):
360.                t = t + tt[int(routes[i,j]), int(routes[i, j+1])]
361.                if t < tws[int(routes[i, j+1])]:
362.                    toa[i,j+1] = t
363.                    tstart[i, j+1] = tws[int(routes[i, j+1])]
364.                    waiting[k] = tws[int(routes[i, j+1])] - t
365.                    customers5[k] = int(routes[i, j+1])
366.                    k = k +1
367.                else:
368.                    toa[i,j+1] = t
369.                    tstart[i, j+1] = t
370.                    waiting[k] = 0
371.                    customers5[k] = int(routes[i, j+1])
372.                t = tstart[i, j+1] + st[int(routes[i, j+1])]
373.                tod[i, j+1] = t
374.        wait, custom = zip(*sorted(zip(waiting, customers5), reverse = True))
375.        noc = random.randint(dl,  ul)
376.        cust = np.zeros(shape = (noc, 1))
377.        routes2 = np.zeros(shape=(n1, 1))
378.        k1 = 0
379.        for i in range(n1):
380.            if routes1[i] not in cust:
```

```
            routes2[k1] = routes1[i]
            k1 = k1 + 1
        p = random.random()
        if p<0.9:
            routes2 = insert1(noc, routes2, cust, nvinitial)
        else:
            routes2 = insert2(noc, routes2, cust, nvinitial)
        if function2(routes2) <= function2(routeinitial) and routes2.sum() == sumroutes:
            routeinitial = np.copy(routes2)
        return routeinitial

    def impdistant(routeinitial):
        nvinitial =function1(routeinitial)
        routes1 = np.copy(routeinitial)
        routes = converter(routeinitial)
        wc1 = np.zeros(shape=(n1, 1))
        wc2 = np.zeros(shape=(n1, 1))
        k = 0
        for i in range(len(routes)):
            sk = np.count_nonzero(routes[i,:])
            for j in range(1,sk+1):
                wc1[k] = routes[i,j]
                wc2[k] = (dm[int(routes[i,j-1]), int(routes[i,j])] + dm[int(routes[i,j]), int(routes[i,j+1])])
                k = k + 1
        dmt, custom = zip(*sorted(zip(wc2, wc1), reverse = True))
        noc = random.randint(dl, ul)
        cust = np.zeros(shape = (noc, 1))
        routes2 = np.zeros(shape=(n1, 1))
        k1 = 0
        for i in range(n1):
            if routes1[i] not in cust:
                routes2[k1] = routes1[i]
                k1 = k1 + 1
        p = random.random()
        if p<0.9:
            routes2 = insert1(noc, routes2, cust, nvinitial)
        else:
            routes2 = insert2(noc, routes2, cust, nvinitial)
        if function2(routes2) < function2(routeinitial) and routes2.sum() == sumroutes:
            routeinitial = np.copy(routes2)
        return routeinitial

    #steady number of customer selected according to the waiting time
    def impmixed(routeinitial):
        nvinitial = function1(routeinitial)
        routes1 = np.copy(routeinitial)
        routes = converter(routeinitial)
        customers5 = np.zeros(shape = (n1,1))
        waiting = np.zeros(shape = (n1,1))
        toa = np.zeros(shape = (len(routes), n1))
        tstart = np.zeros(shape = (len(routes), n1))
        tod = np.zeros(shape = (len(routes), n1))
        k = 0
        for i in range(0, len(routes)):
            t = 0
            smax = np.count_nonzero(routes[i,:])
```

```
437.            for j in range(0, smax+1):
438.              t = t + tt[int(routes[i,j]), int(routes[i, j+1])]
439.              if t < tws[int(routes[i, j+1])]:
440.                toa[i,j+1] = t
441.                tstart[i, j+1] = tws[int(routes[i, j+1])]
442.                waiting[k] = tws[int(routes[i, j+1])] - t
443.                customers5[k] = int(routes[i, j+1])
444.                k = k +1
445.              else:
446.                toa[i,j+1] = t
447.                tstart[i, j+1] = t
448.                waiting[k] = 0
449.                customers5[k] = int(routes[i, j+1])
450.              t = tstart[i, j+1] + st[int(routes[i, j+1])]
451.              tod[i, j+1] = t
452.          custom, wait = zip(*sorted(zip(customers5, waiting), reverse = False))
453.          wc1 = np.zeros(shape=(n1, 1))
454.          wc2 = np.zeros(shape=(n1, 1))
455.          k = 0
456.          for i in range(len(routes)):
457.            sk = np.count_nonzero(routes[i,:])
458.            for j in range(1,sk+1):
459.              wc1[k] = routes[i,j]
460.              wc2[k] = (dm[int(routes[i,j-1]), int(routes[i,j])] + dm[int(routes[i,j]), int(routes[i,j+1])])
461.              k = k + 1
462.          custom, dmt  = zip(*sorted(zip(wc1, wc2), reverse = False))
463.          total = np.zeros(shape=(n1, 1))
464.          for i in range(n1):
465.            total[i] = dmt[i] + wait[i]
466.          total2, custom2 = zip(*sorted(zip(total, custom), reverse = True))
467.          noc = random.randint(dl, ul)
468.          cust = np.zeros(shape = (noc, 1))
469.          k = 0
470.          for i in range(0, noc):
471.            cust[k] = int(custom2[i])
472.            k = k + 1
473.          routes2 = np.zeros(shape=(n1, 1))
474.          k1 = 0
475.          for i in range(n1):
476.            if routes1[i] not in cust:
477.              routes2[k1] = routes1[i]
478.              k1 = k1 + 1
479.          p = random.random()
480.          if p<0.9:
481.            routes2 = insert1(noc, routes2, cust, nvinitial)
482.          else:
483.            routes2 = insert2(noc, routes2, cust, nvinitial)
484.          if function2(routes2) <= function2(routeinitial) and routes2.sum() == sumroutes:
485.            routeinitial = np.copy(routes2)
486.          return routeinitial
487.
488.      def imprandom(routes):
489.          nvinitial = function1(routes)
490.          routes1 = np.copy(routes)
491.          noc = random.randint(dl, ul)
492.          cust = np.zeros(shape = (noc, 1))
```

```
493.        k = 0
494.        while np.count_nonzero(cust)<noc:
495.            ranc = random.randint(0,  n1-1)
496.            while ranc in cust:
497.                ranc = random.randint(0,  n1-1)
498.            cust[k] = ranc
499.            k = k + 1
500.        routes2 = np.zeros(shape=(n1, 1))
501.        k1 = 0
502.        for i in range(n1):
503.            if routes1[i] not in cust:
504.                routes2[k1] = routes1[i]
505.                k1 = k1 + 1
506.        p = random.random()
507.        if p<0.9:
508.            routes2 = insert1(noc, routes2, cust, nvinitial)
509.        else:
510.            routes2 = insert2(noc, routes2, cust, nvinitial)
511.        if function2(routes2) < function2(routes) and routes2.sum() == sumroutes:
512.            routes = np.copy(routes2)
513.        return routes
514.
515.    i = 0
516.    iterations = ...
517.    time_limit = ...
518.    while i in range(iterations) and timeit.default_timer() - start < time_limit:
519.        initialrouteimp = np.copy(routeimp)
520.        kk = np.count_nonzero(f1)
521.        for kk1 in range(1):
522.            routeimp = pareto[kk1, :]
523.            routeimp = impnv1(routeimp)
524.            f1, f2, pareto = paretocheck(routeimp,f1,  f2, pareto)
525.        for kk1 in range(1, kk):
526.            routeimp = pareto[kk1, :]
527.            routeimp = impnv2(routeimp)
528.            f1, f2, pareto = paretocheck(routeimp,f1,  f2, pareto)
529.            routeimp = imprandom(routeimp)
530.            f1, f2, pareto = paretocheck(routeimp,f1,  f2, pareto)
531.            routeimp = impdistant(routeimp)
532.            f1, f2, pareto = paretocheck(routeimp,f1,  f2, pareto)
533.            print(np.trim_zeros(f1),  np.trim_zeros(f2),   i, (timeit.default_timer() - start)//1 )
534.            if i%6 == 0:
535.                f1, f2, pareto = par(f1, f2, pareto)
536.        i = i + 1
537.    stop = timeit.default_timer()
```

Appendix C Supplement to Chapter 4

The new optimal solution so far, proposed by the EA algorithm in Solomon's benchmark instances are:

Table C.1 New Optimal Solution in R202 Instance

Number of Route	Route Sequence
1	0-69-1-30-90-10-32-70-31-52-0
2	0-45-47-36-63-64-11-19-62-88-7-18-6-99-96-95-97-43-2-13-58-0
3	0-92-37-42-15-14-38-44-16-61-86-84-83-8-82-48-49-46-17-60-5-93-85-91-100-98-59-94-89-0
4	0-27-50-33-65-71-29-76-3-79-78-81-9-51-20-66-35-34-68-24-80-77-12-0
5	0-28-26-39-67-23-72-73-21-40-53-87-57-41-22-75-56-4-0
Total Distance	1014.96

Table C.2 New Optimal Solution in R210 Instance

Number of Route	Route Sequence
1	0-2-57-15-42-14-44-38-86-16-61-99-59-87-43-41-22-74-56-4-55-25-54-26-58-0
2	0-95-92-98-85-5-45-36-47-48-82-7-88-62-19-11-64-49-46-8-84-17-83-60-94-13-97-37-100-91-93-96-0
3	0-27-69-1-30-65-71-33-50-76-3-79-81-9-34-78-29-24-80-68-77-0
4	0-28-12-39-67-23-75-72-73-21-40-53-6-89-18-52-31-10-63-90-20-51-35-0
Total Distance	920.13

The new solutions proposed in Salhi and Nagy's benchmark instances are:

Table C.3 New Optimal Solution in CMT2X Instance

Number of Route	Route Sequence
1	0-8-35-53-14-19-54-13-57-15-5-37-20-70-60-71-36-69-61-22-0
2	0-58-10-38-65-11-66-59-0
3	0-26-67-34-68-73-62-28-74-21-47-48-30-0
4	0-4-29-45-27-52-46-7-72-39-9-12-0
5	0-2-33-63-1-43-42-64-41-56-23-24-49-16-0
6	0-75-6-51-17-40-3-44-32-50-18-25-55-31-0
Total Distance	771.01

Table C.4 New Optimal Solution in CMT5X Instance

Number of Route	Route Sequence
1	0-46-11-2-20-35-36-3-28-22-0
2	0-45-33-39-10-30-34-50-9-49-0
3	0-5-38-16-29-21-31-26-7-23-6-0
4	0-32-1-27-48-8-43-24-14-25-13-18-0
5	0-12-47-17-37-15-44-42-19-40-41-4-0
Total Distance	606.33

Table C.5 New Optimal Solution in CMT7X Instance

Number of Route	Route Sequence
1	0-75-4-67-34-46-52-27-13-0
2	0-26-12-40-17-51-6-68-2-0
3	0-7-35-8-54-19-14-53-11-0
4	0-45-29-48-70-47-21-74-28-30-0
5	0-33-73-1-43-41-42-22-62-0
6	0-16-63-23-56-49-24-31-10-0
7	0-38-65-66-59-58-25-55-18-0
8	0-3-44-32-50-9-39-72-71-60-20-0
9	0-57-15-5-37-36-69-61-64-0
Total Distance	1040.67

Table C.6 New Optimal Solution in CMT8X Instance

Number of Route	Route Sequence
1	0-94-59-85-93-95-13-87-97-100-98-37-92-91-0
2	0-27-28-68-79-81-9-71-66-65-51-53-58-0
3	0-40-26-12-76-50-77-3-33-34-78-89-5-0
4	0-96-99-61-16-44-86-17-84-83-18-30-32-90-10-62-52-0
5	0-28-26-39-67-23-72-73-21-40-53-87-57-41-22-75-56-4-0
6	0-2-57-42-15-14-38-43-41-22-23-75-72-0
7	0-73-74-21-56-39-67-25-55-80-88-63-64-11-0
8	0-82-48-47-49-36-46-19-7-60-8-45-0
Total Distance	1036.52

Table C.7 New Optimal Solution (a) in CMT9X Instance

Number	Route Sequence

of Route	
1	0-112-147-6-94-95-59-93-85-98-37-92-00-75-4-67-34-46-52-27-13-0
2	0-97-117-53-105-26-149-109-12-132-69-1-30-62-0
3	0-27-146-52-127-31-88-148-10-108-90-32-111-58-40-0
4	0-21-73-72-74-133-22-75-56-139-110-28-89-18-0
5	0-118-60-5-84-8-17-113-86-141-13-137-87-144-57-115-0
6	0-2-145-41-15-96-99-104-100-138-0
7	0-119-44-16-14-142-42-43-116-77-79-129-76-0
8	0-33-81-102-50-68-101-70-122-20-51-9-120-0
9	0-135-35-103-106-7-82-48-124-47-36-143-49-0
10	0-150-80-134-24-29-121-34-78-3-54-83-114-46-123-19-107-0
11	0-130-55-25-39-23-67-4-91-61-45-0
12	0-38-140-125-11-63-126-64-131-128-71-136-65-66-0
Total Distance	1744.19

Table C.8 New Optimal Solution (b) in CMT9X Instance

Number of Route	Route Sequence
1	0-112-147-94-95-59-93-85-98-37-92-97-0
2	0-89-117-53-105-26-149-109-12-0
3	0-132-69-1-30-62-52-127-31-88-148-10-108-90-32-111-0
4	0-58-40-21-73-72-74-133-22-75-56-139-110-0
5	0-28-27-146-18-118-60-5-84-8-6-0
6	0-17-113-86-141-13-137-87-144-57-115-0
7	0-2-145-41-15-96-99-104-100-138-0
8	0-119-44-16-14-142-42-43-116-77-79-129-76-0
9	0-33-81-102-50-68-101-70-122-20-51-9-120-0
10	0-135-35-103-106-7-82-48-124-47-36-143-49-0
11	0-150-80-134-24-29-121-34-78-3-54-83-114-46-123-19-107-0
12	0-130-55-25-39-23-67-4-91-61-45-0
13	0-38-140-125-11-63-126-64-131-128-71-136-65-66-0
Total Distance	1741.36

Table C.9 New Optimal Solution (c) in CMT9X Instance

Number of Route	Route Sequence
1	0-112-94-95-59-85-93-98-100-37-92-0
2	0-97-117-137-53-105-26-149-12-109-68-150-80-116-0
3	0-40-76-77-102-27-146-0

4	0-89-147-6-96-99-104-118-5-84-61-28-111-138-0
5	0-50-1-132-69-101-32-90-31-127-52-0
6	0-13-87-144-57-115-2-145-41-22-133-0
7	0-21-73-72-74-75-56-39-139-4-110-58-18-106-0
8	0-7-82-124-47-123-19-107-48-60-83-0
9	0-114-8-125-45-17-113-16-141-88-148-62-10-108-0
10	0-131-128-20-30-79-129-120-81-3-0
11	0-9-103-71-135-33-70-122-51-66-65-0
12	0-136-35-78-54-130-134-24-29-121-34-0
13	0-42-142-14-119-44-91-86-15-43-38-140-46-0
14	0-55-25-67-23-126-63-11-64-49-143-36-0
Total Distance	1698.21

Table C.10 New Optimal Solution in CMT10X Instance

Number of Route	**Route Sequence**
1	0-112-156-183-94-95-59-85-93-98-92-91-0
2	0-27-167-190-31-162-101-176-102-157-81-120-0
3	0-53-105-28-111-196-68-150-80-177-170-12-184-154-138-0
4	0-152-58-40-13-87-97-151-100-193-141-44-119-142-15-147-0
5	0-89-166-118-5-84-173-61-16-191-192-37-0
6	0-146-52-127-88-182-148-159-62-10-189-108-0
7	0-180-149-26-195-130-134-163-24-132-69-1-0
8	0-50-76-116-77-158-3-79-129-169-55-109-0
9	0-6-96-99-104-117-137-178-115-2-144-172-42-0
10	0-18-153-106-194-7-82-48-123-19-107-175-0
11	0-198-21-72-197-75-133-22-171-74-73-110-179-0
12	0-155-4-139-39-187-165-60-83-199-114-8-174-125-45-17-113-0
13	0-70-30-160-128-188-161-103-9-164-34-33-0
14	0-185-78-135-35-136-71-66-20-122-51-0
15	0-32-131-181-63-126-90-57-145-41-186-56-23-67-0
16	0-121-29-25-54-65-11-124-47-168-36-143-49-64-0
17	0-43-14-38-140-86-46-0
Total Distance	1937.02

Besides the solutions in instances of well known datasets of the literature, Appendix C also includes the MOEA algorithm that is presented in Chapter 4, and is given in the Python programming language.

```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import timeit
import math

start = timeit.default_timer()

#Maximum Number of Generations
max_gen = 100
#Population Size
pop_size = 30
step1 = 0.05
pososto1 = 0.25
pososto2 = 0.8
pososto3 = 0.9

filename = "r109.xlsx"
dataset = pd.read_excel(filename)
tt = pd.read_excel("travel_times_matrix.xlsx")
dm = pd.read_excel("distance_matrix.xlsx")

def totq(route, demand):
    s = np.count_nonzero(route)
    q=0
    for j in range(1, s+1):
        q = q + demand[int(route[j])]
    return q

def totcroute(route, dm):
    td = 0
    s = np.count_nonzero(route)
    for j in range(0, s+1):
        td = td + dm[int(route[j]),int(route[j+1])]
    return td

#Random customers
def remove1(routes, dl, ul, n1):
    noc = random.randint(dl, ul)
    cust = np.zeros(shape = (noc, 1))
    k = 0
    while np.count_nonzero(cust)<noc:
        ranc = random.randint(0, n1-1)
        while ranc in cust:
            ranc = random.randint(0, n1-1)
        cust[k] = ranc
        k = k + 1
    return cust

```

```
51.     #select entire route
52.     def remove2n(routes, demand, n, capacity, tt, twe, tws, st, dm):
53.       routesc = converter(routes, n, demand,capacity, tt, twe, tws, st)
54.       r1 = np.zeros(shape=(len(routes), 1))
55.       r2 = np.zeros(shape=(len(routes), 1))
56.       for i in range(len(routes)):
57.         r1[i] = i
58.         r2[i] = totcroute(routesc[i, :], dm)/totq(routesc[i, :], demand)
59.       rrr2, rrr1 = zip(*sorted(zip(r2, r1), reverse = True))
60.       randomnumb = random.randint(0, 6)
61.       ii = int(rrr1[randomnumb])
62.       noc = np.count_nonzero(routesc[ii, :])
63.       while noc ==0:
64.         randomnumb = random.randint(0, 6)
65.         ii = int(rrr1[randomnumb])
66.         noc = np.count_nonzero(routesc[ii, :])
67.       cust = np.zeros(shape = (noc, 1))
68.       return cust
69.
70.     #distant customers
71.     def remove3(routes, n1, dm, dl, ul, n, demand,capacity, tt, twe, tws, st):
72.       wc1 = np.zeros(shape=(n1, 1))
73.       wc2 = np.zeros(shape=(n1, 1))
74.       routes1 = converter(routes, n, demand,capacity, tt, twe, tws, st)
75.       k = 0
76.       nvv = function1(routes, n, demand,capacity, tt, twe, tws, st)
77.       for i in range(nvv):
78.         sk = np.count_nonzero(routes1[i,:])
79.         for j in range(1,sk+1):
80.           wc1[k] = routes1[i,j]
81.           wc2[k] = (dm[int(routes1[i,j-1]), int(routes1[i,j])] + dm[int(routes1[i,j]), int(routes1[i,j+1])])
82.           k = k + 1
83.       dem, custom = zip(*sorted(zip(wc2, wc1), reverse = True))
84.       noc = random.randint(dl, ul)
85.       cust = np.zeros(shape = (noc, 1))
86.       for i in range(noc):
87.         cust[i] = int(custom[i])
88.       return cust
89.
90.     #waiting time at customers
91.     def remove4(routeinitial, n1, tws, twe, tt, st, dl, ul, demand, capacity, n):
92.       routes = converter(routeinitial, n, demand, capacity, tt, twe, tws, st)
93.       customers5 = np.zeros(shape = (n1,1))
94.       waiting = np.zeros(shape = (n1,1))
95.       toa = np.zeros(shape = (len(routes), n1))
96.       tstart = np.zeros(shape = (len(routes), n1))
97.       tod = np.zeros(shape = (len(routes), n1))
98.       k = 0
99.       for i in range(0, len(routes)):
100.        t = 0
101.        smax = np.count_nonzero(routes[i,:])
102.        for j in range(0, smax+1):
103.          t = t + tt[int(routes[i,j]), int(routes[i, j+1])]
104.          if t < tws[int(routes[i, j+1])]:
105.            toa[i,j+1] = t
106.            tstart[i, j+1] = tws[int(routes[i, j+1])]
```

```
107.                waiting[k] = tws[int(routes[i, j+1])] - t
108.                customers5[k] = int(routes[i, j+1])
109.                k = k +1
110.            else:
111.                toa[i,j+1] = t
112.                tstart[i, j+1] = t
113.                waiting[k] = 0
114.                customers5[k] = int(routes[i, j+1])
115.            t = tstart[i, j+1] + st[int(routes[i, j+1])]
116.            tod[i, j+1] = t
117.    wait, custom = zip(*sorted(zip(waiting, customers5), reverse = True))
118.    noc = random.randint(dl, ul)
119.    cust = np.zeros(shape = (noc, 1))
120.    k = 0
121.    for i in range(0, noc):
122.        cust[k] = int(custom[i])
123.        k = k + 1
124.    return cust
125.
126. def insert1(routes2, cust, nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st):
127.    s1 = len(cust)
128.    while cust.sum() != 0:
129.        i = random.randint(0, s1-1)
130.        while cust[i] == 0:
131.            i = random.randint(0, s1-1)
132.        customerchosen = int(cust[i])
133.        cust[i] = 0
134.        initialcost = 99999999
135.        sk = np.count_nonzero(routes2)
136.        routes2n = np.copy(routes2)
137.        new = 0
138.        www = False
139.        while new in range(0, sk):
140.            routes2n = np.copy(routes2)
141.            for j in range(0, new):
142.                routes2n[j] = routes2[j]
143.            routes2n[new] = customerchosen
144.            for j in range(new, n1-1):
145.                routes2n[j+1] = routes2[j]
146.            if function2(routes2n, n , dm, demand, capacity, tt, twe, tws, st) <= initialcost and function1(rou
    tes2n, n, demand,capacity, tt, twe, tws, st) <= nvinitial:
147.                routesopt = np.copy(routes2n)
148.                initialcost = function2(routes2n, n , dm, demand, capacity, tt, twe, tws, st)
149.                www = True
150.            new = new + 1
151.        if new == sk and www == False:
152.            cust[i] = customerchosen
153.            nvinitial = nvinitial + 1
154.        if www == True:
155.            routes2 = np.copy(routesopt)
156.    return routes2
157.
158. def insert2(routes2, cust, nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st):
159.    s1 = len(cust)
160.    while cust.sum() != 0:
161.        i = random.randint(0, s1-1)
```

```
        while cust[i] == 0:
            i = random.randint(0, s1-1)
        customerchosen = int(cust[i])
        cust[i] = 0
        initialcost = 99999999
        sk = np.count_nonzero(routes2)
        routes2n = np.copy(routes2)
        new = 0
        www = False
        while new in range(0, sk):
            routes2n = np.copy(routes2)
            for j in range(0, new):
                routes2n[j] = routes2[j]
            routes2n[new] = customerchosen
            for j in range(new, n1-1):
                routes2n[j+1] = routes2[j]
            if function2(routes2n, n , dm, demand, capacity, tt, twe, tws, st) <= initialcost:
                routesopt = np.copy(routes2n)
                initialcost = function2(routes2n, n , dm, demand, capacity, tt, twe, tws, st)
                www = True
            new = new + 1
        if www == True:
            routes2 = np.copy(routesopt)
    return routes2


def insert3(routes2, cust, nvinitial, n1, n, dm, demand, capacity, tt, twe, tws, st):
    s1 = len(cust)
    while cust.sum() != 0:
        i = random.randint(0, s1-1)
        while cust[i] == 0:
            i = random.randint(0, s1-1)
        customerchosen = int(cust[i])
        cust[i] = 0
        initialcost = 99999999
        sk = np.count_nonzero(routes2)
        routes2n = np.copy(routes2)
        new = 0
        www = False
        while new in range(0, sk):
            routes2n = np.copy(routes2)
            for j in range(0, new):
                routes2n[j] = routes2[j]
            routes2n[new] = customerchosen
            for j in range(new, n1-1):
                routes2n[j+1] = routes2[j]
            if function2(routes2n, n , dm, demand, capacity, tt, twe, tws, st) <= initialcost and function1(rou
tes2n, n, demand,capacity, tt, twe, tws, st) <= nvinitial - 1:
                routesopt = np.copy(routes2n)
                initialcost = function2(routes2n, n , dm, demand, capacity, tt, twe, tws, st)
                www = True
            new = new + 1
        if new == sk and www == False:
            cust[i] = customerchosen
            nvinitial = nvinitial + 1
        if www == True:
```

```
217.            routes2 = np.copy(routesopt)
218.        return routes2
219.
220.    def converter(route, n, demand,capacity, tt, twe, tws, st):
221.        oprt = np.zeros(shape=(n,n))
222.        k1=0
223.        t=0
224.        q=0
225.        j=0
226.        s=1
227.        nv = 1
228.        k2 = int(route[j])
229.        oprt[nv, s] = int(k1)
230.        while j in range(n-1):
231.            k2 = int(route[j])
232.            q = q + demand[k2]
233.            if q > capacity or t + tt[k1,k2] > twe[k2]:
234.                k1=0
235.                q=0
236.                t=0
237.                s = 1
238.                k2 = int(route[j])
239.                nv = nv + 1
240.                oprt[nv-1, s] = int(k2)
241.            else:
242.                oprt[nv-1, s] = int(k2)
243.                s = s+1
244.                if t + tt[k1,k2] < tws[k2]:
245.                    t = tws[k2] + st[k2]
246.                else:
247.                    t = t + tt[k1, k2] + st[k2]
248.                k1 = int(route[j])
249.                j = j + 1
250.        return oprt
251.
252.    #First function to optimize - Total Number of Vehicles
253.    def function1(route, n, demand,capacity, tt, twe, tws, st):
254.        cch = converter(route , n, demand,capacity, tt, twe, tws, st)
255.        chrom_nv = 0
256.        for i in range(n):
257.            if cch[i, 1] != 0:
258.                chrom_nv = i+1
259.        return chrom_nv
260.
261.    #Second function to optimize - Total travelled distance
262.    def function2(route, n , dm, demand,capacity, tt, twe, tws, st):
263.        nov1 = function1(route, n, demand,capacity, tt, twe, tws, st)
264.        cch = converter(route, n, demand,capacity, tt, twe, tws, st)
265.        td = 0
266.        for i in range(nov1):
267.            for j in range(n - 1):
268.                td = td + dm[int(cch[i, j]), int(cch[i, j+1])]
269.        return td
270.
271.    def afairesi(chrom, x, n):
272.        child = np.zeros(shape=(n-1, 1))
```

```
273.        k1=0
274.        for i in range(n-1):
275.            if chrom[i] not in x:
276.                child[k1] = chrom[i]
277.                k1 = k1 +1
278.        return child
279.
280.    def crossover1(chromo1, chromo2, demand, n1, sumroutes, n, dl, ul, tws, twe, tt, st, dm, capacity):
281.        r2=random.random()
282.        ss = False
283.        if r2<pososto1:
284.            x2 = remove1(chromo2,  dl, ul, n1)
285.            ss = True
286.        elif r2<pososto2:
287.            x2 = remove2n(chromo2,  demand, n, capacity, tt, twe, tws, st, dm)
288.        elif r2<pososto3:
289.            x2 = remove3(chromo2,  n1, dm, dl, ul, n, demand,capacity, tt, twe, tws, st)
290.            ss = True
291.        else:
292.            x2 = remove4(chromo2,  n1, tws, twe, tt, st, dl, ul, demand, capacity, n)
293.            ss = True
294.        child1 = afairesi(chromo1, x2, n)
295.        if ss == False:
296.            r2 = random.random()
297.            if r2 < 0.1:
298.                nvinitial = function1(chromo1,  n, demand,capacity, tt, twe, tws, st)
299.                chromo1x = insert1(child1, x2,  nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
300.            else:
301.                nvinitial = function1(chromo1,  n, demand,capacity, tt, twe, tws, st)
302.                chromo1x = insert3(child1, x2,  nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
303.        else:
304.            r2 = random.random()
305.            if r2 < 0.5:
306.                nvinitial = function1(chromo1,  n, demand,capacity, tt, twe, tws, st)
307.                chromo1x = insert1(child1, x2,  nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
308.            else:
309.                nvinitial = function1(chromo1,  n, demand,capacity, tt, twe, tws, st)
310.                chromo1x = insert2(child1, x2,  nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
311.        r3 =  random.random()
312.        if r3 <= 0.05:
313.            cust11 = np.zeros(shape = (1, 1))
314.            cust11[0] = random.randint(0,  n1-1)
315.            meion1 = afairesi(chromo1x, cust11, n)
316.            nvinitial  = function1(meion1,  n, demand,capacity, tt, twe, tws, st)
317.            chromo1x = insert1(meion1, cust11, nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
318.        if chromo1x.sum() <sumroutes:
319.            chromo1x = np.copy(chromo1)
320.        return chromo1x
321.
322.    def crossover2(chromo1, chromo2, demand, n1, sumroutes, n, dl, ul, dm, tws, twe, tt, st, capacity):
323.        r2=random.random()
324.        ss = False
325.        if r2<pososto1:
326.            x2 = remove1(chromo1,  dl, ul, n1)
327.            ss = True
328.        elif r2<pososto2:
```

```
329.            #x2 = remove2(chromo1)
330.            x2 = remove2n(chromo1,  demand, n, capacity, tt, twe, tws, st, dm)
331.        elif r2<pososto3:
332.            ss = True
333.            x2 = remove3(chromo1,  n1, dm, dl, ul, n, demand,capacity, tt, twe, tws, st)
334.        else:
335.            x2 = remove4(chromo1,  n1, tws, twe, tt, st, dl, ul, demand, capacity, n)
336.            ss = True
337.        child1 = afairesi(chromo2, x2, n)
338.        if ss ==False:
339.            r2 = random.random()
340.            if r2 < 0.1:
341.                nvinitial = function1(chromo2,  n, demand,capacity, tt, twe, tws, st)
342.                chromo2x  = insert1(child1, x2,  nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
343.            else:
344.                nvinitial = function1(chromo2,  n, demand,capacity, tt, twe, tws, st)
345.                chromo2x  = insert3(child1, x2,  nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
346.        else:
347.            r2 = random.random()
348.            if r2 < 0.5:
349.                nvinitial = function1(chromo1,  n, demand,capacity, tt, twe, tws, st)
350.                chromo2x  = insert1(child1, x2,  nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
351.            else:
352.                nvinitial = function1(chromo1,  n, demand,capacity, tt, twe, tws, st)
353.                chromo2x  = insert2(child1, x2,  nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
354.        r3 =  random.random()
355.        if r3 <= 0.05:
356.            cust11 = np.zeros(shape = (1, 1))
357.            cust11[0] = random.randint(0,  n1-1)
358.            meion1 = afairesi(chromo2x, cust11, n)
359.            nvinitial  = function1(meion1,  n, demand,capacity, tt, twe, tws, st)
360.            chromo2x = insert1(meion1, cust11, nvinitial, n1, n , dm, demand, capacity, tt, twe, tws, st)
361.        if chromo2x.sum() <sumroutes:
362.            chromo2x  = np.copy(chromo2)
363.        return  chromo2x
364.
365.    #Function to carry out NSGA-II's fast non dominated sort
366.    def fast_non_dominated_sort(values1, values2):
367.        S=[[] for i in range(0,len(values1))]
368.        front  = [[]]
369.        n=[0 for i in range(0,len(values1))]
370.        rank = [0 for i in range(0, len(values1))]
371.
372.        for p in range(0,len(values1)):
373.            S[p]=[]
374.            n[p]=0
375.            for q in range(0, len(values1)):
376.                if (values1[p] < values1[q] and values2[p] < values2[q]) or (values1[p] <= values1[q] and values2
    [p] < values2[q]) or (values1[p] < values1[q] and values2[p] <= values2[q]):
377.                    if q not in S[p]:
378.                        S[p].append(q)
379.                elif (values1[q] < values1[p] and values2[q] < values2[p]) or (values1[q] <= values1[p] and value
    s2[q] < values2[p]) or (values1[q] < values1[p] and values2[q] <= values2[p]):
380.                    n[p] = n[p] + 1
381.            if n[p]==0:
382.                rank[p] = 0
```

```
383.                if p not in front[0]:
384.                    front[0].append(p)
385.        i = 0
386.        while(front[i] != []):
387.            Q=[]
388.            for p in front[i]:
389.                for q in S[p]:
390.                    n[q] =n[q] - 1
391.                    if( n[q]==0):
392.                        rank[q]=i+1
393.                        if q not in Q:
394.                            Q.append(q)
395.            i = i+1
396.            front.append(Q)
397.        del front[len(front)-1]
398.        return front
399.
400.    def crowding_distance(values1, values2, front):
401.        distance = [0 for i in range(0,len(front))]
402.        sorted1 = sort_by_values(front, values1[:])
403.        sorted2 = sort_by_values(front, values2[:])
404.        distance[0] = 9999999999999999
405.        distance[len(front) - 1] = 9999999999999999
406.        for k in range(1,len(front)-1):
407.            distance[k] = distance[k]+ (values1[sorted1[k+1]] - values2[sorted1[k-1]])/(max(values1)-
        min(values1))
408.        for k in range(1,len(front)-1):
409.            distance[k] = distance[k]+ (values1[sorted2[k+1]] - values2[sorted2[k-1]])/(max(values2)-
        min(values2))
410.        return distance
411.
412.    def sort_by_values(list1, values):
413.        sorted_list = []
414.        while(len(sorted_list)!=len(list1)):
415.            if index_of(min(values),values) in list1:
416.                sorted_list.append(index_of(min(values),values))
417.            values[index_of(min(values),values)] = math.inf
418.        return sorted_list
419.
420.    #Function to find ton deikti tis listas
421.    def index_of(a,list):
422.        for i in range(0,len(list)):
423.            if list[i] == a:
424.                return i
425.        return -1
426.
427.    def routing(dataset, tt, dm):
428.        #-------------------------seperating the dataset in arrays and matrices----------------------------------
429.        cx = dataset.iloc[:,1].values
430.        cy = dataset.iloc[:,2].values
431.        capacity = int(dataset.iloc[0,7])
432.        print(capacity, filename)
433.        demand  = dataset.iloc[:,3].values
434.        tws = dataset.iloc[:, 4].values
435.        twe = dataset.iloc[:, 5].values
436.        st = dataset.iloc[:, 6].values
```

```
437.      nodes = dataset.iloc[:,0].values
438.      n=len(dataset)
439.      n1 = n - 1
440.
441.      customers1 = np.copy(nodes)
442.      dl = 5
443.      ul = 18
444.      sumroutes = customers1.sum()
445.    #-----attributes of the normalization procedure -----
446.      minv = int(demand.sum()//capacity)
447.      partstw = (twe.max() + tws.min())/2
448.      partsdm = (dm.max() + dm.min())/2
449.      categ = dm[:,:] / partsdm
450.      norm = partstw/partsdm
451.      chrom = np.zeros(shape=(5151, n1))
452.      pos  = np.zeros(shape=(5151, 1))
453.
454.      num_chrom = 0
455.      routeimp = np.zeros(shape=(1,n))
456.      #--------- starting to explore solutions ----------
457.      for w1 in np.arange(0.0, 1.00001, step1):
458.        for w2 in np.arange(0.0, 1.0001-w1, step1):
459.          w3 = 1 - w1 - w2
460.          customers = np.copy(customers1)
461.          s_chrom = 0
462.          q = True
463.          s = 0
464.          i = 0
465.          while q:
466.            if i ==0:
467.              t = 0
468.              sumq = 0
469.              k1 = 0
470.            mindt = 1000000
471.            dok = 0
472.            minf = mindt
473.            for j in range(1, n):
474.              if customers[j] != 0 and sumq + demand[int(customers[j])] <= capacity and t + tt[i,int(custo
    mers[j])] <= twe[int(customers[j])]:
475.                minf = w1*dm[i,int(customers[j])]*norm  + w2*abs(t +tt[i,int(customers[j])] - tws[int(cust
    omers[j])]) +w3*abs(-t-tt[i,int(customers[j])] + twe[int(customers[j])])
476.                if minf < mindt:
477.                  s = j
478.                  mindt = minf
479.                  dok = dok+1
480.            if dok == 0 and j == n1-1:
481.              i = 0
482.              s = 0
483.            if dok != 0:
484.              chrom[num_chrom,  s_chrom] = int(customers[s])
485.              s_chrom = s_chrom + 1
486.            sumq = sumq + demand[int(customers[s])]
487.            if t + tt[i,int(customers[s])] < tws[int(customers[s])]:
488.              t = tws[int(customers[s])] + st[int(customers[s])]
489.            else:
490.              t = t + tt[i, int(customers[s])] + st[int(customers[s])]
```

```
491.                 i = int(customers[s])
492.                 customers[s] = 0
493.                 if customers.sum() == 0:
494.                     break
495.             num_chrom = num_chrom + 1
496.
497.     #Get rid of the duplicate solutions
498.     solutions = np.zeros(shape=(num_chrom, n-1))
499.     solutions[0, :] = chrom[0, :]
500.     kk = 1
501.     for i in range(1, num_chrom):
502.         j = 0
503.         sss = True
504.         while j in range(num_chrom) and sss == True:
505.             if np.array_equal(chrom[i, :], solutions[j, :]) == True:
506.                 sss = False
507.             else:
508.                 j = j + 1
509.         if sss == True:
510.             solutions[kk, :] = chrom[i, :]
511.             kk = kk + 1
512.     solution = np.zeros(shape=(kk - 1, n-1))
513.     for i in range(kk-1):
514.         solution[i, :] = solutions[i, :]
515.     solutionnew = np.zeros(shape=(kk - 1, n-1))
516.     function1_values = [function1(solution[i,:], n, demand,capacity, tt, twe, tws, st) for i in range(0,kk-
        1)]
517.     function2_values = [function2(solution[i,:], n , dm, demand, capacity, tt, twe, tws, st) for i in range(0
        ,kk-1)]
518.     non_dominated_sorted_solution = fast_non_dominated_sort(function1_values[:],function2_values
        [:])
519.     k = 0
520.     i = 0
521.     while k <=pop_size - 1 and i in range(len(non_dominated_sorted_solution)):
522.         j = 0
523.         while k <= pop_size and j in range(len(non_dominated_sorted_solution[i])):
524.             solutionnew[k, :] = solution[non_dominated_sorted_solution[i][j], :]
525.             j = j + 1
526.             k = k + 1
527.         i = i + 1
528.     for i in range(pop_size):
529.         solution[i,:] = solutionnew[i, :]
530.
531.     #Starting the generation
532.     gen_no=0
533.     while(gen_no<max_gen) and timeit.default_timer() - start < 1800:
534.         function1_values = [function1(solution[i,:], n, demand, capacity, tt, twe, tws, st) for i in range(0,p
        op_size)]
535.         function2_values = [function2(solution[i,:], n , dm, demand, capacity, tt, twe, tws, st) for i in range
        (0,pop_size)]
536.         non_dominated_sorted_solution = fast_non_dominated_sort(function1_values[:],function2_valu
        es[:])
537.         print("The best front for Generation number ",gen_no, " is")
538.         for valuez in non_dominated_sorted_solution[0]:
539.             print(function1(solution[valuez], n, demand,capacity, tt, twe, tws, st), function2(solution[valuez
        ], n , dm, demand, capacity, tt, twe, tws, st))
```

```
540.          print((timeit.default_timer()  - start)//1)
541.          crowding_distance_values=[]
542.          for i in range(0,len(non_dominated_sorted_solution)):
543.            crowding_distance_values.append(crowding_distance(function1_values[:],function2_values[:],
      non_dominated_sorted_solution[i][:]))
544.          solution2 = np.zeros(shape=(2*pop_size, n-1))
545.          for i in range(0, pop_size):
546.            solution2[i,:] = solution[i, :]
547.          for i in range(0,pop_size, 2):
548.            s1 = crossover1(solution2[i, :],solution2[i+1, :], demand, n1, sumroutes, n, dl, ul, tws, twe, tt, st,
       dm, capacity)
549.            s2 = crossover2(solution2[i, :], solution2[i+1, :], demand, n1, sumroutes, n, dl, ul, dm, tws, twe,
      tt, st, capacity)
550.            for k in range(n-1):
551.              solution2[i+pop_size, k] = s1[k]
552.              solution2[i + pop_size +1, k] = s2[k]
553.          solutions = np.zeros(shape=(pop_size*2, n-1))
554.          solutions[0, :] = solution2[0, :]
555.          kk = 1
556.          for i in range(1, pop_size*2):
557.            j = 0
558.            sss = True
559.            while j in range(pop_size*2) and sss == True:
560.              if np.array_equal(solution2[i, :], solutions[j, :]) == True:
561.                sss = False
562.                j = pop_size*2
563.              else:
564.                j = j + 1
565.            if sss == True:
566.              solutions[kk, :] = solution2[i, :]
567.              kk = kk + 1
568.          solution = np.zeros(shape=(kk - 1, n-1))
569.          for i in range(kk-1):
570.            solution[i, :] = solutions[i, :]
571.          function1_values2 = [function1(solution[i,:], n, demand,capacity, tt, twe, tws, st)for i in range(0,kk
      -1)]
572.          function2_values2 = [function2(solution[i,:], n , dm, demand, capacity, tt, twe, tws, st)for i in rang
      e(0,kk-1)]
573.          non_dominated_sorted_solution2  = fast_non_dominated_sort(function1_values2[:],function2_va
      lues2[:])
574.          k = 0
575.          i = 0
576.          while k <=pop_size - 1 and i in range(len(non_dominated_sorted_solution2)):
577.            j = 0
578.            while k <= pop_size-1 and j in range(len(non_dominated_sorted_solution2[i])):
579.              solutionnew[k, :] = solution[non_dominated_sorted_solution2[i][j],  :]
580.              k = k + 1
581.              j = j + 1
582.            i = i + 1
583.          solution = np.copy(solutionnew)
584.          for i in range(pop_size):
585.            solution[i,:] = solutionnew[i, :]
586.          gen_no = gen_no + 1
587.        return solution, function1_values, function2_values
588.
589.    solution, function1_values, function2_values = routing(dataset)
```

