Heuristic approaches for the Vehicle Routing Problem with Heterogeneous Fleet and Real Life Attributes

A Thesis Submitted to the University of Kent in the subject of Management Science for the degree of Doctor of Philosophy

By Lina Simeonova

Supervisors:
Dr. Niaz Wassan
Dr. Gabor Nagy

Kent Business School
University of Kent

TABLE OF CONTENTS

List of Figures

List of Tables

Acknowledgment

I would like to express my most sincere gratitude to my supervisor Dr. Niaz Ahmed Wassan, for his unconditional support and guidance during my PhD. I greatly appreciate all he has done for me and I have utmost respect for him as a mentor and as a friend. I would also like to thank my second supervisor Dr. Gabor Nagy for his support and input for my thesis and Prof. Said Salhi for his kind help on different aspects of my research. I would also like to thank the Kent Business School who funded my PhD and gave me the opportunity to experience the University, not only as a student, but also as a Graduate Teaching Assistant and an Academic Ambassador.

I would like to thank my parents, my mother Pepka Simeonova and my father Rosen Simeonov for all their love and support during my PhD and for always believing in me. I especially want to thank my sister Beti Simeonova. I dedicate this thesis to her and I wish her the same unforgettable academic experience at the University of Kent. Last, but not least I would like to thank my fellow PhD colleagues and friends Naveed, Mathew, Stefano, Tommaso, Sheena, Marios, Julie, Jinwoo and everyone at the school. We shared much more than an academic experience, but common goals and aspirations, and I am looking forward to working with them in the future.

Abstract

The Vehicle Routing Problem with all its variants and richness is still one of the most challenging Combinatorial Optimization problems in the Management Science / Operations Research arena since its introduction in the 1950s. In this research we introduce a real life Vehicle Routing Problem, inspired by the Gas Delivery industry in the UK. It has various real life attributes which have not been researched in the past, such as demand-dependent service times, light load requirements and allowable overtime coupled with unlimited vehicle fleet. A Mixed Integer formulation of the problem is presented and the problem is solved to optimality, reporting optimal solutions and lower and upper bounds. After solving the real life routing problem, both optimally and heuristically some interesting observations and practical implications are reported, relating to better fleet utilization and better working time utilization.

We design three initial solution methods, namely the Adapted Sweep, the Adapted Nearest Neighbour and the Parallel Clustering method. They are motivated by the real attributes of the Vehicle Routing Problem under research and show a very good performance in terms of reaching a good initial solution quality as compared to other famous initial solution methods in the literature. Moreover, the Adapted Sweep and the Adapted Nearest Neighbour have computational times of less than one second.

Two new hybrid metaheuristic methods are designed in order to address the real life Vehicle Routing Problem. A Population Variable Neighbourhood Search with Adaptive Memory Procedure is the first method, which aims to incorporate and hybridize the learning principles of Adaptive Memory into a method which does not make use of memory structures in its original form, namely the Variable Neighbourhood Search. Moreover, we use a Population version of the Variable Neighbourhood Search in order to provide diversification to the method and to aid the learning aspect of the method. The Population Variable Neighbourhood Search with Adaptive Memory Procedure was tested extensively on the real life Vehicle Routing Problem, as well as relevant literature benchmark instances and it was

found to perform well in comparison with the optimal solutions we generated. Moreover, the method shows a good performance on the benchmark instances with less than 1% deviation from the Best Known Solutions in the literature.

We later extend the Population Variable Neighbourhood Search with Adaptive Memory Procedure (PVNS_AMP) and hybridize it with aspects from Tabu Search in order to create the second new hybrid metaheuristic method, namely the Population Variable Neighbourhood Search with Adaptive Memory Procedure and Tabu Search principles (TS_PVNS_AMP). The TS_PVNS_AMP was found to have better performance on the RVRP without overtime, and superior performance on the RVRP with overtime as compared to the PVNS_AMP. Moreover, the TS_PVNS_AMP showed a better performance than the PVNS_AMP on the relevant literature benchmark instances reaching Best Known Solutions in the literature with less than 0.5 % deviation from the Best Known Solutions on average.

We have also tested our proposed algorithms on other VRP problems, such as the Heterogeneous Fleet VRP with imposed fleet and the School Bus Routing Problem. We have done this experimentation in order to test the generalizability of the methods and their flexibility in addressing other problems from the Vehicle Routing family. Our methodology showed good performance on the literature benchmarks for both problems in terms of solution quality and computational time, as well as a good degree of flexibility in terms of finding good heuristic solutions.