```
590.
591.      stop = timeit.default_timer()
592.      print('Time: ', stop - start)
```

Appendix D Supplement to Chapter 5

In Appendix D, the analytical results of instance sets that are presented condensed, are given. Therefore, the results of the ALNS algorithm in the FSMVRPTW are analytically presented. Additionally, new best-published solutions that are exploited through the ALNS algorithm are given.

FSMVRPTW

Table D.1, D.2 and D.3 present the analytical results of Table 5.4, according to the cost structure (A, B and C). Additionally, Tables D.4, D.5, D.6, D.7, D.8, D.9 present new-optimal solutions obtained from the proposed ALNS algorithm when tested in the specific instance sets.

Table D.1 Results in Cost Structure A of the FSMVRPTW Dataset

Problem Case	VC	FC	TC	Dev	Composition
R101A	1845.79	2710	4555.79	5.60%	A3, B10, C9, D2
R102A	1834.43	2680	4514.43	8.36%	A6, B6, C10, D2
R103A	1804.85	2670	4380	8.76%	A6, B8, C7, D3
R104A	1510.67	2720	4230.67	7.48%	A5, B4, C10, D3
R105A	1753.22	2670	4423.22	7.29%	A7, B7, C9, D2
R106A	1589.52	2830	4419.52	9.16%	A3, B6, C5, D6
R107A	1637.17	2730	4367.17	9.99%	A7, B5, C7, D4
R108A	1419.64	2860	4279.64	8.95%	A2, B7, C5, D4, E1
R109A	1276.06	3110	4386.06	9.22%	A3, B5, C4, D6, E1
R110A	1530.26	2730	4260.26	7.54%	A5, B8, C6, D4
R111A	1575.72	2750	4325.72	9.10%	A5, B3, C9, D4
R112A	1563.52	2690	4253.52	8.54%	A3, B7, C7, D4
C101A	1110.71	6500	7610.71	7.29%	A11, B4
C102A	1190.58	6300	7490.58	5.80%	A13, B3
C103A	1180.35	6900	8080.35	14.14%	A7, B6
C104A	1076.09	6500	7576.09	7.08%	A11, B4
C105A	983.66	6900	7883.66	11.14%	A7, B6
C106A	1095.56	6700	7795.56	10.05%	A9, B5
C107A	1173.77	6700	7873.77	11.14%	A9, B5
C108A	1348.63	6100	7448.63	5.21%	A15, B2
C109A	1342.36	6300	7642.36	7.98%	A13, B3
RC101A	2304.91	3330	5634.91	9.40%	A13, B9, C4
RC102A	2028.43	3390	5418.43	8.65%	A9, B10, C3, D1
RC103A	1895.88	3360	5255.88	9.39%	A11, B7, C4, D1
RC104A	1387.67	3540	4927.67	4.45%	A4, B2, C7, D2
RC105A	2222.49	3240	5462.49	8.48%	A9, B10, C4
RC106A	1826	3360	5186	5.05%	A6, B8, C6

RC107A	1951.61	3330	5281.61	10.29%	A8, B10, C3, D1
RC108A	1391.83	3510	4901.83	4.10%	A1, B6, C7, D1
R201A	1320.25	2250	3570.25	3.58%	A5
R202A	1197.48	2250	3447.48	4.55%	A5
R203A	1035.98	2250	3285.98	4.61%	A5
R204A	810.23	2250	3060.23	1.39%	A5
R205A	1104.79	2250	3354.79	4.22%	A5
R206A	907.13	2550	3457.13	9.88%	A1, B3
R207A	840.65	2550	3390.65	10.18%	A1, B3
R208A	820.96	2250	3070.96	2.46%	A5
R209A	808.09	2500	3308.09	6.04%	A5
R210A	769.18	2500	3269.18	3.12%	A5
R211A	749.05	2500	3249.05	7.59%	A5
C201A	771.43	5000	5771.43	1.34%	A5
C202A	854.47	5000	5854.47	2.98%	A5
C203A	655.3	5200	5855.3	3.06%	A1, B3
C204A	698.52	5200	5898.52	3.89%	A1, B3
C205A	723.57	5400	6123.57	7.59%	A4, B1
C206A	799.02	5400	6199.02	8.96%	A4, B1
C207A	650.07	5400	6050.07	6.38%	A4, B1
C208A	735.31	5400	6135.31	7.89%	A4, B1
RC201A	1732.82	3050	4782.82	9.34%	A6, B3, C2
RC202A	1485.42	3100	4585.42	8.03%	A7, B2, C1, D1
RC203A	989.42	3250	4239.42	1.66%	A8, B2, C1, D1
RC204A	987.72	3300	4287.72	6.50%	A6, B, C1, D1
RC205A	941.04	3600	4541.04	5.80%	A8, B3, C1, D1
RC206A	1402.36	3050	4452.36	4.72%	A6, B3, C2
RC207A	1124.78	3250	4374.78	4.60%	A8, B2, C1, D1
RC208A	989.56	3250	4239.56	4.04%	A8, B2, C1, D1

Table D.2 Results in Cost Structure B of the FSMVRPTW Dataset

Problem Case	**VC**	**FC**	**TC**	**Dev**	**Composition**
R101B	1799.99	544	2343.99	5.46%	A2,B9, C10, D2
R102B	1670.3	584	2254.3	10.07%	A4, B3, C7, D6
R103B	1394.08	644	2038.08	9.95%	A1, B1, C6, D5, E2
R104B	1110.97	676	1786.97	6.16%	A2, B0, C2, D4, E4
R105B	1484.41	650	2134.41	7.75%	A4, B3, C4, D5, E2
R106B	1408.42	678	2086.42	10.49%	A4, B2, C2, D5, E3
R107B	1255.68	670	1925.68	9.91%	A1, B3, C4, D4, E3

R108B	1217.66	620	1837.66	11.52%	A1, B3, C4, D3, E3
R109B	1342.79	624	1966.79	8.18%	A3, B2, C4, D5, E2
R110B	1374.26	574	1948.26	10.78%	A2, B4, C5, D3, E2
R111B	1308.33	602	1910.33	9.73%	A2, B3, C3, D9
R112B	1137.85	726	1863.85	12.16%	A1, B2, C3, D2, E5
C101B	1140.06	1340	2480.06	5.98%	A9, B5
C102B	1136.48	1340	2476.48	6.48%	A9, B5
C103B	1137.26	1300	2437.26	4.85%	A11, B4
C104B	1177.5	1340	2517.5	8.60%	A9, B5
C105B	1081.6	1380	2461.6	5.19%	A7, B6
C106B	1168.68	1300	2468.68	5.49%	A11, B4
C107B	1008.82	1460	2468.82	5.50%	A3, B8
C108B	1185.55	1300	2485.55	6.28%	A11, B4
C109B	1127.57	1390	2517.57	8.12%	A8, B4, C1
RC101B	1991.47	414	2405.47	-0.08%	A12, B7, C1
RC102B	1673.27	396	2069.27	-6.53%	A3, B12
RC103B	1508.6	450	1958.6	-2.83%	A5, B5, C4
RC104B	1485.47	414	1899.47	0.16%	A2, B7, C3
RC105B	1829.33	414	2243.33	-1.36%	A7, B9, C1
RC106B	1596.62	408	2004.62	-5.98%	A4, B10, C1
RC107B	1565.53	396	1961.53	-1.17%	A3, B10, C1
RC108B	1433.63	486	1919.63	1.25%	A3, B6, C3, D1
R201B	1398.09	500	1898.09	15.26%	A4, B1
R202B	1208.43	510	1718.43	14.42%	A1, B3
R203B	1050.76	450	1500.76	11.91%	A5
R204B	831.64	500	1331.64	9.32%	A4, B1
R205B	996.95	500	1496.95	5.50%	A4, B1
R206B	958.9	560	1518.9	12.82%	A2, B1, C1
R207B	885.25	560	1445.25	13.12%	A2, B1, C1
R208B	831.51	360	1191.51	-0.48%	A4
R209B	945.09	450	1395.09	5.50%	A5
R210B	1044.03	450	1494.03	9.02%	A5
R211B	856.93	410	1266.93	3.85%	A3, B1
C201B	703.08	1080	1783.08	5.20%	A2, B1, C1
C202B	782.34	1080	1862.34	10.51%	A4, B1
C203B	845.92	1000	1845.92	9.77%	A5
C204B	739.18	1080	1819.18	8.44%	A4, B1
C205B	638.26	1200	1838.26	8.69%	A2, C2
C206B	742.96	1080	1822.96	7.91%	A4, B1

C207B	702.29	1080	1782.29	5.63%	A4, B1
C208B	695.46	1080	1775.46	5.27%	A4, B1
RC201B	1433.73	700	2133.73	10.08%	B3, C3, D1
RC202B	1181.36	720	1901.36	7.54%	A2, C2, E2
RC203B	1078.92	700	1778.92	10.94%	B3, C1, D1, E1
RC204B	917.15	730	1647.15	10.60%	B1, C2, E2
RC205B	1318.36	700	2018.36	10.14%	A1, B1, C2, D1, E1
RC206B	1259.49	600	1859.49	7.83%	A1, B2, C1, D2
RC207B	1089.6	670	1759.6	6.88%	A2, B1, C2, D2
RC208B	887.98	760	1647.98	11.11%	A1, B1, E3

Table D.3 Results in Cost Structure C of the FSMVRPTW Dataset

Problem Case	VC	FC	TC	Dev	Composition
R101C	1782.15	319	2101.15	0.084531687	A5, B4, C8, D4, E1
R102C	1597.87	318	1915.87	8.72%	A3, B1, C5, D9
R103C	1349.43	359	1708.43	10.44%	A4, B1, C4, D3, E4, F1
R104C	1135.04	335	1470.04	8.70%	A1, B2, C1, D6, E3
R105C	1512.74	327	1839.74	9.41%	A2, B8, C2, D7, E1
R106C	1416.15	322	1738.15	9.79%	A3, B4, D7, E2
R107C	1208.85	333	1541.85	8.25%	B2, C3, D5, E3
R108C	1094.66	360	1454.66	10.63%	A2, B1, C3, D4, E4
R109C	1264.82	402	1666.82	10.64%	A1, B2, C4, D3, E5
R110C	1285.29	343	1628.29	12.81%	A2, B2, C3, D5, E3
R111C	1256.3	342	1598.3	12.60%	A1, B6, C1, D3, E4
R112C	1125.54	309	1434.54	8.06%	A2, B3, D5, E3
C101C	901.76	760	1661.76	2.01%	A4, B8
C102C	902.38	730	1632.38	2.17%	A3, B8
C103C	1008.06	730	1738.06	8.86%	A3, B8
C104C	1005.55	760	1765.55	10.99%	A3, B5, C2
C105C	958.08	730	1688.08	3.63%	A3, B8
C106C	988.03	740	1728.03	6.08%	A6, B7
C107C	1008.65	740	1748.65	7.35%	A6, B7
C108C	987.69	760	1747.69	7.70%	A3, B5, C2
C109C	993.5	740	1733.5	7.34%	A6, B7
RC101C	1855.55	375	2230.55	9.67%	A5, B6, C7, D1
RC102C	1619.43	378	1997.43	8.09%	A3, B6, C3, D4
RC103C	1407.85	426	1833.85	11.39%	A1, B2, C7, D4
RC104C	1323.49	357	1680.49	10.63%	A2, B1, C8, D2
RC105C	1578.6	387	1965.6	4.28%	A2, B3, C8, D2

RC106C	1473.25	387	1860.25	6.06%	A2, B3, C8, D2
RC107C	1308.5	426	1734.5	8.33%	A1, B2, C7, D4
RC108C	1282.19	372	1654.19	9.09%	A2, B4, C4, D4
R201C	1237.48	275	1512.48	5.80%	A3, B2
R202C	1150.7	275	1425.7	11.99%	A3, B2
R203C	915.96	225	1140.96	2.23%	A5
R204C	839.42	230	1069.42	7.68%	A2, B2
R205C	1095.19	205	1300.19	8.90%	A3, B1
R206C	917.8	305	1222.8	9.05%	A1, B2, C1
R207C	886.71	275	1161.71	10.37%	A3, B2
R208C	790.29	230	1020.29	5.20%	A2, B2
R209C	920.72	305	1225.72	11.94%	A1, B2, C1
R210C	1052.51	205	1257.51	9.78%	A3, B1
R211C	845.98	250	1095.98	10.16%	A4, B1
C201C	706.63	540	1246.63	4.38%	A2, B1, C1
C202C	689.33	540	1229.33	3.72%	A2, B1, C1
C203C	797.55	520	1317.55	12.01%	A1, B3
C204C	736.7	570	1306.7	11.17%	A3, D1
C205C	601.43	670	1271.43	6.81%	C2, D1
C206C	601.04	670	1271.04	6.93%	C2, D1
C207C	704.38	540	1244.38	5.02%	A2, B1, C1
C208C	702.57	540	1242.57	4.73%	A2, B1, C1
RC201C	1507.73	310	1817.73	11.97%	A4, B4, C2
RC202C	1222.88	385	1607.88	11.26%	A2, C1, D1, E2
RC203C	1084.97	345	1429.97	12.28%	A1, B1, C1, D3
RC204C	925.2	270	1195.2	3.22%	A1, B1, E2
RC205C	1313.58	290	1603.58	6.02%	A2, B2, D1, E1
RC206C	1153.55	405	1558.55	11.71%	C1, D3, E1
RC207C	1113.6	355	1468.6	11.73%	C5, D1
RC208C	902.22	360	1262.22	10.71%	A2, C2, E2

Table D.4 New Optimal Solution in RC101B Instance

Number of Route	**Vehicle Type**	**Specific Route**
1	B	0-92-31-29-27-26-34-32-93-80-92-31-29-27-26-34-32-93-80-0
2	A	0-14-47-16-15-14-47-16-15-0
3	A	0-36-39-61-36-39-61-0
4	A	0-64-23-49-64-23-49-0
5	A	0-90-66-56-90-66-56-0

6	B	0-95-62-67-71-94-96-54-95-62-67-71-94-96-54-0
7	B	0-21-19-18-22-20-48-25-21-19-18-22-20-48-25-0
8	A	0-63-76-51-63-76-51-0
9	A	0-98-73-79-3-100-98-73-79-3-100-0
10	A	0-88-53-55-60-88-53-55-60-0
11	A	0-84-85-89-91-84-85-89-91-0
12	B	0-72-42-44-40-38-41-35-37-72-42-44-40-38-41-35-37-0
13	A	0-81-43-68-81-43-68-0
14	A	0-33-28-30-50-33-28-30-50-0
15	B	0-83-52-12-78-10-13-17-83-52-12-78-10-13-17-0
16	A	0-65-59-75-74-65-59-75-74-0
17	B	0-82-99-86-57-24-82-99-86-57-24-0
18	C	0-5-45-2-7-6-8-46-4-1-70-5-45-2-7-6-8-46-4-1-70-0
19	B	0-11-9-87-97-58-77-11-9-87-97-58-77-0
20	A	0-69-0
Cost		2405.47

Table D.5 New Optimal Solution in RC102B Instance

Number of Route	**Vehicle Type**	**Specific Route**
1	A	0-91-95-62-67-84-0
2	B	0-42-39-36-44-38-41-72-54-0
3	B	0-14-47-11-15-16-9-74-52-0
4	B	0-98-73-79-78-60-100-70-0
5	B	0-3-5-45-8-7-6-46-4-0
6	B	0-96-71-40-43-35-37-93-0
7	A	0-90-81-68-55-0
8	B	0-65-99-87-59-97-75-58-0
9	B	0-21-23-57-86-77-25-83-0
10	B	0-69-88-53-10-13-17-12-82-0
11	B	0-92-64-51-76-49-20-24-0
12	B	0-48-19-18-22-56-66-0
13	B	0-50-29-27-26-89-85-0
14	B	0-63-33-28-30-32-31-34-94-80-0
15	A	0-2-1-61-0
Cost		2069.27

Table D.6 New Optimal Solution in RC103B Instance

Number of Route	**Vehicle Type**	**Specific Route**

1	A	0-39-36-44-0
2	A	0-65-64-53-98-80-0
3	C	0-69-82-99-9-10-13-16-17-47-0
4	C	0-2-45-46-8-7-6-4-5-3-1-100-0
5	B	0-90-52-86-59-87-97-75-58-77-0
6	A	0-85-63-33-30-32-95-91-0
7	B	0-20-23-19-49-22-24-74-0
8	B	0-51-76-89-18-48-21-25-83-0
9	A	0-88-79-73-0
10	A	0-68-55-70-0
11	C	0-61-42-43-40-38-41-37-35-72-54-81-0
12	C	0-94-34-31-27-26-28-29-71-93-96-0
13	B	0-12-14-11-15-78-60-0
14	B	0-92-62-50-67-84-56-66-57-0
Cost		1958.6

Table D.7 New Optimal Solution in RC106B Instance

Number of Route	**Vehicle Type**	**Specific Route**
1	B	0-71-94-67-50-56-91-80-0
2	A	0-45-5-81-0
3	B	0-11-12-14-47-15-16-17-13-0
4	B	0-61-42-44-43-40-35-37-70-0
5	B	0-83-57-19-22-20-66-0
6	B	0-88-2-8-46-4-3-1-100-0
7	B	0-63-76-23-21-18-48-49-25-24-0
8	B	0-69-98-73-79-7-6-55-68-0
9	B	0-72-36-38-39-41-54-96-0
10	A	0-92-95-62-33-30-32-93-0
11	B	0-31-29-27-26-28-34-89-0
12	A	0-90-53-78-60-0
13	A	0-64-51-85-84-0
14	C	0-82-99-52-86-87-9-10-0
15	B	0-65-59-75-97-58-77-74-0
Cost		2004.62

Table D.8 New Optimal Solution in RC107B Instance

Number of Route	**Vehicle Type**	**Specific Route**

1	B	0-83-64-51-85-63-76-89-56-91-0
2	B	0-65-52-74-86-57-22-49-20-0
3	B	0-11-9-99-66-84-50-93-0
4	B	0-95-67-72-38-37-36-40-42-0
5	A	0-98-53-73-60-100-68-0
6	B	0-82-12-14-47-17-16-15-13-10-0
7	C	0-88-79-7-6-5-3-45-46-4-1-70-0
8	A	0-39-35-43-44-0
9	A	0-90-92-94-80-0
10	B	0-71-81-61-41-54-96-0
11	B	0-69-87-59-75-97-58-77-0
12	B	0-2-8-78-55-0
13	B	0-62-19-21-18-48-23-25-24-0
14	B	0-31-29-27-28-26-30-32-34-33-0
Cost		1961.53

Table D.9 New Optimal Solution in R208B Instance

Number of Route	Vehicle Type	Specific Route
1	A	0-94-92-42-15-43-14-44-38-86-16-61-5-60-83-7-88-31-70-51-9-81-33-68-80-12-28-53-13-95-59-98-37-100-91-85-93-96-6-0
2	A	0-52-10-32-90-63-11-64-49-36-47-46-8-45-17-84-99-97-87-2-57-41-22-74-73-58-0
3	A	0-40-21-72-75-56-23-67-54-76-50-1-62-19-48-82-18-89-0
4	A	0-27-69-30-20-66-65-71-35-34-78-79-3-77-29-24-55-25-39-4-26-0
Cost		1191.51

HFVRP

Table D.10 present the new optimal solution obtained by the ALNS when tested in HFVRP dataset, in problem instance 16.

Table D.10 New Optimal Solution in Problem Instance 16

Number of Route	Vehicle Type	Specific Route
1	B	0-12-47-0
2	A	0-46-32-27-0
3	B	0-48-8-26-31-28-1-0
4	B	0-11-2-16-38-0
5	B	0-4-41-13-14
6	C	0-22-3-36-35-20-29-21-34-50-9-0
7	C	0-6-23-7-43-24-25-18-0

8	A	0-19-40-42-37-0
9	C	0-5-49-10-30-39-33-45-15-44-17-0
Cost		1150.25

HFVRPTW

Table D.11 New Optimal Solution in HC102 Instance

Number of Route	Vehicle Type	Specific Route
1	A	0-20-24-27-29-30-28-26-23-0
2	B	0-67-65-63-62-74-72-61-64-68-66-69-0
3	B	0-43-42-41-40-44-46-45-48-51-50-52-49-47-0
4	B	0-13-17-18-19-15-16-14-12-1-0
5	B	0-57-55-54-53-56-58-60-59-0
6	B	0-81-78-76-71-70-73-77-79-80-82-83-0
7	A	0-10-2-75-88-91-0
8	A	0-7-8-11-9-6-4-5-0
9	A	0-31-35-37-38-39-0
10	A	0-32-33-36-34-0
11	A	0-90-87-86-84-85-89-0
12	A	0-94-92-93-95-0
13	A	0-21-25-22-0
14	A	0-98-96-97-100-99-3-0
Cost		1848.43

Table D.12 New Optimal Solution in HC204 Instance

Number of Route	Vehicle Type	Specific Route
1	A	0-20-22-24-27-29-6-32-33-34-36-39-38-37-35-31-52-50-51-46-41-42-43-48-0
2	A	0-67-62-74-72-61-64-66-69-68-65-49-55-54-53-56-58-60-59-57-40-44-45-47-0
3	A	0-21-30-28-26-23-17-18-19-16-14-12-15-13-25-9-11-10-8-0
4	B	0-90-91-88-86-84-83-82-85-76-71-70-73-80-79-81-78-77-96-87-63-0
5	A	0-93-5-75-2-1-99-100-97-92-94-95-98-7-3-4-89-0
Cost		1254.25

Table D.13 New Optimal Solution in HRC204 Instance

Number	Vehicle	Specific Route

of Route	Type	
1	A	0-69-55-100-70-68-0
2	A	0-65-99-87-59-9-82-90-0
3	A	0-81-54-72-71-93-94-96-0
4	B	0-10-11-15-16-17-47-14-12-53-98-0
5	A	0-61-2-6-7-88-0
6	B	0-67-34-31-29-27-26-28-30-32-33-50-0
7	B	0-20-49-19-18-48-21-23-25-77-58-57-0
8	B	0-60-78-73-79-8-46-4-45-5-3-1-0
9	B	0-42-44-43-40-36-35-37-38-39-41-0
10	A	0-83-24-22-64-66-0
11	A	0-52-86-74-75-97-13-0
12	A	0-92-62-85-63-89-76-51-0
13	A	0-80-91-95-84-56-0
Cost		4305.16

Table D.14 New Optimal Solution in HR201 Instance

Number of Route	Vehicle Type	Specific Route
1	A	0-28-33-65-71-29-12-76-79-81-51-9-78-34-50-3-68-54-0
2	B	0-2-73-21-40-18-8-46-48-60-17-91-100-93-58-0
3	A	0-27-52-69-31-30-62-11-19-7-88-90-49-10-20-32-66-35-1-0
4	B	0-72-39-67-23-75-22-41-56-74-4-55-25-24-80-77-70-0
5	A	0-95-59-92-42-15-14-98-85-99-6-87-57-43-37-97-96-13-89-0
6	A	0-5-83-45-36-63-64-47-82-61-16-44-38-86-84-94-53-26-0
Cost		1666.2

Table D.15 New Optimal Solution in HR202 Instance

Number of Route	Vehicle Type	Specific Route
1	A	0-27-50-33-65-34-29-71-30-88-7-19-49-10-20-66-35-68-12-0
2	A	0-45-48-47-36-63-64-11-62-31-69-76-3-79-78-81-9-51-90-32-0
3	A	0-94-95-92-42-15-14-44-38-86-18-82-8-46-83-59-97-43-74-2-13-0
4	B	0-96-99-61-16-91-37-87-57-41-22-75-56-4-54-55-25-24-80 -77-1-70-52-0
5	B	0-28-26-39-67-23-72-73-21-40-53-6-5-84-17-85-100 -98-93-60-89-58-0
Cost		1517.77

Appendix E Supplement Chapter 6

```
1.  import random
2.  import numpy as np
3.  import pandas as pd
4.  import timeit
5.  from math import sin, cos, sqrt, atan2, radians
6.
7.  start = timeit.default_timer()
8.
9.  #------- Vehicles' dataset
10. filename1 = "Vehicles.xlsx"
11. datveh = pd.read_excel(filename1)
12.
13. filename2 = "Customers.xlsx"
14. datcust = pd.read_excel(filename2)
15.
16. dminitial = np.loadtxt("distance_matrix.txt")
17. ttinitial = np.loadtxt("travel_time_matrix.txt")
18.
19. #Create tha matrices of vehicles and routes
20. def vehdata(sumnv, availnv, fc, vc, capacity1, capacity2, startingshift, vtype, endingshift):
21.     vehdat = np.zeros(shape = (sumnv,7))
22.     av1 = np.copy(availnv)
23.     k = 0
24.     while av1.sum() != 0:
25.         ran = random.randint(0,len(availnv)-1)
26.         while av1[ran] == 0:
27.             ran = random.randint(0,len(availnv)-1)
28.         vehdat[k, 0] = fc[ran]
29.         vehdat[k, 1] = vc[ran]
30.         vehdat[k, 2] = capacity1[ran]
31.         vehdat[k, 3] = capacity2[ran]
32.         vehdat[k, 4] = startingshift[ran]
33.         vehdat[k, 5] = vtype[ran]
34.         vehdat[k, 6] = endingshift[ran]
35.         av1[ran] = av1[ran]-1
36.         k = k + 1
37.     return vehdat
38.
39. #Set the matrix related to the load of each route, the capacity of the vehicle, etc.
40. def quantity(routes, vehdat, n2, demand1, demand2, dm, toa, tstart, tod, ttravel):
41.     quantity = np.zeros(shape = (len(routes), 15))
42.     for i in range(len(routes)):
43.         numberofcust = 0
44.         quantity[i, 0] = vehdat[i,5]
45.         quantity[i, 1] = 0
```

```
46.             quantity[i, 2] = vehdat[i, 2]
47.             quantity[i, 3] = 0
48.             quantity[i, 4] = vehdat[i, 3]
49.             for j in range(n2-1):
50.                 quantity[i, 1] = quantity[i, 1] + demand1[int(routes[i, j])]
51.                 quantity[i, 3] = quantity[i, 3] + demand2[int(routes[i, j])]
52.                 quantity[i, 5] = quantity[i, 5] + dm[int(routes[i,j]), int(routes[i, j + 1])]
53.                 if routes[i,j]>0:
54.                     numberofcust=numberofcust+1
55.             quantity[i, 6] = vehdat[i,1] * quantity[i, 5]
56.             if quantity[i, 5] != 0:
57.                 quantity[i, 7] = vehdat[i, 0]
58.             else:
59.                 quantity[i, 7] = 0
60.             quantity[i, 8] = quantity[i, 7] + quantity[i, 6]
61.             quantity[i, 9] = ttravel[i,0]
62.             quantity[i, 10] = ttravel[i,1]
63.             quantity[i, 11] = ttravel[i,2]
64.             quantity[i, 12] = vehdat[i, 4]
65.             quantity[i, 13] = vehdat[i, 6]
66.             quantity[i, 14] = numberofcust
67.         return quantity
68.
69.     #Calculation of the total cost of a single route
70.     def totc(route, vehdat, n2, dm):
71.         s = np.count_nonzero(route)
72.         fixedc = vehdat[0]
73.         variablec = 0
74.         td = 0
75.         for j in range(s+2):
76.             td = td + dm[int(route[j]), int(route[j+1])]
77.         variablec = td*vehdat[1]
78.         totalc = fixedc + variablec
79.         if s == 0:
80.             totalc = 0
81.         return totalc
82.
83.     #Calculation of the total cost of all routes
84.     def totalcost(route, vehdat, n2, dm):
85.         fixedc = 0
86.         variablec = 0
87.         totaldist = 0
88.         for i in range(len(route)):
89.             if route[i, 1] != 0:
90.                 fixedc = fixedc + vehdat[i, 0]
91.             td = 0
92.             for j in range(n2 - 1):
93.                 td = td + dm[int(route[i, j]), int(route[i, j+1])]
```

```
            variablec = variablec + td*vehdat[i, 1]
            totaldist = totaldist + td
        totalc = fixedc + variablec
        return totalc

    #Calculation of the fixed and variable cost of all routes seperately
    def fvc(routein, vehdat, n2, dm):
        fixedc = 0
        variablec = 0
        totaldist = 0
        for i in range(len(routein)):
            if routein[i, 1] != 0:
                fixedc = fixedc + vehdat[i, 0]
            td = 0
            for j in range(n2 - 1):
                td = td + dm[int(routein[i, j]), int(routein[i, j+1])]
            variablec = variablec + td*vehdat[i, 1]
            totaldist = totaldist + td
        return variablec,fixedc

    #Calculation of the total fixed cost of all routes
    def fcc(routein, vehdat):
        fixedc = 0
        for i in range(len(routein)):
            if routein[i, 1] != 0:
                fixedc = fixedc + vehdat[i, 0]
        return fixedc

    #Calculation of the load of deliveries of a single route in Kg
    def totq1(route1,  n2, demand1):
        q = 0
        s = np.count_nonzero(route1)
        for i in range(0, s+1):
            q = q + demand1[int(route1[i])]
        return q

    #Calculation of the load of deliveries of a single route in m3
    def totq2(route1,  n2, demand2):
        q = 0
        s = np.count_nonzero(route1)
        for i in range(0, s+1):
            q = q + demand2[int(route1[i])]
        return q

    #Calculation of the load of pickups of a single route in Kg
    def totr1(route1,  n2, reverse1):
        q = 0
        s = np.count_nonzero(route1)
```

```
142.      for i in range(0, s+1):
143.          q = q + reverse1[int(route1[i])]
144.      return q
145.
146.  #Calculation of the load of pickups of a single route in m3
147.  def totr2(route1, n2, reverse2):
148.      q = 0
149.      s = np.count_nonzero(route1)
150.      for i in range(0, s+1):
151.          q = q + reverse2[int(route1[i])]
152.      return q
153.
154.  #Calculation of the total distance traveled from all routes
155.  def totdistance(routes, dm, n2):
156.      td = 0
157.      for i in range(len(routes)):
158.          s = np.count_nonzero(routes[i, :])
159.          for j in range(s+1):
160.              td = td + dm[int(routes[i,j]), int(routes[i, j+1])]
161.      return td
162.
163.  def countcust(customers):
164.      s = 0
165.      for i in range(len(customers)):
166.          if customers[i] != 0:
167.              s = s + 1
168.      return s
169.
170.  #Check in the load on board in each step of the distribution process for a single vehicle
171.  def posotites(routes, position, vehdat, reverse1, reverse2, demand1, demand2, n2):
172.      rrr1 = np.count_nonzero(routes)
173.      q = True
174.      d1 = 0
175.      d2 = 0
176.      r1 = 0
177.      r2 = 0
178.      for i in range(0, rrr1+1):
179.          d1 = d1 + demand1[int(routes[i])]
180.          d2 = d2 + demand2[int(routes[i])]
181.          r1 = r1 + reverse1[int(routes[i])]
182.          r2 = r2 + reverse2[int(routes[i])]
183.      if d1 >vehdat[position,2] or d2 > vehdat[position,3] or r1 > vehdat[position,2] or r2>vehdat[position
      ,3]:
184.          q = False
185.      i = 1
186.      while i in range(1,rrr1+2) and q == True:
187.          d1 = d1 - demand1[int(routes[i])] + reverse1[int(routes[i])]
188.          d2 = d2 - demand2[int(routes[i])] + reverse2[int(routes[i])]
```

```
189.            if d1 > vehdat[position,2] or d2 > vehdat[position,3]:
190.                q = False
191.            i = i+1
192.        return q
193.
194.    #Calculation of the load on board in each step of the distribution for all routes, both in kg and m3
195.    def analytikaposotites(routes, n2, demand1, demand2, reverse1, reverse2):
196.        posotites1 = np.zeros(shape = (len(routes), n2+1))
197.        posotites2 = np.zeros(shape = (len(routes), n2+1))
198.        for i in range(len(routes)):
199.            posotites1[i,0] = totq1(routes[i, :], n2, demand1)
200.            posotites2[i,0] = totq2(routes[i, :], n2, demand2)
201.            sk = np.count_nonzero(routes[i, :])
202.            for j in range(1, sk + 2):
203.                posotites1[i,j] = posotites1[i, j-1]- demand1[int(routes[i,j])] + reverse1[int(routes[i,j])]
204.                posotites2[i,j] = posotites2[i, j-1]- demand2[int(routes[i,j])] + reverse2[int(routes[i,j])]
205.        return posotites1, posotites2
206.
207.    #Check that all time windows of a single route are respected
208.    def twcheck(routes, position, vehdat, tws, tt, twe, st):
209.        r1 = np.count_nonzero(routes)
210.        t = vehdat[position, 4]
211.        q = True
212.        i = 0
213.        while i in range(0,r1+1) and q == True:
214.            t = t + tt[int(routes[i]), int(routes[i+1])]
215.            if t > twe[int(routes[i+1])] or t > vehdat[position, 6]:
216.                q = False
217.            else:
218.                if t < tws[int(routes[i+1])]:
219.                    t = tws[int(routes[i+1])]
220.                t = t + st[int(routes[i+1])]
221.            i = i + 1
222.        return q
223.
224.    #Calculation of the toa, tod, waiting time, and start of serving in each customer
225.    def times(routes, n2, tws, tt, twe, st, vehdat):
226.        customers = np.zeros(shape = (n2-1,1))
227.        waiting = np.zeros(shape = (n2-1,1))
228.        toa = np.zeros(shape = (len(routes), n2+1))
229.        tstart = np.zeros(shape = (len(routes), n2+1))
230.        tod = np.zeros(shape = (len(routes), n2+1))
231.        ttravel = np.zeros(shape = (len(routes), 3))
232.        k = 0
233.        for i in range(0, len(routes)):
234.            t = vehdat[i, 4]
235.            toa[i, 0] = vehdat[i, 4]
236.            tod[i, 0] = vehdat[i, 4]
```

```
237.            smax = np.count_nonzero(routes[i,:])
238.            for j in range(0, smax+1):
239.                t = t + tt[int(routes[i, j]), int(routes[i, j+1])]
240.                if j ==0 and t <= tws[int(routes[i, j+1])]:
241.                    ttravel[i,0] = tws[int(routes[i, j+1])] - tt[int(routes[i, j]), int(routes[i, j+1])]
242.                    tstart[i, j] = ttravel[i, 0]
243.                    tod[i, 0] = tws[int(routes[i, j+1])] - tt[int(routes[i, j]), int(routes[i, j+1])]
244.                elif j == 0 and t > tws[int(routes[i, j+1])]:
245.                    ttravel[i,0] = vehdat[i, 4]
246.                    tstart[i, j] = ttravel[i, 0]
247.                    tod[i, j] = 0
248.                if t < tws[int(routes[i, j+1])]:
249.                    tstart[i, j+1] = tws[int(routes[i, j+1])]
250.                    waiting[k] = tws[int(routes[i, j+1])] - t
251.                    customers[k] = int(routes[i, j+1])
252.                    k = k +1
253.                else:
254.                    tstart[i, j+1] = t
255.                    waiting[k] = 0
256.                    customers[k] = int(routes[i, j+1])
257.                toa[i,j+1] = t
258.                t = tstart[i, j+1] + st[int(routes[i, j+1])]
259.                tod[i, j+1] = t
260.            ttravel[i,1] = toa[i,:].max()
261.            ttravel[i,2] = toa[i,:].max() - ttravel[i,0]
262.        return tstart, toa, tod, ttravel
263.
264.    #Construction procedure for the initial plan of routes
265.    def construction1(sumnv, availnv, vehdat, w1, w2, customers1, n2, tws, tt, twe, st, demand1, reverse1
        , demand2, reverse2, dm):
266.        customers = np.copy(customers1)
267.        routes = np.zeros(shape=(sumnv, n2 + 2))
268.        routes1 = np.zeros(shape=(sumnv, n2 + 2))
269.        dmmax = dm.max()
270.        twemax = twe.max()
271.        r = 0
272.        while r in range(len(routes)) and customers.sum()!=0:
273.            q = True
274.            i = 0
275.            position = 1
276.            dok = 0
277.            t = vehdat[r, 6]
278.            mindt = 999999
279.            s = 0
280.            while q:
281.                mindt = 999999
282.                dok = 0
283.                s = 0
```

```
284.            routes1 = np.copy(routes)
285.            for j in range(0, len(customers)):
286.               routes1[r, position] = int(customers[j])
287.               if int(customers[j])!= 0 and posotites(routes1[r, :], r, vehdat, reverse1, reverse2, demand1, d
       emand2, n2) == True and twcheck(routes1[r,:], r, vehdat, tws, tt, twe, st) == True:
288.                  #minf = w1*dm[i,int(customers[j])]/dmmax + w2*(twe[int(customers[j])] - t - tt[i, int(cust
       omers[j])])/twe[int(customers[j])]
289.                  minf = w1*dm[i,int(customers[j])]/dmmax + w2*(twe[int(customers[j])] - t - tt[i, int(custo
       mers[j])])/twe[int(customers[j])]
290.                  if minf < mindt:
291.                     s = int(customers[j])
292.                     k = j
293.                     mindt = minf
294.                     dok = dok + 1
295.            if dok == 0 or customers.sum() == 0:
296.               q = False
297.            else:
298.               if t + tt[i, s] < tws[s]:
299.                  t = tws[s] + st[s]
300.               else:
301.                  t = t + tt[i, s] + st[s]
302.               i = s
303.               routes[r, position] = s
304.               position = position + 1
305.               customers[k] = 0
306.         r = r + 1
307.      return routes, customers
308.
309.   #function that is integrated into the construction procedure
310.   def incon(routes2, cust, vehdat, sumnv, n2, dm, tws, twe, tt, st, demand1, reverse1, demand2, revers
       e2, customers1, k):
311.      for i in range(len(cust)):
312.         www = False
313.         if cust[i] != 0:
314.            initialcost = 99999999
315.            www = False
316.            if totq1(routes2[k, :], n2, demand1) + demand1[int(cust[i])] <= vehdat[k, 2] and totq2(routes2[k
       , :], n2, demand2) + demand2[int(cust[i])] <= vehdat[k, 3] and totr1(routes2[k, :], n2, reverse1) + rever
       se1[int(cust[i])] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[int(cust[i])] <= vehdat[
       k, 3]:
317.               sk = np.count_nonzero(routes2[k, :])
318.               routes2n = np.copy(routes2)
319.               new = 1
320.               while new in range(1, sk+2):
321.                  routes2n = np.copy(routes2)
322.                  for j in range(0, new):
323.                     routes2n[k, j] = routes2[k, j]
324.                  routes2n[k, new] = cust[i]
```

```
325.                 for j in range(new, sk+1):
326.                     routes2n[k, j+1] = routes2[k, j]
327.                 if totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm) <= initialc
    ost and twcheck(routes2n[k,:], k, vehdat, tws, tt, twe, st) == True and posotites(routes2n[k, :], k, vehd
    at, reverse1, reverse2, demand1, demand2, n2) == True:
328.                     routesopt = np.copy(routes2n)
329.                     initialcost = totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, d
    m)
330.                     www = True
331.                 new = new + 1
332.         if www == True:
333.             cust[i] = 0
334.             routes2 = np.copy(routesopt)
335.     return routes2, cust
336.
337.     #Selection of customers randomly
338.     def selection1(routes, vehdat, dl, ul, sumnv, n2, customers, unrouted):
339.         noc = random.randint(dl, ul)
340.         cust = np.zeros(shape = (noc, 1))
341.         k = 0
342.         while np.count_nonzero(cust) < noc:
343.             ranc = random.randint(0, n2-2)
344.             while customers[ranc] in cust or customers[ranc] in unrouted:
345.                 ranc = random.randint(0, n2-2)
346.             cust[k] = customers[ranc]
347.             k = k + 1
348.         i = 0
349.         routes2 = np.zeros(shape=(sumnv, n2+1))
350.         while i in range(0, sumnv):
351.             if routes[i,:].sum() != 0:
352.                 s = np.count_nonzero(routes[i,:])
353.                 k1 = 1
354.                 for j in range(1, s+1):
355.                     if routes[i, j] not in cust:
356.                         routes2[i,k1] = routes[i, j]
357.                         k1 = k1 + 1
358.             i = i + 1
359.         return routes2, cust, noc
360.
361.     #Selection of the routes according to the highest €/quantity
362.     def selection2(routes, vehdat, sumnv, dm, n2, demand1, demand2, reverse1, reverse2):
363.         r1 = np.zeros(shape=(sumnv, 1))
364.         r2 = np.zeros(shape=(sumnv, 1))
365.         p1 = random.random()
366.         for i in range(len(routes)):
367.             r1[i] = i
368.             if p1<0.5:
```

```
369.            r2[i] = totc(routes[i,:], vehdat[i,:], n2, dm)/(totq1(routes[i, :], n2, demand1)+totr1(routes[i, :], n
    2, reverse1))
370.        else:
371.            r2[i] = totc(routes[i,:], vehdat[i,:], n2, dm)/(totq2(routes[i, :], n2, demand2)+totr2(routes[i, :], n
    2, reverse2))
372.    rrr2, rrr1 = zip(*sorted(zip(r2, r1), reverse = True))
373.    ssss = np.count_nonzero(rrr2)
374.    randomnumb = random.randint(0, ssss//3)
375.    ii = int(rrr1[randomnumb])
376.    noc = np.count_nonzero(routes[ii, :])
377.    cust = np.zeros(shape = (noc, 1))
378.    for i in range(1, noc+1):
379.        cust[i-1] = routes[ii, i]
380.    routes2 = np.copy(routes)
381.    for j in range(0, noc+2):
382.        routes2[ii, j] = 0
383.    return routes2, cust, noc
384.
385. #Selection of distant customers
386. def selection3(routes, vehdat, n2, tws, twe, tt, st, dl, dm, ul, sumnv):
387.    wc1 = np.zeros(shape=(n2-1, 1))
388.    k = 0
389.    wc2 = np.zeros(shape=(n2-1, 1))
390.    for i in range(sumnv):
391.        sk = np.count_nonzero(routes[i,:])
392.        for j in range(1,sk+1):
393.            wc1[k] = routes[i, j]
394.            wc2[k] = (dm[int(routes[i,j-
    1]), int(routes[i,j])] + dm[int(routes[i,j]), int(routes[i,j+1])])*vehdat[i,1]
395.            k = k + 1
396.    dem, custom = zip(*sorted(zip(wc2, wc1), reverse = True))
397.    noc = random.randint(dl, ul)
398.    cust = np.zeros(shape = (noc, 1))
399.    k = 0
400.    for i in range(0, noc):
401.        cust[k] = int(custom[i])
402.        k = k + 1
403.    i = 0
404.    routes2 = np.zeros(shape=(sumnv, n2+1))
405.    while i in range(0, sumnv):
406.        if routes[i,:].sum() != 0:
407.            s = np.count_nonzero(routes[i,:])
408.            k1 = 1
409.            for j in range(1, s+1):
410.                if routes[i,j] not in cust:
411.                    routes2[i,k1] = routes[i,j]
412.                    k1 = k1 + 1
413.        i = i + 1
```

```
414.      return routes2, cust, noc
415.
416.    #Selection of customers in which, vehicles wait the most to deliver
417.    def selection4(routes, vehdat, n2, tws, twe, tt, st, dl, dm, ul, sumnv):
418.      customers5 = np.zeros(shape = (n2-1,1))
419.      waiting = np.zeros(shape = (n2-1,1))
420.      toa = np.zeros(shape = (len(routes), n2+1))
421.      tstart = np.zeros(shape = (len(routes), n2+1))
422.      tod = np.zeros(shape = (len(routes), n2+1))
423.      k = 0
424.      for i in range(0, len(routes)):
425.        t = vehdat[i, 6]
426.        smax = np.count_nonzero(routes[i,:])
427.        for j in range(0, smax+1):
428.          t = t + tt[int(routes[i, j]), int(routes[i, j+1])]
429.          if t < tws[int(routes[i, j+1])]:
430.            if j == 0:
431.              toa[i,j+1] = tws[int(routes[i, j+1])]
432.            else:
433.              toa[i,j+1] = t
434.            tstart[i, j+1] = tws[int(routes[i, j+1])]
435.            if int(routes[i, j+1]) != 0:
436.              waiting[k] = tstart[i, j+1] - toa[i,j+1]
437.              customers5[k] = int(routes[i, j+1])
438.              k = k +1
439.          else:
440.            toa[i,j+1] = t
441.            tstart[i, j+1] = t
442.            if int(routes[i, j+1]) != 0:
443.              waiting[k] = 0
444.              customers5[k] = int(routes[i, j+1])
445.              k = k + 1
446.          t = tstart[i, j+1] + st[int(routes[i, j+1])]
447.          tod[i, j+1] = t
448.      dem, custom = zip(*sorted(zip(waiting, customers5), reverse = True))
449.      noc = random.randint(dl,  ul)
450.      cust = np.zeros(shape = (noc, 1))
451.      k = 0
452.      for i in range(0, noc):
453.        cust[k] = int(custom[i])
454.        k = k + 1
455.      routes2 = np.zeros(shape=(sumnv, n2+1))
456.      i = 0
457.      while i in range(0, sumnv):
458.        if routes[i,:].sum() != 0:
459.          s = np.count_nonzero(routes[i,:])
460.          k1 = 1
461.          for j in  range(1, s+1):
```

```
462.                if routes[i,j] not in cust:
463.                    routes2[i,k1] = routes[i,j]
464.                    k1 = k1 + 1
465.            i = i + 1
466.        return routes2, cust, noc
467.
468.    #Selection of customers in which, vehicles wait the most to deliver
469.    def selection34(routes, vehdat, n2, tws, twe, tt, st, dl, dm, ul, sumnv):
470.        customers5 = np.zeros(shape = (n2-1,1))
471.        waiting = np.zeros(shape = (n2-1,1))
472.        toa = np.zeros(shape = (len(routes), n2+1))
473.        tstart = np.zeros(shape = (len(routes), n2+1))
474.        tod = np.zeros(shape = (len(routes), n2+1))
475.        twemax = twe.max()
476.        dmmax = dm.max()
477.        k = 0
478.        for i in range(0, len(routes)):
479.            t = vehdat[i, 6]
480.            smax = np.count_nonzero(routes[i,:])
481.            for j in range(0, smax+1):
482.                t = t + tt[int(routes[i,j]), int(routes[i, j+1])]
483.                if t < tws[int(routes[i, j+1])]:
484.                    if j == 0:
485.                        toa[i,j+1] = tws[int(routes[i, j+1])]
486.                    else:
487.                        toa[i,j+1] = t
488.                    tstart[i, j+1] = tws[int(routes[i, j+1])]
489.                    if int(routes[i, j+1]) != 0:
490.                        waiting[k] = (tstart[i, j+1] - toa[i,j+1]/twemax)*0.2+((dm[int(routes[i,j-
    1]), int(routes[i,j])] + dm[int(routes[i,j]), int(routes[i,j+1])])/dmmax)*0.8
491.                        customers5[k] = int(routes[i, j+1])
492.                        k = k +1
493.                else:
494.                    toa[i,j+1] = t
495.                    tstart[i, j+1] = t
496.                    if int(routes[i, j+1]) != 0:
497.                        waiting[k] = 0 + ((dm[int(routes[i,j-
    1]), int(routes[i,j])] + dm[int(routes[i,j]), int(routes[i,j+1])])/dmmax)*0.8
498.                        customers5[k] = int(routes[i, j+1])
499.                        k = k + 1
500.                t = tstart[i, j+1] + st[int(routes[i, j+1])]
501.                tod[i, j+1] = t
502.        dem, custom = zip(*sorted(zip(waiting, customers5), reverse = True))
503.        noc = random.randint(dl, ul)
504.        cust = np.zeros(shape = (noc, 1))
505.        k = 0
506.        for i in range(0, noc):
507.            cust[k] = int(custom[i])
```

```
508.            k = k + 1
509.        routes2 = np.zeros(shape=(sumnv, n2+1))
510.        i = 0
511.        while i in range(0, sumnv):
512.            if routes[i,:].sum() != 0:
513.                s = np.count_nonzero(routes[i,:])
514.                k1 = 1
515.                for j in  range(1, s+1):
516.                    if routes[i, j] not in cust:
517.                        routes2[i,k1] = routes[i, j]
518.                        k1 = k1 + 1
519.            i = i + 1
520.        return routes2, cust, noc
521.
522.    #Selection of two routes
523.    def selection5(routes, vehdat, n2, dl, ul, sumnv):
524.        ii1 = random.randint(0,  sumnv-1)
525.        while routes[ii1,:].sum() == 0:
526.            ii1 = random.randint(0,  sumnv-1)
527.        ii2 = random.randint(0,  sumnv-1)
528.        while routes[ii2,:].sum() == 0 or ii1 == ii2:
529.            ii2 = random.randint(0,  sumnv-1)
530.        noc = np.count_nonzero(routes[ii1,  :]) + np.count_nonzero(routes[ii2,  :])
531.        cust= np.zeros(shape = (noc, 1))
532.        routes2 = np.copy(routes)
533.        for i in range(np.count_nonzero(routes[ii1,  :])):
534.            cust[i] = routes[ii1, i+1]
535.            routes2[ii1, i +1] = 0
536.        for j in range(np.count_nonzero(routes[ii1,  :]), noc):
537.            cust[j] = routes[ii2, j+1 - np.count_nonzero(routes[ii1,  :])]
538.            routes2[ii2, j+1 - np.count_nonzero(routes[ii1,  :])] = 0
539.        return routes2, cust, noc
540.
541.    #Selection of 3 routes
542.    def selection55(routes, vehdat, n2, dl, ul, sumnv):
543.        ii1 = random.randint(0,  sumnv-1)
544.        while routes[ii1,:].sum() == 0:
545.            ii1 = random.randint(0,  sumnv-1)
546.        ii2 = random.randint(0,  sumnv-1)
547.        while routes[ii2,:].sum() == 0 or ii1 == ii2:
548.            ii2 = random.randint(0,  sumnv-1)
549.        ii3 = random.randint(0,  sumnv-1)
550.        while routes[ii3,:].sum() == 0 or ii3 == ii2 or ii3 == ii1:
551.            ii3 = random.randint(0,  sumnv-1)
552.        noc = np.count_nonzero(routes[ii1,  :]) + np.count_nonzero(routes[ii2,  :]) + np.count_nonzero(route
        s[ii3, :])
553.        cust= np.zeros(shape = (noc, 1))
554.        routes2 = np.copy(routes)
```