CHAPTER 1

Introduction

1.1. Purpose of the research

The purpose of this research is to formulate and solve a real life Vehicle Routing Problem inspired by the UK Gas delivery industry, as well as design a new hybrid metaheuristic method to address the RVRP. The new method is also tested on other relevant Vehicle Routing Problems from the literature, in order to show algorithmic efficiency and a degree of methodological generalizability.

1.2. Objectives of the research

The three main objectives of this research are as follows:

(i) To introduce a new real life Vehicle Routing Problem (RVRP) variant, which reflects the real routing practices of a gas delivery company in the UK. It is characterized with heterogeneous vehicle fleet, demand-dependent service times, special requirement for light load, maximum allowable overtime and other relevant routing elements.

(ii) To present a new Mixed Integer Formulation of the RVRP and test it using Cplex, where optimal solutions and lower/upper bounds are reported where possible.

(iii) For the RVRP introduced in this research, only small problem instances can be solved to optimality. Therefore, a new metaheuristic algorithm is designed to solve the proposed RVRP. It is based on the classical Variable Neighbourhood Search (VNS), but it is adapted in a population-based manner, hence becoming Population VNS or PVNS. It also makes use of learning principles and mechanisms inspired by Adaptive Memory Programming (AMP). We refer to the new method as PVNS_AMP. The PVNS_AMP was later enhanced by adding principles from Tabu Search, which resulted in a new hybrid metaheuristic method, which we call TS_PVNS_AMP.

1.3. Contribution to Knowledge

The contribution to knowledge stems from the 3 main objectives of this research, namely to introduce a new RVRP to the literature, to design a new methodology to address the RVRP and to demonstrate the methodological efficiency and potential for generalization to other relevant problems from the VRP domain. Therefore, the contribution of this research is threefold and it is stated below:

(i) Researching a Vehicle Routing Problem, which can be applied in real setting, is one of the main drivers in the literature over the years and it fits into the current trend of minimizing the gap between optimization and real life practices. Moreover, adapting the classical Vehicle Routing Problem to better represent real operations is the reason for the steep growth of the Vehicle Routing Problem variants. To the best of our knowledge, there is no VRP problem in the literature which considers the same routing elements, as the one introduced in this research, which makes it a new RVRP variant. Researching a RVRP requires some practical implications of how routing practices can be improved, since it is inspired by real operations. One of the ultimate contributions of researching a RVRP is to be able to show that as a result of the study, there can be some cost savings or ways to improve the current practice. This research offers some interesting insights and analysis of the results of the study, which show great potential for cost savings and more efficient routing.

(ii) Testing the formulation of the RVRP on Cplex is for methodological purposes. It is common in the literature when addressing a RVRP, that either an exact method or heuristic method is used, but rarely both. Moreover, there are no universal literature benchmark instances for RVRPs, because they are so different from one another. Therefore, having an optimal solution and lower/upper bounds, acts as a guide for the efficiency of any proposed heuristic method and serves as a methodological comparability platform. This is an important aspect when dealing with RVRPs and it is not yet addressed in the literature.

(iii) The new method introduced in this research, namely the PVNS_AMP, employs learning mechanisms for extracting good solution sequences with good diversity of candidate solutions used to enhance learning and diversification of the solution space search. The enhanced method, namely the TS_PVNS_AMP shows superior performance to the PVNS_AMP, demonstrating greater intensification of the search space and a good performance on

literature benchmark instances, reaching best known solutions which in most cases are proved to be optimal. Moreover, we test our methodology on other relevant Vehicle Routing Problems, such as the Fleet Size and Mix VRP (FSMVRP), the Heterogeneous fleet VRP with imposed fleet (HFVRP) and the School Bus Routing Problem (SBRP), in order to test the generalizability of our methodology to other problems within the VRP domain. This is not commonly done in the literature, because real-life VRPs are very problem specific. However, in this research the trend for methodological generalizability is incorporated and some interesting results and observations are reported.