```
555.        for i in range(np.count_nonzero(routes[ii1, :])):
556.            cust[i] = routes[ii1, i+1]
557.            routes2[ii1, i +1] = 0
558.        for j in range(np.count_nonzero(routes[ii1, :]), np.count_nonzero(routes[ii1, :])+np.count_nonzero(
        routes[ii2, :])):
559.            cust[j] = routes[ii2, j+1 - np.count_nonzero(routes[ii1, :])]
560.            routes2[ii2, j+1 - np.count_nonzero(routes[ii1, :])] = 0
561.        for j in range(np.count_nonzero(routes[ii1, :])+np.count_nonzero(routes[ii2, :]), noc):
562.            cust[j] = routes[ii3, j+1 - np.count_nonzero(routes[ii1, :])]
563.            routes2[ii3, j+1 - np.count_nonzero(routes[ii1, :])] = 0
564.        return routes2, cust, noc
565.
566.    #Insertion operator 1
567.    def insertion_1(noc, routes2, cust, vehdat, fcc1, sumnv, n2, dm, tws, twe, tt, st, demand1, reverse1, d
        emand2, reverse2):
568.        while cust.sum() != 0:
569.            i = random.randint(0, noc-1)
570.            while cust[i] == 0:
571.                i = random.randint(0, noc-1)
572.            customerchosen = int(cust[i])
573.            cust[i] = 0
574.            initialcost = 99999999
575.            www = False
576.            k = 0
577.            zeros = 0
578.            while k in range(0, sumnv) and zeros < 10:
579.                if routes2[k, 1] == 0:
580.                    zeros = zeros + 1
581.                if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(rou
        tes2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, reve
        rse1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[cus
        tomerchosen] <= vehdat[k, 3]:
582.                    sk = np.count_nonzero(routes2[k, :])
583.                    routes2n = np.copy(routes2)
584.                    new = 1
585.                    while new in range(1, sk+2):
586.                        routes2n = np.copy(routes2)
587.                        for j in range(0, new):
588.                            routes2n[k, j] = routes2[k, j]
589.                        routes2n[k, new] = customerchosen
590.                        for j in range(new, sk+1):
591.                            routes2n[k, j+1] = routes2[k, j]
592.                        if totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm) <= initialco
        st and fcc(routes2n, vehdat) <= fcc1 and twcheck(routes2n[k,:], k, vehdat, tws, tt, twe, st) == True and
         posotites(routes2n[k, :], k, vehdat, reverse1, reverse2, demand1, demand2, n2) == True:
593.                            routesopt = np.copy(routes2n)
594.                            initialcost = totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, d
        m)
```

```
595.                    www = True
596.                new = new + 1
597.            k = k + 1
598.        if www == True:
599.            routes2 = np.copy(routesopt)
600.    return routes2
601.
602. #Insertion operator 2
603. def insertion_2(noc, routes2, cust, vehdat, fcc1, sumnv, n2, dm, tws, twe, tt, st, demand1, reverse1, d
     emand2, reverse2):
604.    while cust.sum() != 0:
605.        i = random.randint(0, noc-1)
606.        while cust[i] == 0:
607.            i = random.randint(0, noc-1)
608.        customerchosen = int(cust[i])
609.        cust[i] = 0
610.        initialcost = 99999999
611.        www = False
612.        k = 0
613.        zeros = 0
614.        while k in range(0, sumnv) and zeros < 10:
615.            if routes2[k, 1] == 0:
616.                zeros = zeros + 1
617.            if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(rou
     tes2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, reve
     rse1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[cus
     tomerchosen] <= vehdat[k, 3]:
618.                sk = np.count_nonzero(routes2[k, :])
619.                routes2n = np.copy(routes2)
620.                new = 1
621.                while new in range(1, sk+2):
622.                    routes2n = np.copy(routes2)
623.                    for j in range(0, new):
624.                        routes2n[k, j] = routes2[k, j]
625.                    routes2n[k, new] = customerchosen
626.                    for j in range(new, sk+1):
627.                        routes2n[k, j+1] = routes2[k, j]
628.                    if totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm) <= initialc
     ost and twcheck(routes2n[k,:], k, vehdat, tws, tt, twe, st) == True and posotites(routes2n[k, :], k, vehd
     at, reverse1, reverse2, demand1, demand2, n2) == True:
629.                        routesopt = np.copy(routes2n)
630.                        initialcost = totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, d
     m)
631.                        www = True
632.                    new = new + 1
633.            k = k + 1
634.        if www == True:
635.            routes2 = np.copy(routesopt)
```

```
636.     return routes2
637.
638.   #Insertion operator 3
639.   def insertion_3(noc, routes2, cust, vehdat, fcc1, sumnv, n2, dm, tws, twe, tt, st, demand1, reverse1, d
       emand2, reverse2):
640.     while cust.sum() != 0:
641.       i = random.randint(0, noc-1)
642.       while cust[i] == 0:
643.         i = random.randint(0, noc-1)
644.       customerchosen = int(cust[i])
645.       cust[i] = 0
646.       initialcost = 99999999
647.       www = False
648.       k = 0
649.       zeros = 0
650.       while k in range(0, sumnv) and zeros < 10:
651.         if routes2[k, 1] == 0:
652.           zeros = zeros + 1
653.         if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(rou
       tes2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, reve
       rse1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[cus
       tomerchosen] <= vehdat[k, 3]:
654.           sk = np.count_nonzero(routes2[k, :])
655.           routes2n = np.copy(routes2)
656.           new = 1
657.           while new in range(1, sk+2):
658.             routes2n = np.copy(routes2)
659.             for j in range(0, new):
660.               routes2n[k, j] = routes2[k, j]
661.             routes2n[k, new] = customerchosen
662.             for j in range(new, sk+1):
663.               routes2n[k, j+1] = routes2[k, j]
664.             if totdistance(routes2n, dm, n2) <= initialcost and twcheck(routes2n[k,:], k, vehdat, tws, tt,
       twe, st) == True and posotites(routes2n[k, :], k, vehdat, reverse1, reverse2, demand1, demand2, n2) =
       = True :
665.               routesopt = np.copy(routes2n)
666.               initialcost = totdistance(routes2n, dm, n2)
667.               www = True
668.             new = new + 1
669.         k = k + 1
670.       if www == True:
671.         routes2 = np.copy(routesopt)
672.     return routes2
673.
674.   #Insertion operator 4
675.   def insertion_4(noc, routes2, cust, vehdat, fcc1, sumnv, n2, dm, tws, twe, tt, st, demand1, reverse1, d
       emand2, reverse2):
676.     while cust.sum() != 0:
```

```
677.            i = random.randint(0, noc-1)
678.            while cust[i] == 0:
679.                i = random.randint(0, noc-1)
680.            customerchosen = int(cust[i])
681.            cust[i] = 0
682.            initialcost = 99999999
683.            www = False
684.            k = 0
685.            zeros = 0
686.            while k in range(0, sumnv) and zeros < 10:
687.                if routes2[k, 1] == 0:
688.                    zeros = zeros + 1
689.                if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(rou
        tes2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, reve
        rse1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[cus
        tomerchosen] <= vehdat[k, 3]:
690.                    sk = np.count_nonzero(routes2[k, :])
691.                    routes2n = np.copy(routes2)
692.                    new = 1
693.                    while new in range(1, sk+2):
694.                        routes2n = np.copy(routes2)
695.                        for j in range(0, new):
696.                            routes2n[k, j] = routes2[k, j]
697.                        routes2n[k, new] = customerchosen
698.                        for j in range(new, sk+1):
699.                            routes2n[k, j+1] = routes2[k, j]
700.                        if dm[int(routes2n[k, new-
        1]), customerchosen]/dm.max() + (twe[customerchosen] + tt[int(routes2n[k, new-
        1]),customerchosen])/twe.max() <= initialcost and twcheck(routes2n[k,:], k, vehdat, tws, tt, twe, st) ==
         True and posotites(routes2n[k, :], k, vehdat, reverse1, reverse2, demand1, demand2, n2) == True:
701.                            routesopt = np.copy(routes2n)
702.                            initialcost = dm[int(routes2n[k, new-
        1]), customerchosen]/dm.max() + (twe[customerchosen] + tt[int(routes2n[k, new-
        1]),customerchosen])/twe.max()
703.                            www = True
704.                        new = new + 1
705.                k = k + 1
706.            if www == True:
707.                routes2 = np.copy(routesopt)
708.        return routes2
709.
710.    #Insertion operator 5
711.    def insertion_5(noc, routes2, cust, vehdat, fcc1, sumnv, n2, dm, tws, twe, tt, st, demand1, reverse1, d
        emand2, reverse2):
712.        while cust.sum() != 0:
713.            i = random.randint(0, noc-1)
714.            while cust[i] == 0:
715.                i = random.randint(0, noc-1)
```

```
716.          customerchosen = int(cust[i])
717.          cust[i] = 0
718.          initialcost = 99999999
719.          www = False
720.          k = 0
721.          zeros = 0
722.          while k in range(0, sumnv) and zeros < 10:
723.             if routes2[k, 1] == 0:
724.                zeros = zeros + 1
725.             if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(rou
       tes2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, reve
       rse1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[cus
       tomerchosen] <= vehdat[k, 3]:
726.                sk = np.count_nonzero(routes2[k, :])
727.                routes2n = np.copy(routes2)
728.                new = 1
729.                while new in range(1, sk+2):
730.                   routes2n = np.copy(routes2)
731.                   for j in range(0, new):
732.                      routes2n[k, j] = routes2[k, j]
733.                   routes2n[k, new] = customerchosen
734.                   for j in range(new, sk+1):
735.                      routes2n[k, j+1] = routes2[k, j]
736.                   criterion1 = totc(routes2[k,:], vehdat[k,:], n2, dm)
737.                   if criterion1 == 0:
738.                      criterion1 = vehdat[k,1]
739.                   if totc(routes2n[k,:], vehdat[k,:], n2, dm) - criterion1 <= initialcost and twcheck(routes2n[k
       ,:], k, vehdat, tws, tt, twe, st) == True and posotites(routes2n[k, :], k, vehdat, reverse1, reverse2, dema
       nd1, demand2, n2) == True:
740.                      routesopt = np.copy(routes2n)
741.                      initialcost = totc(routes2n[k,:], vehdat[k,:], n2, dm) - criterion1
742.                      www = True
743.                   new = new + 1
744.             k = k + 1
745.          if www == True:
746.             routes2 = np.copy(routesopt)
747.       return routes2
748.
749.    #Insertion of unrouted customers to existing routes
750.    def insertion_unrouted(routes2, cust, vehdat, sumnv, n2, dm, tws, twe, tt, st, demand1, reverse1, de
       mand2, reverse2, customers1):
751.       cust1 = np.copy(cust)
752.       while cust1.sum() != 0:
753.          i = random.randint(0, len(cust)-1)
754.          while cust1[i] == 0:
755.             i = random.randint(0, len(cust)-1)
756.          customerchosen = int(cust[i])
757.          cust1[i] = 0
```

```
758.          initialcost = 99999999
759.          www = False
760.          k = 0
761.          while k in range(0, sumnv):
762.             if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(routes2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, reverse1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[customerchosen] <= vehdat[k, 3]:
763.                sk = np.count_nonzero(routes2[k, :])
764.                routes2n = np.copy(routes2)
765.                new = 1
766.                while new in range(1, sk+2):
767.                   routes2n = np.copy(routes2)
768.                   for j in range(0, new):
769.                      routes2n[k, j] = routes2[k, j]
770.                   routes2n[k, new] = customerchosen
771.                   for j in range(new, sk+1):
772.                      routes2n[k, j+1] = routes2[k, j]
773.                   if totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm) <= initialcost and twcheck(routes2n[k,:], k, vehdat, tws, tt, twe, st) == True and posotites(routes2n[k, :], k, vehdat, reverse1, reverse2, demand1, demand2, n2) == True:
774.                      routesopt = np.copy(routes2n)
775.                      initialcost = totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm)
776.                      www = True
777.                      cust[i] = 0
778.                   new = new + 1
779.             k = k + 1
780.          if www == True:
781.             routes2 = np.copy(routesopt)
782.       k1 = np.count_nonzero(cust)
783.       unrouted2 = np.zeros(shape=(k1, 1))
784.       k = 0
785.       for i in range(len(cust)):
786.          if cust[i] != 0:
787.             unrouted2[k] = cust[i]
788.             k = k + 1
789.       return routes2, unrouted2
790.
791.    #Calculation of number of routes
792.    def number_of_routes(routes):
793.       s = 0
794.       for i in range(len(routes)):
795.          if routes[i, 1] != 0:
796.             s = s + 1
797.       return s
798.
799.    #Extra improvement 1
```

```
800.  def route_improvement(routes, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st, demand1, reverse1, de
      mand2, reverse2):
801.    sumroutes = routes.sum()
802.    routes21 = np.copy(routes)
803.    routes22 = np.copy(routes)
804.    kkk1 = 0
805.    i = random.randint(0, len(routes)-1)
806.    while (vehdattt[i, 2] == vehdattt[:, 2].min() or routes[i,1] == 0) and kkk1 <50:
807.      i = random.randint(0, len(routes)-1)
808.      kkk1 = kkk1 + 1
809.    if kkk1 < 50:
810.      cust= np.zeros(shape = (np.count_nonzero(routes[i, :]), 1))
811.      for j in range(np.count_nonzero(routes[i, :])):
812.        cust[j] = routes[i, j+1]
813.        routes21[i, j + 1] = 0
814.        routes22[i, j + 1] = 0
815.      k = 0
816.      www=False
817.      while k in range(len(routes)) and www == False:
818.        if i != k and vehdattt[i, 5] != vehdattt[k, 5] and totq1(routes[i, :], n2, demand1) <= vehdattt[k, 2]
       and posotites(routes22[k, :], k, vehdattt, reverse1, reverse2, demand1, demand2, n2) == True and tot
      c(routes[i,:], vehdattt[i,:], n2, dm) > totc(routes[i,:], vehdattt[k,:], n2, dm):
819.          for x in range(len(cust)):
820.            routes21[k, x+1] = cust[x]
821.          www = twcheck(routes21[k,:], k, vehdattt, tws, tt, twe, st)
822.        k = k + 1
823.    else:
824.      cust = np.zeros(shape=(1, 1))
825.    if routes21.sum() < sumroutes:
826.      routes21 = np.copy(routes)
827.    ss = 0
828.    nn = 1
829.    if number_of_routes(routes) < len(routes) and cust.sum()!=0:
830.      while cust.sum() != 0 and number_of_routes(routes22) < len(routes) and nn<=3:
831.        k = random.randint(0, len(routes)-1)
832.        while vehdattt[i, 5] == vehdattt[k, 5] or routes[k, 1] != 0:
833.          k = random.randint(0, len(routes)-1)
834.        w = True
835.        x = 1
836.        while ss in range(len(cust)) and w == True:
837.          routes22[k, x] = cust[ss]
838.          if twcheck(routes22[k,:], k, vehdattt, tws, tt, twe, st) == True and totq1(routes22[k, :], n2, de
      mand1) <= vehdattt[k, 2] and posotites(routes22[k, :], k, vehdattt, reverse1, reverse2, demand1, dem
      and2, n2) == True:
839.            cust[ss] = 0
840.            ss = ss + 1
841.            x = x + 1
842.          else:
```

```
843.                routes22[k, ss+1] = 0
844.                w = False
845.            nn = nn + 1
846.        noc = len(cust)
847.        if cust.sum() != 0:
848.            routes22 = insertion_5(noc, routes22, cust, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st, dem
    and1, reverse1, demand2, reverse2)
849.    if routes22.sum() < sumroutes:
850.        routes22 = np.copy(routes)
851.    if totalcost(routes21, vehdattt, n2, dm) < totalcost(routes22, vehdattt, n2, dm):
852.        routes_return = np.copy(routes21)
853.    else:
854.        routes_return = np.copy(routes22)
855.    return routes_return
856.
857.  #Extra improvement 2
858.  def route_improvement_correct2(routes, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st, demand1, re
    verse1, demand2, reverse2):
859.    sumroutes = routes.sum()
860.    routes21 = np.copy(routes)
861.    i = 0
862.    for i in range(sumnv):
863.        if (vehdattt[i, 2] != vehdattt[:, 2].min() and routes[i,1] != 0):
864.            cust= np.zeros(shape = (np.count_nonzero(routes[i, :]), 1))
865.            for j in range(np.count_nonzero(routes[i, :])):
866.                cust[j] = routes[i, j+1]
867.                routes21[i, j + 1] = 0
868.            k = 0
869.            www=False
870.            while k in range(len(routes)) and www == False:
871.                if i != k and vehdattt[k, 2] < vehdattt[i, 2] and totq1(routes[i, :], n2, demand1) >= vehdattt[k,
    2] and totc(routes[i,:], vehdattt[i,:], n2, dm) > totc(routes[i,:], vehdattt[k,:], n2, dm):
872.                    for x in range(len(cust)):
873.                        routes21[k, x+1] = cust[x]
874.                    www = twcheck(routes21[k,:], k, vehdattt, tws, tt, twe, st)
875.                k = k + 1
876.        if routes21.sum() < sumroutes:
877.            routes21 = np.copy(routes)
878.        routes_return = np.copy(routes21)
879.    return routes_return
880.
881.  #Routing procedure
882.  def routing(datveh, datcust, dminitial, ttinitial):
883.    #-------------------- Vehices' data ----------------------
884.    vtype = datveh.iloc[:,0].values
885.    capacity1 = datveh.iloc[:,2].values
886.    capacity2 = datveh.iloc[:,1].values
887.    fc = datveh.iloc[:,3].values
```

```
888.        vc = datveh.iloc[:,4].values
889.        availnv =datveh.iloc[:,5].values
890.        startingshift = datveh.iloc[:, 6]
891.        endingshift = datveh.iloc[:, 7]
892.        #----------------- Customers dataset ---------------------
893.        n2 = len(datcust)
894.        id = datcust.iloc[:,0].values
895.        cx = datcust.iloc[:,1].values
896.        cy = datcust.iloc[:,2].values
897.        demand1  = datcust.iloc[:,3].values
898.        demand2  = datcust.iloc[:,4].values
899.        reverse1 = datcust.iloc[:,5].values
900.        reverse2 = datcust.iloc[:,6].values
901.        tws = datcust.iloc[:, 7].values
902.        twe = datcust.iloc[:, 8].values
903.        st = datcust.iloc[:, 9].values
904.        #------------- Travel time and distance matrix -----------
905.        dm = np.copy(dminitial)
906.        tt = np.copy(ttinitial)
907.        #-----------------------------------------------------------
908.        sumnv=availnv.sum()
909.        customers1 = np.zeros(shape = (n2-1, 1))
910.        for i in range(1, n2):
911.            customers1[i-1] = id[i]
912.        sumroutes = customers1.sum()
913.        #----------------- Construction procedure ------------------
914.        customnew = np.zeros(shape =(n2-1, 1))
915.        vehdattt = vehdata(sumnv, availnv, fc, vc,capacity1, capacity2, startingshift, vtype, endingshift)
916.        routes, customers = construction1(sumnv, availnv, vehdattt, 0.85, 0.15, customers1, n2, tws, tt, twe
        , st, demand1, reverse1, demand2, reverse2, dm)
917.        #In case customers have not been integrated into routes are saved in array unrouted
918.        if routes.sum() == sumroutes and customers.sum() == 0:
919.            customnew = np.zeros(shape = (1,1))
920.            customnew[0] = 0
921.            unrouted = np.zeros(shape = (1, 1))
922.            unrouted[0] = 0
923.        else:
924.            mincustsum = countcust(customers)
925.            customnew = np.copy(customers)
926.            sumroutes = routes.sum()
927.            unrouted = np.zeros(shape = (mincustsum, 1))
928.            k = 0
929.            for i in range(len(customers1)):
930.                if customnew[i] != 0:
931.                    unrouted[k] = customnew[i]
932.                    k = k + 1
933.
934.        optcost = totalcost(routes, vehdattt, n2, dm)
```

```
935.        print("Construction",  optcost, unrouted.sum())
936.
937.        #Set of parameters
938.        if n2-1 < 10:
939.            dl = 1
940.            ul = 2
941.        elif n2-1 < 51:
942.            dl = 3
943.            ul = 8
944.        else:
945.            dl = 2
946.            ul = 14
947.
948.        pososto1 = 0.25
949.        pososto2 = 0.25
950.        pososto3 = 0.17
951.        pososto4 = 0.17
952.        pososto5 = 0.10
953.
954.        #Computation time limit
955.        timelimit = 900
956.
957.        r = 0.15
958.        select1 = 0
959.        select2 = 0
960.        select3 = 0
961.        select4 = 0
962.        select5 = 0
963.        select6 = 0
964.        ssss = 1
965.
966.        #Starting of the improvement  procedure
967.        i = 0
968.        while timeit.default_timer()  - start < timelimit:
969.            fcc1 = fcc(routes, vehdattt)
970.            p = random.random()
971.            if p <= pososto1:
972.                routes2, cust, noc = selection1(routes, vehdattt, dl, ul, sumnv, n2, customers1, customnew)
973.                s1 = 1
974.            elif p <=pososto1 + pososto2:
975.                routes2, cust, noc = selection2(routes, vehdattt, sumnv, dm, n2, demand1, demand2, reverse1,
        reverse2)
976.                s1 = 2
977.            elif p <= pososto1 + pososto2 + pososto3:
978.                routes2, cust, noc = selection3(routes, vehdattt, n2, tws, twe, tt, st, dl, dm, ul, sumnv)
979.                s1 = 3
980.            elif p <= pososto1 + pososto2 + pososto3+pososto4:
981.                routes2, cust, noc = selection4(routes, vehdattt, n2, tws, twe, tt, st, dl, dm, ul, sumnv)
```

```
982.            s1 = 4
983.        elif p <= pososto1 + pososto2 + pososto3 + pososto4 + pososto5:
984.            routes2, cust, noc = selection5(routes, vehdattt, n2, dl, ul, sumnv)
985.            s1 = 5
986.        else:
987.            routes2, cust, noc = selection55(routes, vehdattt, n2, dl, ul, sumnv)
988.            s1 = 55
989.        p1 = random.random()
990.        if p1 < 0.25:
991.            routes2 = insertion_1(noc, routes2, cust, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st, deman
    d1, reverse1, demand2, reverse2)
992.        elif p1 < 0.5:
993.            routes2 = insertion_2(noc, routes2, cust, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st, deman
    d1, reverse1, demand2, reverse2)
994.        elif p1 < 0.75:
995.            routes2 = insertion_3(noc, routes2, cust, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st, deman
    d1, reverse1, demand2, reverse2)
996.        else:
997.            routes2 = insertion_5(noc, routes2, cust, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st, deman
    d1, reverse1, demand2, reverse2)
998.
999.        #Adating the parameters
1000.       if totalcost(routes2, vehdattt, n2, dm) < optcost - 0.001 and routes2.sum() == sumroutes:
1001.           routes = np.copy(routes2)
1002.           optcost = (totalcost(routes, vehdattt, n2, dm))
1003.           if s1 == 1:
1004.               select1 = select1 +1
1005.           elif s1 == 2:
1006.               select2 = select2 +1
1007.           elif s1 == 3:
1008.               select3 = select3 +1
1009.           elif s1 == 4:
1010.               select4 = select4 +1
1011.           elif s1 == 5:
1012.               select5 = select5 + 1
1013.           else:
1014.               select6 = select6 + 1
1015.           if ssss%9==0:
1016.               pososto1 = pososto1*(1 - r) + r*select1/(ssss)
1017.               pososto2 = pososto2*(1 - r) + r*select2/(ssss)
1018.               pososto3 = pososto3*(1 - r) + r*select3/(ssss)
1019.               pososto4 = pososto4*(1 - r) + r*select4/(ssss)
1020.               pososto5 = pososto5*(1 - r) + r*select5/(ssss)
1021.           ssss = ssss + 1
1022.
1023.       if i%5==0 and timeit.default_timer() - start > 0.55*timelimit:
1024.           routes2 = route_improvement_correct2(routes, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st,
    demand1, reverse1, demand2, reverse2)
```

```
1025.             if totalcost(routes2, vehdattt, n2, dm) < optcost - 0.001 and routes2.sum() == sumroutes:
1026.                 routes = np.copy(routes2)
1027.                 optcost = (totalcost(routes, vehdattt, n2, dm))
1028.                 print("route_improvement_correct2")
1029.         if i%5==0 and timeit.default_timer() - start > 0.55*timelimit:
1030.             routes2 = route_improvement(routes, vehdattt, fcc1, sumnv, n2, dm, tws, twe, tt, st, demand1,
      reverse1, demand2, reverse2)
1031.             if totalcost(routes2, vehdattt, n2, dm) < optcost - 0.001 and routes2.sum() == sumroutes:
1032.                 routes = np.copy(routes2)
1033.                 optcost = (totalcost(routes, vehdattt, n2, dm))
1034.                 print("route_improvement")
1035.
1036.         #Try to integrate the unrouted customers into the routes
1037.         if i%15 == 0 and unrouted.sum() > 0:
1038.             routes, unrouted = insertion_unrouted(routes, unrouted, vehdattt, sumnv, n2, dm, tws, twe, tt,
      st, demand1, reverse1, demand2, reverse2, customers1)
1039.             sumroutes = routes.sum()
1040.             optcost = (totalcost(routes, vehdattt, n2, dm))
1041.             print(unrouted.sum())
1042.         i=i+1
1043.     optimalroutexx = np.copy(routes)
1044.     tstart, toa, tod, ttravel = times(optimalroutexx, n2, tws, tt, twe, st, vehdattt)
1045.     vehicles = quantity(optimalroutexx, vehdattt, n2, demand1, demand2, dm, toa, tstart, tod, ttravel)
1046.     posotites1, posotites2 = analytikaposotites(optimalroutexx, n2, demand1, demand2, reverse1, reve
     rse2)
1047.     print(optcost)
1048.     #data that are returned after the routing procedure
1049.     return vehicles, optimalroutexx, toa, tod, vehdattt, optcost
1050.
1051. stop = timeit.default_timer()
```

Dynamic Routing – Rerouting

```
import random
import numpy as np
import pandas as pd
import timeit

start = timeit.default_timer()

#Initial plan of routes, created by the ALNS algorithm
filename1 = "static_routes.xlsx"
routes = pd.read_excel(filename1)

#Data of vehicles
filename2 = "vehdattt.xlsx"
datveh = pd.read_excel(filename2)

#Data of Customers
filename3 = "customers1.xlsx"
datcust = pd.read_excel(filename3)

#Customers that have been served
filename4 = "servedcustomers.xlsx"
servedcustomers = pd.read_excel(filename4)

#Current time
tnow = 410

#Number of route we focus on
nroute = 4

#Check if the load on board of each vehicle, in every stage of the distribution is respected or not
def posotitesrerouitng(routes, position, vehdat, reverse1, reverse2, demand1, demand2, n2, q1, q2):
    rrr1 = np.count_nonzero(routes)
    q = True
    d1 = q1
    d2 = q2
    i = 1
    while i in range(1,rrr1+2) and q == True:
        d1 = d1 - demand1[int(routes[i])] + reverse1[int(routes[i])]
        d2 = d2 - demand2[int(routes[i])] + reverse2[int(routes[i])]
        if d1 > vehdat[position,2] or d2 > vehdat[position,3]:
            q = False
            #print(d1, vehdat[position,2], d2, vehdat[position,3])
        i = i+1
    return q

#Calculation of load on truck in every step of the distribution
```

```
47. def analytikaposotitesrerouting(routes, n2, demand1, demand2, reverse1, reverse2):
48.     posotites1 = np.zeros(shape = (n2+1, 1))
49.     posotites2 = np.zeros(shape = (n2+1, 1))
50.     posotites1[0] = totq1(routes, n2, demand1)
51.     posotites2[0] = totq2(routes, n2, demand2)
52.     sk = np.count_nonzero(routes)
53.     for j in range(1, sk + 2):
54.         posotites1[j] = posotites1[j-1]- demand1[int(routes[j])] + reverse1[int(routes[j])]
55.         posotites2[j] = posotites2[j-1]- demand2[int(routes[j])] + reverse2[int(routes[j])]
56.     return posotites1, posotites2
57. #Second Calculation of load on truck in every step of the distribution
58. def analytikaposotitesrerouting2(routes, n2, demand1, demand2, reverse1, reverse2, q1, q2):
59.     posotites1 = np.zeros(shape = (n2+1, 1))
60.     posotites2 = np.zeros(shape = (n2+1, 1))
61.     posotites1[0] = q1
62.     posotites2[0] = q2
63.     sk = np.count_nonzero(routes)
64.     for j in range(1, sk + 2):
65.         posotites1[j] = posotites1[j-1]- demand1[int(routes[j])] + reverse1[int(routes[j])]
66.         posotites2[j] = posotites2[j-1]- demand2[int(routes[j])] + reverse2[int(routes[j])]
67.     return posotites1, posotites2
68.
69. #Check that all time windows of a single route are respected
70. def twcheckrerouting(routes, position, vehdat, tws, tt, twe, st, tnow):
71.     r1 = np.count_nonzero(routes)
72.     t = tnow
73.     tstart = tnow
74.     q = True
75.     i = 0
76.     while i in range(0,r1+1) and q == True:
77.         t = t + tt[int(routes[i]), int(routes[i+1])]
78.         if t > twe[int(routes[i+1])]:
79.             q = False
80.         else:
81.             if t < tws[int(routes[i+1])]:
82.                 t = tws[int(routes[i+1])]
83.             t = t + st[int(routes[i+1])]
84.         i = i + 1
85.     if t - tstart > vehdat[position, 4]:
86.         q = False
87.     return q
88.
89. #Calculation of delays
90. def delays(routes, vehdat, tws, tt, twe, st, tnow):
91.     r1 = np.count_nonzero(routes)
92.     t = tnow
93.     delay = 0
94.     i = 0
```

```
    while i in range(0,r1+1):
        t = t + tt[int(routes[i]), int(routes[i+1])]
        if t > twe[int(routes[i+1])]:
            delay = delay + t - twe[int(routes[i+1])]
        else:
            if t < tws[int(routes[i+1])]:
                t = tws[int(routes[i+1])]
            t = t + st[int(routes[i+1])]
        i = i + 1
    return delay

#New calculation of the toa, tod, waiting time, and start of serving in each customer
def timesrerouting(routes, n2, tws, tt, twe, st, tnow):
    waiting = np.zeros(shape = (n2-1,1))
    toa = np.zeros(shape = (n2+1, 1))
    tstart = np.zeros(shape = (n2+1,1))
    tod = np.zeros(shape = (n2+1, 1))
    ttravel = np.zeros(shape = (3, 1))
    k = 0
    t = tnow
    toa[0] = tnow
    tod[0] = tnow
    smax = np.count_nonzero(routes)
    for j in range(0, smax+1):
        t = t + tt[int(routes[j]), int(routes[j+1])]
        if t < tws[int(routes[j+1])]:
            tstart[j+1] = tws[int(routes[j+1])]
            waiting[k] = tws[int(routes[j+1])] - t
            k = k +1
        else:
            tstart[j+1] = t
            waiting[k] = 0
        toa[j+1] = t
        t = tstart[j+1] + st[int(routes[j+1])]
        tod[j+1] = t
    ttravel[1] = toa.max()
    ttravel[2] = toa.max() - ttravel[0]
    return tstart, toa, tod, ttravel

def rerouting(datcust, datveh, routesall, nroute, servedcustomers, tnow, dm, tt):
    #----------------- Vehices' data --------------
    vehdattt = datveh.iloc[:,:].values
    #----------------- Routes ----------------------
    routes = routesall.iloc[:,:].values
    #-------------- Customers dataset -------------
    n2 = len(datcust)
    id = datcust.iloc[:,0].values
    cx = datcust.iloc[:,1].values
```

```
cy = datcust.iloc[:,2].values
demand1  = datcust.iloc[:,3].values
demand2  = datcust.iloc[:,4].values
reverse1 = datcust.iloc[:,5].values
reverse2 = datcust.iloc[:,6].values
tws = datcust.iloc[:, 7].values
twe = datcust.iloc[:, 8].values
st = datcust.iloc[:, 9].values
#-------------------------------------------------
customers1 = np.zeros(shape = (n2-1, 1))
for i in range(1, n2):
    customers1[i-1] = id[i]
#-------------------------------------------------
checkroute = np.copy(routes[nroute,:])
#---- find the unserved customers ---------
unserved = np.zeros(shape=(n2+1, 1))
nuns = len(servedcustomers)
s = np.count_nonzero(routes[nroute,:])
for j in range(nuns+1, s+1):
    unserved[j-nuns-1] = checkroute[j]
#--- find the unserved customers and the last served one ------
unservedwithstartpoint = np.zeros(shape=(n2+1, 1))
for j in range(nuns, s+1):
    unservedwithstartpoint[j-nuns] = checkroute[j]
sumroutes = unservedwithstartpoint.sum()
#---- initial quantities in the route -------
posotites1, posotites2 = analytikaposotitesrerouting(checkroute, n2, demand1, demand2, reverse1,
reverse2)
q1 = posotites1[nuns]
q2 = posotites2[nuns]
#print(q1, q2)
#-------------------------------------------------------------
if twcheckrerouting(unservedwithstartpoint, nroute, vehdattt, tws, tt, twe, st, tnow) == True:
    print("You can proceed as planned")
    routenew = np.copy(unservedwithstartpoint)
    tstart, toa, tod, ttravel=timesrerouting(routenew, n2, tws, tt, twe, st, tnow)
    posotites1, posotites2 = analytikaposotitesrerouting2(routenew, n2, demand1, demand2, reverse
1, reverse2, q1, q2)
else:
    print("There will be delays")
    routenew = np.zeros(shape=(n2+1, 1))
    cust = np.trim_zeros(unserved)
    noc = np.count_nonzero(unserved)
    initialcost2 = 999999999
    for i in range(1000):
        cust1 = np.copy(cust)
        routes2 = np.zeros(shape=(n2+1, 1))
        routes2[0] = unservedwithstartpoint[0]
```

```
189.          while cust1.sum() != 0:
190.              i = random.randint(0,  noc-1)
191.              while cust1[i] == 0:
192.                  i = random.randint(0,  noc-1)
193.              www = False
194.              customerchosen = int(cust1[i])
195.              cust1[i] = 0
196.              initialcost = 99999999
197.              sk = np.count_nonzero(routes2)
198.              routes2n = np.copy(routes2)
199.              new = 1
200.              while new in range(1, sk+1):
201.                  routes2n = np.copy(routes2)
202.                  for j in range(0, new):
203.                      routes2n[j] = routes2[j]
204.                  routes2n[new]  = customerchosen
205.                  for j in range(new, sk+1):
206.                      routes2n[j+1] = routes2[j]
207.                  if totc(routes2n, vehdattt[nroute,:],  n2, dm) - totc(routes2, vehdattt[nroute,:],  n2, dm) <= i
    nitialcost and twcheckrerouting(routes2n,  nroute,  vehdattt, tws, tt, twe, st, tnow) == True and posotit
    esrerouitng(routes2n, nroute, vehdattt, reverse1, reverse2, demand1, demand2, n2, q1, q2) == True:
208.                      routesopt = np.copy(routes2n)
209.                      initialcost = totc(routes2n, vehdattt[nroute,:],  n2, dm) - totc(routes2, vehdattt[nroute,:],
    n2, dm)
210.                      www = True
211.                  new = new + 1
212.              if www == True:
213.                  routes2 = np.copy(routesopt)
214.          tcnew = totc(routes2, vehdattt[nroute,:],  n2, dm)
215.          if routes2.sum() == sumroutes and cust1.sum() == 0 and tcnew < initialcost2:
216.              routenew = np.copy(routes2)
217.              initialcost2 = totc(routes2, vehdattt[nroute,:],  n2, dm)
218.      if routenew.sum()  == 0:
219.          for i in range(10):
220.              cust1 = np.copy(cust)
221.              routes2 = np.zeros(shape=(n2+1, 1))
222.              routes2[0] = unservedwithstartpoint[0]
223.              while cust1.sum() != 0:
224.                  i = random.randint(0,  noc-1)
225.                  while cust1[i] == 0:
226.                      i = random.randint(0,  noc-1)
227.                  www = False
228.                  customerchosen = int(cust1[i])
229.                  cust1[i] = 0
230.                  initialcost = 99999999
231.                  routes2n = np.copy(routes2)
232.                  sk = np.count_nonzero(routes2)
233.                  new = 1
```

```
234.                while new in range(1, sk+1):
235.                    routes2n = np.copy(routes2)
236.                    for j in range(0, new):
237.                        routes2n[j] = routes2[j]
238.                    routes2n[new] = customerchosen
239.                    for j in range(new, sk+1):
240.                        routes2n[j+1] = routes2[j]
241.                    if delays(routes2n, vehdattt[nroute,:], tws, tt, twe, st, tnow) - delays(routes2, vehdattt[n
    route,:], tws, tt, twe, st, tnow)<= initialcost and posotitesrerouitng(routes2n, nroute, vehdattt, revers
    e1, reverse2, demand1, demand2, n2, q1, q2) == True:
242.                        routesopt = np.copy(routes2n)
243.                        initialcost = delays(routes2n, vehdattt[nroute,:], tws, tt, twe, st, tnow) - delays(routes
    2, vehdattt[nroute,:], tws, tt, twe, st, tnow)
244.                        www = True
245.                    new = new + 1
246.                if www == True:
247.                    routes2 = np.copy(routesopt)
248.            tcnew = delays(routes2, vehdattt[nroute,:], tws, tt, twe, st, tnow)
249.            if routes2.sum() == sumroutes and cust1.sum() == 0 and tcnew < initialcost2:
250.                routenew = np.copy(routes2)
251.                initialcost2 = totc(routes2, vehdattt[nroute,:], n2, dm)
252.          print("solution with the minimum delays is: ", routenew, "with, ", initialcost2)
253.       else:
254.          print("new route was found")
255.       tstart, toa, tod, ttravel=timesrerouting(routenew, n2, tws, tt, twe, st, tnow)
256.       posotites1, posostites2 = analytikaposotitesrerouting2(routenew, n2, demand1, demand2, revers
    e1, reverse2, q1, q2)
257.    return vehdattt,routenew, posotites1, posostites2, toa, tod, ttravel
258.
259.vehdattt,routenew, posotites1, posostites2, toa, tod, ttravel = rerouting(datcust, datveh, routes, nrou
    te, servedcustomers, tnow, dm , tt)
260.
261.stop = timeit.default_timer()
```

Drag and Drop Procedure

```
1.  import random
2.  import numpy as np
3.  import pandas as pd
4.  import timeit
5.
6.  start = timeit.default_timer()
7.
8.  #Initial plan of routes, created by the ALNS algorithm
9.  filename1 = "static_routes.xlsx"
10. routes = pd.read_excel(filename1)
11.
12. #Data of vehicles
13. filename2 = "vehdattt.xlsx"
14. datveh = pd.read_excel(filename2)
15.
16. #Data of Customers
17. filename3 = "customers1.xlsx"
18. datcust = pd.read_excel(filename3)
19.
20. #Customers that are selected (dragged)
21. filename4 = "draged.xlsx"
22. sel_cust = pd.read_excel(filename4)
23. nroute = 22
24.
25. def times(routes, n2, tws, tt, twe, st, vehdat):
26.     customers = np.zeros(shape = (n2-1,1))
27.     waiting = np.zeros(shape = (n2-1,1))
28.     toa = np.zeros(shape = (len(routes), n2+1))
29.     tstart = np.zeros(shape = (len(routes), n2+1))
30.     tod = np.zeros(shape = (len(routes), n2+1))
31.     ttravel = np.zeros(shape = (len(routes), 3))
32.     k = 0
33.     for i in range(0, len(routes)):
34.         t = vehdat[i, 4]
35.         toa[i, 0] = vehdat[i, 4]
36.         tod[i, 0] = vehdat[i, 4]
37.         smax = np.count_nonzero(routes[i,:])
38.         for j in range(0, smax+1):
39.             t = t + tt[int(routes[i,j]), int(routes[i, j+1])]
40.             if j ==0 and t <= tws[int(routes[i, j+1])]:
41.                 ttravel[i,0] = tws[int(routes[i, j+1])] - tt[int(routes[i, j]), int(routes[i, j+1])]
42.                 tstart[i, j] = ttravel[i, 0]
43.                 tod[i, 0] = tws[int(routes[i, j+1])] - tt[int(routes[i, j]), int(routes[i, j+1])]
44.             elif j == 0 and t > tws[int(routes[i, j+1])]:
45.                 ttravel[i,0] = vehdat[i, 4]
```

```
46.                 tstart[i, j] = ttravel[i, 0]
47.                 tod[i, j] = 0
48.             if t < tws[int(routes[i, j+1])]:
49.                 tstart[i, j+1] = tws[int(routes[i, j+1])]
50.                 waiting[k] = tws[int(routes[i, j+1])] - t
51.                 customers[k] = int(routes[i, j+1])
52.                 k = k +1
53.             else:
54.                 tstart[i, j+1] = t
55.                 waiting[k] = 0
56.                 customers[k] = int(routes[i, j+1])
57.             toa[i,j+1] = t
58.             t = tstart[i, j+1] + st[int(routes[i, j+1])]
59.             tod[i, j+1] = t
60.         ttravel[i,1] = toa[i,:].max()
61.         ttravel[i,2] = toa[i,:].max() - ttravel[i,0]
62.     return tstart, toa, tod, ttravel
63.
64. def din1(routes2, cust, vehdat, fcc1, nroute, n2, dm, tws, twe, tt, st, demand1, reverse1, demand2, re
    verse2):
65.     custreturn = np.copy(cust)
66.     while cust.sum() != 0:
67.         i = random.randint(0, len(cust)-1)
68.         while cust[i] == 0:
69.             i = random.randint(0, len(cust)-1)
70.         customerchosen = int(cust[i])
71.         cust[i] = 0
72.         initialcost = 99999999
73.         www = False
74.         k = nroute
75.         if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(route
    s2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, revers
    e1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[custo
    merchosen] <= vehdat[k, 3]:
76.             sk = np.count_nonzero(routes2[k, :])
77.             routes2n = np.copy(routes2)
78.             new = 1
79.             while new in range(1, sk+2):
80.                 routes2n = np.copy(routes2)
81.                 for j in range(0, new):
82.                     routes2n[k, j] = routes2[k, j]
83.                 routes2n[k, new] = customerchosen
84.                 for j in range(new, sk+1):
85.                     routes2n[k, j+1] = routes2[k, j]
86.                 if totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm) <= initialcost
    and fcc(routes2n, vehdat) <= fcc1 and twcheck(routes2n[k,:], k, vehdat, tws, tt, twe, st) == True and p
    ososites(routes2n[k, :], k, vehdat, reverse1, reverse2, demand1, demand2, n2) == True:
87.                     routesopt = np.copy(routes2n)
```

```
88.                 initialcost = totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm)
89.                 www = True
90.             new = new + 1
91.         if www == True:
92.             routes2 = np.copy(routesopt)
93.             custreturn[i]=0
94.     return routes2, custreturn
95.
96. def din2(routes2, cust, vehdat, fcc1, nroute, n2, dm, tws, twe, tt, st, demand1, reverse1, demand2, re
    verse2):
97.     custreturn = np.copy(cust)
98.     while cust.sum() != 0:
99.         i = random.randint(0, len(cust)-1)
100.        while cust[i] == 0:
101.            i = random.randint(0, len(cust)-1)
102.        customerchosen = int(cust[i])
103.        cust[i] = 0
104.        initialcost = 99999999
105.        www = False
106.        k = nroute
107.        if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(route
    s2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, revers
    e1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[custo
    merchosen] <= vehdat[k, 3]:
108.            sk = np.count_nonzero(routes2[k, :])
109.            routes2n = np.copy(routes2)
110.            new = 1
111.            while new in range(1, sk+2):
112.                routes2n = np.copy(routes2)
113.                for j in range(0, new):
114.                    routes2n[k, j] = routes2[k, j]
115.                routes2n[k, new] = customerchosen
116.                for j in range(new, sk+1):
117.                    routes2n[k, j+1] = routes2[k, j]
118.                if totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm) <= initialcost
     and twcheck(routes2n[k,:], k, vehdat, tws, tt, twe, st) == True and posotites(routes2n[k, :], k, vehdat, r
    everse1, reverse2, demand1, demand2, n2) == True:
119.                    routesopt = np.copy(routes2n)
120.                    initialcost = totc(routes2n[k,:], vehdat[k,:], n2, dm) - totc(routes2[k,:], vehdat[k,:], n2, dm)
121.                    www = True
122.                new = new + 1
123.        if www == True:
124.            routes2 = np.copy(routesopt)
125.            custreturn[i] = 0
126.    return routes2, custreturn
127.
128. def din3(routes2, cust, vehdat, fcc1, nroute, n2, dm, tws, twe, tt, st, demand1, reverse1, demand2, re
    verse2):
```

```
129.        custreturn = np.copy(cust)
130.        while cust.sum() != 0:
131.            i = random.randint(0, len(cust)-1)
132.            while cust[i] == 0:
133.                i = random.randint(0, len(cust)-1)
134.            customerchosen = int(cust[i])
135.            cust[i] = 0
136.            initialcost = 99999999
137.            www = False
138.            k = int(nroute)
139.            if totq1(routes2[k, :], n2, demand1) + demand1[customerchosen] <= vehdat[k, 2] and totq2(route
        s2[k, :], n2, demand2) + demand2[customerchosen] <= vehdat[k, 3] and totr1(routes2[k, :], n2, revers
        e1) + reverse1[customerchosen] <= vehdat[k, 2] and totr2(routes2[k, :], n2, reverse2) + reverse2[custo
        merchosen] <= vehdat[k, 3]:
140.                sk = np.count_nonzero(routes2[k, :])
141.                routes2n = np.copy(routes2)
142.                new = 1
143.                while new in range(1, sk+2):
144.                    routes2n = np.copy(routes2)
145.                    for j in range(0, new):
146.                        routes2n[k, j] = routes2[k, j]
147.                    routes2n[k, new] = customerchosen
148.                    for j in range(new, sk+1):
149.                        routes2n[k, j+1] = routes2[k, j]
150.                    if totdistance(routes2n, dm, n2) <= initialcost and twcheck(routes2n[k,:], k, vehdat, tws, tt, t
        we, st) == True and posotites(routes2n[k, :], k, vehdat, reverse1, reverse2, demand1, demand2, n2) ==
         True :
151.                        routesopt = np.copy(routes2n)
152.                        initialcost = totdistance(routes2n, dm, n2)
153.                        www = True
154.                    new = new + 1
155.            if www == True:
156.                routes2 = np.copy(routesopt)
157.                custreturn[i] = 0
158.        return routes2, custreturn
159.
160.    #Since we have selected the customers, we need to transfer them in a specific route, and therefore
        we check the feasibility
161.    #Input 1: The initial plan of routes
162.    #Input 2: the initial data of vehicles
163.    #Input 3: Customers' Data
164.    #Input 4: Array- Matrix of Distances
165.    #Input 5: Array- Matrix of Travel Times between Customers
166.    #Input 6: The id number of the route in which we want to tranfer the customers
167.    #Input 7: The selected customers
168.
169.    def dragndrop(routesall, datveh, datcust, dm, tt, nroute, sel_cust):
170.        #----------------- Vehices' data ----------------------
```

```
171.        vehdattt = datveh.iloc[:,:].values
172.        #---------- Routes ----------
173.        routes = routesall.iloc[:,:].values
174.        #----------
175.        cust = sel_cust.iloc[:, :].values
176.        nroute = int(nroute)
177.        #---------- Customers dataset ----------
178.        n2 = len(datcust)
179.        id = datcust.iloc[:,0].values
180.        cx = datcust.iloc[:,1].values
181.        cy = datcust.iloc[:,2].values
182.        demand1  = datcust.iloc[:,3].values
183.        demand2  = datcust.iloc[:,4].values
184.        reverse1 = datcust.iloc[:,5].values
185.        reverse2 = datcust.iloc[:,6].values
186.        tws = datcust.iloc[:, 7].values
187.        twe = datcust.iloc[:, 8].values
188.        st = datcust.iloc[:, 9].values
189.        #----------
190.        customers1 = np.zeros(shape = (n2-1, 1))
191.        for i in range(1, n2):
192.            customers1[i-1] = id[i]
193.        #----------
194.        #print(totalcost(routes, vehdattt, n2, dm))
195.        sssss=0
196.        routes2 = np.copy(routes)
197.        for sssss in range(120):
198.            p1 = random.random()
199.            fcc1 = fcc(routes, vehdattt)
200.            if p1 < 0.33:
201.                routes2, cust = din1(routes2, cust, vehdattt, fcc1, nroute, n2, dm, tws, twe, tt, st, demand1, rev
    erse1, demand2, reverse2)
202.            elif p1 < 0.67:
203.                routes2, cust = din2(routes2, cust, vehdattt, fcc1, nroute, n2, dm, tws, twe, tt, st, demand1, rev
    erse1, demand2, reverse2)
204.            else:
205.                routes2, cust = din3(routes2, cust, vehdattt, fcc1, nroute, n2, dm, tws, twe, tt, st, demand1, rev
    erse1, demand2, reverse2)
206.        routesfinal = np.zeros(shape=(len(vehdattt), len(datcust) + 1))
207.
208.    #Since we have completed the check, we export the final plan
209.    #If a customer has the ability to be tranfered in the selected route, it is removed from the existing one
210.        for i in range(len(routes)):
211.            if i != nroute and routes[i,:].sum() != 0:
212.                k1 = 1
213.                for j in range(1, np.count_nonzero(routes2[i,:])+1):
214.                    if routes[i, j] not in routes2[nroute, :]:
215.                        routesfinal[i,k1] = routes[i, j]
```