1.4. Background of the Research

Transportation is an inseparable part of any society. It has a very close relation with other aspects of life ranging from personal lifestyle and status to the general ability to consume and distribute utilities, goods, commodities and skills. The world today is more economically, socially and politically integrated than any other time in history. Advances in the area of transportation not only shorten the distance between countries, companies and people but are a major landmark for humanity. Therefore, researchers still aim to improve the way we travel and transport. Some academic publications are concerned with the benefits of transportation and how it improves and helps societies, especially in economic and political context. However, most of the publications in many academic areas, such as marketing, supply chain, economics and management science/ operational research, are concerned with minimizing the cost of travel and the significance of finding new concepts and methods to make transportation more effective and cost efficient ranging from innovative supply chain management principles to specific algorithms for distribution.

The reason why businesses are concerned with improvement in transportation capability is because it allows them to build strategic competences based on just-in-time management, added value, consumer relations, as well as improving the business flexibility, effectiveness and efficiency (Morash, 1997). Companies can have cost advantage before competitors, differentiation advantage or both (Jobber, 2008). Therefore, continuous academic and practical efforts are directed towards achieving cost advantage through transportation optimisation and in practice many companies have based their strategic competence on distribution optimisation, such as Tesco Plc (Jobber, 2008). Nowadays, there is little economic growth in the different sectors of transportation in the UK and the delivery cost of goods and

services for the end user is increasing, as well as pushing up the Consumer Price Index (Office of National Statistics, 2012). The main reason for this is the sharp increase in fuel cost and operational costs. Therefore the first key area of consideration in the transportation sector outlined by the Freight Transport Association is minimizing costs of distribution, followed by delivering high quality service, human resource considerations and minimizing carbon emissions (The Logistics Report, 2012).

Transportation has many definitions, depending on the area it is applied to. For the purpose of this research transportation is defined as "transportation of materials and products to and from markets and suppliers" (Transportation Research Board). Moreover, the focus here is distribution of goods by road, which in the UK is responsible for 60% of the total movement of goods (Transportation Statistics Great Britain, 2011).

Road distribution and transportation falls under the umbrella term "logistics". Logistics is the management of resource flow from an origin point to various destination points, in order to meet a set of needs, either individual or corporate. The resource flow can be food, materials, liquids, utilities, materials and many more. Depending on the transported commodity, the nature of the business, the fleet size and other operational and strategic aspects of the business, the logistics system can be optimized in a way to fit the companies' objectives and minimize costs. Logistics is a part of the supply chain management process and there are various types of logistics such as procurement, production, distribution, disposal and reverse. The traditional logistics management encompasses decisions ranging from production timing, warehousing, inventory management and control, just-in-time management, purchasing, vehicle maintenance. It is a full managerial concept for complete understanding of production, distribution channels and after sales management. Some researchers focus on inventory optimisation, others on just-in-time delivery, whereas this research fits into the stream of vehicle routing and scheduling optimisation.

Optimising logistics is not only an issue for individual companies in their attempt to achieve competitive advantage and maximum efficiency. Advance in the area of logistics is also a national and international priority. For instance in the UK there is a Chartered Institute for Logistics and Transport, which supports knowledge and advances in the area, providing membership for major logistics companies. Moreover, there is a European Union body

Eurodecision Operational Research, which provides solutions for various aspects of logistics optimisation, such as route optimisation, planning optimisation, staff optimisation. Additionally, it provides information on software and models for solving various logistics problems using linear programming, non-linear programming, metaheuristics methods, artificial intelligence and many more (EUOR, 2012). The logistics management involves strategic and operational decisions. The operational decisions are location, production, inventory and distribution. The distribution element is the single most important element for companies and a key area for decision making in the logistics mix, because for most firms transportation incurs the greatest costs (Marinakis, 2012) and driving costs down is beneficial both for companies and consumers (Toth and Vigo, 2002). Moreover, decision making for distribution has to be fast and timely, as it has shorter planning horizons. Therefore, designing new efficient and fast algorithms to better optimize distribution is paramount for advances in economy and academia.

This research is focused on road transport optimization, which is inspired by real life company operations. It falls into the family of Vehicle Routing Problems. The Vehicle Routing Problem (VRP) is a Combinatorial Optimisation (CO) problem which belongs to the area of Management Science and Operations Research (MS/OR). There are two fundamental approaches known in the literature used to address the VRP. On one hand there is the classical heavily quantitative approach, which deals with VRP in a more 'laboratory' manner and the aim is to design quantitative solution methods for exact or heuristic optimization. On the other hand there are the Soft Systems Methodologies and Problem Structuring Methods, which argue to be alternative to the classical approaches and aim to capture real world uncertainties and complexities (Kirby, 2007), as well as some qualitative aspects. This research belongs to the former approach, but it occupies an area in the MS/OR literature, which is gaining much popularity nowadays, namely the real life Vehicle Routing Problems (RVRP). Real life problems still do not have a widely accepted definition, but what is common for them is that they follow the classical approach and provide efficient quantitative solutions for RVRPs, but also incorporate routing characteristics that are usually informed by real routing operations.