```
216.                    k1 = k1 + 1
217.            else:
218.                routesfinal[nroute] = np.copy(routes2[nroute, :])
219.            tstart, toa, tod, ttravel=times(routesfinal, n2, tws, tt, twe, st, vehdattt)
220.            posotites1, pososti tes2 = analytikaposotites(routesfinal, n2, demand1, demand2, reverse1, rever
        se2)
221.        #Return the final plan of routes
222.        return vehdattt,routesfinal, posotites1, pososti tes2, toa, tod, ttravel
223.
224.    vehdattt,routenew, posotites1, pososti tes2, toa, tod, ttravel = dragndrop(routes, datveh, datcust, dm,
        tt, nroute, sel_cust)
225.    stop = timeit.default_time
```

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Μέθοδοι Βελτιστοποίησης για Ένα Σύνολο Προβλημάτων Δρομολόγησης Οχημάτων και Προγραμματισμού Παραδόσεων

Διδακτορική Διατριβή – Εκτενής Περίληψη

Συντάκτης

Γρηγόριος Δ. Κωνσταντακόπουλος

Επιβλέπων

Δρ. Ηλίας Π. Τατσιόπουλος

Καθηγητής ΕΜΠ

Εργαστήριο Οργάνωσης Παραγωγής

Τομέας Βιομηχανικής Διοίκησης & Επιχειρησιακής Έρευνας

Σχολή Μηχανολόγων Μηχανικών

Ιούλιος 2021

Περιεχόμενα

Περίληψη

Στο πλαίσιο της παρούσας διδακτορικής διατριβής μελετάται και επιλύεται μία σειρά προβλημάτων δρομολόγησης οχημάτων και προγραμματισμού παραδόσεων, ενώ ταυτόχρονα αναπτύσσονται αποδοτικοί εξελικτικοί και τοπικής αναζήτησης μεταευρετικοί αλγόριθμοι οι οποίοι έχουν ως στόχο την βελτιστοποίηση της διανομής, προτείνοντας το βέλτιστο πλάνο δρομολογίων και παραδόσεων. Το γεγονός ότι η δρομολόγηση οχημάτων επηρεάζεται από πολλές μεταβλητές και περιορισμούς, οι οποίες έχουν προέλθει από τη φύση των εταιρειών διανομής, το εξωτερικό περιβάλλον αυτών, και από τις ανάγκες και τις απαιτήσεις των πελατών, έχει οδηγήσει τόσο στο εμπορικό/επιχειρηματικό όσο και ερευνητικό ενδιαφέρον. Προκειμένου να συνδεθούν οι δύο αυτοί τομείς, η παρούσα διατριβή μελετά και προτείνει την ενσωμάτωση των προτεινόμενων αλγόριθμων σε ένα cloud σύστημα δρομολόγησης.

Επιπλέον, οι διάφορες μεταβλητές και περιορισμοί του Προβλήματος της Δρομολόγησης Οχημάτων (ΠΔΟ) αντιστοιχούν σε διαφορετικές παραλλαγές του προβλήματος. Στην παρούσα διδακτορική διατριβή, οι παραλλαγές του προβλήματος που λαμβάνονται υπόψιν έχουν προκύψει μέσα από την αναλυτική ανασκόπηση της βιβλιογραφίας, καθώς και μέσω της συνεργασίας με εταιρείες διανομής και ανάπτυξης λογισμικού. Πιο συγκεκριμένα, καθορίζονται οι παραλλαγές που απασχολούν πιο συχνά τους ερευνητές και τους επαγγελματίες που δραστηριοποιούνται στο χώρο, όσο και οι παραλλαγές που έχουν τη δυναμική να προσφέρουν στο άμεσο μέλλον. Οι παραλλαγές του προβλήματος της δρομολόγησης που τελικά επιλέχθηκαν προς μελέτη είναι:

1. το ΠΔΟ με Χρονικά Παράθυρα (ΠΔΟΧΠ),
2. το ΠΔΟ με Ταυτόχρονες Περισυλλογές και Παραδόσεις (ΠΔΟΤΠΠ),
3. το ΠΔΟ με Ετερογενή Στόλο Οχημάτων (ΠΔΟΕΣΟ), και
4. το Συνεργατικό ΠΔΟ (ΣΠΔΟ).

Τα εξεταζόμενα προβλήματα δρομολόγησης, τα οποία ανήκουν στην κατηγορία των NP-hard προβλημάτων, παρουσιάζουν εξαιρετική πολυπλοκότητα, ειδικότερα όταν αντιμετωπίζονται ταυτόχρονα. Στην παρούσα διατριβή αντιμετωπίζονται όλες οι παραπάνω παραλλαγές είτε μεμονωμένα είτε συνδυαστικά προκειμένου να καλύπτουν τις ανάγκες και απαιτήσεις των εταιρειών διανομής που δραστηριοποιούνται στις εμπορευματικές μεταφορές. Συγκεκριμένα, το ΠΔΟΧΠΠ λαμβάνει υπόψιν το χρονικό εύρος που έχουν ορίσει οι πελάτες προκειμένου να πραγματοποιηθεί η παράδοση ή παραλαβή των εμπορευμάτων. Στο ΠΔΟΤΤΠ, όπως υποδεικνύει και η ονομασία, επικεντρώνεται σε παραδόσεις και επιστροφές που εκτελούνται ταυτόχρονα προς και από τους πελάτες. Όσον αφορά το ΠΔΟΕΣΟ, ο στόλος οχημάτων που διατηρούν και χρησιμοποιούν οι εταιρείες θεωρείται ετερογενής, που σημαίνει ότι τα οχήματα διαφέρουν όσον αφορά τη χωρητικότητά τους, και τα σταθερά και μεταβλητά κόστη διανομής. Τέλος, το ΣΠΔΟ επικεντρώνεται στις περιπτώσεις όπου εταιρείες διανομής που βρίσκονται

και λειτουργούν στις ίδιες περιοχές, συνεργάζονται στο κομμάτι της δρομολόγησης και διανομής.

Το γεγονός ότι με την ταυτόχρονη αντιμετώπιση πολλών παραλλαγών του προβλήματος δυσκολεύει η επίλυση του, καθώς και η ανάγκη των εταιρειών για τη δημιουργία αποδοτικών διαδρομών, έχουν οδηγήσει στην ανάγκη για ανάπτυξη αλγορίθμων βελτιστοποίησης. Αρχικά προτάθηκαν οι ακριβείς αλγόριθμοι από τους ερευνητές. Το πλεονέκτημά τους, το οποίο είναι η βέλτιστη αντιμετώπιση και επίλυση του ΠΔΟ συνοδεύεται από έναν μεγάλο περιορισμό ο οποίος είναι ο περιορισμένος αριθμός πελατών που μπορούν να διαχειριστούν καθώς και ο αυξημένος χρόνος επεξεργασίας που απαιτείται για την επίλυση. Αυτός ο περιορισμός επηρεάζει σημαντικά, οδηγώντας στην μη εφαρμογή αυτών σε συστήματα δρομολόγησης. Επίσης, είχε ως αποτέλεσμα το ερευνητικό ενδιαφέρον να στραφεί στους ευρετικούς αλγορίθμους (heuristic algorithms) οι οποίοι προσφέρουν ποιοτικές λύσεις σε ελάχιστο υπολογιστικό χρόνο. Στη συνέχεια οι ερευνητές προχώρησαν στην ανάπτυξη των μεταευρετικών αλγορίθμων (metaheuristic algorithms) οι οποίοι κατάφεραν να εξισορροπήσουν την ποιότητα της λύσης και τον υπολογιστικό χρόνο που απαιτείται.

Στη συγκεκριμένη έρευνα, μελετώνται και προτείνονται διάφοροι μεταευρετικοί αλγόριθμοι προκειμένου να επιλύσουν τα πολλαπλά προβλήματα δρομολόγησης που αντιμετωπίζουν οι εταιρείες του κλάδου των διανομών. Πιο συγκεκριμένα μελετάται ένας Εξελικτικός Αλγόριθμος (ΕΑ), καθώς και δύο αλγόριθμοι Τοπικής Αναζήτησης (ΤΑ). Οι αλγόριθμοι αυτοί αντιμετωπίζουν είτε μεμονωμένα, είτε συνδυαστικά ορισμένες παραλλαγές του προβλήματος της δρομολόγησης οχημάτων. Πιο συγκεκριμένα, ο ΕΑ αντιμετωπίζει το πρόβλημα της δρομολόγησης οχημάτων με ταυτόχρονες παραδόσεις και επιστροφές και με χρονικά παράθυρα. Επιπλέον, ο αλγόριθμος εφαρμόζεται σε περιπτώσεις διανομής με ηλεκτρικά οχήματα τα οποία έχουν μειωμένη αυτονομία. Η καινοτομία στον συγκεκριμένο αλγόριθμο συνδέεται με τον τελεστή εναλλαγής, καθώς επίσης και με το γεγονός ότι το πρόβλημα αντιμετωπίζεται ως πολύ-αντικειμενικό. Στην πολύ-αντικειμενική βελτιστοποίηση υπάρχουν δύο ή παραπάνω αντικρουόμενοι αντικειμενικοί στόχοι οι οποίοι πρέπει να βελτιστοποιηθούν. Στο συγκεκριμένο πρόβλημα είναι η ελαχιστοποίηση (i) του αριθμού των οχημάτων που απαιτούνται για την εκτέλεση των παραγγελιών και (ii) της συνολικά διανυόμενης απόστασης.

Όσον αφορά τους αλγόριθμους τοπικής αναζήτησης, ο πρώτος που αναπτύχθηκε και αντιμετωπίζει το πρόβλημα της δρομολόγησης οχημάτων με χρονικά παράθυρα ως πολύ-αντικειμενικό είναι αυτός της Αναζήτησης Ευρείας Γειτονιάς (ΑΕΓ). Ο δεύτερος αλγόριθμος τοπικής αναζήτησης που αναπτύχθηκε είναι αυτός της Προσαρμοστικής Αναζήτησης Ευρείας Γειτονιάς (ΠΡΑΕΓ), ο οποίος αντιμετωπίζει όλες τις προαναφερθείσες παραλλαγές του προβλήματος και πιο συγκεκριμένα το Πρόβλημα της Δρομολόγησης Οχημάτων με Ετερογενή Στόλο Οχημάτων, Ταυτόχρονες Περισυλλογές και Παραδόσεις και Χρονικά Παράθυρα. Ο αλγόριθμος ΠΡΑΕΓ, μετά από προσαρμογές, εφαρμόζεται στο Συνεργατικό ΠΔΟ προκειμένου να εκτιμηθούν τα οικονομικά και περιβαλλοντικά οφέλη από τη συνεργασία εταιρειών διανομής. Στο πλαίσιο της μελέτης παραλλαγών του ΠΔΟ και της αντιμετώπισής τους μέσω

αλγορίθμων βελτιστοποίησης διατυπώθηκε το μαθηματικό τους μοντέλο, καθώς αποτελεί βασικό μέρος για την αποτελεσματική αντιμετώπιση του προβλήματος.

Τέλος, οι προτεινόμενοι αλγόριθμοι έχουν αναπτυχθεί στη γλώσσα προγραμματισμού Python και έχουν δοκιμαστεί σε πολλαπλά δεδομένα της βιβλιογραφίας, προκειμένου να εξασφαλιστεί η αποδοτικότητά τους και η ικανότητας του να ενσωματωθούν σε ένα cloud σύστημα δρομολόγησης. Από τη στιγμή που και οι τρεις προτεινόμενοι αλγόριθμοι προσφέρουν αποτελέσματα υψηλής ποιότητας αποθηκεύονται σε έναν cloud διακομιστή (server) προκειμένου να είναι προσπελάσιμοι μέσω των κατάλληλων αιτημάτων. Παρ' όλα αυτά, στη συγκεκριμένη μορφή του συστήματος μόνο ο αλγόριθμος ΠΡΑΕΓ είναι προσπελάσιμος καθώς καλύπτει τις περισσότερες περιπτώσεις στη διανομή φορτίων. Το σύστημα το οποίο είναι ικανό να επιλύσει ένα μεγάλο εύρος παραλλαγών του προβλήματος, πρέπει να τροφοδοτηθεί με τα κατάλληλα δεδομένα προκειμένου να προσφέρει αποδοτικά πλάνα εξυπηρέτηση πελατών. Προκειμένου να εξασφαλιστεί αυτό, το σύστημα εκμεταλλεύεται τεχνολογίες που εξασφαλίζουν την ποιότητα και την ακρίβεια των δεδομένων. Όσον αφορά τη γεωκωδικοποίηση και τα δεδομένα χρόνων μετακίνησης και αποστάσεων μεταξύ των σημείων παραγγελίας, χρησιμοποιούνται χάρτες του διαδικτύου. Τα οφέλη από τη χρήση ενός τέτοιου συστήματος το οποίο παρέχεται μέσω διαδικτύου και δεν απαιτεί την εγκατάστασή του στους υπολογιστές της κάθε εταιρείας ξεχωριστά είναι πολλά και δεν περιορίζονται μόνο στο οικονομικό σκέλος. Πέρα από τη εξάλειψη του κόστους για την εγκατάσταση και συντήρηση του συστήματος, οι χρήστες έχουν τη δυνατότητα να εξάγουν αποδοτικά και με μεγάλη ακρίβεια πλάνα δρομολογίων τα οποία οπτικοποιούνται σε χάρτες του διαδικτύου, να κάνουν αλλαγές στα προτεινόμενα δρομολόγια, καθώς επίσης και να εκτελούν διαδικασίες αναδρομολόγησης μετά την έναρξη της διανομής. Επιπλέον, οι τεχνολογίες και τα χαρτογραφικά δεδομένα δίνουν τη δυνατότητα στους χρήστες να αντιμετωπίσουν και τη δυναμική φύση του προβλήματος, καθώς ορισμένα δεδομένα ενδέχεται να αλλάξουν κατά τη διάρκεια του δρομολογίου και να επηρεάσουν την ομαλή εκτέλεσή του. Όλες οι παραπάνω τεχνολογίες και οι λειτουργίες του συστήματος μπορούν να ενισχύσουν σημαντικά τις προσπάθειες των εταιρειών διανομής και των υπευθύνων δρομολόγησης, τόσο πριν όσο και μετά την έναρξη της διανομής.

Συμπερασματικά, η έρευνα επικεντρώνεται σε διάφορους πυλώνες και ξεκινά με τη μελέτη των παραλλαγών του ΠΔΟ οι οποίες λαμβάνονται υπόψιν είτε μεμονωμένα είτε συνδυαστικά, και για τα οποία διαμορφώνεται το μαθηματικό μοντέλο. Επιπρόσθετα, η παρούσα διδακτορική διατριβή προτείνει και αναπτύσσει αλγορίθμους βελτιστοποίησης οι οποίοι αντιμετωπίζουν τις συγκεκριμένες περιπτώσεις διανομής και δοκιμάζονται τόσο σε δεδομένα της βιβλιογραφίας όσο και σε πραγματικά, προκειμένου να διαπιστωθεί η ικανότητά τους να ενσωματωθούν σε πληροφορικά συστήματα δρομολόγησης. Ο τελευταίος πυλώνας έρευνας αφορά τη δυνατότητα των αλγορίθμων να ενσωματωθούν σε πληροφοριακά συστήματα και να συνεργαστούν με τεχνολογίες που είναι απαραίτητες για την ανάπτυξη πληροφοριακών συστημάτων δρομολόγησης.

1. Εισαγωγή

Το κεφάλαιο της εισαγωγής παρέχει τα κίνητρα, τους στόχους, τη συμβολή και τη σύνοψη της διατριβής. Αρχικά εισάγεται το ερευνητικό θέμα, το οποίο είναι το Πρόβλημα Δρομολόγησης Οχημάτων (ΠΔΟ), ενώ παρουσιάζονται τα κίνητρα που οδήγησαν στην αναλυτική μελέτη του ΠΔΟ. Κατά συνέπεια, δίνεται και η συμβολή της παρούσας εργασίας και ακολουθεί η σύνοψη της διατριβής.

1.1 Ερευνητικά Κίνητρα

Τα δίκτυα της εφοδιαστικής αλυσίδας περιλαμβάνουν πολλαπλές λειτουργίες όπως η προμήθεια πρώτων υλών σε βιομηχανίες, η αποθήκευση εμπορευμάτων και η διανομή αυτών σε καταστήματα λιανικής πώλησης και καταναλωτές. Οι διαδικασίες αυτές παίζουν ουσιαστικό ρόλο στη ζωή μας, καθώς όντας αποτελεσματικές προσφέρουν εμπορεύματα και υπηρεσίες υψηλής ποιότητας με ελάχιστο κόστος. Η παρούσα διατριβή ενισχύει αυτή την προσπάθεια εστιάζοντας σε ένα από τα τελευταία μέρη της εφοδιαστικής αλυσίδας, τη διανομή, όπου πολλαπλές μεταβλητές και περιορισμοί επηρεάζουν τη διαδικασία.

Η σημασία της συγκεκριμένης έρευνας ενισχύεται και από αριθμητικά δεδομένα καθώς οι οδικές εμπορευματικές μεταφορές, που περιλαμβάνουν εθνικές, διεθνείς, διασταυρούμενες εμπορικές και ενδομεταφορές, αυξήθηκαν κατά 0,2% από το 2017 έως το 2018 στις χώρες της Ευρωπαϊκής Ένωσης (ΕΕ). Πιο συγκεκριμένα, οι συνολικές εμπορευματικές μεταφορές στην ΕΕ έφθασαν το 2018 τα 1.925 δισεκατομμύρια τονοχιλιόμετρα ενώ για την Ελλάδα 28,4 δισ. Επιπλέον, ο τομέας των μεταφορών και των logistics στην Ελλάδα αναπτύσσεται με ταχείς ρυθμούς και συνεισφέρει περίπου το 9% του ακαθάριστου εγχώριου προϊόντος (ΑΕΠ) (19 δισεκατομμύρια). Παρά τη σημασία του τομέα στην εθνική οικονομία, μια άλλη έρευνα (EEL 2018) που περιελάμβανε 181 εταιρείες που δραστηριοποιούνται σε αυτόν τον τομέα έδειξε ότι μόνο το 29% των εταιρειών χρησιμοποιεί συστήματα δρομολόγησης και μόνο το 12% τηλεματικά συστήματα. Το γεγονός αυτό δεν περιορίζεται μόνο στην Ελλάδα, καθώς άλλες μελέτες όπως αυτή των Leyerer et al. (2019) υποδεικνύουν ότι οι περισσότερες εταιρείες δεν χρησιμοποιούν ακόμη λογισμικό δρομολόγησης, παρά τις σημαντικές δυνατότητες εξοικονόμησης όσον αφορά τις αποστάσεις οδήγησης, το κόστος και τις εκπομπές ρύπων. Αυτά τα δεδομένα μπορούν να ερμηνευθούν είτε ως ότι οι εταιρείες έχουν δυσπιστία όσον αφορά τη χρήση συστημάτων είτε ότι τα συστήματα και τα πακέτα λογισμικού αποτελούν ακριβές λύσεις για μικρές και μεσαίες επιχειρήσεις και για αυτόν τον λόγο δεν είναι πρώτη επιλογή.

Επιπλέον, την τελευταία δεκαετία, το ηλεκτρονικό εμπόριο, οι διαδικτυακές αγορές και η αστικοποίηση έχουν αυξηθεί. Το γεγονός αυτό έχει οδηγήσει σε διευρυμένο αριθμό τελικών πελατών και καταστημάτων λιανικής, ειδικά στα κέντρα των πόλεων. Αυτό το γεγονός επηρεάζει σημαντικά τη διαδικασία διανομής καθώς οι εταιρείες διανομής πρέπει να εξυπηρετούν ακόμη και πάνω από 2.000 παραγγελίες καθημερινά. Επιπλέον, οι εταιρείες

διανομής πρέπει να χειρίζονται και να ικανοποιούν τις απαιτήσεις των πελατών τους, όπως τα χρονικά παράθυρά τους και τις ανάγκες τους για ταυτόχρονη παραλαβή και επιστροφή αγαθών, καθώς και τα χαρακτηριστικά της εταιρείας όπως τα διαφορετικά οχήματα που εκτελούν διανομές. Ως αποτέλεσμα, οι προηγμένες αλγοριθμικές προσεγγίσεις που ενσωματώνονται σε συστήματα δρομολόγησης φαίνεται να είναι η μόνη λύση για τις εταιρείες. Όλες αυτές οι ανάγκες των εταιρειών διανομής και των λοιπών εμπλεκόμενων οδήγησαν στη μελέτη του ΠΔΟ και των παραλλαγών του, στην ανάπτυξη αλγορίθμων που μπορούν να χειριστούν μεγάλο όγκο δεδομένων και πολλαπλές μεταβλητές και περιορισμούς. Τα παραπάνω, μαζί με την ενσωμάτωση των αλγορίθμων σε ένα cloud σύστημα δρομολόγησης αποτελούν βασικά πεδία μελέτης της παρούσας διατριβής.

Ωστόσο, πέρα από τις ανάγκες των εταιρειών διανομής και τα οφέλη που μπορούν να προκύψουν από αυτή την έρευνα, το πρωταρχικό κίνητρο προκύπτει από το κενό που υπάρχει στην ερευνητική κοινότητα. Συγκεκριμένα, ελάχιστες αλγοριθμικές προσεγγίσεις που προτείνονται και δημοσιεύονται από ερευνητές ενσωματώνονται σε πακέτα λογισμικού δρομολόγησης, πόσο μάλλον σε cloud συστήματα δρομολόγησης, όπως προκύπτει από την βιβλιογραφική ανασκόπηση στο Κεφάλαιο 2. Αυτή είναι μια σημαντική διαπίστωση δεδομένου ότι οι ερευνητές μελετούν το ΠΔΟ για περισσότερα από 60 χρόνια όταν και προτάθηκε από τους Dantzig και Ramser (1959). Με τα χρόνια, διαφορετικές παραλλαγές του ΠΔΟ έχουν δημιουργηθεί, οι οποίες συσχετίζονται με συγκεκριμένες παραμέτρους ή περιορισμούς που εμφανίζονται σε πραγματικές περιπτώσεις διανομής.

Μία από τις πιο γνωστές παραλλαγές του ΠΔΟ είναι το Πρόβλημα της Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (ΠΔΟΧΠ), όπου γεωγραφικά διάσπαρτοι πελάτες πρέπει να εξυπηρετούνται σε ένα προκαθορισμένο χρονικό διάστημα (χρονικό παράθυρο), μόνο μία φορά και από ένα μόνο όχημα. Η συνολική ποσότητα εμπορευμάτων που διανέμονται σε κάθε διαδρομή δεν μπορεί να υπερβαίνει τη χωρητικότητα του οχήματος, ενώ κάθε όχημα ξεκινά και τελειώνει τη διαδρομή του σε μια συγκεκριμένη αποθήκη. Ένα όχημα μπορεί να φτάσει σε έναν πελάτη πριν από την έναρξη του χρονικού παραθύρου και να περιμένει μέχρι τη συμφωνημένη ώρα εξυπηρέτησης, αλλά δεν επιτρέπεται η παράδοση εμπορευμάτων εάν φτάνει μετά τη λήξη του χρονικού παραθύρου. Μια άλλη παραλλαγή του ΠΔΟ είναι το Πρόβλημα της Δρομολόγησης Οχημάτων με Ταυτόχρονες Περισυλλογές και Παραδόσεις (ΠΔΟΤΠΠ), όπου οι πελάτες μπορούν να παραλαμβάνουν, όσο και να επιστρέφουν αγαθά. Αυτό δημιουργεί έναν επιπλέον περιορισμό καθώς σε οποιοδήποτε στάδιο της διανομής, είτε παραδίδονται είτε συλλέγονται τα αγαθά, η χωρητικότητα του οχήματος δεν μπορεί να παραβιαστεί. Στο Πρόβλημα Δρομολόγησης Οχημάτων με Ετερογενή Στόλο Οχημάτων (ΠΔΟΕΣΟ) τα οχήματα θεωρούνται διαφορετικά ως προς τη χωρητικότητα, το μεταβλητό και το σταθερό κόστος όσο και τις εκπομπές. Αυτό είναι αρκετά συνηθισμένο στο δίκτυο διανομής καθώς λίγες εταιρείες διατηρούν και χρησιμοποιούν αποκλειστικά όμοια οχήματα. Τέλος, στο Συνεργατικό Πρόβλημα Δρομολόγησης Οχημάτων (ΣΠΔΟ), οι εταιρείες διανομής που εδρεύουν και διανέμουν αγαθά στις ίδιες περιοχές συνεργάζονται στη διαδικασία της διανομής

και κατά συνέπεια στη δρομολόγηση. Ένα μείγμα αυτών των παραλλαγών μπορεί να αντιπροσωπεύει ένα πρακτικό και ρεαλιστικό πρόβλημα δρομολόγησης που αντιμετωπίζουν καθημερινά οι περισσότερες εταιρείες διανομής.

Επιπλέον, καθώς οι παραλλαγές ΠΔΟ αυξάνονταν με τα χρόνια, ενώ ταυτόχρονα εμφανίζονται και νέες, έχει προκύψει η ανάγκη για πιο αποτελεσματικούς αλγόριθμους που μπορούν να χειριστούν πολλές παραλλαγές του ΠΔΟ ταυτόχρονα. Αρχικά, οι ακριβείς αλγόριθμοι προτείνονταν συχνότερα από τους ερευνητές. Ωστόσο, το γεγονός ότι αυτές οι μέθοδοι μπορούν να χειριστούν μόνο έναν περιορισμένο αριθμό πελατών (περίπου 100) λόγω της υπολογιστικής πολυπλοκότητας του ΠΔΟ ήταν καθοριστικός παράγοντας για τη μη συχνή μελέτη και εφαρμογή τους. Κατά συνέπεια, οι ερευνητές εστίασαν στην ανάπτυξη προσεγγιστικές μεθόδους που προσφέρουν την καλύτερη αναλογία μεταξύ ποιότητας λύσεων και υπολογιστικού χρόνου. Οι ευρετικοί αλγόριθμοι αποτελούν τέτοιες μεθόδους που παράγουν λύσεις καλής ποιότητας σε ελάχιστο υπολογιστικό χρόνο. Οι πρώτοι ευρετικοί που αναπτύχθηκαν επικεντρώθηκαν στην κατασκευή μιας εφικτής λύσης, ενώ σε μεταγενέστερο χρόνο επικεντρώθηκαν στη βελτίωση της λύσης με τη μεταφορά ενός πελάτη σε άλλη διαδρομή. Μετά τους ευρετικούς, η προσοχή δόθηκε στους μεταευρετικούς που θεωρούνται πιο εξεζητημένοι ευρετικοί βασισμένοι σε έννοιες μαθηματικών και βιολογίας. Η ικανότητά τους να παρέχουν βελτιωμένες λύσεις σε σύγκριση με τους ευρετικούς αλγορίθμους σε παρόμοιο υπολογιστικό χρόνο τους έκανε πιο δημοφιλείς.

Συμπερασματικά, ενώ το κύριο χάσμα μεταξύ της έρευνας και της βιομηχανίας, όσον αφορά τη διανομή, παρατηρείται στους περιορισμένους αλγόριθμους που αναπτύσσονται και ενσωματώνονται σε πραγματικά και λειτουργικά συστήματα, η παρούσα διατριβή στοχεύει στη μελέτη του ΠΔΟ από πολλαπλές οπτικές γωνίες. Συγκεκριμένα, οι στόχοι της έρευνας είναι η μελέτη του ΠΔΟΧΠ, του ΠΔΟΤΠΠ, του ΠΔΟΕΣΟ και του ΣΠΔΟ, είτε μεμονωμένα είτε συνδυαστικά, η διαμόρφωση των σύνθετων μαθηματικών μοντέλων, η πρόταση και η ανάπτυξη αποτελεσματικών μεταευρετικών αλγορίθμων, και τέλος η μελέτη του τρόπου με τον οποίο οι προτεινόμενοι αλγόριθμοι μπορούν να ενσωματωθούν σε cloud σύστημα δρομολόγησης το οποίο εκμεταλλεύεται τις τελευταίες τεχνολογικές εξελίξεις. Επιπλέον, η μελέτη ενός cloud συστήματος δρομολόγησης το οποίο ενσωματώνει νέες αλγοριθμικές προσεγγίσεις και εκμεταλλεύεται προηγμένες τεχνολογίες που προσφέρουν τα απαραίτητα δεδομένα γρήγορα και με υψηλή ακρίβεια αποτελούν έναν ακόμα βασικό πυλώνα της έρευνας. Τέλος, η παρούσα διατριβή καλύπτει ένα ευρύ φάσμα λειτουργιών για την ανάπτυξη ενός cloud συστήματος δρομολόγησης, με στόχο να φέρει πιο κοντά την ερευνητική κοινότητα και τη βιομηχανία.

1.2 Στόχοι και Συνεισφορές της Διατριβής

Ο κύριος στόχος της παρούσας διατριβής είναι να καλύψει το ερευνητικό κενό, το οποίο είναι οι περιορισμένοι αλγόριθμοι που αναπτύσσονται από ερευνητές και αντιμετωπίζουν ταυτόχρονα πολλαπλές μεταβλητές και περιορισμούς που εμπλέκονται στη διαδικασία της

διανομής, καθώς και που ενσωματώνονται σε πραγματικές εφαρμογές. Για την εκπλήρωση αυτού του στόχου και την προσφορά προηγμένων τεχνολογικών λύσεων, είναι απαραίτητο να εντοπιστούν αρχικά οι μεταβλητές και οι περιορισμοί που αντιμετωπίζονται πιο συχνά από τις εταιρείες διανομής και κατά συνέπεια να οριστεί η μαθηματική διατύπωση του προβλήματος. Οι μεταβλητές και οι περιορισμοί που αντιμετωπίζονται πιο συχνά αντιστοιχούν στις ανάγκες των περισσότερων εταιρειών διανομής και αυτός είναι ο κύριος λόγος για την εστίαση σε αυτές. Στη συνέχεια, η έρευνα επικεντρώνεται στην ανάπτυξη προηγμένων μεταευρετικών αλγορίθμων που επιλύουν τις περίπλοκες περιπτώσεις διανομής που έχουν δημιουργηθεί και προσφέροντας μειωμένο κόστος διανομής. Τέλος, ένας άλλος κρίσιμος στόχος της παρούσας μελέτης είναι η ενσωμάτωση των προτεινόμενων μεταευρετικών σε ένα σύστημα δρομολόγησης νέφους που μπορεί να εφαρμοστεί σε πραγματικές λειτουργίες διανομής.

Όπως είναι φυσικό, οι στόχοι της διατριβής συσχετίζονται με τις συνεισφορές της, και χωρίζονται σε 5 κύριους πυλώνες που είναι:

i. η βιβλιογραφική ανασκόπηση για την εύρεση των τάσεων που ακολουθούν οι παραλλαγές του ΠΔΟ και οι αλγόριθμοι, για τη συσχέτισή τους, και για τους αλγορίθμους που οδηγούν σε πρακτικές εφαρμογές,

ii. οι μεθοδολογίες που αναπτύσσονται για την αντιμετώπιση εύρους παραλλαγών του ΠΔΟ,

iii. η μαθηματική διατύπωση των προβλημάτων που αντιμετωπίζονται, ακόμα και σε περιπτώσεις που συνδυάζονται πολλαπλές μεταβλητές και περιορισμοί των αστικών εμπορευματικών μεταφορών,

iv. τα υπολογιστικά αποτελέσματα, καθώς δοκιμάζονται τόσο δεδομένα της βιβλιογραφίας, με τους αλγόριθμους να προσφέρουν νέα βέλτιστα, όσο και με πραγματικά δεδομένα διανομής, εταιρειών που δραστηριοποιούνται στο χώρο, και

v. η πρακτική εφαρμογή και ενσωμάτωση των αλγορίθμων σε ένα cloud σύστημα δρομολόγησης

1.3 Σύνοψη της Διδακτορικής Διατριβής

Η παρούσα διδακτορική διατριβή στη συνέχεια οργανώνεται ως εξής:

Το **Κεφάλαιο 2** παρουσιάζει τις τελευταίες εξελίξεις στο ΠΔΟ και στις πολλαπλές παραλλαγές που υπάρχουν, καθώς και πώς αυτές οι παραλλαγές συνδέονται με πραγματικές περιπτώσεις αστικών εμπορευματικών μεταφορών. Αυτό είναι ένα θεμελιώδες μέρος της έρευνας, καθώς οι παραλλαγές του ΠΔΟ που αντιμετωπίζονται πιο συχνά ορίζονται και συγκρίνονται με τις μεταβλητές και τους περιορισμούς που οι εταιρείες του κλάδου θεωρούν πιο σημαντικούς. Επιπρόσθετα, στο Κεφάλαιο 2 παρουσιάζεται μια βιβλιογραφική ανασκόπηση που σχετίζεται με τις αλγοριθμικές προσεγγίσεις που υπάρχουν τις τάσεις που ακολουθούν. Επιπλέον, ορίζεται η συσχέτιση μεταξύ των αλγοριθμικών προσεγγίσεων και των παραλλαγών του ΠΔΟ καθώς επίσης αναλύονται τα άρθρα που αναπτύσσουν αλγορίθμους οι οποίοι εντάσσονται σε συστήματα αποφάσεων.

Το **Κεφάλαιο 3** αντιμετωπίζει το Πρόβλημα της Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (ΠΔΟΧΠ). Πιο συγκεκριμένα, πραγματοποιείται βιβλιογραφική ανασκόπηση για το ΠΔΟΧΠ. Επιπλέον, παρουσιάζεται η μαθηματική διατύπωση του προβλήματος και η προτεινόμενη αλγοριθμική προσέγγιση (Πολυαντικειμενική Αναζήτηση Ευρείας Γειτονιάς - ΠΑΕΓ) η οποία το αντιμετωπίζει. Τέλος, ο αλγόριθμος δοκιμάζεται σε δεδομένα της βιβλιογραφίας, και τα αποτελέσματά που προσφέρει συγκρίνονται με τα καλύτερα δημοσιευμένα στη βιβλιογραφία.

Το **Κεφάλαιο 4** πραγματεύεται το ΠΔΟ με Ταυτόχρονες Περισυλλογές και Παραδόσεις, καθώς και με Χρονικά Παράθυρα (ΠΔΟΤΠΠΧΠ). Η δομή του Κεφαλαίου 4 ακολουθεί αυτή του Κεφάλαιου 3. Στο συγκεκριμένο κεφάλαιο ωστόσο η προτεινόμενη αλγοριθμική προσέγγιση είναι ένας πολυαντικειμενικός εξελικτικός αλγόριθμος ο οποίος δοκιμάζεται σε δύο σύνολα δεδομένων της βιβλιογραφίας για να καθοριστεί η αποτελεσματικότητά του.

Το **Κεφάλαιο 5** πραγματεύεται το ΠΔΟ με Ετερογενή Στόλο Οχημάτων, Ταυτόχρονες Περισυλλογές και Παραδόσεις και με Χρονικά Παράθυρα (ΠΔΟΕΣΟΤΠΠΧΠ), το οποίο συνδυάζει το ΠΔΟΧΠ, το ΠΔΟΤΠΠ και το ΠΔΟΕΣΟ. Η βιβλιογραφική ανασκόπηση αυτού του προβλήματος ακολουθείται από τη μαθηματική διατύπωση και την περιγραφή του. Στη συνέχεια παρουσιάζεται ο προτεινόμενος αλγόριθμος Προσαρμοστικής Αναζήτησης Ευρείας Γειτονιάς (ΠΡΑΕΓ), ενώ η αποτελεσματικότητά του ορίζεται δοκιμάζοντάς τον σε πολλαπλά σύνολα δεδομένων της βιβλιογραφίας. Επιπλέον, ο αλγόριθμος δοκιμάζεται και στο Συνεργατικό ΠΔΟ για να εκτιμηθούν τα οφέλη όσον αφορά το κόστος και τις εκπομπές ρύπων, όταν τρεις εταιρείες Third Party Logistics (3PL) συνεργάζονται στη διαδικασία της διανομής.

Το **Κεφάλαιο 6** περιγράφει το cloud σύστημα δρομολόγησης που αποθηκεύει τους αλγόριθμους, που έχουν αναπτυχθεί στα Κεφάλαιο 3, 4 και 5, στη βάση δεδομένων. Επιπλέον, αναλύονται οι τεχνολογίες που αξιοποιούνται για την ανάπτυξη του cloud συστήματος, καθώς και οι λειτουργίες του. Τέλος, το σύστημα δοκιμάζεται και επικυρώνεται σε τρεις πραγματικές περιπτώσεις διανομής για να εκτιμηθεί η αποτελεσματικότητά του και τα οφέλη που προσφέρονται στις εταιρείες διανομής και στους υπεύθυνους προγραμματισμού.

Τέλος, το **Κεφάλαιο 7** συνοψίζει τη συμβολή της τρέχουσας έρευνας, τους στόχους που έχουν επιτευχθεί και τα μελλοντικά ερευνητικά βήματα που θα ακολουθήσουν για την αντιμετώπιση ακόμη περισσότερων παραλλαγών που αφορούν τους ερευνητές και τους επαγγελματίες του χώρου, καθώς και τις βελτιώσεις που μπορούν να γίνουν στη μαθηματική διατύπωση και τους αλγόριθμους για την εξαγωγή ακόμη πιο αποτελεσματικών πλάνων δρομολογίων.

2. Βιβλιογραφική Επισκόπηση

2.1 Εισαγωγή

Το Πρόβλημα της Δρομολόγησης Οχημάτων (ΠΔΟ) αποτελεί μία από τις κύριες προκλήσεις που έχουν να αντιμετωπίσουν οι εταιρείες διανομής. Οι ερευνητές μελετάνε το πρόβλημα από το 1959 όταν οι Dantzig και Ramser (1959) εισήγαγαν το Truck Dispatching Problem. Έκτοτε, οι παράγοντες που επηρεάζουν τη διανομή έχουν μετατραπεί κατά τη μοντελοποίηση είτε σε μεταβλητές είτε σε περιορισμούς, διαμορφώνοντας τις διάφορες σε παραλλαγές του ΠΔΟ. Στις περισσότερες περιπτώσεις στόχος είναι η ελαχιστοποίηση του συνολικού κόστους διανομής. Προκειμένου να επιτευχθεί κάτι τέτοιο είναι βασικό να διαπιστωθούν οι παράγοντες που επηρεάζουν τη διανομή κάθε εταιρείας καθώς και να χρησιμοποιηθούν οι κατάλληλοι αλγόριθμοι και συστήματα.

Σε αυτήν την κατεύθυνση, το Κεφάλαιο αναγνωρίζει τις τάσεις που χαρακτηρίζουν τις παραλλαγές του ΠΔΟ, τους προτεινόμενους αλγόριθμους και τη συσχέτισή τους. Οι διάφορες παραλλαγές ΠΔΟ ταξινομούνται σε δεκαέξι (16) κατηγορίες που αντιπροσωπεύουν τις κύριες περιπτώσεις που εμφανίζονται στη διανομή εμπορευματικών μεταφορών. Στη συνέχεια, παρουσιάζονται οι αλγόριθμοι που επιλύουν τις παραλλαγές του ΠΔΟ χρησιμοποιώντας μια συστηματική βιβλιογραφική ανασκόπηση όπου αναλύονται οι σχέσεις μεταξύ των παραλλαγών ΠΔΟ και των διαφόρων αλγορίθμων. Τέλος, το τρέχον Κεφάλαιο παρουσιάζει επίσης μελέτες και εργασίες που οδηγούν σε πρακτικές εφαρμογές (πακέτα λογισμικού και πληροφοριακά συστήματα) στη δρομολόγηση οχημάτων.

Το υπόλοιπο μέρος αυτού του κεφαλαίου οργανώνεται ως εξής: Η ενότητα 2.2 παρουσιάζει τις κύριες παραλλαγές ΠΔΟ που σχετίζονται με τις αστικές εμπορευματικές μεταφορές. Η ενότητα 2.3 ταξινομεί και αναλύει τους διάφορους αλγόριθμους βελτιστοποίηση που εφαρμόζονται στο ΠΔΟ. Στην ενότητα 2.4 παρουσιάζεται η μεθοδολογία έρευνας που ακολουθήθηκε και στην ενότητα 2.5 τα ποσοτικά αποτελέσματα μετά την ανάλυση. Στην ενότητα 2.6 παρουσιάζονται οι πρακτικές εφαρμογές για την αντιμετώπιση του ΠΔΟ. Τέλος, στην ενότητα 2.7 παρουσιάζονται οι συμπερασματικές παρατηρήσεις.

2.2 Το Πρόβλημα της Δρομολόγησης Οχημάτων και οι Παραλλαγές του

Σε αυτή την ενότητα, αναλύονται οι παραλλαγές που αντιμετωπίζουν οι περισσότερες εταιρείες διανομής στις καθημερινές τους δραστηριότητες, σχετικά με τη δρομολόγηση των οχημάτων και τον προγραμματισμό των παραδόσεων. Αρχικά, η χωρητικότητα των οχημάτων αποτελεί βασικό παράγοντα του προβλήματος δρομολόγησης του οχήματος (ΠΔΟ) (Kim et al. 2015) και αυτόν που πρώτο οι ερευνητές και οι επαγγελματίες του χώρου έλαβαν υπόψη. Η χωρητικότητα σχηματίζει δύο παραλλαγές του ΠΔΟ: 1) το ΠΔΟ με Περιορισμούς Χωρητικότητα (ΠΔΟΠΧ) στο οποίο όλα τα οχήματα είναι πανομοιότυπα και έχουν την ίδια χωρητικότητα και 2) ΠΔΟ με Ετερογενή Στόλο (ΠΔΟΕΣΟ), στον οποίο υπάρχουν πολλοί τύποι οχημάτων, καθένας από τους οποίους χαρακτηρίζεται από διαφορετική χωρητικότητα,

σταθερό και μεταβλητό κόστος (Prins 2009). Και στις δύο περιπτώσεις, κατά τη διατύπωση του προβλήματος, θεωρείται ότι τα οχήματα θα ξεκινήσουν και θα επιστρέψουν στην κεντρική αποθήκη μετά την ολοκλήρωση της διαδρομής τους. Ωστόσο, κυρίως στις περιπτώσεις που εταιρείες συνεργάζονται με τρίτους για την διανομή, τα οχήματα των συνεργατών δεν επιστρέφουν απαραίτητα στην αποθήκη μετά την ολοκλήρωση της διαδρομής. Αυτή η περίπτωση αναγνωρίζεται ως το Ανοιχτό ΠΔΟ (ΑΠΔΟ) στην ερευνητική κοινότητα. Επίσης, ανεξάρτητα από το αν ο στόλος των οχημάτων είναι ετερογενής ή ομοιογενής, ορισμένοι ερευνητές, την τελευταία δεκαετία, εξετάζουν είτε δισδιάστατους (2Δ-ΠΔΟ) είτε τρισδιάστατους (3Δ-ΠΔΟ) περιορισμούς φόρτωσης για να διασφαλίσουν ότι τα εμπορεύματα μπορούν να φορτωθούν στα οχήματα (Zachariadis et al. 2016).

Υπάρχουν αρκετές περιπτώσεις που μπορούν να εκτελέσουν περισσότερα από ένα ταξίδια κατά τη διάρκεια της εργάσιμης ημέρας, οδηγώντας στο ΠΔΟ με Πολλαπλές Διαδρομές (ΠΔΟΠΔ). Για τη βελτιστοποίηση της διαδικασίας και την εκτέλεση του μέγιστου αριθμού ταξιδιών κατά τις βάρδιες των οδηγών, χρησιμοποιούνται συνήθως Δορυφορικές Εγκαταστάσεις (ΠΔΟΔΕ) για την αναπλήρωση των οχημάτων. Σε αυτό το πλαίσιο εντάσσεται και το ΠΔΟ με δύο κλιμάκια και η διαχείριση της παράδοσης αγαθών από την κεντρική αποθήκη στους πελάτες γίνεται με δρομολόγηση οχημάτων μέσω δορυφορικών εγκαταστάσεων (Grangier et al. 2016).

Οι απαιτήσεις των πελατών καθορίζουν σε μεγάλο βαθμό τη ρύθμιση των παραμέτρων του ΠΔΟ. Μία από τις πιο κοινές παραλλαγές του προβλήματος είναι το ΠΔΟ με Χρονικά Παράθυρα (ΠΔΟΧΠ), όπου κάθε πελάτης καθορίζει ένα χρονικό διάστημα στο οποίο πρέπει να παραδοθεί η παραγγελία (César and Oliveira 2010). Αυτή η παραλλαγή έχει συνδεθεί άρρηκτα με την ακρίβεια των παραδόσεων και την ικανοποίηση των πελατών.

Είναι επίσης σημαντικό να τονιστεί ότι οι λειτουργίες διανομής δεν τελειώνουν στη φάση της παράδοσης αγαθών, καθώς το φαινόμενο των πελατών να επιστρέφουν προϊόντα είναι συνηθισμένο στην πράξη. Τόσο το ΠΔΟ με Επιστροφές (ΠΔΟΕ) όσο και το ΠΔΟ με Ταυτόχρονες Περισυλλογές και Παραδόσεις (ΠΔΟΤΠΠ) μελετούν την περίπτωση των παραδόσεων και των παραλαβών κατά την εκτέλεση των δρομολογίων.

Η περιβαλλοντική ρύπανση ανάγκασε τις κυβερνήσεις να θεσπίσουν περιβαλλοντικούς κανονισμούς για τη μείωση του θορύβου, της κυκλοφορίας, των εκπομπών CO2 και κατά συνέπεια τη βελτίωση της ποιότητας ζωής των πολιτών. Σε αυτή την κατεύθυνση, οι εταιρείες διανομής στοχεύουν τόσο στην ελαχιστοποίηση του κόστους μεταφοράς, όσο και των εκπομπών CO2, ένα πρόβλημα που προσδιορίζεται στη βιβλιογραφία ως Πράσινο ΠΔΟ (ΠΠΔΟ) (Lin et al. 2014) ή ως Πρόβλημα Ρύπανσης Δρομολόγησης (ΠΡΔ).

Επιπλέον, τα ηλεκτρικά οχήματα και τα υβριδικά οχήματα που μπορούν να λειτουργούν τόσο ηλεκτρικά όσο και με παραδοσιακά καύσιμα συμβάλλουν στην ελαχιστοποίηση της ρύπανσης και των εκπομπών CO2. Αυτές οι εκδοχές του προβλήματος είναι γνωστές ως Ηλεκτρικό ΠΔΟ

(ΗΠΔΟ) και Υβριδικό ΠΔΟ (ΥΠΔΟ) αντίστοιχα και έχουν προσελκύσει πρόσφατα την προσοχή εταιρειών και ερευνητών (Mancini 2017).

Η ακρίβεια των παραδόσεων εξαρτάται σε μεγάλο βαθμό από τον χρόνο ταξιδιού μεταξύ των σημείων. Αυτό το πρόβλημα έχει αντιμετωπιστεί μέσω της χρήσης μιας συνάρτησης, στην οποία ο χρόνος αναχώρησης είναι η ανεξάρτητη μεταβλητή και είναι γνωστή ως Χρονικά Εξαρτώμενο ΠΔΟ (ΧΕΠΔΟ) (Figliozzi 2012). Ωστόσο, παραγγελίες και απρόβλεπτα γεγονότα μπορεί να εμφανιστούν δυναμικά κατά την εκτέλεση της διαδρομής, δημιουργώντας έτσι το Δυναμικό ΠΔΟ (ΔΠΔΟ) και το Στοχαστικό ΠΔΟ (ΣΤΠΔΟ).

Μια άλλη μεταβλητή που εξαρτάται σε μεγάλο βαθμό από τον αριθμό των κέντρων διανομής που συνεργάζονται για τη διανομή αγαθών είναι το ΠΔΟ με Πολλαπλές Αποθήκες (ΠΔΟΠΑ) (Renaud et al. 1996). Μία επέκταση αυτού είναι και το Συνεργατικό ΠΔΟ (ΣΠΔΟ), στο οποίο ένας όμιλος εταιρειών συνεργάζεται για να ελαχιστοποιήσει το κόστος λειτουργίας και να βελτιστοποιήσει την αποτελεσματικότητα του δικτύου (Wang et al. 2017).

Με τα χρόνια, οι ερευνητές μελετούν όλο και περισσότερες παραλλαγές του ΠΔΟ για να αντιμετωπίσουν περιπτώσεις που αντιμετωπίζουν οι εταιρείες διανομής στις καθημερινές τους λειτουργίες. Οι παραλλαγές του ΠΔΟ που αναλύονται σε αυτήν την ενότητα αντικατοπτρίζουν τις περισσότερες από τις προκλήσεις που αντιμετωπίζονται στη διανομή εμπορευμάτων. Τα Πληροφοριακά Συστήματα μπορούν να ενισχύσουν την αντιμετώπιση του προβλήματος ενσωματώνοντας στη λειτουργικότητά τους τη χρήση αλγορίθμων.

2.3 Μέθοδοι Επίλυσης

Στην συγκεκριμένη ενότητα μελετώνται οι αλγόριθμοι που αντιμετωπίζουν αποτελεσματικά ένα σύνολο παραλλαγών του ΠΔΟ. Η ταξινόμηση των αλγορίθμων βασίζεται στην έρευνα των Labadie et al. (2016) και Lin et al. (2014) και περιλαμβάνει τους ακριβείς, τους ευρετικούς και τους μεταευρετικους αλγορίθμους. Έμφαση και μεγαλύτερη ανάλυση γίνεται στους ευρετικούς και μεταευρετικούς αλγορίθμους, καθώς και στις επιμέρους κατηγορίες τους.

2.3.1 Ακριβείς Αλγόριθμοι

Οι ακριβείς αλγόριθμοι υπολογίζουν κάθε δυνατή λύση μέχρι να επιτευχθεί η καλύτερη, με πολύ κακή απόδοση σε υπολογιστικό χρόνο, ακόμη και σε αρκετά μικρές περιπτώσεις (El-Sherbeny 2010). Οι ακριβείς μέθοδοι ταξινομούνται σε τρεις κατηγορίες: Μέθοδοι χαλάρωσης Lagrange, δημιουργία στηλών και δυναμικός προγραμματισμός.

2.3.2 Ευρετικοί Αλγόριθμοι

Οι εταιρείες διανομής που αντιμετωπίζουν το ΠΔΟ χρειάζονται μεθόδους ικανές να παράγουν λύσεις υψηλής ποιότητας σε περιορισμένο υπολογιστικό χρόνο. Ως εκ τούτου, έχουν προταθεί πολλαπλοί ευρετικοί αλγόριθμοι, χωρισμένοι σε κατασκευαστικούς, τοπικής αναζήτησης και δύο φάσεων. Οι κατασκευαστικοί χωρίζονται σε διαδοχικές (sequential) και παράλληλες (parallel) μεθόδους. Στην πρώτη μέθοδο, οι κόμβοι επιλέγονται διαδοχικά μέχρι να

κατασκευαστεί μια εφικτή λύση λαμβάνοντας υπόψιν τη χωρητικότητα του οχήματος και τα χρονικά παράθυρα. Από την άλλη πλευρά, οι παράλληλες μέθοδοι δημιουργούν πολλές λύσεις ταυτόχρονα. Στη δεύτερη μέθοδο κατασκευάζονται διαδρομές διαδοχικά και η διαδικασία τελειώνει όταν δεν περισσεύουν άλλοι πελάτες εκτός των δρομολογίων (Solomon 1987). Οι τοπικές μέθοδοι αναζήτησης, αντίθετα, τροποποιούν την αρχική λύση για βελτιωμένες γειτονικές λύσεις. Η βελτίωση της λύσης μπορεί να είναι εντός μιας διαδρομής, που ονομάζεται εντός διαδρομής, ή μεταξύ διαφορετικών διαδρομών.

2.3.3 Μεταευρετικοί Αλγόριθμοι

Σε αντίθεση με τους ευρετικούς, οι μεταευρετικοί σκοπεύουν να ξεφύγουν από το τοπικό βέλτιστο καθώς εξερευνά ένα πιο σημαντικό υποσύνολο του χώρου λύσης. Οι μεταευρετικοί αλγόριθμοι είναι πιο προηγμένες διαδικασίες και περιέχουν μεθόδους αναζήτησης πληθυσμού και τοπικής αναζήτησης. Αυτές οι δύο υποκατηγορίες μεταευρετικών αναλύονται περαιτέρω στις ενότητες 2.3.3.1 και 2.3.3.2.

2.3.3.1 Μεταευρετικοί Αναζήτησης Πληθυσμού

Οι μεταευρετικοι που βασίζονται στον πληθυσμό λύσεων στοχεύουν στη δημιουργία μιας νέας λύσης, είτε συνδυάζοντας υπάρχουσες λύσεις είτε κάνοντας τες να συνεργάζονται μέσω μιας μαθησιακής διαδικασίας. Οι γενετικοί αλγόριθμοι (ΓΑ) είναι οι πιο συνηθισμένοι μεταευρετικοί αλγόριθμοι και εμπνέονται από τη βιολογική εξέλιξη. Οι μεμετικοί αλγόριθμοι (ΜΑ) είναι μια "προηγμένη" έκδοση των ΓΑ καθώς για κάθε παιδί εφαρμόζεται μια τοπική μέθοδος βελτίωσης. Και οι δύο αλγόριθμοι ανήκουν στην ευρύτερη κατηγορία των Εξελικτικών Αλγορίθμων (ΕΑ) (Labadie et al. 2016).

Η γενική κατηγορία αλγορίθμων που βασίζονται σε σμήνη περιλαμβάνει τη βελτιστοποίηση αποικίας μυρμηγκιών (ΒΑΜ) και τη βελτιστοποίηση μέσω σμήνους σωματιδίων (ΒΣΣ). Στο ΒΑΜ, τα μυρμήγκια κινούνται χτίζοντας μονοπάτια σύμφωνα με πιθανολογικούς κανόνες. Οι (Gambardella et al. 1999) ανέπτυξαν έναν τέτοιο αλγόριθμο για το ΠΔΟΧΠ, ενώ ένας αποτελεσματικός και καινοτόμος αλγόριθμους ΒΑΜ που έχει αναπτυχθεί για την αντιμετώπιση των παραλλαγών ΠΔΟ είναι αυτός των Kalayci and Kaya (2016). Επιπλέον, στο ΒΣΣ υπάρχει ένα πλήθος «σωματιδίων» και το καθένα μετακινείται από τη θέση του σε μια άλλη στον χώρο αναζήτησης. Η κίνηση κάθε σωματιδίου επηρεάζεται τόσο από την καλύτερη θέση που έχει επιτύχει το συγκεκριμένο σωματίδιο όσο και από την καλύτερη λύση όλων των σωματιδίων.

2.3.3.2 Μεταευρετικοί Τοπικής Αναζήτησης

Από την άλλη πλευρά, οι μεταευρετικοί τοπικής αναζήτησης στοχεύουν στην εξερεύνηση του χώρου λύσης μεταφέροντας την τρέχουσα λύση σε μια άλλη γειτονιά. Η αναζήτηση Tabu (Glover 1986) είναι ένας από τους πιο συχνά εφαρμοζόμενους αλγόριθμους και εξαρτάται από την αρχή της συνέχισης της αναζήτησης ακόμα κι αν έχει επιτευχθεί ένα τοπικό βέλτιστο. Επομένως, εφαρμόζεται μια κίνηση στην τρέχουσα λύση ακόμη και αν ο στόχος επιδεινωθεί.

Ωστόσο, θα πρέπει να σημειωθεί ότι δεν είναι δυνατή η επανεξέταση μιας λύσης που έχει επισκεφτεί στο παρελθόν καθώς η λίστα tabu καταγράφει τις πρόσφατες αναζητήσεις.

Ένας άλλος αλγόριθμος είναι αυτός της Αναζήτησης Ευρείας Γειτονιάς (ΑΕΓ) εξερευνά έναν ευρύ χώρο αναζήτησης. Στον συγκεκριμένο αλγόριθμο, η αρχική λύση καταστρέφεται μερικώς σύμφωνα με τους τελεστές καταστροφής και τελικά επισκευάζεται, σύμφωνα με τους τελεστές επανεισαγωγής. Η μέθοδος Προσαρμοστικής Αναζήτησης Ευρείας Γειτονιάς (ΠΡΑΕΓ) είναι παρόμοια, εκτός από το ότι οι πιο αποτελεσματικοί τελεστές που υπόκεινται σε αλλαγές κατά την αναζήτηση εφαρμόζονται σε κάθε επανάληψη (Hornstra et al. 2020).

Ένα άλλος μεταευρετικός, αυτός της Κατευθυνόμενης Τοπικής Αναζήτησης λειτουργεί εκχωρώντας ποινές στην αντικειμενική συνάρτηση έτσι ώστε ο χώρος αναζήτησης να γίνει πιο εκτεταμένος και να αποφευχθεί η παγίδευση σε ένα τοπικό βέλτιστο. Αυτό συμβαίνει επειδή η αναζήτηση θα μεταφερθεί σε μια νέα περιοχή του χώρου λύσεων. Η μέθοδος της Προσομοιωμένης Ανόπτησης (ΠΑ) ανήκει επίσης στους μεταευρετικούς τοπικής αναζήτησης, εμπνέεται από τη συμπεριφορά των ατόμων και των στερεών όταν αλλάζει η θερμοκρασία, και προτάθηκε από τους (Kirkpatrick et al. 1983).

Όσον αφορά την Αναζήτηση Μεταβλητής Γειτονιάς (ΑΜΓ), την οποία πρότειναν οι βασίζεται στο ότι επιτρέπεται μια συστηματική αλλαγή γειτονιάς όταν ένας αλγόριθμος τοπικής αναζήτησης δεν βελτιώνει (Mladenović and Hansen 1997). Επιπλέον, η διαδικασία της Άπληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης (ΑΤΠΑ) είναι ένας ακόμα μεταευρετικός τοπικής αναζήτησης που εφαρμόζεται στο ΠΔΟ αλλά λιγότερο συχνά σε σύγκριση με άλλες. Τέλος, ο αλγόριθμος Επαναλαμβανόμενη Τοπική Αναζήτηση (ΑΤΑ) βασίζεται στη δημιουργία μιας ακολουθίας μιας τοπικά βέλτιστης λύσης κάνοντας μια τροποποίηση, που ονομάζεται διαταραχή, στο τρέχον τοπικό βέλτιστο, καθώς και εκτελώντας διαδικασίες τοπικής αναζήτησης μετά την εκκίνηση από την τροποποιημένη λύση (Silva et al. 2014).