The area of RVRP is gaining much popularity mainly because there is a trend in the literature to bring academia closer to industry practices. Some criticisms that the OR community

experiences an 'over-mathematization' and simplification of the optimization problems (Kirby, 2007) lead to a quest for more real life oriented VRP variants, which incorporate elements of real logistics operations. Nowadays, capacities of vehicles, demands and distances are not sufficient elements to build a true representation of the VRP and much more is demanded from authors, in order to make a significant contribution to the literature. The new problem introduced here and the motivations behind it fit into the trend of minimizing the gap between literature and operations and make the application of the RVRP much more practical. Contrary to common approaches to VRP research, here an exploratory research is conducted prior to the formulation of the research question with the aim to discover the important VRP elements to be modelled.

1.5. Thesis outline

This thesis begins with a detailed literature review in Chapter 2 of any relevant VRP variants and solution methods used to address VRPs. The focus is placed on any VRP variants and solution methods, which are relevant to this research. Chapter 3 provides a full description of the RVRP problem under study, as well as formulation and optimal solutions achieved by Cplex. Chapter 4 details the initial solution methods we designed in order to generate a starting point for our metaheuristic algorithms, with relevant descriptions and computational experience. Chapter 5 and 6 give details on the hybrid metaheuristic methods we designed, namely PVNS_AMP and TS_PVNS_AMP. Description of the algorithmic steps is provided for each method, as well as any relevant parameter testing, methodological justification and detailed results on the RVRP and literature benchmark instances. Chapter 7 offers an extension to the application of our methodology, where we test it on other relevant VRP problems for a degree of generalizability. Chapter 8 provides a rounded conclusion detailing the main findings and interesting observations from this research. Moreover, we discuss how we have achieved the main objectives of this research and our contribution to the body of knowledge. Finally, we provide directions for further research, which are motivated by our findings and computational experience.

CHAPTER 2

Literature Review

This literature review will focus on two key aspects.

(i) Some of the most researched variants of the VRP, as well as the classical VRP are described, with the focus being on VRPs relevant to the RVRP in this study. RVRPs are discussed in more depth and some critical perspective is offered on their definition and methodological justification.

(ii) A review of the exact methods for the VRP is offered, but the focus is on heuristic methods, because this study aims to design a new hybrid metaheuristic method to address the RVRP. A classification of the heuristic methods is presented, as well some details and successful applications of the methods. The methods which are used in this study are discussed in-depth, as well as some of the most powerful methods in the literature which are responsible for various best known solutions.

2.1. The history of the Vehicle Routing Problem

Combinatorial Optimization (CO) is a topic, which aims to find optimal objects from a finite set of objects, for instance finding a shortest path in an undirected graph. Usually, there is a set of discrete feasible solutions and the goal is to find the best solution. Combinatorial means that all possible combinations of the decision variables must be exploited in order to find an optimal solution (Gambardella, 2005). For this type of problems explicit enumeration and exhaustive search is not always feasible, because with the increase of possible decision moves the complexity of the problem increases exponentially, as well as the time necessary to solve it. According to computational complexity theory, the VRP belongs to the class of NP-hard problems (Non-Deterministic Polynomial time hard), which classifies how easy or hard certain types of decision problems are to solve. There is still some debate on whether certain CO problems belong to NP or NP-hard or P. Put simply, VRP being a NP-hard problem, means that

there exists an algorithm which can solve the problem on a non-deterministic machine (with unlimited parallelism) and a deterministic machine can verify that the solution is correct in polynomial time. An algorithm for solving CO problems is good if it can be solved in polynomial time. For smaller instances exhaustive search (exact) methods can be used to solve a VRP in polynomial time, whereas larger instances are impossible to solve exactly, because of the intrinsic complexity of the problems. This is why there is great academic effort to design heuristic (approximate) algorithms, which are polynomial and provide quick good solutions to large VRP problems. Some authors even state that the NP-hard nature of VRPs is what pushed researchers to explore and use heuristic methods (Yeun, 2008). For further insight on computational complexity, readers are referred to Lawler (1976).