Τέλος, στην προσπάθειά τους να λύσουν βέλτιστα το ΠΔΟ, οι ερευνητές συνδυάζουν αλγόριθμους για να εκμεταλλευτούν τα πλεονεκτήματα κάθε διαδικασίας. Ειδικά τα τελευταία χρόνια, αναπτύσσονται υβριδικοί αλγόριθμοι που συνδυάζουν μεταευρετικές μεθόδους για την επίτευξη καλύτερων αποτελεσμάτων. Αυτό είναι ένα χαρακτηριστικό που μελετήθηκε στην ανασκόπηση της βιβλιογραφίας για να προσδιοριστεί ποιοι αλγόριθμοι χρησιμοποιούνται για την επίλυση πολλαπλών παραλλαγών ΠΔΟ.

2.4 Μεθοδολογία Έρευνας

Η έρευνα επικεντρώνεται στη τάση των παραλλαγών του ΠΔΟ καθώς και στους αλγόριθμους. Προκειμένου να επιτευχθεί η επιθυμητή ανάλυση καθορίστηκαν αρχικά οι παραλλαγές του ΠΔΟ, όσο και οι αλγόριθμοι που τις επιλύουν. Για τον στόχο αυτό χρησιμοποιήθηκε η βάση δεδομένων Scopus (https://www.scopus.com) καθώς καλύπτει ένα ευρύ φάσμα ακαδημαϊκών δημοσιεύσεων με κριτές. Πιο συγκεκριμένα, λαμβάνονται υπόψη μόνο άρθρα που δημοσιεύτηκαν σε περιοδικά, στην αγγλική γλώσσα και την τελευταία δεκαετία, καθώς επίσης και τα οποία περιέχουν τουλάχιστον μία από τις προηγουμένως καθορισμένες παραλλαγές του ΠΔΟ, προτείνοντας ή παρουσιάζοντας αλγόριθμο.

Επίσης, δεν συμπεριλαμβάνονται οι όροι, όπως «απόβλητα», «μετρητά», «επικίνδυνα», «σκυρόδεμα», «υγειονομική περίθαλψη», «φροντίδα», «αίμα», «λεωφορείο» και «μεταφορά ανθρώπων», καθώς δεν είναι σχετικά με τις εμπορευματικές μεταφορές. Οι όροι «ποδήλατο», «ιπτάμενα ρομπότ» και «UAV» εξαιρέθηκαν καθώς αναφέρονται σε διαφορετικά μέσα μεταφοράς από τα φορτηγά. Οι όροι "δρομολόγηση τοποθεσίας" και "δρομολόγηση αποθέματος" εξαιρέθηκαν επίσης από η έρευνα καθώς αποφασίστηκε να εξαιρεθούν τυχόν συνδυασμένα προβλήματα. Επίσης, η έρευνα αυτή περιορίζεται στους Θεματικούς Τομείς «Επιστήμη Υπολογιστών», «Μηχανική», «Μαθηματικά», «Επιστήμες Αποφάσεων» και «Επιχειρήσεις, Διοίκηση και Λογιστική». Ο όρος έρευνας περιλαμβάνει μόνο συγκεκριμένα περιοδικά που επιλέγονται σύμφωνα με το Scimago Journal Rank (https://www.scimagojr.com), περιλαμβάνοντας μόνο αυτά που ανήκουν στο πρώτο τεταρτημόριο (Q1).

Η συγκεκριμένη ερευνητική μέθοδος οδήγησε σε 334 δημοσιεύσεις, καθεμία από τις οποίες μελετήθηκε διεξοδικά για να προσδιοριστεί η συνάφειά της. Επιπλέον, σε αυτήν την έρευνα δεν υπάρχει πρόθεση να επανεξεταστεί η πλήρης βιβλιογραφία του TSP ή τα ερευνητικά θέματα που σχετίζονται με το ΠΔΟ. Αντίθετα, εστιάζει στο ΠΔΟ και στις παραλλαγές που περιγράφονται στην ενότητα 2.2. Ακολουθώντας την περιγραφόμενη διαδικασία 263 άρθρα κατηγοριοποιήθηκαν σύμφωνα με την ταξινόμηση των προτεινόμενων αλγορίθμων.

2.5 Ανάλυση ερευνητικών Αποτελεσμάτων

Στην ενότητα αυτή παρουσιάζονται τα αριθμητικά αποτελέσματα που προέκυψαν από την κατηγοριοποίηση των σχετικών εργασιών. Η ταξινόμηση των άρθρων έχει χωριστεί σε δύο σκέλη, τις παραλλαγές του ΠΔΟ και τους εφαρμοσμένους αλγόριθμους για την επίλυση αυτών των παραλλαγών. Η παρούσα έρευνα επικεντρώνεται σε δημοσιευμένα άρθρα από το 2010 έως το πρώτο τρίμηνο του 2020 για να εντοπίσει τις τάσεις της τελευταίας δεκαετίας.

Στην προσπάθειά τους να αντιμετωπίσουν προβλήματα της πραγματικής ζωής, οι περισσότερες δημοσιεύσεις εξετάζουν περισσότερες από μία παραλλαγές του προβλήματος. Το ΠΔΟΠΧ είναι μακράν η πιο μελετημένη παραλλαγή (82,89%) λόγω της απλότητάς της. Οι δημοσιεύσεις που δεν αντιμετωπίζουν το ΠΔΟΠΧ ασχολούνται με το ΠΔΟΕΣΟ (17,11%). Οι παραλλαγές που χρησιμοποιούν χρονικά παράθυρα μελετώνται συχνά καθώς αντικατοπτρίζουν επαρκώς τις πραγματικές περιπτώσεις και έτσι εμφανίζονται σχεδόν στις μισές περιπτώσεις (46,39%). Αντίστοιχα, το ΠΔΟΠΠ εμφανίζεται στο ένα πέμπτο των αναθεωρημένων εργασιών λόγω της υψηλής συσχέτισής του με τις πραγματικές περιπτώσεις διανομής. Το ΠΠΔΟ φαίνεται να προσελκύει την προσοχή των ερευνητών τα τελευταία 6 χρόνια λόγω της τρέχουσας ανάγκης για ελαχιστοποίηση των εκπομπών CO_2.

Η ανάγκη για καλύτερη διαχείριση του δικτύου και μείωση του κόστους οδήγησε το ΠΔΟΠΑ και το ΣΠΔΟ να καταταχθούν μεταξύ των οι πιο μελετημένων παραλλαγών του ΠΔΟ. Επίσης, σε σύγκριση με άλλες μελέτες ταξινόμησης το ΠΔΟΠΚ φαίνεται να έχει κερδίσει το ενδιαφέρον τα τελευταία χρόνια λόγω της ανάγκης των εταιρειών να βελτιστοποιήσουν τη δρομολόγηση σε κάθε στάδιο της εφοδιαστικής αλυσίδας.

Επιπλέον, το ΔΠΔΟ και το ΧΕΠΔΟ έχουν εκμεταλλευτεί νέες τεχνολογίες, οι οποίες προσφέρουν σε πραγματικό χρόνο και μεγάλα δεδομένα. Από την άλλη πλευρά, παραλλαγές όπως το ΑΠΔΟ, το ΠΔΟΧΩΠ και το ΠΔΟΠΔ, φαίνεται να έχουν σχετικά χαμηλό αλλά σταθερό ενδιαφέρον από τους ερευνητές την τελευταία δεκαετία λόγω της συσχέτισής τους με πραγματικές περιπτώσεις logistics που δεν εμφανίζονται συχνά.

Όσον αφορά τους τους αλγορίθμους, οι μεταευρετικοί κυριαρχούν στους προτεινόμενους αλγόριθμους με 252 περιπτώσεις, ακολουθούμενοι από τους Κλασσικούς Ευρετικούς (122) και τους Υβριδικούς (53). Οι ακριβείς αλγόριθμοι (32) αναπτύσσονται λιγότερο συχνά λόγω της υπολογιστικής πολυπλοκότητάς τους και του περιορισμού τους στην επιτυχή εφαρμογή τους μόνο σε προβλήματα μικρής κλίμακας.

Σε μια πιο λεπτομερή ανάλυση, οι κατασκευαστικοί αλγόριθμοι φαίνεται να εμφανίζονται πιο συχνά καθώς εφαρμόζονται για την επίλυση του προβλήματος και τη δημιουργία της αρχικής λύσης σε μεταευρετικούς αλγόριθμους. Όσον αφορά τους μεταευρετικούς αλγόριθμους, οι ΓΑ έχουν προσελκύσει το ενδιαφέρον των ερευνητών την τελευταία δεκαετία. Συνολικά, οι ΕΑ, που περιέχουν ΓΑ και ΜΑ, αναπτύσσονται και εφαρμόζονται πιο συχνά από τους αλγόριθμους που βασίζονται σε σμήνη (ΒΑΜ και ΒΣΣ). Όσον αφορά τους μεταευρετικούς τοπικής αναζήτησης, οι ερευνητές εφαρμόζουν και αναπτύσσουν εξίσου αυτούς τους αλγόριθμους, εκτός από τον ΑΤΠΑ ο οποίος είναι λιγότερο μελετημένος.

Επίσης, όπως ήδη αναφέρθηκε παραπάνω, οι παραλλαγές του ΠΔΟ σπάνια μελετώνται μεμονωμένα. Στις περισσότερες περιπτώσεις, οι ερευνητές αντιμετωπίζουν έναν συνδυασμό αυτών των παραλλαγών ταυτόχρονα για μια καλύτερη κατά προσέγγιση περίπτωση της πραγματικής ζωής. Το ΠΔΟΠΧ, το ΠΔΟΕΣΟ και το ΠΔΟΧΠ είναι από τις παραλλαγές που συνδυάζονται με όλες σχεδόν τις λοιπές παραλλαγές. Το ΠΔΟΠΧ συνδυάζεται περισσότερο με άλλες παραλλαγές καθώς απλοποιεί σημαντικά το πρόβλημα. Όσον αφορά τις υπόλοιπες παραλλαγές, πολλαπλοί συνδυασμοί μεταξύ των ΠΔΟΧΠ, ΠΔΟΠΠ, ΠΔΟΕΣΟ, ΠΔΟΠΑ και ΠΠΔΟ παρατηρούνται και σχηματίζονται από ερευνητές.

Επιπρόσθετα, πολλές δημοσιεύσεις προτείνουν υβριδικούς αλγόριθμους επίλυσης για όλες σχεδόν τις παραλλαγές του ΠΔΟ, υποδεικνύοντας μια τάση στην ερευνητική κοινότητα προς αυτούς τους αλγόριθμους. Οι ευρετικές τεχνικές κατασκευής και τοπικής βελτίωσης εμπλέκονται στη λύση όλων των παραλλαγών ΠΔΟ. Όσον αφορά τον ΠΔΟΧΠ, ενώ ο Tabu και ο ΠΑ ήταν μεταξύ των πιο κοινά εφαρμοζόμενων αλγορίθμων, φαίνεται να έχει διαμορφωθεί μια νέα κατεύθυνση προς τους ΕΑ. Σε περιπτώσεις που λαμβάνονται υπόψη μόνο το ΠΔΟΠΧ και το ΠΔΟΧΠ, η δομή του χρωμοσώματος στους ΓΑ συμβάλλει στην ταχύτερη εκτέλεση των τελεστών διασταύρωσης και μετάλλαξης και στην αποθήκευση του πληθυσμού.

Το ΠΔΟΕΣΟ είναι μια σημαντική παραλλαγή για εταιρείες διανομής, καθώς οι περισσότερες από αυτές διαθέτουν μεικτό στόλο οχημάτων. Ωστόσο, μόνο το 17,11% των άρθρων εξετάζει τη συγκεκριμένη παραλλαγή. Οι πιο συνηθισμένοι αλγόριθμοι που εφαρμόζονται στο ΠΔΟΕΣΟ είναι i) οι ΓΑ, ii) οι ΑΜΓ, iii) οι αλγόριθμοι ΠΡΑΕΓ και ΠΑΕΓ και iv) ο αλγόριθμος ΑΤΑ. Οι τρεις μεταευρετικοί αλγόριθμοι τοπικής αναζήτησης εφαρμόζονται σε μια ενιαία λύση που οδηγεί σε μικρότερους υπολογιστικούς χρόνους. Αντίθετα, οι ΕΑ που εφαρμόζονται συχνά, χρειάζονται λύσεις αποθήκευσης οδηγώντας σε αυξημένους υπολογιστικούς χρόνους.

Τέλος, το Πράσινο, το Ηλεκτρικό, το Υβριδικό ΠΔΟ και το Πρόβλημα Ρύπανσης Δρομολόγησης θεωρούνται μία κατηγορία στην ανάλυση. Αυτές οι παραλλαγές έχουν προταθεί και εμφανιστεί τα τελευταία δεκαπέντε χρόνια και έχουν μελετηθεί διεξοδικά από ερευνητές. Σε αυτές τις πρώτες προσπάθειες για λύσεις που μειώνουν τις εκπομπές CO_2 και την κατανάλωση καυσίμου, οι αλγόριθμοι ΠΡΑΕΓ και ΠΑΕΓ εφαρμόζονται ως επί το πλείστων.

2.6 Πρακτικές Εφαρμογές

Η παραπάνω έρευνα επιβεβαιώνει ότι το ΠΔΟ αντιμετωπίζεται ευρέως από τους ερευνητές και αποτελεί ένα από τα πιο μελετημένα προβλήματα στην εφοδιαστική. Ωστόσο, υπάρχει περιορισμένος αριθμός ερευνητικών εργασιών που οδηγούν στην ενσωμάτωση αλγορίθμων σε συστήματα δρομολόγησης. Η δήλωση ενισχύεται από τη βιβλιογραφική ανασκόπηση, η οποία αξιοποιεί τη βάση δεδομένων SCOPUS. Ο όρος εφαρμόζεται στον τίτλο, την περίληψη, τις λέξεις κλειδια και είναι ("πρόβλημα δρομολόγησης οχήματος" και («πληροφοριακά συστήματα» ή «σύστημα» ή «λογισμικό») και («μεταευρετικοί» ή «ευρετικοί» ‘η «ακριβείς») και «μελέτη περίπτωσης "). Ο αριθμός των εργασιών που προκύπτουν από την έρευνα είναι 92. Ωστόσο, μόνο 5 από αυτές συζητούν την ανάπτυξη ενός συστήματος δρομολόγησης ή την ενσωμάτωση ενός αλγορίθμου σε μια πραγματική εφαρμογή δρομολόγησης.

Δύο από αυτές τις μελέτες (Fanti et al. 2014; Abbatecola et al. 2016) στοχεύουν στην ανάπτυξη Συστημάτων Υποστήριξης Αποφάσεων (ΣΥΑ), εστιάζοντας κυρίως στην αρχιτεκτονική των συστημάτων και στη μαθηματική διατύπωση του προβλήματος τους, χωρίς να παρουσιάζουν αναλυτικά τις αλγοριθμικές προσεγγίσεις. Επιπλέον, οι Tarantilis και Kiranoudis (2007) πρότειναν έναν ευέλικτο προσαρμοστικό αλγόριθμο βασισμένο στη μνήμη για την αντιμετώπιση του ΠΔΟΕΣΟ και ο οποίος εντάχθηκε σε ένα ΣΥΑ. Τέλος, οι Tarantilis et al. (2004) πρότειναν ένα ΣΥΑ που χρησιμοποιεί έναν μεταευρετικό αλγόριθμο για την επίλυση του ΑΠΔΟ. Το προτεινόμενο ΣΥΑ γεωκωδικοποιεί και χαρτογραφεί τις τοποθεσίες των πελατών μέσω σύνδεσης σε ένα Γεωγραφικό Σύστημα Πληροφοριών (ΓΣΠ).

Όλες οι προαναφερθείσες έρευνες επικεντρώθηκαν σε περιορισμένο αριθμό παραλλαγών ΠΔΟ, με αποτέλεσμα να περιορίσουν τις πιθανές πραγματικές περιπτώσεις που μπορούν να ωφεληθούν από τη χρήση των συστημάτων τους. Η έρευνα του (Erdoğan 2017) είναι η μόνη που πλησιάζει την παρουσιαζόμενη έρευνα, καθώς έχει αναπτυχθεί ένα υπολογιστικό φύλλο ανοιχτού κώδικα για την αντιμετώπιση περισσότερων παραλλαγών. Ο ερευνητής συνδέει διαδικτυακούς χάρτες με το Microsoft Excel και εξάγει σημαντικά δεδομένα που σχετίζονται με τη δρομολόγηση. Ο περιορισμός αυτής της λύσης είναι ο αυξημένος υπολογιστικός χρόνος που απαιτείται καθώς η γλώσσα προγραμματισμού Visual Basic (VB), δεν είναι μια πολύ αποτελεσματική γλώσσα προγραμματισμού για τη βελτιστοποίηση.

Παρά την πρόοδο της τεχνολογίας, μόνο λίγες μελέτες που προτείνουν αλγόριθμους οδηγούν σε πραγματικές εφαρμογές και συστήματα. Ένας λόγος είναι η εξειδικευμένη γνώση που απαιτείται για την ανάπτυξη ενός λειτουργικού συστήματος. Επιπλέον, οι εφαρμογές και τα πακέτα λογισμικού της πραγματικής ζωής δημιουργούνται κυρίως από εταιρείες λογισμικού για τις οποίες οι αλγόριθμοι αποτελούν πνευματική ιδιοκτησία και δεν αποκαλύπτονται.

2.7 Συμπεράσματα

Αυτό το Κεφάλαιο μελετά και παρουσιάζει πολλαπλές παραλλαγές ΠΔΟ που έχουν μελετηθεί όλα αυτά τα χρόνια και συσχετίζονται με τη διανομή εμπορευμάτων. Επιπλέον, κατηγοριοποιούνται και παρουσιάζονται οι αλγοριθμικές προσεγγίσεις που έχουν αναπτυχθεί και προταθεί για την επίλυση αυτών των περιπτώσεων. Τόσο οι παραλλαγές όσο και οι αλγόριθμοι είναι τα κύρια πεδία μελέτης και η εκτενής βιβλιογραφική ανασκόπηση στοχεύει ως πρώτο βήμα στη συσχέτισή τους.

Η βιβλιογραφική ανασκόπηση προέκυψε από μια καλά καθορισμένη μεθοδολογική προσέγγιση και ερευνητικό πρωτόκολλο. Μια αρχική ομάδα 334 άρθρων που εξήχθη από τη βάση δεδομένων Scopus αξιολογήθηκε και περιορίστηκε περαιτέρω σε μια ομάδα 263 σχετικών άρθρων μετά την εφαρμογή κριτηρίων επιλογής. Έμφαση δόθηκε στους αλγόριθμους που προτάθηκαν και αναπτύχθηκαν για το ΠΔΟ στη διανομή εμπορευμάτων. Επομένως, δημοσιεύσεις που δεν είναι σχετικά με τη μεταφορά εμπορευμάτων εξαιρέθηκαν.

Τα άρθρα ταξινομήθηκαν σύμφωνα με τις παραλλαγές ΠΔΟ και την μέθοδο που προτάθηκε για τη λύση τους για τον εντοπισμό των τάσεων σε κάθε περίπτωση και την παρουσίαση της συσχέτισής τους. Μια τάση προς τους ΕΑ και τους Μεταευρετικούς Τοπικής Αναζήτησης φαίνεται να διαμορφώνεται, καθώς εφαρμόζονται σε πολλές παραλλαγές του προβλήματος και προσφέρουν πολύ αποτελεσματικά αποτελέσματα. Επιπλέον, οι υβριδικοί αλγόριθμοι εφαρμόζονται όλο και πιο συχνά σε πολλές παραλλαγές του ΠΔΟ.

Τέλος, η ανασκόπηση της βιβλιογραφίας δεν περιορίστηκε στη μελέτη των παραλλαγών ΠΔΟ, των προτεινόμενων αλγορίθμων και της συσχέτισής τους, αλλά επεκτάθηκε και στις ερευνητικές μελέτες που προτείνουν αλγόριθμους βελτιστοποίησης για την αντιμετώπιση του ΠΔΟ και που ενσωματώνονται επίσης στη δρομολόγηση της πραγματικής ζωής. εφαρμογές. Τα αποτελέσματα οδήγησαν σε περιορισμένο μόνο αριθμό άρθρων που προτείνουν τόσο αλγόριθμους βελτιστοποίησης όσο και την ενσωμάτωσή τους σε συστήματα δρομολόγησης. Το γεγονός αυτό, μαζί με την απουσία συστημάτων δρομολόγησης cloud επιβεβαιώνουν το ερευνητικό χάσμα που εξακολουθεί να υπάρχει μεταξύ των ερευνητικών μελετών και των εφαρμογών της πραγματικής ζωής.

3. Το Πρόβλημα της Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα

3.1 Εισαγωγή

Το πρόβλημα δρομολόγησης οχημάτων με τα παράθυρα χρόνου (ΠΔΟΧΠ) είναι μια πολύ γνωστή παραλλαγή του ΠΔΟ, η οποία έχει λάβει μεγάλη προσοχή τα τελευταία χρόνια και αντικατοπτρίζει πολλά σενάρια πραγματικού κόσμου στις λειτουργίες διανομής. Το ΠΔΟΧΠ αποτελείται από γεωγραφικά διασκορπισμένους πελάτες που πρέπει να εξυπηρετηθούν μέσα σε ένα προκαθορισμένο χρονικό διάστημα (χρονικό παράθυρο), μόνο μία φορά και από ένα μόνο όχημα. Η συνολική ποσότητα εμπορευμάτων που διανέμεται σε κάθε διαδρομή δεν μπορεί να υπερβαίνει τη χωρητικότητα του οχήματος, ενώ κάθε όχημα ξεκινά και τελειώνει τη διαδρομή του στο αμαξοστάσιο. Ένα όχημα μπορεί να φτάσει σε έναν πελάτη πριν από το άνοιγμα του χρονικού παραθύρου και να περιμένει μέχρι τον συμφωνημένο χρόνο σέρβις, αλλά δεν επιτρέπεται να παραδώσει τα αγαθά εάν φτάσει μετά το κλείσιμο του χρονικού παραθύρου (César and Oliveira 2010). Αντίθετα, πολλές ερευνητικές εργασίες (Taş et al. 2014) θεωρούν ότι ορισμένοι ή όλοι οι πελάτες έχουν «χαλαρά» χρονικά παράθυρα και μπορεί να εξυπηρετηθεί πριν από το μετά το κλείσιμο του χρονικού παραθύρου, πληρώνοντας ποινή κάθε φορά που παραβιάζεται το χρονικό παράθυρο.

Για την επίλυσή τους αρχικά προτάθηκαν ακριβείς αλγόριθμοι οι οποίοι υπολογίζουν κάθε δυνατή λύση έως ότου επιτευχθεί η καλύτερη, ενώ έχουν πολύ κακή απόδοση σε υπολογιστικό χρόνο, ακόμη και όταν ασχολούνται με σχετικά μικρές περιπτώσεις (El-Sherbeny 2010). Παρά αυτόν τον περιορισμό, έχουν προταθεί ακριβείς αλγόριθμοι από ερευνητές για την αντιμετώπιση του ΠΔΟΧΠ, όλοι εκ των οποίων επιλύουν περιπτώσεις διανομής με έως και 100 πελάτες παραγγελίες. Κατά συνέπεια, οι ερευνητές κατευθύνθηκαν προς τους ευρετικούς αλγόριθμους οι οποίοι παρέχουν καλύτερη αντιστάθμιση μεταξύ ποιότητας λύσης και χρόνου υπολογισμού. Αυτή η θεωρία ενισχύεται από ορισμένα παραδείγματα επιτυχούς εφαρμογής ευρετικών αλγορίθμων που επιλύουν το ΠΔΟΧΠ. Πιο συγκεκριμένα, η εργασία του Solomon (1987) θεωρείται πρωτοποριακή, όπως και αυτή των Pisinger και Ropke (2007) πρότειναν ένα ευρετικό που μπορεί να λύσει μια κατηγορία παραλλαγών ΠΔΟ, συμπεριλαμβανομένων των χρονικών παραθύρων.

Παρόλο που οι ευρετικοί προσφέρουν λύσεις καλής ποιότητας, οι τελευταίες εξελίξεις στην τεχνολογία επέτρεψαν στους μεταευρετικούς αλγόριθμους να αναζητήσουν έναν ευρύτερο χώρο λύσεων σε περιορισμένο χρόνο CPU, ενώ προσφέρουν ποιοτικά αποτελέσματα. Το ΠΔΟΧΠ συνήθως αντιμετωπίζεται θέτοντας μία αντικειμενική συνάρτηση. Παρ' όλα αυτά τα τελευταία χρόνια αντιμετωπίζεται και ως πολύ-αντικειμενικό, με τους (Tan et al. 2006) και (Ombuki et al. 2006) να είναι οι πρώτοι που θεώρησαν το ΠΔΟΧΠ ως πολυαντικειμενικό, εφαρμόζοντας την έννοια της βελτιστοποίησης Pareto.

Σε αυτή την κατεύθυνση, στο συγκεκριμένο Κεφάλαιο παρουσιάζεται η ανάπτυξη ενός αλγορίθμου Πολυαντικειμενικής Αναζήτησης Ευρείας Γειτονιάς (ΠΑΕΓ) που εκμεταλλεύεται τη λύση που προκύπτει από τον ευρετικό αλγόριθμο κατασκευής. Ο προτεινόμενος αλγόριθμος ΠΑΕΓ αντιμετωπίζει το ΠΔΟΧΠ ως πολύ-αντικειμενικό και στοχεύει στην ελαχιστοποίηση τόσο του αριθμού των οχημάτων όσο και της συνολικά διανυόμενης απόστασης εφαρμόζοντας τελεστές «καταστροφής» και «επισκευής», διατηρώντας ταυτόχρονα την έννοια της βέλτιστης Pareto. Τα αποτελέσματα του αλγορίθμου συγκρίνονται με τα καλύτερα δημοσιευμένα αποτελέσματα της βιβλιογραφίας, υποδεικνύοντας έναν πολύ αποτελεσματικό αλγόριθμο.

Στο υπόλοιπο του Κεφαλαίου, το ΠΔΟΤW ορίζεται και διατυπώνεται στην ενότητα 3.2, ενώ στην ενότητα 3.3 παρουσιάζεται διεξοδικά ο προτεινόμενος αλγόριθμος MOLNS που στοχεύει στην επίλυση πραγματικών περιπτώσεων ΠΔΟΤW. Τα υπολογιστικά αποτελέσματα του προτεινόμενου αλγορίθμου στις περιπτώσεις αναφοράς του Solomon παρουσιάζονται στην ενότητα 3.4. Στην ίδια ενότητα, συζητείται η αναμενόμενη εφαρμογή του αλγορίθμου σε λογισμικό δρομολόγησης και προγραμματισμού. Τέλος, η ενότητα 3.5 περιέχει τα συμπεράσματα του Κεφαλαίου.

3.2 Περιγραφή Προβλήματος

Από τη στιγμή που έχουν καθοριστεί οι παράμετροι του προβλήματος, πρέπει να διαμορφωθεί και το μαθηματικό μοντέλο του τελικού προβλήματος. Πιο συγκεκριμένα, το πρόβλημα καθορίζεται από ένα δίκτυο G (N, A), όπου N το σύνολο των κόμβων και and $A = (i, j): i \neq j, i, j \in N$ το σύνολο των τόξων (arcs). Ο κόμβος {0} αντιπροσωπεύει την αποθήκη, ενώ το σύνολο $N^* = \{N/0\}$ περιγράφει το σύνολο των πελατών. Κάθε τόξο (i, j), το οποίο είναι η διαδρομή από τον κόμβο i προς τον κόμβο j, χαρακτηρίζεται από την απόσταση d_{ij}, καθώς και από τον χρόνο μετακίνησης t_{ij}. Επιπλέον, κάθε πελάτης χαρακτηρίζεται από τις ποσότητες προς παράδοση, q_i, ενώ επίσης ο πελάτης ορίζει ένα χρονικό παράθυρο $[e_i, l_i]$, που αντιπροσωπεύει την ενωρίτερη και την αργότερη δυνατή έναρξη φόρτωσης/ εκφόρτωσης εμπορευμάτων στον πελάτη. Οι πελάτες πρέπει αν εξυπηρετηθούν ακριβώς μια φορά και από ένα μοναδικό όχημα. Ο χρόνος άφιξης δίνεται από το a_i. Επιπλέον, αν το όχημα φτάσει πριν την άνοιξη του χρονικού παραθύρου (e_i), τότε πρέπει να περιμένει w_i^k χρόνο, μέχρι η εξυπηρέτηση να είναι εφικτή, ενώ κανένα όχημα δεν μπορεί να καταφτάσει μετά την ολοκλήρωση του χρονικού παραθύρου (l_i).

3.3 Μέθοδος Επίλυσης – Πολυαντικειμενική Αναζήτηση Ευρείας Γειτονιάς

Ο προτεινόμενος αλγόριθμος Πολυαντικειμενικής Αναζήτησης Ευρείας Γειτονιάς (ΠΑΕΓ), ως πρώτο βήμα, λαμβάνει την αρχική λύση εφαρμόζοντας τον αλγόριθμο του πλησιέστερου γείτονα με προσανατολισμό στον χρόνο, ο οποίος είναι ευρετικός κατασκευαστικός. Κατά συνέπεια, σε κάθε επανάληψη, εφαρμόζονται οι χειριστές καταστροφής και επισκευής για τη βελτίωση της λύσης. Εάν η πρόσφατα ληφθείσα λύση είναι καλύτερη από την τρέχουσα

καλύτερη, τότε την αντικαθιστά και χρησιμοποιείται ως είσοδος στην επόμενη επανάληψη. Για να αξιολογηθεί εάν μια λύση είναι καλύτερη από μια άλλη, εφαρμόζεται η έννοια της βελτιστοποίησης Pareto και αναλύεται διεξοδικά στην ενότητα 3.3.1. Ο κατασκευαστικός ευρετικός αλγόριθμος κατασκευής περιγράφεται στην ενότητα 3.3.2. Στην ενότητα 3.3.3 και 3.3.4, παρουσιάζονται οι κινήσεις καταστροφής και επισκευής αντίστοιχα. Τέλος, το γενικό πλαίσιο του αλγορίθμου MOLNS παρουσιάζεται στην ενότητα 3.3.5.

3.3.1 Αποτελεσματικότητα κατά Pareto

Οι παραδόσεις σε τελικούς πελάτες και καταναλωτές αποτελούν βασικό μέρος της αλυσίδας εφοδιασμού και των λειτουργιών logistics. Εμπλέκονται πολλά ενδιαφερόμενα μέρη, όπως εταιρείες διανομής τρίτων (3PL), εταιρείες μεταφορών ή ακόμα και εταιρείες παραγωγής που διαθέτουν δικό τους στόλο οχημάτων. Ανάλογα με την περίπτωση, είτε ο αριθμός των οχημάτων είτε η συνολικά διανυόμενη απόσταση είναι πιο σημαντικός ή συμβάλλει περισσότερο στο κόστος. Ως εκ τούτου, είναι σημαντικό να επιτευχθεί κάθε πιθανή λύση χωρίς κυριαρχία, και κάθε ενδιαφερόμενος παίρνει την απόφαση και επιλέγει τη λύση που είναι πιο κερδοφόρα (Ehrgott 2005). Πιο συγκεκριμένα, κατά τη μετάβαση από μια βέλτιστη λύση Pareto σε μια άλλη, υπάρχει πάντα μείωση σε έναν στόχο και ταυτόχρονη αύξηση στον δεύτερο στόχο.

Τόσο στη φάση της κατασκευής όσο και στη φάση του αλγορίθμου LNS, κάθε λύση που προκύπτει ελέγχεται αν ανήκει είτε στις λύσεις που κυριαρχούν είτε στις μη κυριαρχούμενες λύσεις. Εάν μια λύση που λαμβάνεται από τον αλγόριθμο ανήκει στις μη κυριαρχούμενες λύσεις, τότε το βέλτιστο σύνολο Pareto ενημερώνεται. Επομένως, κάθε λύση που λαμβάνεται από τον προτεινόμενο αλγόριθμο συγκρίνεται με τις μη κυριαρχούμενες λύσεις για να διασφαλιστεί ότι οι βέλτιστες λύσεις διατηρούνται και ενημερώνονται συνεχώς.

3.3.2 Κατασκευαστική Διαδικασία

Έχουν αναπτυχθεί αρκετοί ευρετικοί αλγόριθμοι για τη δημιουργία λύσεων για το ΠΔΟ. Στην παρούσα ενότητα επιλέγεται ο αλγόριθμος του πλησιέστερου γείτονα με προσανατολισμό στον χρόνο (ΠΓΠΧ) για τη λήψη των αρχικών λύσεων λόγω της απλότητας και της ταχύτητάς του. Ο αλγόριθμος λειτουργεί δημιουργώντας διαδρομές που ξεκινούν από την αποθήκη (κόμβος 0), και κατά συνέπεια, προστίθεται ο "πλησιέστερος" στον τελευταίο κόμβο που επισκέφθηκε ο μη δρομολογημένος πελάτης. Κατά την αναζήτηση του "πλησιέστερου" πελάτη, πρέπει να τηρούνται οι περιορισμοί χρονικού παραθύρου και οι περιορισμοί χωρητικότητας. Διαφορετικά, δεν μπορεί να προστεθεί πελάτης. Ξεκινά μια νέα διαδρομή εάν δεν βρεθεί πελάτης, εκτός εάν δεν έχουν μείνει άλλοι πελάτες για προσθήκη, που σημαίνει ότι όλες οι διαδρομές έχουν κατασκευαστεί και η διαδικασία έχει ολοκληρωθεί. Ο Solomon (Solomon 1987) πρότεινε αυτή τη μέθοδο σε μια προσπάθεια να λύσει το ΠΔΟΧΠ βέλτιστα.

3.3.3 Πολυαντικειμενική Αναζήτηση Ευρείας Γειτονιάς

Όπως περιγράφεται στην ενότητα 3.3.2, οι πελάτες προστίθενται σε κάθε διαδρομή, με βάση την "εγγύτητα" τους με τον τελευταίο προστιθέμενο πελάτη. Ωστόσο, τα χρονικά παράθυρα επηρεάζουν σε μεγάλο βαθμό την αρχική λύση, η οποία σε πολλές περιπτώσεις αποκλίνει από τη βέλτιστη. Επομένως, αφού κατασκευαστούν οι αρχικές λύσεις, εφαρμόζεται ο αλγόριθμος ΠΑΕΓ που καταστρέφει εν μέρει και στη συνέχεια επισκευάζει τη λύση για να ελαχιστοποιηθεί τόσο ο αριθμός των οχημάτων όσο και η συνολική απόσταση που διανύθηκε. Οι τελεστές καταστροφής είναι 5 ενώ οι κινήσεις επισκευής είναι 2.

3.3.4 Γενικό Πλαίσιο Αλγορίθμου

Αρχικά, οι βέλτιστες λύσεις Pareto που κατασκευάζονται από τον αλγόριθμο ΠΓΠΧ, χρησιμοποιούνται ως είσοδος στον αλγόριθμο ΠΑΕΓ. Σε κάθε επανάληψη που εφαρμόζεται ο αλγόριθμος ΠΑΕΓ, οι κινήσεις καταστροφής και επισκευής εφαρμόζονται σε κάθε μία από τις βέλτιστες λύσεις Pareto. Κατά συνέπεια, για κάθε λύση, λαμβάνεται μια νέα - X_{new}. Εάν η λύση X_{new} αποτελεί βέλτιστη λύση Pareto, καθώς ανήκει στις μη κυριαρχούμενες λύσεις, τότε το βέλτιστο σύνολο Pareto ανανεώνεται, διαφορετικά δεν γίνονται αλλαγές.

3.4 Υπολογιστική Μελέτη

Αυτή η ενότητα περιγράφει τα αποτελέσματα των υπολογιστικών πειραμάτων που πραγματοποιήθηκαν για την αξιολόγηση της απόδοσης του προτεινόμενου αλγορίθμου. Ο αλγόριθμος αναπτύχθηκε σε Python και έτρεχε σε προσωπικό υπολογιστή με τις ακόλουθες προδιαγραφές: Intel Core i7 - 8550U 1,8 GHz με μνήμη 16 GB. Ο προτεινόμενος αλγόριθμος δοκιμάστηκε σε περιπτώσεις αναφοράς ΠΔΟΧΠ του Solomon με 100 πελάτες (Solomon 1987). Στο συγκεκριμένο σύνολο δεδομένων, τα προβλήματα C1 και C2 έχουν γεωγραφικά συγκεντρωμένους πελάτες. Στα R1 και R2, τα γεωγραφικά δεδομένα δημιουργούνται τυχαία, ενώ στα σύνολα προβλημάτων RC1 και RC2, περιλαμβάνεται ένας συνδυασμός τυχαίων και ομαδοποιημένων πελατών. Τα σύνολα R1, C1 και RC1 έχουν μικρό χρονικό ορίζοντα προγραμματισμού, μικρού εύρους χρονικά παράθυρα και χαμηλή χωρητικότητα οχημάτων, ενώ τα σετ R2, C2 και RC2 έχουν μεγάλο χρονικό ορίζοντα προγραμματισμού, μεγάλου εύρους χρονικά παράθυρα και μεγάλη χωρητικότητα οχημάτων, που καθορίζουν τον αριθμό των πελατών που εξυπηρετείται από το ίδιο όχημα.

Κάθε περίπτωση εκτελέστηκε τρεις φορές, ενώ το υπολογιστικό όριο έχει οριστεί στα 15 λεπτά. Φυσικά, η CPU και η γλώσσα προγραμματισμού είναι επίσης σημαντικοί παράγοντες που επηρεάζουν τον υπολογιστικό χρόνο που απαιτείται. Στις περισσότερες περιπτώσεις, τα αποτελέσματα απαιτούν λιγότερο από τα 15 λεπτά που είναι το όριο. Δεδομένου ότι το ΠΔΟΤW αντιμετωπίζεται ως πολυαντικειμενικό, σε ορισμένες περιπτώσεις, ελήφθησαν περισσότερες από μία λύσεις. Για αυτόν τον λόγο τα αποτελέσματα συγκρίνονται μόνο όταν έχουν ίδιο αριθμό οχημάτων σαν λύση. Εφόσον ισχύει η προϋπόθεση αυτή, η απόκλιση υπολογίζεται μεταξύ της συνολικά διανυόμενης απόστασης που προκύπτει μεταξύ των

βέλτιστων λύσεων της βιβλιογραφίας και των λύσεων που προκύπτουν από τον προτεινόμενο αλγόριθμο. Όσον αφορά τα αποτελέσματα, στις περιπτώσεις με γεωγραφικά ομαδοποιημένους πελάτες (C1 και C2) τα αποτελέσματα είναι καλύτερα σε σχέση με αυτές που έχουν τυχαία γεωγραφικά δεδομένα, καθώς η μέση απόκλιση, σε αυτήν την περίπτωση, είναι 0,30%. Στα τυχαία γεωγραφικά δεδομένα (R), η μέση απόκλιση είναι μεγαλύτερη και φτάνει το 3,37%, ενώ στα μικτά γεωγραφικά δεδομένα πελατών (RC) είναι 3,47%. Τέλος, η μέση απόκλιση για όλες τις περιπτώσεις είναι μικρότερη από 2,85%, η οποία μπορεί να θεωρηθεί πολύ αποτελεσματική αφού τα βέλτιστα αποτελέσματα έχουν προκύψει από πολλούς ερευνητές με την πάροδο του χρόνου. Συνολικά, το 32,47% των συγκρίσιμων λύσεων έχουν απόκλιση μικρότερη από 1%, οι περισσότερες από αυτές σε περιπτώσεις C. Ταυτόχρονα, το 98,70% των αποτελεσμάτων αποκλίνει λιγότερο από 10% από το βέλτιστο. Τέλος, τρεις νέες βέλτιστες λύσεις Pareto προτείνονται, στα RC202, RC207 και R201. Στο πρόβλημα RC202, επιτυγχάνεται μείωση της συνολικής διανυόμενης απόστασης, αλλά σε βάρος ενός επιπλέον οχήματος. Στις περιπτώσεις R201 και RC202, ο αλγόριθμος κατάφερε να βελτιώσει τις λύσεις μειώνοντας τη συνολική απόσταση, ενώ ο αριθμός των οχημάτων παρέμεινε σταθερός.

3.5 Συμπεράσματα

Τα αποτελέσματα που προέκυψαν από τον αλγόριθμο ΠΑΕΓ συγκρίθηκαν με τις βέλτιστες δημοσιευμένες λύσεις στα προβλήματα δίνοντας σχεδόν κοντινά στις βέλτιστες λύσεις αποτελέσματα, περιλαμβάνοντας ακόμη και τρεις νέες λύσεις Pareto. Τα υπολογιστικά πειράματα δείχνουν ότι τα αποτελέσματα είναι αποτελεσματικά και λαμβάνονται σε πολύ ρεαλιστικό χρόνο, καθώς τόσο η αποτελεσματικότητα όσο και η ταχύτητα είναι κρίσιμοι παράγοντες κατά την εφαρμογή ενός αλγορίθμου σε λύσεις λογισμικού, ο προτεινόμενος αλγόριθμος MOLNS μπορεί να εξεταστεί για εφαρμογή.

Επιπλέον, το γεγονός ότι ο αλγόριθμος είναι πολυαντικειμενικός δίνει τη δυνατότητα επιλογής της λύσης που ταιριάζει καλύτερα στις ανάγκες. Αυτό είναι σημαντικό σε περιπτώσεις που πρέπει να βρεθεί η καλύτερη αντιστάθμιση μεταξύ δύο στόχων. Τα οφέλη του αλγόριθμου φαίνεται να είναι σημαντικά και γι' αυτό ο αλγόριθμος αποθηκεύεται στη βάση δεδομένων cloud της Oracle και είναι προσβάσιμος από το σύστημα που παρουσιάζεται στο Κεφάλαιο 6, με τις κατάλληλες αλλαγές στο αίτημα.

Επιπλέον, για να καθοριστεί η αποδοτικότητα του αλγορίθμου ως προς το χρόνο εκτέλεσης, αναπτύσσεται ένας άλλος πολυσκοπικός μεταευρετικός αλγόριθμος που μπορεί να λύσει το ΠΔΟΧΠ. Ο αλγόριθμος είναι ένας Εξελικτικός Αλγόριθμος (ΕΑ) που χειρίζεται ταυτόχρονα χρονικά παράθυρα, καθώς και ταυτόχρονες παραλαβές και παραδόσεις. Ο ΕΑ παρουσιάζεται παρακάτω στο Κεφάλαιο 4.

4. Πρόβλημα της Δρομολόγησης Οχημάτων με Ταυτόχρονες Περισυλλογές και Παραδόσεις και με Χρονικά Παράθυρα

4.1 Εισαγωγή

Το δίκτυο εφοδιαστικής αλυσίδας περιλαμβάνουν πολλές διαδικασίες, όπως διανομή αγαθών σε λιανοπωλητές και καταναλωτές, καθώς και η αποθήκευση προϊόντων και η προμήθεια πρώτων υλών σε κατασκευαστές. Επιπλέον, περιλαμβάνουν τη συλλογή προϊόντων από τελικούς πελάτες για ανακύκλωση, ανακατασκευή ή επισκευή (Govindan et al. 2015). Δεδομένου του εύρους των λειτουργιών της εφοδιαστικής αλυσίδας, η δρομολόγηση των οχημάτων και ο προγραμματισμός των παραδόσεων αποτελούν ένα μικρό αλλά ουσιαστικό κομμάτι αυτής. Επομένως, η αποτελεσματική δρομολόγηση των οχημάτων και ο προγραμματισμός των παραδόσεων διασφαλίζει ότι θα μειωθεί το κόστος μεταφοράς και θα βελτιωθούν οι παρεχόμενες υπηρεσίες.

Εκτός από το Πρόβλημα της Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (ΠΔΟΧΠ), το οποίο αποτελεί μία από τις πιο συχνά εμφανιζόμενες παραλλαγές του ΠΔΟ, οι ερευνητές εστιάζουν επίσης στο ΠΔΟ με Ταυτόχρονες Περισυλλογές και Παραδόσεις (ΠΔΟΤΠΠ) την έρευνα που παρουσιάζεται στο Κεφάλαιο 2. Στο ΠΔΟΤΠΠ, η κύρια παραλλαγή από το αρχικό πρόβλημα είναι ότι κάθε πελάτης μπορεί να παραλάβει και να επιστρέψει αγαθά (Nagy and Salhi 2005). Αυτή η παραλλαγή δημιουργεί έναν επιπλέον περιορισμό, καθώς σε οποιοδήποτε στάδιο του δρομολογίου, όπου τα αγαθά παραδίδονται και συλλέγονται, η χωρητικότητα του οχήματος δεν μπορεί να παραβιαστεί. Συνδυάζοντας αυτές τις δύο παραλλαγές, αυτό το περίπλοκο πρόβλημα έρχεται πιο κοντά σε πραγματικές περιπτώσεις διανομής.

Επιπλέον, η μετατροπή πραγματικών περιπτώσεων διανομής σε παραλλαγές του ΠΔΟ δεν θα είχε καμία αξία εάν δεν αξιοποιηθεί στο έπακρο η πρόοδος στην τεχνολογία και την επιστήμη των υπολογιστών. Και οι δύο τομείς επέτρεψαν στους ερευνητές να αναπτύξουν πιο αποτελεσματικούς αλγόριθμους για την αντιμετώπιση των παραλλαγών του ΠΔΟ. Περιορισμένες μελέτες έχουν προτείνει ακριβείς αλγόριθμους για αυτήν την παραλλαγή (El-Sherbeny 2010), οι περισσότερες από τις οποίες αντιμετωπίζουν μέχρι 100 παραγγελίες.

Όσον αφορά τους ευρετικούς αλγόριθμους οι οποίοι προσφέρουν λύσεις καλής ποιότητας σε σύντομο χρονικό διάστημα καθώς επίσης έχουν τη δυνατότητα να επιλύουν προβλήματα οποιουδήποτε μεγέθους, αναπτύσσονται συχνότερα από τους ακριβείς αλγόριθμους. Μερικά από τα πιο μελετημένουν ευρετικούς που εφαρμόζονται για διάφορες παραλλαγές του ΠΔΟ, συμπεριλαμβανομένου του ΠΔΟΤΠΠ, είναι (i) η αναζήτηση του πλησιέστερου γείτονα με προσανατολισμό στον χρόνο (ΠΓΠΧ) (Solomon 1987), (ii) ο Sweep (Gillett and Miller 1974) και (iii) τον αλγόριθμο Savings (Clarke and Wright 1964). Παρά το γεγονός ότι είναι άνω των 30 ετών, αυτοί οι αλγόριθμοι παραμένουν η βάση άλλων πιο εξελιγμένων και βελτιωμένων αλγορίθμων και εξακολουθούν να χρησιμοποιούνται σε εμπορικά λογισμικά δρομολόγησης (Gayialis and Tatsiopoulos 2004).

Ωστόσο, το κέντρο της προσοχής έχει μετατοπιστεί στους μεταευρετικούς αλγόριθμους καθώς συνδυάζουν πλεονεκτήματα τόσο από ακριβείς όσο και από ευρετικούς αλγόριθμους. Σύμφωνα με την έρευνα που παρουσιάζεται στο Κεφάλαιο 2, μεταξύ των πιο συχνά αναπτυσσόμενων αλγορίθμων για αυτές τις παραλλαγές του ΠΔΟ είναι οι εξελικτικοί και γενετικοί αλγόριθμοι, η βελτιστοποίηση μέσω σμήνους σωματιδίων, η βελτιστοποίηση αποικίας μυρμηγκιών, προσομοιωμένης ανόπτησης, αναζήτηση ευρείας γειτονιάς και αναζήτηση tabu.

Στις περισσότερες περιπτώσεις, οι αλγόριθμοι που απευθύνονται στο ΠΔΟΧΠ και το ΠΔΟΤΠΠ παράγουν μεροληπτικές λύσεις είτε ως προς τον αριθμό των οχημάτων είτε για τη συνολική απόσταση που διανύθηκε, που είναι οι δύο στόχοι. Ωστόσο, τα τελευταία χρόνια, ορισμένοι ερευνητές θεωρούν το ΠΔΟ ως πολυαντικειμενικό, πράγμα που σημαίνει ότι περισσότερες από μία λύσεις μπορούν να θεωρηθούν βέλτιστες σύμφωνα με τις τιμές των δύο αντικειμενικών συναρτήσεων. Πιο συγκεκριμένα, οι Ombuki et al. (2006) και Tan et al. (2006) ήταν οι πρώτοι που πρότειναν έναν πολυσκοπικό γενετικό και εξελικτικό αλγόριθμο, αντίστοιχα, για το ΠΔΟΤW, προσφέροντας νέες μη κυριαρχούμενες λύσεις. Αντίθετα, οι ερευνητές δεν έχουν μελετήσει την πολυαντικειμενική φύση του ΠΔΟΤΠΠ στον ίδιο βαθμό, με αυτή των Gong et al. (2018) να είναι μία από τις κύριες, καθώς πρότεινε έναν αποτελεσματικό εξελικτικό αλγόριθμο μελισσών για αυτό το πρόβλημα.

Στο τρέχον Κεφάλαιο, ένας Πολυαντικειμενικό Εξελικτικός Αλγόριθμος (ΠΕΑ) που χρησιμοποιεί τη διαδικασία ταξινόμησης χωρίς κυριαρχία που προτείνεται από τους (Deb et al. 2002), προτείνεται για την επίλυση του Προβλήματος Δρομολόγησης Οχημάτων με Ταυτόχρονες Περισυλλογές και Παραδόσεις και με Χρονικά Παράθυρα (ΠΔΟΤΠΠΧΠ), ενώ τόσο ο αριθμός των οχημάτων όσο και η συνολικά διανυόμενη απόσταση πρέπει να ελαχιστοποιηθούν. Ο ΠΕΑ ελέγχεται και στις περιπτώσεις της βιβλιογραφίας προσφέρονται ανταγωνιστικά υπολογιστικά αποτελέσματα βέλτιστα της βιβλιογραφίας. Επομένως, η συνεισφορά αυτού του Κεφαλαίου δεν περιορίζεται στην αντιμετώπιση του ΠΔΟΤΠΠΧΠ ως πολυαντικειμενικό, στην παραλλαγή του αλγορίθμου ΠΑΕΓ και στον τελεστή διασταύρωσης που είναι ενσωματωμένος στον αλγόριθμο. Η προσφορά της έρευνας επεκτείνεται στην πλήρη παρουσίαση όλων των μη κυριαρχούμενων λύσεων στα σύνολα δεδομένων που μελετήθηκαν.

Επιπλέον, ο αλγόριθμος ενσωματώνει και αντιμετωπίζει την ανάγκη των οχημάτων να ανεφοδιάζουν το ντεπόζιτό τους ή να επαναφορτίζουν τις μπαταρίες τους. Αυτό το χαρακτηριστικό σπάνια αντιμετωπίζεται σε πραγματικές περιπτώσεις, καθώς τα οχήματα βενζίνης και ντίζελ έχουν υψηλή αυτονομία, ενώ ο χρόνος που απαιτείται για τον ανεφοδιασμό είναι ελάχιστος. Η αυτονομία οδήγησης και ο χρόνος ανεφοδιασμού έχουν μεγάλο αντίκτυπο στα ηλεκτρικά οχήματα. Ως εκ τούτου, η μαθηματική διατύπωση έχει δημιουργηθεί έτσι ώστε να εξετάζεται η αυτονομία κάθε τύπου οχήματος.

Οι επόμενες ενότητες του Κεφαλαίου περιλαμβάνουν τη διατύπωση του προβλήματος, την παρουσίαση του προτεινόμενου αλγορίθμου, τα αποτελέσματα επικύρωσής του και, τέλος, τα

συμπεράσματα της έρευνας. Πιο συγκεκριμένα, η μαθηματική διατύπωση του ΠΔΟSPDTW παρουσιάζεται στην ενότητα 4.2. Η ενότητα 4.3 αναλύει τη δομή του προτεινόμενου αλγορίθμου, τη βελτιωμένη παραλλαγή του αλγορίθμου TONN, τον νέο προτεινόμενο τελεστή διασταύρωσης και τον τελεστή μετάλλαξης. Τα υπολογιστικά αποτελέσματα παρουσιάζονται και συγκρίνονται με τις καλύτερα δημοσιευμένες λύσεις σε περιπτώσεις αναφοράς στην ενότητα 4.4. Στην ενότητα 4.5 συζητούνται τα οικονομικά και περιβαλλοντικά οφέλη από τη χρήση βενζίνης, ντίζελ και ηλεκτρικών οχημάτων. Τέλος, τα συμπεράσματα συζητούνται στην ενότητα 4.6.

4.2 Περιγραφή Προβλήματος

Από τη στιγμή που έχουν καθοριστεί οι παράμετροι του προβλήματος, πρέπει να διαμορφωθεί και το μαθηματικό μοντέλο του τελικού προβλήματος. Πιο συγκεκριμένα, το πρόβλημα καθορίζεται από ένα δίκτυο $= \{F \cup N, A\}$, όπου N το σύνολο των κόμβων, $F = \{n+1, n+2, \dots n+s\}$ το σύνολο των σημείων επαναφόρτισης και τροφοδότησης καυσίμου, και $A = (i,j): i \neq j, i, j \in N$ το σύνολο των τόξων (arcs). Ο κόμβος $\{0\}$ αντιπροσωπεύει την αποθήκη, ενώ το σύνολο $N^* = \{N/0\}$ περιγράφει το σύνολο των πελατών. Κάθε τόξο (i,j), το οποίο είναι η διαδρομή από τον κόμβο i προς τον κόμβο j, χαρακτηρίζεται από την απόσταση d_{ij}, καθώς και από τον χρόνο μετακίνησης t_{ij}. Επιπλέον, κάθε πελάτης χαρακτηρίζεται από τις ποσότητες προς παράδοση, q_i, τις ποσότητες προς επιστροφή, r_i, οι οποίες φορτώνονται από τους πελάτες και επιστρέφουν στην κεντρική αποθήκη. Ακόμα, κάθε πελάτης ορίζει ένα χρονικό παράθυρο $[e_i, l_i]$, που αντιπροσωπεύει την ενωρίτερη και την αργότερη δυνατή έναρξη φόρτωσης/ εκφόρτωσης εμπορευμάτων στον πελάτη. Αντίστοιχα, χρονικό παράθυρο έχει και η κεντρική αποθήκη (η οποία συμβολίζεται με 0) είναι $[e_0, l_0]$ = [E, L], και ουσιαστικά ορίζει το διάστημα στο οποίο μπορούν να φορτώνουν και να εκφορτώνουν εμπορεύματα από και προς την αποθήκη τα φορτηγά/ οχήματα. Αντίστοιχα, χρονικά παράθυρα έχουν και τα σημεία επαναφόρτισης και τροφοδότηση καυσίμου, τα οποία χαρακτηρίζονται και από χρόνο εξυπηρέτησης.

Η έναρξη της εξυπηρέτηση του οχήματος k στον κόμβο j, συμβολίζεται με a_j^k και εξαρτάται από τον χρόνο άφιξης του οχήματος, καθώς ένα όχημα που μπορεί να φτάσει πριν την έναρξη του χρονικού παραθύρου θα πρέπει να περιμένει μέχρι να είναι εφικτή η εξυπηρέτηση (e_i). Αν το όχημα φτάσει μεταξύ e_i και l_i, η εξυπηρέτηση ξεκινάει άμεσα, ενώ κανένα όχημα δεν μπορεί να εξυπηρετήσει εάν φτάσει μετά το κλείσιμο του χρονικού παραθύρου l_i. Επιπλέον, κάθε όχημα χαρακτηρίζεται από την αυτονομία του dr^k και από ρυθμό κατανάλωσης βάσει απόστασης f^k. Η μεταβλητή απόφασης x_{ij}^k η οποία ισούται με 1 αν το όχημα k εκτελεί τη διαδρομή από τον κόμβο i στον κόμβο j, και 0 σε διαφορετική περίπτωση, και συνεχίζοντας με την y_{ij}^k η οποία αφορά την συνολική ποσότητα εντός του οχήματα κατά την μετακίνηση από τον κόμβο i στον κόμβο j. Η Τρίτη μεταβλητή απόφασης είναι η g_i^k , η οποία δίνει το εναπομένον επίπεδο καυσίμου όταν φτάνει στον πελάτη i, σύμφωνα με τον τύπο του οχήματος. Στόχος του ΠΔΟΤΠΠΧΠ είναι η εξυπηρέτηση όλων των πελατών στο N^*, καθώς ελαχιστοποιείται ταυτόχρονα η συνολικά διανυόμενη απόσταση και ο αριθμός των οχημάτων.

4.3 Μέθοδος Επίλυσης – Εξελικτικός Αλγόριθμος

Για την αντιμετώπιση του ΠΔΟΤΠΠΧΠ που παρουσιάζεται παραπάνω, αναπτύχθηκε ένας αποτελεσματικός πολυαντικειμενικός εξελικτικός αλγόριθμος. Πιο συγκεκριμένα, η ενότητα 4.3.1 συζητά τη σημασία της πολυαντικειμενικής βελτιστοποίησης και την έννοια της βελτιστοποίησης Pareto. Η ενότητα 4.3.2 παρουσιάζει τη βελτιωμένη έκδοση του αλγορίθμου πλησιέστερου γείτονα με προσανατολισμό στον χρόνο (ΠΓΠΧ) που χρησιμοποιείται για τη δημιουργία του αρχικού πληθυσμού λύσεων. Η ενότητα 4.3.3 περιγράφει τη δομή του χρωμοσώματος που εφαρμόζεται στο ΠΕΑ. Στην ενότητα 4.3.4, περιγράφεται διεξοδικά η μεθοδολογική διαδικασία του ΠΕΑ. Τέλος, ο νέος προτεινόμενος τελεστής διασταύρωσης και ο τελεστής μετάλλαξης περιγράφονται στην ενότητα 4.3.5.

4.3.1 Πολυαντικειμενική Βελτιστοποίηση

Ένα σύνολο βέλτιστων λύσεων μπορεί να ληφθεί σε προβλήματα βελτιστοποίησης πολλαπλών στόχων, αντί για μία μόνο λύση. Οι λύσεις που ανήκουν σε αυτό το σύνολο δεν κυριαρχούν μεταξύ τους. Το κύριο χαρακτηριστικό των λύσεων που δεν κυριαρχούν είναι ότι ενώ μετακινούνται από τη μια λύση στην άλλη, υπάρχει πάντα ένα ορισμένο ποσό θυσίας σε έναν στόχο για να επιτευχθεί ένα συγκεκριμένο ποσό κέρδους σε έναν άλλο στόχο.

Στην περίπτωση του ΠΔΟ, οι διαφορετικές λύσεις που λαμβάνονται μεταφράζονται σε διαφορετικά σχέδια δρομολογίων και παραδόσεων. Αυτά τα διαφορετικά σχέδια εξυπηρετούν τα συμφέροντα πολλών ενδιαφερομένων που επιλέγουν το πιο κερδοφόρο. Η επιλογή βασίζεται σε περαιτέρω πληροφορίες και στη φύση της κάθε εταιρείας. Ως εκ τούτου, είναι κρίσιμο να θεωρηθεί το ΠΔΟΤΠΠΧΠ ως πολυαντικειμενικό, όπου οι δύο στόχοι είναι ο αριθμός των οχημάτων και η συνολική απόσταση που διανύθηκε για την κάλυψη των αναγκών πολλών ενδιαφερομένων. Μέσω αυτής της προσέγγισης και σε αναζήτηση ενός οικονομικά αποδοτικού σχεδίου, μπορεί να προκύψουν περισσότερες από μία λύσεις. Επομένως, εναπόκειται στη διακριτική ευχέρεια του υπεύθυνου λήψης αποφάσεων να επιλέξει το πιο κερδοφόρο σχέδιο.

4.3.2 Κατασκευαστική Διαδικασία

Ο αρχικός πληθυσμός λύσεων συμβάλλει σημαντικά στην αποτελεσματικότητα του προτεινόμενου ΠΕΑ. Επομένως, απαιτείται ένας αποτελεσματικός αλγόριθμος, από άποψη χρόνου και ποιότητας λύσης. Το γεγονός ότι ο αλγόριθμος κατασκευής ΠΓΠΧ πληροί αυτές τις απαιτήσεις είναι ο κύριος λόγος που επιλέγεται για τη διαδικασία κατασκευής. Ο συγκεκριμένος κατασκευαστικός αλγόριθμος είναι ίδιος με αυτόν που παρουσιάζεται στην ενότητα 3.3.2 έχοντας μόνο μία διαφορά. Η διαφορά αυτή βασίζεται στο γεγονός ότι ο χρόνος, η απόσταση και η χωρητικότητα δεν είναι στην ίδια κλίμακα, επηρεάζοντας τις κατασκευασμένες λύσεις. Επομένως, εφαρμόστηκε η διαδικασία της κανονικοποίησης στα σύνολα $(t_{ij}, d_{ij}, e_i, l_i, s_i, q_i)$. Αυτή η κανονικοποίησης παρέχει γραμμικό μετασχηματισμό και προσαρμόζει δεδομένα σε ένα προκαθορισμένο όριο, το οποίο είναι το [0, 1]. Αυτή η

διαδικασία δεδομένων προεπεξεργασίας βελτιώνει την ακρίβεια και επιτυγχάνει καλύτερη απόδοση στο δοκιμασμένο σύνολο δεδομένων.

4.3.3 Δομή Χρωμοσώματος

Η δομή του χρωμοσώματος επηρεάζει σημαντικά τον αλγόριθμο όσον αφορά την αποτελεσματικότητα και την πολυπλοκότητα του χρόνου, καθώς εμπλέκεται στους τελεστές διασταύρωσης και μετάλλαξης. Επομένως, αποφασίστηκε να αναπαρασταθεί το χρωμόσωμα ως πίνακας με μήκος ίσο με τον συνολικό αριθμό πελατών. Κάθε αντικείμενο του χρωμοσώματος είναι ένα γονίδιο και αντιπροσωπεύεται από έναν ακέραιο που αναφέρεται στον κόμβο (id) του πελάτη. Η αλληλουχία των γονιδίων καθορίζει τη σειρά επίσκεψης των πελατών. Όταν μια ακολουθία δεν μπορεί να συνεχιστεί λόγω παραβίασης των περιορισμών χωρητικότητας ή χρονικού παραθύρου, ξεκινά μια νέα διαδρομή. Στην περίπτωση οχημάτων που έχουν περιορισμένη αυτονομία, ο αλγόριθμος λαμβάνει επίσης υπόψη την ανάγκη είτε της επαναφόρτισης πριν αποφορτιστεί η μπαταρία (ηλεκτρικά οχήματα), είτε του ανεφοδιασμού του ντεπόζιτο (συμβατικά οχήματα). Επομένως, μεταξύ δύο διαδοχικών γονιδίων, μπορεί να παρέμβει ένα σημείο φόρτιση ή ανεφοδιασμού καυσίμου (ΣΦΑΚ).

4.3.4 Προτεινόμενος Εξελικτικός Αλγόριθμος

Αφού ο ΠΓΠΧ δημιουργήσει τον αρχικό πληθυσμό (R_0) μεγέθους N, ο ΠΕΑ εφαρμόζεται για να εξελίξει τις λύσεις και να αποκτήσει το σύνολο Pareto. Πιο συγκεκριμένα, σε κάθε επανάληψη t, λαμβάνονται υπόψιν των σύνολο γονέων R_t, ο αλγόριθμος εφαρμόζει τις κινήσεις; διασταύρωσης σε όλους τους N γονείς και παράγουν N παιδιά τα οποία αποθηκεύονται στο Q_t (πληθυσμός παιδιών). Στη συνέχεια, τα χρωμοσώματα που ανήκουν στο σύνολο $R_t \cup Q_t$ ταξινομούνται σύμφωνα με τη μη κυριαρχούμενη ταξινόμηση και δημιουργούν σύνολα Pareto $F_1, F_2, \dots F_K$. Αυτή η διαδικασία χρησιμοποιείται για ταξινόμηση των λύσεων σε σύνολα $(F_1, F_2, \dots F_K)$σύμφωνα με τον κανόνα του Pareto. Αυτό σημαίνει ότι κάθε σύνολο F_i περιέχει λύσεις, όπου F_1 οι καλύτερες μη κυριαρχούμενες λύσεις. Αντίστοιχα, το σύνολο F_2 περιέχει λύσεις που κυριαρχούνται μόνο από τις λύσεις του συνόλου F_1. Το σύνολό R_{t+1}, που περιέχει τους γονείς της επόμενης επανάληψης γεμίζεται με λύσεις των συνόλων F_1, F_2, κλπ, μέχρις ότου N χρωμοσώματα γεμίσουν το σύνολο R_{t+1}. Μέσω αυτής της διαδικασίας, οι καλύτερες λύσεις σε κάθε επανάληψη διατηρούνται και χρησιμοποιούνται για την εξέλιξη των λύσεων. Η περιγραφόμενη διαδικασία ακολουθείται για συγκεκριμένο αριθμό επαναλήψεων, που είναι ο μέγιστος αριθμός γενεών.

4.3.5 Τελεστές Διασταύρωσης και Μετάλλαξης

Πολλές κινήσεις διασταύρωσης έχουν αναπτυχθεί με τα χρόνια για το ΠΔΟ. Ο προτεινόμενος και εφαρμοσμένος τελεστής διασταύρωσης ονομάζεται Διαδοχική Διασταύρωση (ΔΔ). Στην ΔΔ από το σύνολο P επιλέγονται δύο γονείς (P_i, P_{i+1}) και από τον καθένα επιλέγεται ένας τυχαίος αριθμός διαδοχικών γονιδίων από κάθε χρωμόσωμα. Στη συνέχεια, οι πελάτες που έχουν επιλεγεί αφαιρούνται από τον αντίθετο γονέα. Δεδομένου ότι όλοι οι πελάτες πρέπει να

εξυπηρετούνται, πρέπει επίσης να περιέχονται στο χρωμόσωμα. Η εισαγωγή πελάτη γίνεται στη θέση του χρωμοσώματος που ελαχιστοποιεί το κόστος. Μετά τη διαδικασία εισαγωγής, δημιουργούνται δύο νέα χρωμοσώματα, γνωστά ως απόγονοι.

Ο τελεστής μετάλλαξης που ακολουθείται είναι ένας τελεστής κόμβου ανταλλαγής. Σε αυτή την περίπτωση, δύο πελάτες (γονίδια) του χρωμοσώματος επιλέγονται τυχαία και ανταλλάσσονται. Η κύρια συμβολή αυτής της διαδικασίας είναι η διαφοροποίηση των χρωμοσωμάτων, η οποία μπορεί να βοηθήσει στη διαφυγή από το τοπικό βέλτιστο. Τέλος, τόσο η μετάλλαξη όσο και ο τελεστής διασταύρωσης δεν εφαρμόζονται σε κάθε γενιά και σε κάθε σύνολο γονέων. Αντίθετα, κάθε κίνηση έχει μια συγκεκριμένη πιθανότητα να εφαρμοστεί.

4.4 Υπολογιστικά Αποτελέσματα

Ο προτεινόμενος αλγόριθμος δεν μπορεί να δοκιμαστεί ταυτόχρονα στο ΠΔΟΧΠ και το ΠΔΟΤΠΠ λόγω της έλλειψης ενός συνόλου δεδομένων που περιέχει και τα δύο χαρακτηριστικά. Επομένως, ο αλγόριθμος ελέγχεται σε δύο διαφορετικά σύνολα δεδομένων, του Solomon, για το ΠΔΟΧΠ και του Salhi και του Nagy, για το ΠΔΟΤΠΠ. Ο προτεινόμενος αλγόριθμος αναπτύχθηκε σε Python και έτρεχε σε υπολογιστή με τις ακόλουθες προδιαγραφές: Intel Core i7 - 8550U 1,8 GHz με μνήμη 16 GB.

Τέλος, οι παράμετροι του εξελικτικού αλγορίθμου που χρησιμοποιούνται στον προτεινόμενο αλγόριθμο είναι οι ίδιες με αυτές που παρουσιάζονται στο Ombuki et al. (2006) ώστε τα αποτελέσματα να είναι αντιπροσωπευτικά και συγκρίσιμα.

- Μέγεθος πληθυσμού = 300
- Αριθμός γενεών = 350
- Πιθανότητα διασταύρωσης = 0,80
- Πιθανότητα μετάλλαξης = 0,10

Μια άλλη παράμετρος που ορίζεται είναι ο μέγιστος υπολογιστικός χρόνος. Στην έρευνα, το χρονικό όριο ορίζεται στα 15 λεπτά που θεωρείται εύλογος υπολογιστικός χρόνος για 100 πελάτες.

4.4.1 Αποτελέσματα στο ΠΔΟΧΠ

Τα αποτελέσματα του ΠΕΑ θεωρούνται ανταγωνιστικά σε σύγκριση με τα καλύτερα δημοσιευμένα στη βιβλιογραφία. Η μέση απόκλιση μεταξύ των αποτελεσμάτων που προκύπτουν από τον αλγόριθμο και των βέλτιστων της βιβλιογραφίας είναι 2,64%, κριτήριο που αποδεικνύει την αποτελεσματικότητα του αλγορίθμου. Ωστόσο, το ΠΕΑ φαίνεται να έχει καλύτερη αποτελεσματικότητα με γεωγραφικά ομαδοποιημένα δεδομένα πελατών (σύνολα δεδομένων C), ανεξάρτητα από τον ορίζοντα προγραμματισμού και το εύρος των χρονικών παραθύρων, καθώς η απόκλιση των αποτελεσμάτων από τα βέλτιστα της βιβλιογραφίας κυμαίνεται από 0,00% έως 0,73%.

Ο ΠΕΑ, ο οποίος ενσωματώνει τον προτεινόμενο χειριστή crossover, μπορεί να αναζητήσει έναν τεράστιο χώρο λύσεων για να βρει λύσεις που δεν κυριαρχούν. Ως αποτέλεσμα, δύο νέες λύσεις χωρίς κυριαρχία προτείνονται στις περιπτώσεις R202 και R210 του Solomon. Στην περίπτωση R202, επιτυγχάνεται μείωση της συνολικά διανυόμενης απόστασης, ενώ στην περίπτωση R210, δύο λύσεις χωρίς κυριαρχία ενημερώνονται. Πιο συγκεκριμένα, στην περίπτωση R210, η προτεινόμενη λύση (NV=4, TD=920,13) βελτιώνει τόσο τη λύση (NV=4, TD=924,79), καθώς η απόσταση μειώνεται, όσο και η λύση (NV=5, TD=922,30), αφού τόσο ο αριθμός των οχημάτων όσο και η συνολικά διανυόμενη απόσταση μειώνονται.

4.4.2 Αποτελέσματα στο ΠΔΟΤΠΠ

Το ΠΔΟΤΠΠ μελετάται λιγότερο συχνά από το ΠΔΟΧΠ. Ο προτεινόμενος ΠΕΑ, ο οποίος είναι σε θέση να αντιμετωπίσει το ΠΔΟΤΠΠΧΠ, δοκιμάζεται στο σύνολο δεδομένων των Salhi και Nagy και τα αποτελέσματα που προκύπτουν από τον προτεινόμενο αλγόριθμο συγκρίνονται με τα καλύτερα δημοσιευμένα. Όπως και στην περίπτωση των χρονικών παραθύρων, ο αλγόριθμος καταφέρνει να παράγει νέες βέλτιστες λύσεις Pareto στις περιπτώσεις CMT2X, CMT6X, CMT7X, CMT8X, CMT9X, CMT10X. Σε κάθε μία από αυτές τις 6 περιπτώσεις, ο αριθμός των οχημάτων μειώνεται σε βάρος της αυξημένης συνολικής απόστασης που διανύθηκε. Επιπλέον, η μέση απόκλιση μεταξύ των καλύτερων δημοσιευμένων αποτελεσμάτων και των καλύτερων αποτελεσμάτων από τον ΠΕΑ όσον αφορά την συνολικά διανυόμενη απόσταση είναι 8,50%.

4.5 Συμβατικά έναντι Ηλεκτρικών Οχημάτων

Ενώ η διανομή αγαθών έχει προσελκύσει το ενδιαφέρον των ερευνητών για περισσότερα από 60 χρόνια, είναι μόνο τα τελευταία χρόνια που οι εκπομπές αερίων του θερμοκηπίου εξετάζονται στο ΠΔΟ, λόγω της οικολογικής συνείδησης της βιωσιμότητας των μεταφορών, που οδηγεί σε πράσινη εφοδιαστική. Στην περίπτωση του Ηλεκτρικού ΠΔΟ (ΗΠΔΟ), τα οχήματα που χρησιμοποιούνται για τη διαδικασία διανομής διαθέτουν ηλεκτρικές μπαταρίες. Αυτό συνεπάγεται περιορισμένη εμβέλεια οδήγησης και επισκέψεις στους σταθμούς φόρτισης (ΣΦΑΚ) για την επαναφόρτιση των μπαταριών, όταν χρειάζεται, για την εκτέλεση των διαδρομών (Erdelić et al. 2019). Ωστόσο, το ΗΠΔΟ μπορεί να είναι πολύ πιο δύσκολο και πολύπλοκο εάν η κατανάλωση της μπαταρίας και οι εκπομπές CO_2 σχετίζονται με το φορτίο και την ταχύτητα του οχήματος. Οι (Erdelić and Carić 2019) παρουσιάζουν μια έρευνα για το ΗΠΔΟ, εστιάζοντας στα μοντέλα κατανάλωσης και σε πρόσθετες παραλλαγές ΗΠΔΟ που προέκυψαν, όπως υβριδικά οχήματα, σταθμοί φόρτισης, λειτουργίες φόρτισης και δυναμικές συνθήκες κυκλοφορίας. Στην τρέχουσα έρευνα, υποτίθεται ότι η κατανάλωση και οι εκπομπές CO_2 είναι σταθερές, ανεξάρτητα από το φορτίο και την ταχύτητα του οχήματος.

Επιπλέον, οι περιπτώσεις διανομής που μελετώνται στην παρούσα υποενότητα, αντιμετωπίζονται από εταιρεία logistics στην πόλη της Αθήνας. Ωστόσο, τα δεδομένα και άλλα χαρακτηριστικά προσαρμόζονται στις απαιτήσεις της υπό μελέτη περίπτωσης. Ως εκ τούτου,

τα οχήματα θεωρούνται πανομοιότυπα σε όλες τις περιπτώσεις διανομής, ενώ στην πράξη, η εταιρεία διαθέτει διαφορετικούς τύπους οχημάτων. Λαμβάνοντας υπόψη τα πανομοιότυπα οχήματα, τρεις περιπτώσεις διακρίνονται ανάλογα με τον τύπο καυσίμου, (i) Βενζίνη, (ii) Ντίζελ και (iii) Ηλεκτρικά οχήματα. Ωστόσο, για να εξαχθούν αξιόπιστα συμπεράσματα από την έρευνα, τα οχήματα είναι της ίδιας μάρκας (Nissan) και του ίδιου μοντέλου (NV200). Κατά συνέπεια, οι περιπτώσεις διανομής ελέγχονται για τους τρεις διαφορετικούς τύπους οχημάτων. Σε κάθε περίπτωση, ο στόλος των οχημάτων αποτελείται μόνο από τον ίδιο τύπο οχημάτων. Επιπλέον, υπολογίστηκαν τα πάγια έξοδα, λαμβάνοντας υπόψη τον χρόνο απόσβεσης (10 έτη) και την ετήσια χρήση του οχήματος (250 εργάσιμες ημέρες ετησίως). Το αυξημένο πάγιο κόστος των ηλεκτρικών οχημάτων είναι τουλάχιστον αναμενόμενο αφού η τιμή αγοράς είναι υψηλότερη από τον άλλο τύπο οχημάτων.

Από την άλλη πλευρά, το μεταβλητό κόστος είναι αντιστρόφως ανάλογο με το πάγιο κόστος, αφού εξαρτάται από την αξία του καυσίμου. Επιπλέον, η αυτονομία οδήγησης στα ηλεκτρικά οχήματα είναι σημαντικά χαμηλότερη, γεγονός που οδηγεί στην ανάγκη επαναφόρτισης της μπαταρίας, η οποία είναι επίσης χρονοβόρα (45 λεπτά για να επαναφορτιστεί πλήρως η άδεια μπαταρία).

Τα υπολογιστικά αποτελέσματα της έρευνας, που περιέχουν τον αριθμό των οχημάτων που απαιτούνται (αριθμός διαδρομών) για την εκτέλεση του σχεδίου διαδρομών, τη συνολική απόσταση, τις εκπομπές CO_2 και το κόστος (μεταβλητό, σταθερό και συνολικό), υπολογίζονται για κάθε τύπο οχήματος. Από τα αποτελέσματα παρατηρείται ότι τα βενζινοκίνητα οχήματα έχουν τις υψηλότερες εκπομπές CO_2. Τα πετρελαιοκίνητα οχήματα έχουν επίσης αυξημένες εκπομπές ρύπων, αλλά κερδίζουν σημαντικό ενδιαφέρον λόγω του χαμηλού κόστους διανομής τους. Όσον αφορά τα ηλεκτρικά οχήματα, επιτυγχάνονται μηδενικές εκπομπές ρύπων, αλλά σε βάρος του υψηλού κόστους διανομής σε σύγκριση με άλλες περιπτώσεις οχημάτων. Ο κύριος παράγοντας που επηρεάζει τις παραδόσεις με ηλεκτρικά οχήματα, που οδηγεί σε αυξημένο κόστος, είναι ότι χρειάζονται περισσότερα οχήματα για την εκτέλεση των παραδόσεων σε σύγκριση με τους άλλους τύπους οχημάτων σε κάθε περίπτωση διανομής. Αυτό οφείλεται στην περιορισμένη χωρητικότητα των ηλεκτρικών οχημάτων, καθώς οι μπαταρίες έχουν αυξημένο βάρος και όγκο, γεγονός που τους εμποδίζει να έχουν την ίδια χωρητικότητα με άλλους τύπους οχημάτων.

Η μείωση των εκπομπών και η αύξηση του κόστους στην περίπτωση των ηλεκτρικών οχημάτων μπορεί να είναι παραπλανητική. Και αυτό γιατί η παραγωγή ηλεκτρικής ενέργειας στην Ελλάδα βασίζεται, σε μεγάλο βαθμό, στη χρήση καυσίμων β' κατηγορίας όπως ο λιγνίτης και λιγότερο σε ανανεώσιμες πηγές. Κατά συνέπεια, τα ηλεκτρικά οχήματα μπορεί να έχουν μηδενικές εκπομπές, ωστόσο η παραγωγή ηλεκτρικής ενέργειας έχει αυξήσει τις εκπομπές που μπορούν να ελαχιστοποιηθούν μόνο με τη χρήση ανανεώσιμων πηγών ενέργειας. Επιπλέον, είναι σαφές ότι το κόστος απόκτησης ηλεκτρικών οχημάτων είναι υψηλότερο σε σύγκριση με τη βενζίνη και το ντίζελ, όπως συνέβη με όλες τις νέες τεχνολογίες. Ωστόσο, με την πάροδο

των ετών, το κόστος κτήσης αναμένεται να μειωθεί και να είναι πιο αποδοτικό από πλευράς κόστους για τη διαδικασία διανομής.

4.6 Συμπεράσματα

Συνοψίζοντας, σε αυτό το Κεφάλαιο, εξετάστηκε το πρόβλημα δρομολόγησης οχημάτων με ταυτόχρονες παραλαβές και παραδόσεις και χρονικά παράθυρα, με στόχο την εύρεση μιας αποτελεσματικής λύσης στο πρόβλημα προτείνοντας έναν εξελικτικό αλγόριθμο που λύνει το πολυαντικειμενικό ΠΔΟΤΠΠΧΠ. Το πρώτο βήμα για την επίλυση αυτού του προβλήματος ήταν η εφαρμογή ενός βελτιωμένου κατασκευαστικού αλγόριθμου ΠΓΠΧ για τη δημιουργία του αρχικού πληθυσμού λύσεων. Κατά συνέπεια, αυτές οι λύσεις ενσωματώθηκαν στον προτεινόμενο ΠΕΑ προκειμένου να βρεθεί το καλύτερο σχέδιο διαδρομών που ελαχιστοποιεί τις δύο αντικειμενικές συναρτήσεις: τον αριθμό των οχημάτων και τη συνολική απόσταση που διανύθηκε. Εκτός από την αντιμετώπιση του ΠΔΟΤΠΠΧΠ ως πολυαντικειμενικό, μια άλλη συμβολή του Κεφαλαίου αποδείχθηκε ότι είναι ο τελεστής διαστάυρωσης, ο οποίος επηρεάζει σημαντικά τα αποτελέσματα που λαμβάνονται από το ΠΑΕ.

Η αποτελεσματικότητα του αλγορίθμου επικυρώθηκε χρησιμοποιώντας το σύνολο δεδομένων του Solomon για το ΠΔΟΧΠ και το σύνολο δεδομένων των Salhi και Nagy (1999) για το ΠΔΟΤΠΠ, και τα αποτελέσματα που προέκυψαν συγκρίθηκαν με τα καλύτερα δημοσιευμένα αποτελέσματα, υποδεικνύοντας την υψηλή απόδοση του αλγορίθμου.

Ο αλγόριθμος δοκιμάστηκε σε διάφορες περιπτώσεις όπου ο στόλος αποτελείται από διαφορετικούς τύπους οχημάτων (βενζίνης, ντίζελ και ηλεκτρικού) ώστε να εκτιμηθεί ο οικονομικός και περιβαλλοντικός αντίκτυπος. Μεταξύ των ευρημάτων της έρευνας είναι ότι σε περιπτώσεις διανομής με αγαθά χαμηλού όγκου οι περιορισμοί των ηλεκτρικών οχημάτων (εμβέλεια οδήγησης) δεν επηρεάζουν πολύ το κόστος. Ωστόσο, καθώς αυξάνεται ο όγκος των εμπορευμάτων, απαιτούνται περισσότερα ηλεκτρικά οχήματα σε σύγκριση με άλλους τύπους οχημάτων, γεγονός που οδηγεί σε αυξημένο κόστος διανομής. Αυτό οφείλεται στη μειωμένη χωρητικότητα των ηλεκτρικών οχημάτων. Ανεξάρτητα από το κόστος, οι εκπομπές CO_2 εξαλείφονται με τη χρήση ηλεκτρικών οχημάτων.

5. Πρόβλημα της Δρομολόγησης Οχημάτων με Ετερογενή Στόλο Οχημάτων, Ταυτόχρονες Περισυλλογές και Παραδόσεις και με Χρονικά Παράθυρα

5.1 Εισαγωγή

Στο τρέχον Κεφάλαιο, αναπτύσσεται ένας αλγόριθμος για την ενίσχυση των προσπαθειών των εταιρειών που δραστηριοποιούνται στον τομέα της διανομής εμπορευμάτων σε αστικές περιοχές. Σύμφωνα με την έρευνα που παρουσιάστηκε στο Κεφάλαιο 2, καθώς και με την έρευνα των Elshaer και Awad (2020) οι παραλλαγές του ΠΔΟ που αντιμετωπίζονται συχνότερα είναι (i) η ΠΔΟΧΠ, (ii) το ΠΔΟΕΣΟ και (iii) το ΠΔΟΤΠΠ. Οι παραλλαγές αυτές επομένως αντιμετωπίζονται και από τον προτεινόμενο αλγόριθμο. Οι προαναφερθείσες παράμετροι και περιορισμοί, όταν ληφθούν υπόψη ταυτόχρονα, καλύπτουν τις ανάγκες των περισσότερων εταιρειών διανομής και καλύπτουν το ΠΔΟ με Ετερογεννή Στόλο Οχημάτων, Ταυτόχρονες Περισυλλογές και Παραδόσεις και με Χρονικά Παράθυρα (ΠΔΟΕΣΟΤΠΠΧΠ).

Επιπλέον, οι βάρδιες των οδηγών αποτελούν μια επιπλέον σημαντική παράμετρο τόσο στο τρέχον Κεφάλαιο όσο και στις πραγματικές περιπτώσεις, καθώς οι οδηγοί έχουν συγκεκριμένες ώρες εργασίας. Αυτός ο περιορισμός έχει κερδίσει έδαφος τα τελευταία χρόνια, (αυστηρότερη εργατική νομοθεσία), χωρίς ωστόσο να έχει προσελκύσει την κατάλληλη προσοχή από τους ερευνητές που μελετούν το ΠΔΟ. Στην τρέχουσα έρευνα, και για να απλοποιηθεί αυτός ο περιορισμός, κάθε βάρδια έχει μετατραπεί στον ενωρίτερο χρόνο εκκίνησης και τον τελευταίο χρόνο επιστροφής του οχήματος.

Για την αντιμετώπιση αυτού του σύνθετου προβλήματος προτείνεται ένας μεταευρετικός αλγόριθμος Προσαρμοστικής Αναζήτησης Ευρείας Γειτονιάς (ΠΡΑΕΓ) ο οποίος μέσω τελεστών καταστροφής και επισκευής επιδιώκει στην ελαχιστοποίηση του συνολικού κόστους διανομής. Επιπλέον, ο αλγόριθμος ΠΡΑΕΓ εφαρμόζεται σε μια περίπτωση του Συνεργατικού ΠΔΟ, όπου τρεις εταιρείες 3PL που εδρεύουν και διανέμουν αγαθά στην ίδια περιοχή,. Αυτή η προσέγγιση μπορεί να φαίνεται δύσκολο να εφαρμοστεί στην Ελλάδα λόγω του εδάφους που δεν είναι ακόμη πρόσφορο για τέτοιες συνέργειες. Ωστόσο, τα οικονομικά και περιβαλλοντικά οφέλη αυτής της θεωρητικής προσέγγισης είναι αρκετά για να πείσουν τις εταιρείες να κινηθούν προς αυτή την κατεύθυνση στο μέλλον.

Συνοψίζοντας, η έρευνα που διεξάγεται σε αυτό το Κεφάλαιο έρχεται να παρουσιάσει έναν μεταευρετικό αλγόριθμο που είναι αποθηκευμένος σε μια βάση δεδομένων έτσι ώστε να είναι προσβάσιμος από ένα cloud σύστημα δρομολόγησης, το οποίο παρουσιάζεται στο Κεφάλαιο 6. Προκειμένου να ενισχυθεί αυτή η προσπάθεια και να παρασχεθούν τα επιθυμητά αποτελέσματα, οι μεταβλητές και οι περιορισμοί που εμπλέκονται στη διανομή εμπορευμάτων και η μαθηματική διατύπωση ορίζονται στη συνέχεια στην ενότητα 5.2. Στην ενότητα 5.3, παρουσιάζεται η αλγοριθμική προσέγγιση, η οποία περιλαμβάνει τον αλγόριθμο κατασκευής και τον μεταευρετικό ΠΡΑΕΓ. Τα υπολογιστικά αποτελέσματα που λαμβάνονται από τον

αλγόριθμο, όταν ελέγχονται σε σύνολα δεδομένων της βιβλιογραφίας για το ΗΠΔΟSPDTW, παρουσιάζονται στην ενότητα 5.4. Η ενότητα 5.5 περιγράφει το Συνεργατικό ΠΔΟ και πώς η εφαρμογή του αλγόριθμου ALNS μπορεί να συμβάλει οικονομικά και περιβαλλοντικά στις συνεργαζόμενες εταιρείες. Τέλος, οι καταληκτικές παρατηρήσεις του Κεφαλαίου παρουσιάζονται στην ενότητα 5.6.

5.2 Περιγραφή Προβλήματος

Από τη στιγμή που έχουν καθοριστεί οι παράμετροι του προβλήματος, πρέπει να διαμορφωθεί και το μαθηματικό μοντέλο του τελικού προβλήματος. Πιο συγκεκριμένα, το πρόβλημα καθορίζεται από ένα δίκτυο G (N, A), όπου N το σύνολο των κόμβων και and $A = (i,j): i \neq j, i,j \in N$ το σύνολο των τόξων (arcs). Ο κόμβος $\{0\}$ αντιπροσωπεύει την αποθήκη, ενώ το σύνολο $N^* = \{N/0\}$ περιγράφει το σύνολο των πελατών. Κάθε τόξο (i,j), το οποίο είναι η διαδρομή από τον κόμβο i προς τον κόμβο j, χαρακτηρίζεται από την απόσταση d_{ij}, καθώς και από τον χρόνο μετακίνησης t_{ij}. Επιπλέον, κάθε πελάτης χαρακτηρίζεται από τις ποσότητες προς παράδοση, q_i, οι οποίες ξεκινούν από την αποθήκη και καταλήγουν στους πελάτες, καθώς και από τις ποσότητες προς επιστροφή, r_i, οι οποίες φορτώνονται από τους πελάτες και επιστρέφουν στην κεντρική αποθήκη. Ακόμα, κάθε πελάτης ορίζει ένα χρονικό παράθυρο $[e_i, l_i]$, που αντιπροσωπεύει την ενωρίτερη και την αργότερη δυνατή έναρξη φόρτωσης/ εκφόρτωσης εμπορευμάτων στον πελάτη. Αντίστοιχα, χρονικό παράθυρο έχει και η κεντρική αποθήκη (η οποία συμβολίζεται με 0) είναι $[e_0, l_0]$ = [E, L], και ουσιαστικά ορίζει το διάστημα στο οποίο μπορούν να φορτώνουν και να εκφορτώνουν εμπορεύματα από και προς την αποθήκη τα φορτηγά/ οχήματα.

Ακόμα, το πρόβλημα που μελετάται, χαρακτηρίζεται από ένα σύνολο ετερογενών οχημάτων που συμβολίζεται με K, ενώ κάθε όχημα συμβολίζεται με k (όπου k∈K). Κάθε όχημα k αντιστοιχεί σε ένα μοναδικό τύπο οχήματος c∈C. Επομένως, προκειμένου να «συσχετίσουμε» μαθηματικά τα δύο σύνολα, θέτουμε το σύνολο S_c το οποίο αναπαριστά το σύνολο των οχημάτων τύπου c. Επιπλέον, τα οχήματα που ανήκουν σε κάθε τύπο χαρακτηρίζονται από την χωρητικότητα Qc, το σταθερό κόστος f_c, το μεταβλητό κόστος v_c, καθώς και τον αριθμό των διαθέσιμων οχημάτων κάθε τύπου, n_c. Όσον αφορά τους οδηγούς, αντιστοιχίζεται ένας οδηγός σε κάθε όχημα και επομένως η έναρξη της βάρδιας κάθε οδηγού, p_k, και η λήξη της βάρδιας κάθε οδηγού, m_k, αντιστοιχίζεται και «μεταφέρεται» σε κάθε όχημα k ξεχωριστά.

Τέλος, το συγκεκριμένο πρόβλημα απαιτεί κάποιες επιπλέον μεταβλητές προκειμένου η μαθηματική μοντελοποίηση να είναι πλήρης. Ξεκινώντας με τη μεταβλητή απόφασης x_{ij}^k η οποία ισούται με 1 αν το όχημα k εκτελεί τη διαδρομή από τον κόμβο i στον κόμβο j, και 0 σε διαφορετική περίπτωση, και συνεχίζοντας με την y_{ij}^k η οποία αφορά την συνολική ποσότητα εντός του οχήματα κατά την μετακίνηση από τον κόμβο i στον κόμβο j. Επιπλέον, οι χρονικές μεταβλητές περιλαμβάνουν i) την a_i^k η οποία αναφέρεται στην ώρα άφιξης του οχήματος k στον κόμβο i, ii) την w_i^k η οποία αναφέρεται στον χρόνο αναμονής του οχήματος k στον κόμβο i, και iii) την b_i^k η οποία αναφέρεται στην ώρα αναχώρησης του οχήματος k από τον κόμβο i.

5.3 Μέθοδος Επίλυσης – Προσαρμοστική Αναζήτηση Ευρείας Γειτονιάς

Ο προτεινόμενος ΠΡΑΕΓ ενσωματώνεται σε ένα σύστημα δρομολόγησης και ανήκει στην κατηγορία των μεταευρετικών τοπικής αναζήτησης. The algorithm is applied in a solution that is exploited by a construction heuristic algorithm that is analyzed in Section 5.3.1. Consequently, the individual pieces of the ALNS algorithm are presented in Section 5.3.2.

5.3.1 Κατασκευαστική Διαδικασία

Ο κατασκευαστικός ευρετικός αλγόριθμος που εφαρμόζεται για την παραγωγή της αρχικής λύσης είναι ο ίδιος με αυτόν που παρουσιάζεται στο Κεφάλαιο 4. Αφορά τον ΠΓΠΧ ο οποίος κανονικοποιεί όλα τα δεδομένα προτού εφαρμοστεί.

5.3.2 Γενικό Πλαίσιο Αλγορίθμου

Ο αλγόριθμος ΑΕΓ εξερευνά μια μεγάλη γειτονιά καταστρέφοντας και επισκευάζοντας λύσεις Shaw (1998). Ο αλγόριθμος ΠΡΑΕΓ είναι μια επέκταση που λειτουργεί επίσης με διάφορους τελεστές καταστροφής και επισκευής. Και οι δύο διαδικασίες χρειάζονται μια αρχική λύση, όπως αυτή που παράγεται από τον αλγόριθμο TONN, για να εφαρμοστούν και να λειτουργήσουν. Ωστόσο, η κύρια διαφορά τους παρατηρείται στην πιθανότητα να εφαρμοστεί ο κάθε τελεστής. Στο LNS, η πιθανότητα να εφαρμοστεί κάθε χειριστής καταστροφής (π^d) και κάθε χειριστή επισκευής (π^r) είναι σταθερή και ορίζεται στην αρχή της διαδικασίας. Αντίθετα, στον αλγόριθμο ALNS, οι πιθανότητες κάθε χειριστή καταστροφής και επισκευής (π^d και π^r αντίστοιχα) να επιλεγεί αυξάνονται όσο περισσότερο ο χειριστής έχει συμβάλει στη βελτίωση της λύσης. Σε κάθε επανάληψη του αλγορίθμου, μια νέα λύση κατασκευάζεται (X') μετά τον τελεστή καταστροφής και επισκευής, που συγκρίνεται με τη βέλτιστη μέχρι τώρα λύση (X^*). Εάν η κατασκευασμένη λύση X' είναι καλύτερη από τη βέλτιστη X^*, τότε η βέλτιστη λύση ανανεώνεται.

5.3.3 Μηχανισμός Προσαρμοστικότητας

Σε κάθε επανάληψη του αλγορίθμου ΠΡΑΕΓ, εφαρμόζεται ένας τελεστής καταστροφής και επισκευής. Για την προσαρμογή της πιθανότητας εφαρμογής κάθε τελεστή, η αναζήτηση χωρίζεται σε χρονικά τμήματα του υπολογιστικού χρόνου *t*. Η πιθανότητα κάθε χειριστή υπολογίζεται λαμβάνοντας υπόψη την απόδοση του χειριστή και τη βελτίωση που έχει προσφέρει στη λύση, κατά τη διάρκεια κάθε χρονικού διαστήματος. Πιο συγκεκριμένα, η πιθανότητα κάθε τελεστή καταστροφής για το επόμενο χρονικό διάστημα (*t+1*) υπολογίζεται με $\pi^d_{t+1} = \pi^d_t(1 - w^d) + w^d * ({s^d}/{i^d})$,, όπου π^d_t είναι η πιθανότητα να εφαρμοστεί κάθε τελεστής στο προηγούμενο τμήμα (*t*), s^d ο αριθμός των φορών που ο τελεστής εφαρμόστηκε και πρόσφερε βελτιωμένη λύση, i^d ο αριθμός των φορών που οποιοσδήποτε χειριστής εφαρμόστηκε και προσέφερε βελτιωμένα αποτελέσματα και, τέλος, w^d το βάρος των επιτυχημένων βελτιώσεων στο συγκεκριμένο χρονικό τμήμα. Η αντίστοιχη εξίσωση για τους τελεστές επισκευής δίνεται από, $\pi^r_{t+1} = \pi^r_t(1 - w^r) + w^r * ({s^r}/{ir})$.

5.3.4 Τελεστές Καταστροφής

Στο τρέχον Κεφάλαιο, έξι τελεστές καταστροφής και τρεις επισκευής εφαρμόζονται για τη βελτίωση της αρχικά παραγόμενης λύσης. Μέσω αυτής της προσέγγισης, και δεδομένου ότι η πιθανότητα κάθε χειριστή υπολογίζεται ανάλογα με την επιτυχία του, η παραγόμενη λύση βελτιώνεται σημαντικά ενώ αποφεύγεται το «κόλλημα» στο τοπικό βέλτιστο. Επιπλέον, πριν από την επιλογή των τελεστών καταστροφής, δοκιμάστηκαν πολλαπλοί τελεστές της βιβλιογραφίας. Ωστόσο, επιλέχθηκαν αυτοί που προσφέρουν την καλύτερη ισορροπία μεταξύ της ταχύτητας υπολογισμού και των αποτελεσμάτων. Πιο συγκεκριμένα, οι χειριστές καταστροφής που εφαρμόστηκαν στην τρέχουσα έρευνα είναι οι i) τυχαίας επιλογής κόμβων, ii) μεμονωμένης διαδρομής, iii) δύο μεμονωμένων διαδρομών, iv) μακρινών κόμβων, v) αυξημένου χρόνου αναμονής και vi) λιγότερο ωφέλιμοι. Οι τελεστές ουσιαστικά αποτελούν κριτήρια μέσω των οποίων επιλέγονται οι πελάτες και αφαιρούνται από τις υπάρχουσες διαδρομές.

5.3.5 Τελεστές Επισκευής

Μετά την εφαρμογή ενός τελεστή καταστροφής, πρέπει να εφαρμοστεί μια μέθοδος επισκευής για την ανακατασκευή της λύσης. Στην τρέχουσα έρευνα, υλοποιούνται τρεις τελεστές επισκευής για να βρουν το καλύτερο σχέδιο δρομολογίων. Ένας που εισάγει τους πελάτες στο σημείο που ελαχιστοποιείται το κόστος, ένας που τοποθετεί τους πελάτες μόνο σε υπάρχοντα δρομολόγια και σε σημεία που ελαχιστοποιείται το κόστος, και ένας τελευταίος ο οποιος εισάγει τους πελάτες σε σημεία όπου ελαχιστοποιείται η συνολικά διανυόμενη απόσταση. Όπως και στην περίπτωση των τελεστών καταστροφής, έτσι και εδώ οι τελεστές έχουν πιθανότητες να εφαρμοστούν σύμφωνα με την επιτυχία τους για να βελτιώσουν τη λύση.

5.4 Υπολογιστική Μελέτη – Μελέτες Περίπτωσης

5.4.1 Δεδομένα της Βιβλιογραφίας

Πριν από την ενσωμάτωση του μεταευρετικού αλγόριθμου στο σύστημα, ο αλγόριθμος δοκιμάστηκε σε σύνολα δεδομένων της βιβλιογραφίας. Το γεγονός ότι δεν υπάρχει σύνολο δεδομένων λαμβάνοντας υπόψη όλες τις παραλλαγές ΠΔΟ που μελετώνται στην τρέχουσα έρευνα ταυτόχρονα, μας οδήγησε να ασχοληθούμε με πολλαπλά σύνολα δεδομένων.

Η μελέτη και η αξιολόγηση του αλγορίθμου ξεκινά με το ΠΔΟΕΣΟ το οποίο με τη σειρά του χωρίζεται σε δύο περιπτώσεις. Στην πρώτη ο στόλος των οχημάτων θεωρείται απεριόριστος, ενώ στη δεύτερη είναι περιορισμένος. Η πρώτη περίπτωση προτάθηκε από τους Golden et al. (1984), ενώ η δεύτερη από τον Taillard (1999). Και στις δύο περιπτώσεις ο στόχος είναι η ελαχιστοποίηση του κόστους διανομής το οποίο υπολογίζεται είτε μεμονωμένα από το σταθερό κόστος, είτε μεμονωμένα από το μεταβλητό κόστος, είτε από τον συνδυασμό των δύο.

Με τα χρόνια, μόνο λίγοι ερευνητές προσπάθησαν να λύσουν το ΠΔΟΕΣΟ μαζί με το ΠΔΟΧΠ. Οι Liu και Shen (1999) ήταν οι πρώτοι που αντιμετώπισαν το ΠΔΟΕΣΟ με χρονικά παράθυρα

(ΠΔΟΕΣΟΧΠ), χρησιμοποιώντας τα δεδομένα αναφοράς του Solomon. Ωστόσο, αυτή η παραλλαγή λαμβάνει υπόψη μόνο διαφορετικές χωρητικότητες και σταθερό κόστος για κάθε τύπο οχήματος, δεν λαμβάνει υπόψη κανένα μεταβλητό κόστος ενώ ο αριθμός διαθέσιμων οχημάτων είναι απεριόριστος, απλοποιώντας σημαντικά το πρόβλημα. Με αυτή την ευκαιρία, οι Jiang et al. (2014) χρησιμοποίησαν τα δεδομένα αναφοράς του Solomon για τη δημιουργία ενός συνόλου δεδομένων για το ΠΔΟΕΣΟΧΠ με περιορισμένο αριθμό οχημάτων, ενώ το κόστος διαμορφώνεται και από το σταθερό και από το μεταβλητό κόστος. Στην τρέχουσα μελέτη, ο αλγόριθμος δοκιμάζεται σε όλα αυτά τα σύνολα δεδομένων και σε περιπτώσεις διανομής σε πραγματικό χρόνο για να αποκτήσει μια πιο ολοκληρωμένη άποψη του αλγορίθμου.

Πρέπει να σημειωθεί ότι τα σύνολα δεδομένων που αναφέρονται παραπάνω καλύπτουν μόνο τις παραλλαγές του ΠΔΟΕΣΟ και του ΠΔΟΧΠ χωρίς να λαμβάνεται υπόψη το ΠΔΟΤΠΠ. Επομένως, για την επικύρωση του αλγόριθμου σε αυτήν την παραλλαγή, αξιοποιήθηκε το σύνολο δεδομένων των Salhi and Nagy (1999) που περιλαμβάνει 14 περιπτώσεις προβλημάτων. Με αυτόν τον τρόπο, η αποτελεσματικότητα του αλγορίθμου δοκιμάστηκε σε κάθε παραλλαγή που εξετάστηκε στην έρευνα.

5.4.1.1 ΠΔΟΕΣΟ με Απεριόριστο Αριθμό Οχημάτων

Ξεκινώντας από το ΠΔΟΕΣΟ με Απεριόριστο Αριθμό Οχημάτων (ΠΔΟΕΣΟΑΑΟ) λύθηκαν τρεις περιπτώσεις, μία που λαμβάνει υπόψιν μόνο μεταβλητό κόστος (V), μία που λαμβάνει μόνο σταθερό (F) και μία που λαμβάνει και σταθερό και μεταβλητό κόστος (FV). Ανεξάρτητα από την περίπτωση, ο αλγόριθμος παρουσιάζει παρόμοια αποτελέσματα καθώς η μέση απόκλιση μεταξύ των καλύτερα δημοσιευμένων και των καλύτερων αποτελεσμάτων στο ΠΔΟΕΣΟ-F είναι 2,02%, στο ΠΔΟΕΣΟ-FV είναι 3,18% και τέλος στο ΠΔΟΕΣΟ-V είναι 1,90%. Τέλος, η απόκλιση σε οποιουδήποτε από τα παραδείγματα που επιλύονται δεν υπερβαίνει το 5,95%, το οποίο επιβεβαιώνει την αποτελεσματικότητα του αλγορίθμου σε αυτό το σύνολο δεδομένων.

5.4.1.2 ΠΔΟΕΣΟ Απεριόριστο Αριθμό Οχημάτων και Χρονικά Παράθυρα

Όπως έχει ήδη αναφερθεί, οι Liu και Shen (1999) αξιοποίησαν και συνδύασαν τα δεδομένα των Golden et al. (1984) για το ΠΔΟΕΣΟΑΑΟ, και το σύνολο δεδομένων του Solomon για το ΠΔΟΧΠ. Όπως και στο Κεφάλαιο 3 έτσι και εδώ Οι ονομασίες των προβλημάτων ακολουθούν αυτές που περιγράφονται στο Κεφάλαιο 3 που σημαίνει ότι τα προβλήματα C1 και C2 έχουν γεωγραφικά συγκεντρωμένους πελάτες ενώ τα R1 και R2, τα γεωγραφικά διάσπαρτους πελάτες. Στα σύνολα δεδομένων RC1 και RC2, περιλαμβάνεται ένας συνδυασμός τυχαίων και ομαδοποιημένων πελατών. Τα σύνολα R1, C1 και RC1 έχουν μικρό χρονικό ορίζοντα προγραμματισμού, μικρού εύρους χρονικά παράθυρα και χαμηλή χωρητικότητα οχημάτων, ενώ τα σετ R2, C2 και RC2 έχουν μεγάλο χρονικό ορίζοντα προγραμματισμού, μεγάλου

εύρους χρονικά παράθυρα και μεγάλη χωρητικότητα οχημάτων, που καθορίζουν τον αριθμό των πελατών που εξυπηρετείται από το ίδιο όχημα.

Τα αποτελέσματα του αλγορίθμου συγκρίνονται με αυτά που προέρχονται από αλγόριθμους που προσφέρουν τα βέλτιστα μέχρι στιγμής αποτελέσματα. Ο αναπτυγμένος ΠΡΑΕΓ αλγόριθμος είναι αποδοτικός σύμφωνα με τα αποτελέσματά του, καθώς η απόκλιση των αποτελεσμάτων του αλγορίθμου σε σχέση με τα βέλτιστα της βιβλιογραφίας, σε κάθε πρόβλημα, δεν ξεπερνά το 8.83%. Επιπλέον, στο σύνολο δεδομένων RC1B ο αλγόριθμος καταφέρνει να προσφέρει καλύτερα αποτελέσματα σε 6 περιπτώσεις προβλημάτων, ενώ στο σύνολο προβλημάτων R2B 1 από αυτά.

5.4.2 ΠΔΟΕΣΟ με Περιορισμένο Αριθμό Οχημάτων

5.4.2.1 ΠΔΟΕΣΟ

Πολλοί ερευνητές θεωρούν ότι η αντιμετώπιση του ΠΔΟΕΣΟ με Περιορισμένο Αριθμό Οχημάτων (ΠΔΟΕΣΟΠΑΟ) είναι πιο ρεαλιστική περίπτωση στη διανομή. Για αυτόν τον λόγο, ο αλγόριθμος ΠΡΑΕΓ δοκιμάζεται και σε ένα τέτοιο σύνολο δεδομένων στο οποίο καταφέρνει να προσφέρει μία νέα βέλτιστη λύση καθώς μειώνει το κόστος διανομής κατά 0.06%. Στα υπόλοιπα προβλήματα ο αλγόριθμος αποδεικνύει καλά αποτελέσματα με την απόκλιση να διακυμαίνεται από 1.34% μέχρι 7.85%,

5.4.3 ΠΔΟΕΣΟ με Περιορισμένο Αριθμό Οχημάτων και Χρονικά Παράθυρα

Το τελευταίο σύνολο δεδομένων που δοκιμάστηκε ο αλγόριθμος είναι των Jiang et al. (2014), χωρίς ωστόσο να έχει προσελκύσει το αναμενόμενο ενδιαφέρον των ερευνητών. Οι συγγραφείς άντλησαν το σύνολο δεδομένων του Solomon και δημιούργησαν διαφορετικούς τύπους οχημάτων που χαρακτηρίζονται από τη χωρητικότητά τους, το σταθερό και μεταβλητό κόστος, τον αριθμό των διαθέσιμων οχημάτων και τον αργότερο χρόνο επιστροφής στην αποθήκη. Ο προτεινόμενος αλγόριθμος αποδίδει εξαιρετικά αποτελεσματικά καθώς, σε 9 από τις 56 περιπτώσεις που υπάρχουν, προτείνοντας νέες βέλτιστες λύσεις. Η απόκλιση μεταξύ των λύσεων που προκύπτουν από τον προτεινόμενο αλγόριθμο και από τα βέλτιστα της βιβλιογραφίας κυμαίνεται από -5,64% έως 7,57, με τη μέση απόκλιση να είναι 2,19%.

5.4.4 ΠΔΟΤΠΠ

Η δοκιμή του αλγορίθμου στο ΠΔΟΤΠΠ αποτελεί ένα εξίσου σημαντικό στάδιο της επικύρωσης του αλγορίθμου. Το σύνολο δεδομένων των Salhi και Nagy (1999) είναι το μόνο από τα σύνολα δεδομένων που μελετήθηκαν που απευθύνονται στο ΠΔΟΤΠΠ. Τα αποτελέσματα που προκύπτουν από τον αλγόριθμο επικυρώνουν την αποτελεσματικότητά του, καθώς η μέση απόκλιση μεταξύ των λύσεων που λαμβάνονται και των βέλτιστων λύσεων της βιβλιογραφίας είναι μόλις 3,72%. Τέλος, σε όλες τις περιπτώσεις προβλημάτων, η απόκλιση μεταξύ της λύσης που προκύπτει από τον αλγόριθμο και των βέλτιστων της βιβλιογραφίας δεν υπερβαίνει το 6,18%.

5.5 Η Περίπτωση των Συνεργατικής Δρομολόγησης

5.5.1 Οφέλη από την Εφαρμογή του ΠΡΑΕΓ στο Συνεργατικό ΠΔΟ

Όπως αναφέρθηκε στο Κεφάλαιο 2, το ΠΔΟ με Πολλαπλές Αποθήκες (ΠΔΟΠΑ) και το Συνεργατικό ΠΔΟ (ΣΠΔΟ) είναι από τις πιο κοινές παραλλαγές. Αυτό οφείλεται κυρίως στο ότι οι εταιρείες διανομής κατέχουν και διαχειρίζονται περισσότερες από μία αποθήκες από τις οποίες εκτελείται η διανομή. Επίσης, στο Συνεργατικό ΠΔΟ (ΣΠΔΟ) οι εταιρείες συνεργάζονται και μοιράζονται τους πόρους τους, όπως τα οχήματά τους, για να μειώσουν το κόστος εφοδιαστικής και να ελαχιστοποιήσουν τις περιβαλλοντικές επιπτώσεις. Δεδομένης της συγκεκριμένης ανάγκης και των συνεχώς αυξανόμενων απαιτήσεων που αφαιρούν την μείωση του περιβαλλοντικού αντίκτυπου και του κόστους μελετάται το Συνεργατικό ΠΔΟ.

Προϋπόθεση για να συνεργαστούν οι εταιρείες διανομής στη διαδικασία της δρομολόγησης είναι να εκτελούν παρόμοιες διαδικασίες και να λειτουργούν στο ίδιο επίπεδο της εφοδιαστικής αλυσίδας (Ferrell et al. 2020). Οι ερευνητές προσδιορίζουν αυτό το είδος συνεργασίας ως οριζόντια. Στο τρέχον Κεφάλαιο της διατριβής μελετάται η οριζόντια συνεργασία μεταξύ τριών εταιρειών 3PL στη διαδικασία δρομολόγησης και διανομής. Αυτό σημαίνει ότι τα αιτήματα των πελατών όλων των εταιρειών που αντιπροσωπεύουν μια αποστολή είτε από το κέντρο διανομής της εταιρείας στον πελάτη είτε το αντίθετο μοιράζονται, μαζί με τους πόρους τους (οχήματα).

Στην παρούσα διατριβή μελετώνται οι περιπτώσεις τριών 3PL εταιρειών που είτε λειτουργούν μεμονωμένα (στο AS-IS σενάριο) είτε συνεργατικά. Σε κάθε περίπτωση οι παραλλαγές του ΠΔΟ που λαμβάνονται υπόψιν είναι (i) το ΠΔΟΕΣΟ, (ii) το ΠΔΟΧΠ και (iii) το ΠΔΟΤΠΠ. Στο Συνεργατικό ΠΔΟ η διαφορά είναι ότι οι πόροι των εταιρειών και τα οχήματα που συμμετέχουν στη διανομή προέρχονται από όλες τις εταιρείες 3PL και μπορεί να χρειαστεί να φορτώσουν εμπορεύματα από όλες τις αποθήκες πριν από την εκτέλεση των δρομολογίων. Επίσης το μαθηματικό μοντέλο διαφέρει όσον αφορά το σύνολο $G = (D \cup N, A)$ όπου D το σύνολο των αποθηκών. Βάσει των δεδομένων και των περιπτώσεων εκτιμώνται τα οφέλη της μετάβασης από την τρέχουσα κατάσταση (οι εταιρείες λειτουργούν μεμονωμένα) στη συνεργατική δρομολόγησης που υπολογίζονται μέσω του μεταευρετικού αλγόριθμου που παρουσιάστηκε στις προηγούμενες ενότητες του Κεφαλαίου 5. Ο στόχος και στις δύο περιπτώσεις είναι να ελαχιστοποιηθεί το συνολικό κόστος διανομής ενώ τηρούνται όλοι οι περιορισμοί.

5.5.2 Μελέτη Περίπτωσης – Υπολογιστικά Αποτελέσματα

Αρχικά, για κάθε εταιρεία και για πολυάριθμες ημερήσιες περιπτώσεις διανομής υπολογίζεται το κόστος διανομής και οι εκπομπές CO_2. Ωστόσο, δεδομένου ότι οι τρεις εταιρείες διανέμουν αγαθά στην Αθήνα και εδρεύουν στην Μάνδρα Αττικής, οι ημερήσιες περιπτώσεις διανομής που αντιμετωπίζει κάθε εταιρεία, συνδυάζονται και παράγουν ένα ενιαίο καθημερινό σύνολο, με πολλαπλές αποθήκες, κοινά οχήματα και δεδομένα πελατών που προκύπτουν από όλες τις εταιρείες, προκειμένου να σχηματιστεί η υπόθεση της συνεργατικής δρομολόγησης. Το κόστος

διανομής και οι εκπομπές CO_2 υπολογίζονται επιπρόσθετα και τα αποτελέσματα συγκρίνονται με εκείνα που αξιοποιήθηκαν από την πρώτη περίπτωση για να εκτιμηθούν τα οφέλη αυτής της στρατηγικής.

Όσον αφορά τους πελάτες, ο καθένας ξεχωριστά χαρακτηρίζεται από τον αριθμό ταυτότητας (ID), τη διεύθυνση, την ποσότητα παράδοσης και παραλαβής και το χρονικό παράθυρο. Οι διευθύνσεις είναι απαραίτητες για την εξαγωγή όλων των απαραίτητων αποστάσεων και χρόνων μετακίνησης μεταξύ των πελατών. Όλα αυτά τα δεδομένα, μαζί με τις ποσότητες παράδοσης και παραλαβής, καθώς και με τα χρονικά παράθυρα, αξιοποιούνται περαιτέρω από τον μεταευρετικό αλγόριθμο ως μεταβλητές και περιορισμοί του προβλήματος για τη βελτιστοποίηση του πλάνου δρομολογίων. Πιο συγκεκριμένα, οι ποσότητες παράδοσης και παραλαβής λαμβάνονται υπόψη σε κάθε στάδιο της διαδικασίας διανομής, έτσι ώστε τα εμπορεύματα επί του σκάφους να μην υπερβαίνουν τη χωρητικότητα του οχήματος, ενώ τα χρονικά παράθυρα θεωρούνται για την αποφυγή καθυστερήσεων. Όσον αφορά τα οχήματα, το μεταβλητό (κατανάλωση καυσίμου) και το σταθερό κόστος (αποσβέσεις, συντήρηση και επισκευές) έχουν υπολογιστεί με βάση τα δεδομένα που παρέχονται από κάθε μία από τις εταιρείες 3PL.

Όσον αφορά τις εκπομπές CO_2 λόγω της χρήσης των οχημάτων οι περισσότεροι κατασκευαστές προσφέρουν σχετικές πληροφορίες όπως τα γραμμάρια CO_2 που εκπέμπονται για κάθε τόνο που μεταφέρεται για απόσταση ενός χιλιομέτρου. Συνεπώς, οι εκπομπές εξαρτώνται από τον παράγοντα c^k (gr of CO_2/tkm), το καθαρό βάρος του οχήματος w^k, το το βάρος των εμπορευμάτων εντός του οχήματος σε κάθε τόξο (i, j), u_{ij}^k, καθώς και την απόσταση του τόξου, d_{ij}. Η μαθηματική έκφραση που υπολογίζει τις εκπομπές CO_2 είναι $\sum_{(i,j)\in A}\left[\left(u_{ij}^k + w^k\right)\cdot c^k \cdot d_{ij} \cdot x_{ij}^k\right]$, και είναι έγκυρο για κάθε όχημα $k \in K$. Αυτή η μέθοδος προσφέρει ακριβή αποτελέσματα εκπομπών CO_2 ακόμη κι αν η ταχύτητα, η ηλικία και άλλοι παράγοντες επηρεάζουν τις εκπομπές (Koç et al. 2014).

Όσον αφορά τις εταιρείες, η πρώτη (3PL C1) εξυπηρετεί κατά μ.ο 137 πελάτες ημερησίως, η δεύτερη (3PL C2) 385 και η Τρίτη (3PL C3) 226. Στην περίπτωση που οι εταιρείες συνεργάζονται ο μέσος αριθμός εξυπηρετούμενων πελατών είναι 748 ενώ οι αποθήκες από τις οποίες τα οχήματα φορτώνουν τα εμπορεύματα είναι τρεις. Στη σύγκριση μεταξύ των αποτελεσμάτων που προέκυψαν από τις μεμονωμένες και συνεργατικές περιπτώσεις φαίνεται ότι η συνεργασία μεταξύ εταιρειών 3PL μπορεί να προσφέρει βελτιώσεις σε πολλαπλά επίπεδα. Η πρώτη πτυχή που βελτιώνεται σημαντικά και αποτελεί τον κύριο στόχο είναι το κόστος διανομής, στο οποίο παρατηρείται μείωση 3,28%. Επιπλέον, μειώνονται και οι εκπομπές (-1,75%), μειώνοντάς τες κατά 42,45 κιλά ημερησίως. Αν και η μείωση φαίνεται μικρή, και λαμβάνοντας υπόψη ότι οι εταιρείες λειτουργούν τουλάχιστον 300 ημέρες το χρόνο, ετησίως μπορούν να αποφευχθούν 12.735 τόνοι CO_2. Αυτό θα ήταν δύσκολο να συμβεί χωρίς τη μείωση της συνολικής διανυθείσας απόστασης (-10,11%) και του αριθμού των απαιτούμενων οχημάτων (-5,71%). Τέλος, ο μόνος παράγοντας που επιδεινώνεται μέσω της συνεργασίας είναι ο συντελεστής φορτίου και πιο συγκεκριμένα, από 87,16%, ο συντελεστής

φορτίου μειώθηκε στο 83,38%, που σημαίνει ότι υπάρχει περισσότερος αχρησιμοποίητος χώρος στα οχήματα.

Επιπρόσθετα, παρατηρείται σημαντική μείωση στο συνολικό κόστος διανομής, καθώς οι τρεις εταιρείες εξοικονομούν σωρευτικά 85,78 € καθημερινά, που αντιστοιχούν σε 25.725,00 € ετησίως. Διαφέρει όμως το μερίδιο της κάθε εταιρείας στο συνολικό κόστος διανομής και κατά συνέπεια η εξοικονόμηση. Πολλοί παράγοντες μπορεί να επηρεάσουν την κοινή χρήση κάθε εταιρείας, όπως ο αριθμός των πελατών και των οχημάτων που ανήκουν σε κάθε εταιρεία και η εμπλοκή στις συνεργατικές διαδικασίες δρομολόγησης και προγραμματισμού. Ως εκ τούτου, μια συμφωνία μεταξύ των συνεργαζόμενων εταιρειών που καθορίζει τον επιμερισμό της κάθε εταιρείας στο συνολικό κόστος είναι απαραίτητη.

5.6 Συμπεράσματα

Στο τρέχον Κεφάλαιο εξετάζονται πολλαπλές παραλλαγές του ΠΔΟ, που έχουν προκύψει από την έρευνα του Κεφαλαίου 2, καθώς και από την εμπειρία συνεργαζόμενων εταιρειών διανομής και ανάπτυξης λογισμικού. Ο στόχος είναι να εξεταστούν οι παραλλαγές που αντιμετωπίζονται πιο συχνά στη διανομή εμπορευμάτων. Επομένως, το ΠΔΟΧΠ, το ΠΔΟΤΠΠ και το ΠΔΟΕΣΟ επιλέγονται και αντιμετωπίζονται ταυτόχρονα.

Ακολουθεί η μαθηματική μοντελοποίηση του προβλήματος η οποία αξιοποιείται για την ανάπτυξη του προτεινόμενου αλγόριθμου ΠΡΑΕΓ, ο οποίος προσφέρει αποτελέσματα υψηλής απόδοσης, όπως διαπιστώνεται από πολλαπλά σύνολα δεδομένων της βιβλιογραφίας στα οποία δοκιμάστηκε. Ο αλγόριθμος εφαρμόζεται επίσης, μετά από συγκεκριμένες τροποποιήσεις, σε μια ακόμη παραλλαγή, που είναι το Συνεργατικό ΠΔΟ. Στην περίπτωση αυτή, οι προηγούμενες παραλλαγές ΠΔΟ συνδυάζονται με τα χαρακτηριστικά του Συνεργατικού ΠΔΟ. Στόχος, σε αυτή την περίπτωση, είναι να προσδιοριστούν τόσο οι οικονομικές όσο και οι περιβαλλοντικές επιπτώσεις από τη συνεργασία εταιρειών που βρίσκονται και δραστηριοποιούνται στις ίδιες περιοχές, στη διαδικασία διανομής. Τα αποτελέσματα εφαρμογής είναι ενθαρρυντικά, δίνοντας μεγάλο χώρο στις εταιρείες να μειώσουν τόσο το κόστος διανομής όσο και τις εκπομπές.

Τέλος, ο προτεινόμενος αλγόριθμος ΠΡΑΕΓ μπορεί να χειριστεί μεγάλες ποσότητες δεδομένων και πολλαπλές μεταβλητές και περιορισμούς. Τα οφέλη από την εφαρμογή του σε πραγματικές περιπτώσεις μπορεί να είναι σημαντικά, λαμβάνοντας υπόψη τα αποτελεσματικά αποτελέσματα που επιτυγχάνονται, σε σύγκριση με τα καλύτερα δημοσιευμένα. Κατά συνέπεια, η ενσωμάτωσή του σε ένα πραγματικό σύστημα cloud θα μπορούσε να προσφέρει μεγάλα οφέλη στις εταιρείες logistics και διανομής.

6. Πρακτική Εφαρμογή των Αναπτυγμένων Αλγορίθμων σε Cloud Σύστημα Δρομολόγησης

6.1 Εισαγωγή

Από τότε που εισήχθη για πρώτη φορά το ΠΔΟ, οι ερευνητές και οι επαγγελματίες επικεντρώθηκαν στη λύση του. Όπως ήταν αναμενόμενο, λόγω των περιορισμένων μεθόδων και των τεχνολογικών περιορισμών, οι λύσεις επικεντρώθηκαν σε μη υπολογιστικές μεθόδους. Ωστόσο, με τα χρόνια, η πρόοδος στην τεχνολογία και οι μέθοδοι βελτιστοποίησης βοήθησαν τους ερευνητές να χειριστούν πολλαπλές παραλλαγές του ΠΔΟ και να αναπτύξουν προηγμένους αλγόριθμους για την αντιμετώπισή τους. Αν και οι μέθοδοι βελτιστοποίησης που προτείνονται από ερευνητές δοκιμάζονται σε σύνολα δεδομένων της βιβλιογραφίας, υποδεικνύοντας υψηλή απόδοση, σπάνια ενσωματώνονται σε πληροφοριακά συστήματα ή λογισμικό. Αυτό συμβαίνει επειδή οι ερευνητές επικεντρώνονται στην ανάπτυξη νέων αλγορίθμων και μαθηματικών μοντέλων που μπορούν να ενισχύσουν κυρίως το θεωρητικό μέρος της μελέτης.

Αντίθετα, οι εταιρείες λογισμικού διαθέτουν την τεχνογνωσία και τα μέσα να αναπτύξουν προηγμένους αλγόριθμους που μπορούν να καλύψουν τις ανάγκες κάθε εταιρείας διανομής. Παρ' όλα αυτά οι αλγόριθμοι που εντάσσονται σε αυτούς δεν είναι διαθέσιμοι καθώς είναι η «καρδιά» του συστήματος και επιφέρει έσοδα στην εταιρεία. Ωστόσο, στην συγκεκριμένη διατριβή οι αλγόριθμοι που προτείνονται στο Κεφάλαιο 3, 4 και 5 αξιοποιούνται από ένα cloud σύστημα δρομολόγησης, το οποίο στην τρέχουσα μορφή του επικοινωνεί μόνο με τον αλγόριθμο ΠΡΑΕΓ ο οποίος λαμβάνει υπόψιν του τις περισσότερες παραλλαγές του ΠΔΟ.

Τέλος, η παρούσα διατριβή συνδυάζει μέρη και έννοιες που ενδιαφέρουν τόσο την έρευνα όσο και τη βιομηχανία, και παρουσιάζει τόσο τους αναπτυγμένους αλγόριθμους όσο και την ενσωμάτωσή τους στο cloud σύστημα. Στο υπόλοιπο Κεφάλαιο παρουσιάζονται αρχικά τα κύρια δεδομένα που τροφοδοτούνται στο σύστημα για να είναι λειτουργικό, στο Κεφάλαιο 6.2. Στο Κεφάλαιο 6.3 και 6.4 παρουσιάζονται η στατική και η δυναμική δρομολόγηση αντίστοιχα. Οι τεχνολογίες που αξιοποιεί το σύστημα και οι λειτουργίες του συστήματος παρουσιάζονται στο Κεφάλαιο 6.5. Τέλος, η αξιολόγηση και τα οφέλη του συστήματος παρουσιάζονται στο Κεφάλαιο 6.6, και ακολουθούν τα συμπεράσματα στο Κεφάλαιο 6.7.

6.2 Στατικά Δεδομένα - Βασικά Αρχεία

Το σύστημα, για να είναι αποτελεσματικό, χρειάζεται πρώτα να τροφοδοτηθεί με όλα τα απαραίτητα δεδομένα. Κατά συνέπεια, ο αλγόριθμος θα αξιοποιηθεί για τη δημιουργία του πλάνου δρομολογίων. Αρχικά, πρέπει να μεταφορτωθεί ένα αρχείο excel, που περιλαμβάνει τα δεδομένα των πελατών, και το οποίο έχει συγκεκριμένη δομή η οποία παρέχεται στην αρχική σελίδα του cloud συστήματος (http://smarttransplanner.azurewebsites.net/).

Επιπλέον, οι εταιρείες διανομής πρέπει να καθορίσουν ορισμένα στοιχεία τους, όπως η διεύθυνση της κεντρικής αποθήκης, τα διαθέσιμα οχήματα, οι χωρητικότητες τους (σε kg και m^3), το σταθερό και μεταβλητό κόστος των οχημάτων, οι διαθέσιμοι οδηγοί και οι βάρδιες του οδηγούς. Τα περισσότερα από αυτά τα δεδομένα αποτελούν είτε μεταβλητές είτε περιορισμούς του προβλήματος, χωρίς τα οποία ο αλγόριθμος δεν μπορεί να προσφέρει ένα πλάνο δρομολογίων. Στην ενότητα 6.3, αναλύεται διεξοδικά ο τρόπος εκμετάλλευσης των παραγγελιών των πελατών για την εξαγωγή των δεδομένων που απαιτούνται.

6.3 Τεχνολογίες και Λειτουργικότητα Συστήματος

Προκειμένου το σύστημα να είναι λειτουργικό, πολλές τεχνολογίες πρέπει να συνδυαστούν. Αρχικά, το σύστημα πρέπει να φιλοξενηθεί σε μια cloud πλατφόρμα, η οποία είναι το Microsoft Azure καθώς μπορεί να δημιουργήσει, να εκτελέσει, να διαχειριστεί και να υποστηρίξει μεγάλες και πολύπλοκες εφαρμογές, διασφαλίζοντας την ασφάλεια των εταιρειών. Επιπλέον, οι εταιρείες που αποκτούν πρόσβαση στις υπηρεσίες του συστήματος πρέπει πρώτα να ανεβάσουν τα κύρια δεδομένα τα οποία αποθηκεύονται, δομούνται και να διαχειρίζονται από την κατάλληλη βάση δεδομένων. Επιλέγεται η Oracle Database λόγω των πρόσθετων δυνατοτήτων διαχείρισης δεδομένων, εργαλείων μοντελοποίησης και οπτικοποίησης δεδομένων, καθώς και λόγω του περιβάλλοντος ανάπτυξης κώδικα.

Επιπλέον, το σύστημα χρησιμοποιεί τις υπηρεσίες web που προσφέρει η εταιρεία HERE Global B. V. Πιο συγκεκριμένα, το σύστημα καλεί μέσω του Postman τα Azure REST API για να ζητήσει και να λάβει δεδομένα από τις υπηρεσίες HERE. Συγκεκριμένα, τα αιτήματα HTTP περιλαμβάνουν τις απαραίτητες πληροφορίες όπως το URL, το αναγνωριστικό, τη μέθοδο αιτήματος και το κλειδί API. Η απάντηση από την HERE περιλαμβάνει ένα αρχείο JSON με τα δεδομένα που ζητούνται. Τέτοια αιτήματα υποβάλλονται στην υπηρεσία HERE Geocoding & Search για εξαγωγή των συντεταγμένων των πελατών μέσω των διευθύνσεών τους.

Ο αλγόριθμος που είναι επίσης αποθηκευμένος στο cloud έχει αναπτυχθεί σε γλώσσα προγραμματισμού Python είναι και αυτός προσβάσιμος μέσω αιτήματος HTTP. Τέλος, τα αποτελέσματα της διαδικασίας δρομολόγησης, που είναι συγκεκριμένες διαδρομές με συγκεκριμένη σειρά παραγγελιών, απεικονίζονται σε διαδικτυακό χάρτη, που παρέχεται για άλλη μια φορά από την HERE.

6.4 Στατική-Αρχική Δρομολόγηση

Για την αρχική δρομολόγηση, στην οποία τα δεδομένα που σχετίζονται με τους πελάτες είναι γνωστά πριν από την έναρξη της διανομής, πρέπει να ληφθούν οι χρόνοι μετακίνησης και οι αποστάσεις μεταξύ των σημείων παραγγελίας και των αποθηκών. Επομένως, ένα αίτημα HTTP που ακολουθεί το RESTful API και περιλαμβάνει τις διευθύνσεις των πελατών αποστέλλεται στην υπηρεσία HERE Geocoding & Search. Η απάντηση από την HERE περιλαμβάνει τις συντεταγμένες των πελατών. Σε περιπτώσεις όπου η διεύθυνση δεν δηλώνεται με σαφήνεια, η HERE δίνει τη δυνατότητα χειροκίνητης επιλογής της διεύθυνσης.

Μετά την ολοκλήρωση της πρώτης φάσης το σύστημα μπορεί να οπτικοποιήσει τα σημεία παράδοσης στον χάρτη. Επιπλέον, το σύστημα μπορεί να εκτελέσει τη διαδικασία δρομολόγησης αφού πρώτα λάβει τους χρόνους μετακίνησης και τις αποστάσεις μεταξύ των σημείων παράδοσης. Αυτά τα δεδομένα λαμβάνονται και πάλι μέσω της HERE Technologies, μέσω αιτήματος HTTP (HERE Matrix Routing API v8), το οποίο περιλαμβάνει τις συντεταγμένες των σημείων παράδοσης. Το αρχείο JSON επιστρέφει ως απάντηση στο αίτημα πίνακες αποστάσεων (M x M) και χρόνων μετακίνησης (M x M), όπου M είναι ο αριθμός των σημείων παράδοσης και των αποθηκών. Αυτοί οι πίνακες, μαζί με δεδομένα ποσοτήτων παράδοσης και παραλαβής (σε kg και m^3), χρονικών παραθύρων, χωρητικότητας και το κόστος των οχημάτων κ.λπ., τροφοδοτούνται στον αλγόριθμο δρομολόγησης. Ένα άλλο αίτημα, συμπεριλαμβανομένων των προαναφερθέντων δεδομένων, υποβάλλεται για λήψη του σχεδίου διαδρομών και του χρονοδιαγράμματος των παραδόσεων μέσω του αλγόριθμου ALNS.

Σαν αποτέλεσμα λαμβάνεται το πλάνο δρομολογίων, στο οποίο κάθε διαδρομή περιλαμβάνει συγκεκριμένα σημεία διανομής. Η ακολουθία των σημείων καθορίζει το συνολικό κόστος και τη διανυθείσα απόσταση. Το συνολικό κόστος διανομής προέρχεται από το σταθερό και το μεταβλητό κόστος κάθε διαδρομής. Επιπλέον, κάθε χρήστης επιλέγοντας μία διαδρομή μπορεί να την απομονώσει και να προβάλει αναλυτικά τα διάφορα δεδομένα που σχετίζονται με αυτήν. Πιο συγκεκριμένα, ο εκτιμώμενος χρόνος άφιξης σε κάθε πελάτη/σημείο παράδοσης, η ποσότητα που παραδόθηκε ή παραλήφθηκε, το χρονικό του παράθυρο, καθώς και η κατάσταση της παραγγελίας εμφανίζονται στο σύστημα.

Επιπλέον, το σύστημα συνδέεται με ένα σύστημα απόδειξης παράδοσης (PoD), που βρίσκεται στα οχήματα των οδηγών και επιτρέπει τον εντοπισμό της κατάστασης των παραγγελιών σε κάθε βήμα της διαδικασίας διανομής. Ως αποτέλεσμα, ο οδηγός έχει τη δυνατότητα να προβεί σε ορισμένες ενέργειες με την ολοκλήρωση μιας παραγγελίας μέσω της εφαρμογής SMARTRANS PoD (https://aberon-pod.azurewebsites.net).

6.5 Διαδικασία Επαναδρομολόγησης

Όπως έχει ήδη αναφερθεί, μπορεί να χρειαστεί να τροποποιηθούν τα δρομολόγια ακόμη και αν το αρχικό σχέδιο διαδρομών έχει ήδη εξαχθεί. Κατά συνέπεια, το σύστημα δίνει στον χρήστη τη δυνατότητα να κάνει τέτοιες τροποποιήσεις στις διαδρομές. Αυτή η ενέργεια εκτελείται συχνά από εταιρείες διανομής, καθώς οι απαιτήσεις τους ενδέχεται να αλλάξουν μετά το αρχικό σχέδιο. Προκειμένου να επιτραπεί η προσθήκη μιας παραγγελίας σε μια υπάρχουσα διαδρομή, το σύστημα ελέγχει πρώτα ότι τηρούνται τα χρονικά παράθυρα και ότι η χωρητικότητα του οχήματος δεν παραβιάζεται. Εφόσον δεν παραβιάζεται κανένας περιορισμός, η σειρά εκτέλεσης και ο χρόνος άφιξης των παραγγελιών επαναϋπολογίζεται.

Επιπλέον, κατά την εκτέλεση των παραδόσεων, οι εταιρείες ενδέχεται να χρειαστεί να τροποποιήσουν τη διαδρομή των οχημάτων, είτε λόγω ακυρώσεων παραγγελιών και αλλαγών στα χρονικά παράθυρα, είτε λόγω απροσδόκητων γεγονότων (κυκλοφοριακή συμφόρηση) που μπορεί να προκύψουν κατά την εκτέλεση των παραδόσεων. Σε τέτοιες περιπτώσεις, όταν είναι

απαραίτητο, το σύστημα υλοποιεί εργασίες αναδρομολόγησης χρησιμοποιώντας ενημερωμένα δεδομένα πελατών και τα δεδομένα κίνησης σε πραγματικό χρόνο (ενημερωμένοι χρόνοι μετακίνησης μεταξύ πελατών) που επικρατούν εκείνη τη στιγμή, εκτελώντας ένα νέο αίτημα HTTP προς την HERE Technologies. Μέσω αυτής της προσέγγισης, η σειρά των παραδόσεων μπορεί να αλλάξει έτσι ώστε οι καθυστερήσεις να ελαχιστοποιηθούν.

Όπως γίνεται σαφές, το σύστημα δρομολόγησης που παρέχεται μέσω του νέφους μπορεί να αντιμετωπίσει αποτελεσματικά το περίπλοκο πρόβλημα της δρομολόγησης είτε αυτό αφορά την αρχική ή στατική δρομολόγηση, είτε την δυναμική.

6.6 Αξιολόγηση και Οφέλη

Όπως έχει ήδη αναφερθεί, οι αλγόριθμοι που παρουσιάζονται στα Κεφάλαια 3, 4 και 5 είναι ενσωματωμένοι στο παρουσιαζόμενο σύστημα, ενώ μόνο ο αλγόριθμος ΠΡΑΕΓ (Κεφάλαιο 5) επικοινωνεί αυτή τη στιγμή, με το σύστημα. Δεδομένου ότι το σύστημα είναι έτοιμο να χρησιμοποιηθεί από εταιρείες logistics για τη διευκόλυνση των εργασιών διανομής τους, αποφασίστηκε να δοκιμαστεί περαιτέρω σε πραγματικές περιπτώσεις διανομής, όπου μεγάλος όγκος δεδομένων διαχειρίζεται δυναμικά. Αυτό το βήμα αποφεύγεται στις περισσότερες ερευνητικές μελέτες, καθώς μόνο λίγοι αλγόριθμοι ενσωματώνονται σε πραγματικά συστήματα. Αυτό οφείλεται στο γεγονός ότι η ανάπτυξη ενός συστήματος δρομολόγησης είναι άρρηκτα συνδεδεμένη με την ανάπτυξη λογισμικού και σχετικές τεχνολογίες όπως το cloud computing, η διαχείριση βάσεων δεδομένων, οι διεπαφές προγραμματισμού εφαρμογών (API) και οι υπηρεσίες web που παρουσιάζονται στην ενότητα 6.3.

Προκειμένου να αξιολογηθεί η αποτελεσματικότητα του αλγορίθμου και του συστήματος συνολικά, εξετάζονται τρεις διαφορετικές περιπτώσεις διανομής στην πραγματική ζωή. Η πρώτη υπόθεση διανομής αφορά την αναπλήρωση καταστημάτων αλυσίδας σούπερ μάρκετ και περιλαμβάνει 141 παραγγελίες μεγάλου μεγέθους. Η δεύτερη περίπτωση αφορά παραδόσεις ηλεκτρικών συσκευών σε καταναλωτές που περιλαμβάνει 195 παραγγελίες. Τέλος, η τρίτη περίπτωση αφορά τη διανομή αγαθών σε κανάλι ηλεκτρονικού εμπορίου που περιλαμβάνει 748 παραγγελίες. Σε κάθε περίπτωση το σχέδιο των διαδρομών που προκύπτει από τις υπάρχουσες προσεγγίσεις που χρησιμοποιούν οι εταιρείες, συγκρίθηκε με αυτό που προκύπτει από το σύστημα. Το γεγονός ότι οι υποθέσεις διανομής ανήκουν σε διαφορετικούς τομείς και ο αριθμός και το μέγεθος των παραγγελιών ποικίλλουν ενισχύουν την έρευνα αφού μελετάται ένα ευρύτερο φάσμα λειτουργιών.

Όσον αφορά τα αποτελέσματα, σε κάθε εταιρεία/περίπτωση η χρήση του συστήματος μπορεί να επιφέρει μείωση του κόστους διανομής. Ωστόσο, το κέρδος στην περίπτωση αναπλήρωσης καταστημάτων είναι μικρό καθώς μειώνεται μόνο κατά 6 € ημερησίως κατά μέσο όρο, ενώ ο αριθμός των δρομολογίων που δημιουργούνται είναι ίδιος. Αυτό οφείλεται κυρίως στο ότι οι δρομολογητές έχουν βελτιστοποιήσει τη διαδικασία και βασίζουν το σχέδιο των διαδρομών στην εμπειρία τους και τη σταθερότητα των παραγγελιών σε καθημερινή φάση, κάτι που τους επιτρέπει να κάνουν μερικές τροποποιήσεις πριν διαμορφώσουν το τελικό σχέδιο. Σε

περίπτωση παράδοσης συσκευών από πόρτα σε πόρτα, το κόστος διανομής μειώνεται κατά 12,56 € με τη χρήση του συστήματος, ενώ τα δρομολόγια παραμένουν ίδια. Τέλος, η περίπτωση του ηλεκτρονικού εμπορίου παρουσιάζει τα πιο ενδιαφέροντα αποτελέσματα καθώς τόσο ο αριθμός των δρομολογίων όσο και το κόστος διανομής μειώνονται σημαντικά. Πιο συγκεκριμένα, ο αριθμός των δρομολογίων μειώνεται κατά 1, ενώ το κόστος κατά 102,08 € ημερησίως, που αντιστοιχεί σε 35.728 € ετησίως (350 εργάσιμες ημέρες). Τα οικονομικά οφέλη φαίνονται σημαντικά, ειδικά στην αλυσίδα του ηλεκτρονικού εμπορίου όπου εξυπηρετούνται πολλαπλές παραγγελίες καθημερινά. Ωστόσο, για να αποφασιστεί εάν η χρήση του συστήματος είναι κερδοφόρα για μια εταιρεία, πρέπει να καθοριστεί το κόστος χρήσης. Στην τρέχουσα κατάσταση του συστήματος, όπου έχει οριστεί μόνο το μοντέλο υπηρεσιών (μοντέλο SaaS), το ακριβές οικονομικό κέρδος των εταιρειών δεν μπορεί να καθοριστεί με σαφήνεια.

Ωστόσο, τα οφέλη που αποκομίζουν οι εταιρείες από τη χρήση του συστήματος δρομολόγησης δεν περιορίζονται στο μειωμένο κόστος διανομής και τον αριθμό των οχημάτων, αλλά επεκτείνονται και στην οπτικοποίηση των διαδρομών και στη δυνατότητα δυναμικής τροποποίησης τους ακόμη και κατά τη διάρκεια των εργασιών διανομής όπως ήδη αναφέρθηκαν στα κεφάλαια 6.4 και 6.5.

6.7 Συμπεράσματα

Τα συστήματα δρομολόγησης είναι απαραίτητα για τις σημερινές απαιτητικές λειτουργίες διανομής, και ιδιαίτερα στα κέντρα των πόλεων όπου οι καθημερινές παραδόσεις και η κυκλοφοριακή συμφόρηση αυξάνονται δραματικά. Ο αλγόριθμος και το cloud σύστημα δρομολόγησης που αναπτύχθηκε στοχεύουν σε αυτήν ακριβώς την ανάγκη, καταφέρνοντας να καλύψει τις ανάγκες και τις απαιτήσεις των περισσότερων εταιρειών που δραστηριοποιούνται στις αστικές εμπορευματικές μεταφορές. Εκτός από τη δημιουργία αποτελεσματικών διαδρομών, το σύστημα διαθέτει επίσης πολλαπλές λειτουργίες για την υποστήριξη των εταιρειών και τη βελτίωση της διαδικασίας δρομολόγησης. Τέτοιες λειτουργίες είναι κυρίως η απεικόνιση των διαδρομών στο χάρτη και η παρουσίαση σχετικών με τις παραγγελίες πληροφοριών, καθώς και οι τροποποιήσεις και προσαρμογές που μπορούν να γίνουν στο πλάνο δρομολογίων, με βάση την εμπειρία των δρομολογητών.

Προκειμένου να υποστηριχθούν όλες αυτές οι λειτουργίες και να δημιουργηθεί ένα τέτοιο λειτουργικό σύστημα, έχουν αξιοποιηθεί και συνδυαστεί πολλαπλές τεχνολογίες, όπως υπολογιστικό νέφος, διαχείριση βάσεων δεδομένων, API, υπηρεσίες web κ.λπ., ενώ ταυτόχρονα ο πυρήνας του συστήματος είναι ο αλγόριθμος ΠΡΑΕΓ που έχει αναπτυχθεί. Ο αλγόριθμος έχει δοκιμαστεί τόσο σε σύνολα δεδομένων της βιβλιογραφίας όσο και σε πραγματικές περιπτώσεις διανομής και αποδείχθηκε αποτελεσματικός τόσο από την άποψη των αποτελεσμάτων όσο και από τον υπολογιστικό χρόνο.

7. Ερευνητικά Αποτελέσματα και Μελλοντικά Βήματα

7.1 Ερευνητική Συνεισφορά

Το ΠΔΟ, το οποίο αποτελεί ένα από τα πιο συχνά προβλήματα που αντιμετωπίζονται στην εφοδιαστική αλυσίδα, προτάθηκε για πρώτη φορά πριν από περισσότερα από 60 χρόνια. Με τα χρόνια, έχουν προταθεί πολλαπλές παραλλαγές, καθεμία από τις οποίες εστιάζει σε συγκεκριμένες πτυχές και μεταβλητές του προβλήματος, οδηγώντας σε συγκεκριμένες παραλλαγές στη γενική μαθηματική διατύπωση. Στις περισσότερες παραλλαγές του προβλήματος, ο στόχος είναι να βρεθούν οι πιο οικονομικές διαδρομές που εξυπηρετούν τη ζήτηση. Κατά συνέπεια, στις περισσότερες περιπτώσεις τίθεται η αντικειμενική συνάρτηση, η οποία είναι η ελαχιστοποίηση του κόστους διανομής. Ωστόσο, τα τελευταία χρόνια, ορισμένοι ερευνητές επικεντρώνονται στην ελαχιστοποίηση των εκπομπών αερίων του θερμοκηπίου λόγω της αυξημένης ανάγκης.

Στην παρούσα διατριβή παρουσιάζεται, μελετάται και αντιμετωπίζεται μια κατηγορία προβλημάτων δρομολόγησης οχημάτων. Οι παραλλαγές που εξετάζονται έχουν προκύψει μέσα από μια ερευνητική μελέτη και τη συνεργασία με εταιρείες του κλάδου (εταιρείες διανομής και ανάπτυξης λογισμικού). Επιπλέον, το ΠΔΟ είναι ζωτικής σημασίας για τη βιομηχανία αφού επηρεάζει τα περιθώρια κέρδους των εταιρειών παραγωγής και διανομής. Το γεγονός αυτό τα έχει αναδείξει στα πιο δημοφιλή και ευρέως μελετημένα προβλήματα στην Εφοδιαστική Αλυσίδα.

Το γεγονός ότι τόσα πολλά διαφορετικά πεδία ενδιαφέρονται για το ΠΔΟ και τις παραλλαγές του έχει οδηγήσει σε πολλαπλούς αλγόριθμους βελτιστοποίησης. Κάθε αλγόριθμος μπορεί να εστιάζει σε διαφορετικές παραλλαγές και πτυχές του προβλήματος. Στις περισσότερες περιπτώσεις, το ΠΔΟΧΠ αντιμετωπίζεται λόγω της σύνδεσής του με πραγματικές υποθέσεις. Ενώ αυτή η παραλλαγή έχει μελετηθεί ευρέως και αντιμετωπίζεται από ερευνητές και επαγγελματίες, οι ανεπτυγμένοι αλγόριθμοι ΠΑΕΓ και ΠΕΑ καταφέρνουν να είναι εξαιρετικά αποδοτικοί και προσφέρουν νέες βέλτιστες λύσεις μέχρι στιγμής. Αυτό δεν είναι εύκολο έργο, κυρίως όταν τόσοι πολλοί ερευνητές έχουν επικεντρωθεί γύρω από το θέμα.

Το ίδιο ισχύει για το ΠΔΟΤΠΠ και το ΠΔΟΕΣΟ, τα οποία σχετίζονται επίσης σε μεγάλο βαθμό με τις πραγματικές περιπτώσεις διανομής. Αρχικά, η συλλογή αγαθών που μπορούν να ανακατασκευαστούν, να επισκευαστούν ή να ανακυκλωθούν, από λιανοπωλητές, τελικούς πελάτες και καταστήματα, είναι σημαντική και μπορεί να οδηγήσει σε οικονομικά και περιβαλλοντικά οφέλη τόσο για τους καταναλωτές όσο και για τις εταιρείες. Αυτός είναι ο λόγος για τον οποίο το ΠΔΟΤΠΠ είναι σημαντικό τόσο για την έρευνα όσο και για τη βιομηχανία. Όσον αφορά τους τύπους των οχημάτων, ορισμένοι πελάτες έχουν πανομοιότυπα οχήματα. Στις περισσότερες περιπτώσεις οι εταιρείες έχουν διαφορετικούς τύπους για να ικανοποιήσουν τις απαιτήσεις της διανομής. Αυτός είναι ο κύριος λόγος για την επιλογή του ΠΔΟΕΣΟ. Επιπλέον, η αντιμετώπιση του ΠΔΟΕΣΟ καλύπτει τις περιπτώσεις στις οποίες οι

εταιρείες διαθέτουν πανομοιότυπα οχήματα και κατά συνέπεια μπορούν να καλύψουν τις ανάγκες τους. Επιπρόσθετα, πέρα από τις παραλλαγές του ΠΔΟ που λαμβάνονται υπόψιν και αντιμετωπίζονται, στο μαθηματικό μοντέλο προστίθενται ορισμένες μεταβλητές και περιορισμοί προκειμένου το πρόβλημα να έρθει ακόμα πιο κοντά στην πραγματικότητα. Τέτοιες μεταβλητές και περιορισμοί είναι τα ωράρια των οδηγών καθώς και η αυτονομία των οχημάτων.

Επιπλέον, ο αλγόριθμος ΠΡΑΕΓ εκτός από την προσφορά πολλαπλών νέων βέλτιστων λύσεων σε διάφορα σύνολα δεδομένων της βιβλιογραφίας, εξετάζει και αντιμετωπίζει πολλαπλές μεταβλητές και περιορισμούς, καθιστώντας τον προτεινόμενο αλγόριθμο την κατάλληλη μέθοδο για ενσωμάτωση στο σύστημα δρομολόγησης cloud. Όπως αναφέρθηκε ήδη, οι τρεις αλγόριθμοι είναι όλοι αποθηκευμένοι στη cloud βάση δεδομένων, που διαχειρίζεται η εταιρεία ανάπτυξης λογισμικού, αλλά το σύστημα στην τρέχουσα μορφή του μπορεί να επικοινωνήσει και να λάβει το σχέδιο των διαδρομών μόνο από τον αλγόριθμο ΠΡΑΕΓ. ο ΠΡΑΕΓ επιλέχθηκε σε αυτήν την αρχική μορφή του συστήματος cloud, καθώς μπορεί να αντιμετωπίσει περισσότερες παραλλαγές του ΠΔΟ σε σύγκριση με τους άλλους δύο αλγόριθμους.

Ωστόσο, σε περιπτώσεις που ένας αλγόριθμος που είναι ο πυρήνας ενός συστήματος, η αποτελεσματικότητά και ο αριθμός των μεταβλητών που λαμβάνει υπόψιν δεν μπορούν να είναι τα μόνα που λαμβάνονται υπόψη. Ο υπολογιστικός χρόνος και ο αριθμός των παραγγελιών που μπορεί να χειριστεί το σύστημα είναι επίσης σημαντικοί. Γι' αυτό και ο συγκεκριμένος αλγόριθμος είναι ενσωματωμένος στο cloud system. Ο ΠΡΑΕΓ έχει δοκιμαστεί σε πολλαπλά δεδομένα με τον αριθμό των παραγγελιών που κυμαίνεται από 100 έως 748. Σε όλες τις περιπτώσεις, συμπεριλαμβανομένων των συνόλων δεδομένων της βιβλιογραφίας και των περιπτώσεων διανομής στην πραγματική ζωή, ο αλγόριθμος μπορεί να δημιουργήσει μια λύση πολύ γρήγορα (λιγότερο από 80 δευτερόλεπτα) ενώ όσο περισσότερο εκτελείται, τόσο περισσότερο βελτιώνεται η λύση (μείωση του κόστους διανομής).

Επιπρόσθετα, πέρα από την ποιότητα των αλγορίθμων που αναπτύσσονται και των αποτελεσμάτων που προσφέρονται, μεγάλη συνεισφορά της έρευνας είναι η αξιοποίηση πραγματικών δεδομένων αποστάσεων και χρόνων που καθιστούν τη δρομολόγηση και το πλάνο παραδόσεων ακόμα πιο αξιόπιστο τόσο σε θέμα εκτίμησης χρόνων άφιξης όσο και εκτίμησης κόστους. Τα δεδομένα χρόνων κατά τη στατική δρομολόγηση αποτελούν στατιστική εκτίμηση παρόχων χαρτογραφικών δεδομένων. Αντίθετα, κατά την δυναμική δρομολόγηση τα χρονικά δεδομένα προκύπτουν από την κίνηση που επικρατεί τη συγκεκριμένη στιγμή στο οδικό δίκτυο και επομένως οποιαδήποτε διαδικασία αναδρομολόγησης γίνεται με πολύ μεγάλη ακρίβεια.

Πέραν των δεδομένων χρόνων μετακίνησης και αποστάσεων που αξιοποιούνται από τους κατάλληλους παρόχους, στην παρούσα διατριβή αξιοποιούνται και χάρτες στους οποίους γίνεται οπτικοποίηση των διαδρομών είτε μαζικά είτε μεμονωμένα. Επιπλέον, καθορίζεται

γραφικά η σειρά επίσκεψης και εκπλήρωσης των παραδόσεων, λειτουργία που μπορεί να βοηθήσει σημαντικά τους οδηγούς.

7.2 Μελλοντικά Βήματα Έρευνας

Οι στόχοι που τέθηκαν αρχικά, είναι (i) η μελέτη μιας κατηγορίας παραλλαγών ΠΔΟ, (ii) η μαθηματική τους διατύπωση, (iii) η ανάπτυξη μεταευρετικών αλγορίθμων που απευθύνονται στις υπό μελέτη παραλλαγές ΠΔΟ και (iv) η ενσωμάτωση των αλγορίθμων σε ένα cloud σύστημα δρομολόγησης, στόχοι που έχουν επιτευχθεί κατά τη διάρκεια των διδακτορικών σπουδών. Ωστόσο, αυτοί οι στόχοι και τα επιμέρους στοιχεία της μελέτης μπορούν να αποτελέσουν αφετηρία για μελλοντικές ερευνητικές δραστηριότητες. Στο υπόλοιπο τμήμα της ενότητας 7.2, δίνονται διάφορες κατευθύνσεις σχετικά με τις μελλοντικές μελέτες.

Πρώτον, τα τελευταία χρόνια, οι ερευνητές επικεντρώνονται στους υβριδικούς αλγόριθμους. Αυτοί οι αλγόριθμοι συνδυάζουν το πλήρες δυναμικό πολλαπλών μεταευρετικών αλγορίθμων. Κάθε αλγόριθμος εστιάζει σε συγκεκριμένες πτυχές που περιέχουν πλεονεκτήματα και μειονεκτήματα. Από την άλλη πλευρά, ο συνδυασμός πολλαπλών μεταευρετικών αλγορίθμων στοχεύει στην εξάλειψη των αδύναμων στοιχείων κάθε μεθόδου ενώ παράλληλα ενισχύει τα δυνατά χαρακτηριστικά τους. Λαμβάνοντας υπόψη τα οφέλη, οι προτεινόμενοι αλγόριθμοι των κεφαλαίων 3, 4 και 5 θα αποτελέσουν τη βάση της νέας προτεινόμενης υβριδικής μεταευρετικής. Ωστόσο, θα μελετηθούν πρόσθετα μεταευρετικά στοιχεία προκειμένου να επιλεγεί η καταλληλότερη για κάθε περίπτωση.

Μια άλλη ερευνητική κατεύθυνση αφορά την πτυχή των εκπομπών CO_2 που θα ληφθούν υπόψη και στους τρεις προτεινόμενους αλγόριθμους. Στην περίπτωση των δύο πολυαντικειμενικών αλγορίθμων, δηλαδή του ΠΕΑ και του ΠΑΕΓ, οι εκπομπές CO_2 θα εμπλέκονται ως μία από τις δύο αντικειμενικές συναρτήσεις. Πιο συγκεκριμένα, η ελαχιστοποίηση των εκπομπών CO_2 θα είναι η μία αντικειμενική συνάρτηση, ενώ η δεύτερη το κόστος διανομής. Αυτό μπορεί να βοηθήσει τις εταιρείες που στη σημερινή απαιτητική αγορά πρέπει να ακολουθήσουν τους εθνικούς περιορισμούς σχετικά με τις εκπομπές αερίων του θερμοκηπίου. Όσον αφορά τον αλγόριθμο ΠΡΑΕΓ, αυτό το χαρακτηριστικό θα πρέπει να συμπεριληφθεί ως εκτίμηση. Με αυτόν τον τρόπο κάθε εταιρεία θα μπορεί να υπολογίζει με μεγάλη ακρίβεια τις εκπομπές που προκαλούνται από τη διαδικασία διανομής.

Επιπλέον, κάθε μία από τις προτεινόμενες μεθόδους βελτιστοποίησης μπορεί να βελτιωθεί περαιτέρω προκειμένου να επιτευχθούν ακόμη καλύτερα αποτελέσματα. Όσον αφορά τον αλγόριθμο ΠΑΕΓ και τον αλγόριθμο ΠΡΑΕΓ που έχουν την ίδια βάση, δηλαδή τον αλγόριθμο ΑΕΓ, θα εξεταστούν και θα μελετηθούν νέοι και αποτελεσματικοί τελεστές καταστροφής και επισκευής. Όσον αφορά το ΠΕΑ, τα επόμενα βήματα της ερευνητικής προσέγγισης θα επικεντρωθούν στους τελεστές διασταύρωσης και μετάλλαξης.

Παρά τα επόμενα βήματα της έρευνας που αφορούν τους αλγόριθμους, υπάρχουν αρκετές ενέργειες που μπορούν να γίνουν για τη μετατροπή του cloud συστήματος δρομολόγησης σε

εμπορικό προϊόν. Αρχικά, για να γίνει το σύστημα εμπορικό προϊόν που υποστηρίζει πλήρως τις εταιρείες logistics που δραστηριοποιούνται στη διανομή εμπορευμάτων, θα πρέπει επίσης να εξεταστούν περισσότερες παραλλαγές ΠΔΟ από τους προτεινόμενους αλγόριθμους και πιο συγκεκριμένα από τον αλγόριθμο ΠΡΑΕΓ που εφαρμόζεται στην τρέχουσα μορφή του συστήματος. Πιο συγκεκριμένα, μετά από δοκιμή του αλγόριθμου σε πραγματικές περιπτώσεις διανομής, διαπιστώθηκε ότι ένας περιορισμένος αριθμός διαδρομών, κυρίως λόγω του περιορισμού της χωρητικότητας του οχήματος, διαρκούν λιγότερο από τη μισή βάρδια του οδηγού. Αυτό σημαίνει ότι ο ίδιος οδηγός μπορεί να εκτελέσει άλλη διαδρομή πριν τελειώσει η βάρδια του. Αυτή η μεταβλητή αναγνωρίζεται ως η παραλλαγή ΠΔΟΠΔ. Η βελτίωση του αλγορίθμου λαμβάνοντας υπόψη αυτή την παραλλαγή θα οδηγήσει σε άριστη κάλυψη των αναγκών των εταιρειών logistics, ανεξάρτητα από το μέγεθός τους.

Επιπλέον, δεδομένου ότι ο ΠΡΑΕΓ δοκιμάστηκε στο Συνεργατικό ΠΔΟ, το σύστημα ενδέχεται στο μέλλον να ενσωματώσει τη συγκεκριμένη παραλλαγή. Κατά συνέπεια, μια περαιτέρω ερευνητική μελέτη θα πρέπει να επικεντρωθεί στον μηχανισμό επιμερισμού των κερδών που πρέπει να καθορίσουν οι συνεργαζόμενες εταιρείες πριν συνεργαστούν. Σε κάθε περίπτωση συνεργασίας, η προσέγγιση του επιμερισμού των κερδών πρέπει να είναι δίκαιη για όλα τα μέρη, ενώ ταυτόχρονα είναι κερδοφόρα. Ωστόσο, το γεγονός ότι ο επιμερισμός των κερδών αποτελεί ένα άλλο ερευνητικό θέμα και έχουν ήδη προταθεί πολλαπλές προσεγγίσεις μας οδήγησε να μην το θεωρήσουμε ως μέρος της ερευνητικής μελέτης. Τέλος, το σύστημα πρέπει να χειρίζεται δεδομένα από πολλούς παρόχους logistics σε συμμόρφωση με την Προστασία Προσωπικών Δεδομένων (GDPR). Αυτό είναι απαραίτητο, καθώς οι πελάτες από διαφορετικές εταιρείες πρέπει να ανεβαίνουν σε μια κοινή πλατφόρμα για την αντιμετώπιση του Συνεργατικού ΠΔΟ. Αυτή η περαιτέρω έρευνα μπορεί να προσφέρει στις εταιρείες logistics ένα πολύτιμο εργαλείο για πιο αποτελεσματική διανομή του τελευταίου μιλίου.

Επιπλέον, ενώ οι τρεις προτεινόμενοι αλγόριθμοι είναι αποθηκευμένοι στη βάση δεδομένων cloud της Oracle, όπως περιγράφεται στην ενότητα 6.3, μόνο ο αλγόριθμος ΠΡΑΕΓ συνδέεται με το σύστημα δρομολόγησης cloud μέσω των κατάλληλων αιτημάτων HTTP, προκειμένου να εξαχθεί το πλάνο δρομολογίων. Δεδομένου ότι ο αλγόριθμος ΠΡΑΕΓ δεν καλύπτει τις ανάγκες των εταιρειών που πρέπει να βελτιστοποιήσουν ταυτόχρονα δύο αντικειμενικές συναρτήσεις, ένα μελλοντικό βήμα της έρευνας είναι η ενσωμάτωση των υπαρχόντων αλγορίθμων με το cloud σύστημα δρομολόγησης. Συγκεκριμένα, μελλοντικός στόχος αυτής της μελέτης είναι να δώσει στους χρήστες τη δυνατότητα να επιλέξουν τον αλγόριθμο που θα εφαρμοστεί και θα εξάγει το πλάνο δρομολογίων. Προκειμένου να επιτύχει αυτό το βήμα αποτελεσματικά, οι στόχοι που αφορούν περισσότερο τις εταιρείες διανομής πρέπει να καθοριστούν και επίσης να ενσωματωθούν στους αλγόριθμους πολλαπλών στόχων. Στην τρέχουσα μορφή τους, ο αλγόριθμος ΠΕΑ και ο ΠΑΕΓ δεν αντιμετωπίζουν εξίσου πολλές παραλλαγές του ΠΔΟ όπως ο αλγόριθμος ΠΡΑΕΓ.

Καταλήγοντας, γίνεται σαφές ότι ο ανεπτυγμένος αλγόριθμος ΠΡΑΕΓ, ο οποίος υλοποιείται από το σύστημα δρομολόγησης cloud, μπορεί να προσφέρει σημαντικά οφέλη στις εταιρείες

διανομής. Το αποδοτικό σχέδιο διαδρομών που παράγεται από τον αλγόριθμο ΠΡΑΕΓ ενισχύεται από άλλες λειτουργίες του συστήματος, προσφέροντας μια ολοκληρωμένη λύση δρομολόγησης που υποστηρίζει σθεναρά τους σχεδιαστές. Ωστόσο, αυτό δεν σημαίνει ότι το σύστημα δεν μπορεί να βελτιωθεί περαιτέρω. Όπως ήδη αναφέρθηκε, η ενσωμάτωση περισσότερων μεταβλητών και περιορισμών στον αλγόριθμο, και στη συνέχεια στο σύστημα δρομολόγησης cloud, θα αυξήσει σημαντικά τη χρηστικότητα και την κάλυψη αναγκών, καθώς θα καλυφθούν περισσότερες απαιτήσεις και ανάγκες των χρηστών.

Βιβλιογραφία

Abbatecola L, Fanti MP, Mangini AM, Ukovich W (2016) A Decision Support Approach for Postal Delivery and Waste Collection Services. IEEE Trans Autom Sci Eng 13:1458–1470. https://doi.org/10.1109/TASE.2016.2570121

César H, Oliveira B De (2010) A hybrid search method for the vehicle routing problem with time windows. Ann Oper Res 180:125–144. https://doi.org/10.1007/s10479-008-0487-y

Clarke G, Wright JW (1964) Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Oper Res 12:568–581. https://doi.org/10.1287/opre.12.4.568

Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manage Sci 6:80–91. https://doi.org/10.1287/mnsc.6.1.80

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197. https://doi.org/10.1109/4235.996017

EEL (2018) Logistics in Greece Today

Ehrgott M (2005) Multicriteria optimization: Second edition

El-Sherbeny NA (2010) Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. J King Saud Univ 22:123–131. https://doi.org/10.1016/j.jksus.2010.03.002

Elshaer R, Awad H (2020) A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Comput Ind Eng 140:106242. https://doi.org/https://doi.org/10.1016/j.cie.2019.106242

Erdelić T, Carić T (2019) A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches. J Adv Transp 2019:5075671. https://doi.org/10.1155/2019/5075671

Erdelić T, Carić T, Erdelić M, Tišljarić L (2019) Electric vehicle routing problem with single or multiple recharges. Transp Res Procedia 40:217–224. https://doi.org/https://doi.org/10.1016/j.trpro.2019.07.033

Erdoğan G (2017) An open source Spreadsheet Solver for Vehicle Routing Problems. Comput Oper Res 84:62–72. https://doi.org/10.1016/j.cor.2017.02.022

Fanti MP, Laraspata R, Iacobellis G, et al (2014) A Decision Support System approach for the postal delivery operations. In: 2014 IEEE International Conference on Automation Science and Engineering (CASE). pp 588–593

Ferrell W, Ellis K, Kaminsky P, Rainwater C (2020) Horizontal collaboration: opportunities for improved logistics planning. Int J Prod Res 58:4267–4284. https://doi.org/10.1080/00207543.2019.1651457

Figliozzi MA (2012) The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. Transp Res Part E Logist Transp Rev 48:616–636. https://doi.org/10.1016/j.tre.2011.11.006

Gambardella LM, Taillard É, Agazzi G (1999) Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In: New Ideas In Optimization. pp 63–76

Gayialis SP, Tatsiopoulos IP (2004) Design of an IT-driven decision support system for vehicle

routing and scheduling. Eur J Oper Res 152:382–398. https://doi.org/10.1016/S0377-2217(03)00031-6

Gillett BE, Miller LR (1974) A Heuristic Algorithm for the Vehicle-Dispatch Problem. Oper Res 22:340–349. https://doi.org/10.1287/opre.22.2.340

Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13:533–549. https://doi.org/10.1016/0305-0548(86)90048-1

Golden B, Assad A, Levy L, Gheysens F (1984) The fleet size and mix vehicle routing problem. Comput Oper Res 11:49–66. https://doi.org/10.1016/0305-0548(84)90007-8

Gong G, Deng Q, Gong X, et al (2018) A Bee Evolutionary Algorithm for Multiobjective Vehicle Routing Problem with Simultaneous Pickup and Delivery. Math Probl Eng 2018:2571380. https://doi.org/10.1155/2018/2571380

Govindan K, Soleimani H, Kannan D (2015) Reverse logistics and closed-loop supply chain: A comprehensive review to explore the future. Eur J Oper Res 240:603–626. https://doi.org/10.1016/j.ejor.2014.07.012

Grangier P, Gendreau M, Lehuédé F, Rousseau L-M (2016) An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. Eur J Oper Res 254:80–91. https://doi.org/https://doi.org/10.1016/j.ejor.2016.03.040

Hornstra RP, Silva A, Roodbergen KJ, Coelho LC (2020) The vehicle routing problem with simultaneous pickup and delivery and handling costs. Comput Oper Res 115:104858. https://doi.org/10.1016/j.cor.2019.104858

Jiang J, Ng KM, Poh KL, Teo KM (2014) Vehicle routing problem with a heterogeneous fleet and time windows. Expert Syst Appl 41:3748–3760. https://doi.org/10.1016/j.eswa.2013.11.029

Kalayci CB, Kaya C (2016) An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. Expert Syst Appl 66:163–175. https://doi.org/https://doi.org/10.1016/j.eswa.2016.09.017

Kirkpatrick S, Gelatt CD, Vecchi MP, P. VM (1983) Optimization by Simulated Annealing. Science (80-) 220:671 LP-680. https://doi.org/10.1126/science.220.4598.671

Koç Ç, Bektaş T, Jabali O, Laporte G (2014) The fleet size and mix pollution-routing problem. Transp Res Part B Methodol 70:239–254. https://doi.org/https://doi.org/10.1016/j.trb.2014.09.008

Labadie N, Prins C, Prodhon C, Prins C (2016) Metaheuristics for Vehicle Routing Problems. John Wiley and Sons

Leyerer M, Sonneberg M-O, Heumann M, et al (2019) Individually Optimized Commercial Road Transport: A Decision Support System for Customizable Routing Problems. Sustainability 11:. https://doi.org/10.3390/su11205544

Lin C, Choy KL, Ho GTS, et al (2014) Survey of Green Vehicle Routing Problem : Past and future trends. Expert Syst Appl 41:1118–1138

Liu F-H, Shen S-Y (1999) The Fleet Size and Mix Vehicle Routing Problem with Time Windows. J Oper Res Soc 50:721–732. https://doi.org/10.2307/3010326

Mancini S (2017) The Hybrid Vehicle Routing Problem. Transp Res Part C Emerg Technol

78:1–12. https://doi.org/10.1016/j.trc.2017.02.004

Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100. https://doi.org/https://doi.org/10.1016/S0305-0548(97)00031-2

Nagy G, Salhi S (2005) Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. Eur J Oper Res 162:126–141. https://doi.org/https://doi.org/10.1016/j.ejor.2002.11.003

Ombuki B, Ross BJ, Hanshar F (2006) Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl Intell 24:17–30. https://doi.org/10.1007/s10489-006-6926-z

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34:2403–2435. https://doi.org/10.1016/j.cor.2005.09.012

Prins C (2009) Two memetic algorithms for heterogeneous fleet vehicle routing problems. Eng Appl Artif Intell 22:916–928. https://doi.org/10.1016/j.engappai.2008.10.006

Renaud J, Laporte G, Boctor FF (1996) A tabu search heuristic for the multi-depot vehicle routing problem. Comput Oper Res 23:229–235. https://doi.org/10.1016/0305-0548(95)O0026-P

Salhi S, Nagy G (1999) A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. J Oper Res Soc 50:1034–1042. https://doi.org/10.1057/palgrave.jors.2600808

Silva MM, Subramanian A, Ochi LS, et al (2014) An iterated local search heuristic for the split delivery vehicle routing problem. Eur J Oper Res 53:454–464. https://doi.org/https://doi.org/10.1016/j.cor.2014.08.005

Solomon MM (1987) Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Oper Res 35:254–265. https://doi.org/10.1287/opre.35.2.254

Taillard ÉD (1999) A heuristic column generation method for the heterogeneous fleet VRP. RAIRO-Operations Res Opérationnelle 33:1–14. https://doi.org/10.1051/ro:1999101

Tan KC, Chew YH, Lee LH (2006) A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem. Comput Optim Appl 34:115–151. https://doi.org/10.1007/s10589-005-3070-3

Tarantilis CD, Diakoulaki D, Kiranoudis CT (2004) Combination of geographical information system and efficient routing algorithms for real life distribution operations. Eur J Oper Res 152:437–453. https://doi.org/https://doi.org/10.1016/S0377-2217(03)00035-3

Tarantilis CD, Kiranoudis CT (2007) A flexible adaptive memory-based algorithm for real-life transportation operations: Two case studies from dairy and construction sector. Eur J Oper Res 179:806–822. https://doi.org/https://doi.org/10.1016/j.ejor.2005.03.059

Wang Y, Ma X, Li Z, et al (2017) Profit distribution in collaborative multiple centers vehicle routing problem. J Clean Prod 144:203–219. https://doi.org/https://doi.org/10.1016/j.jclepro.2017.01.001

Zachariadis EE, Tarantilis CD, Kiranoudis CT (2016) The Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries and Two-Dimensional Loading Constraints. Eur J Oper Res 251:369–386. https://doi.org/https://doi.org/10.1016/j.ejor.2015.11.018