The VRP is based on the famous CO problem The Travelling Salesman Problem (TSP). The TSP is an NP-hard problem as well, however until present day much larger instances are solved to optimality. The origin of the TSP dates back to 1800, when Sir William Hamilton created the Icosian Game. It was in the form of a pegboard with 20 holes and each vertex was required to be visited only once, where the ending point is the same as the starting point. The resulting path that connects all vertices is referred to as a Hamiltonian Circuit. Therefore, the optimal solution to the TSP is a Hamiltonian circuit in a complete weighted graph (graph, where each vertex is connected to one another by a single edge, which carries a specific value, usually representing a cost, distance or other value), which has the smallest sum of edges value (Fields, 2004).

Following the principle of the Icosian game, the purpose of the TSP is to find the shortest tour from a starting point that visits all vertices (cities) exactly once and returns to the origin. Each possible tour can be described as a cyclic permutation π, which represents the order in which a salesperson visits all cities. The objective is to find the minimum length permutation π. For an instance with n cities, there are *(n - 1)!* possible permutations that have to be compared, in order to find the shortest one (Fields 2004). Figure 2.1 shows possible solutions for a small TSP. Finding the shortest permutation requires examining all possible combination of sequences of all cities. Figure 2.1 only shows three possible solutions, whereas in fact for a problem with 7 nodes, there are 720 possible solutions that all need to be explored in order to find the optimal.

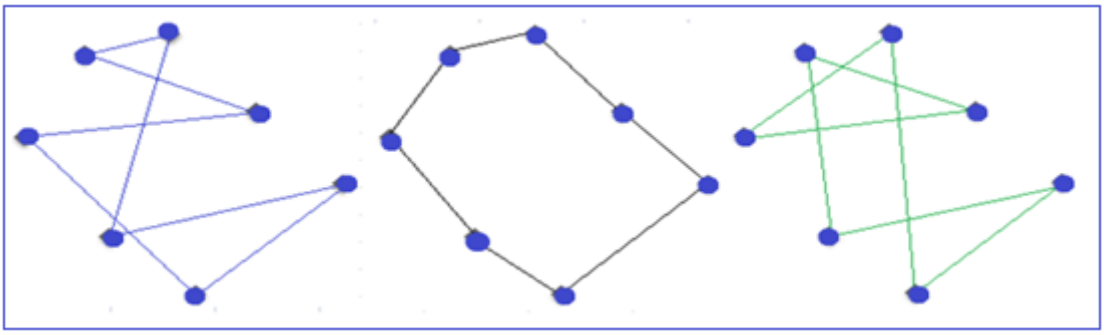

Figure 2.1: Possible Solutions for a TSP

The more cities there are, the more permutations there are to be explored, which increases computational time and effort. Therefore, the larger the instance there is no guarantee that an optimal solution will be found within reasonable time (Dantzig and Ramser, 1959). The TSP is one of the most researched problems and there is software which can guarantee optimality for up to 2 million nodes. However, the VRP problem is more complex and optimal solution can be guaranteed for up to 135 nodes for its classical form (Semet, Toth and Vigo, 2014).

The first academic publication on the Vehicle Routing Problem (VRP) is in 1959 by Dantzig and Ramser, who pioneered the problem in its classical form under the name of the "Truck Dispatching Problem". The approach is based on linear programming and it aims to find near optimal solution to the VRP. Considering a complete graph $G = (V, A)$ the VRP has a set of vertices $V = (0,...,n)$, where vertex 0 refers to the depot and vertices $(1,...,n)$ to the customers. There is a set of arcs $A = (1,...,n)$ and each arc (i, j) has an associated cost $c_{ij} \in A$, which represents the cost of travel from customer i to j. Local subtours where customers are revisited are not allowed, therefore c_{ii} = + ∞ for all $i \in V$ (Toth and Vigo, 2002), except for the global subtour starting and ending at the depot. There are 3 fundamentally different types of VRP depending on the nature of the graph G. It can be symmetric (where the distance from i to j is the same as from j to i), asymmetric (where the distance from i to j is not the same as from j to i) and Euclidean (where the distance between i and j is the Euclidean distance), which is a type of a symmetric problem.

Each customer $i, (i = 1,...,n)$ is associated with known nonnegative demand q_i, where the demand of the depot is $q_o = 0$. Each vehicle has an equal capacity Q, and it is assumed that each $q_i \leq Q$ .The minimum number of vehicles to be used in the VRP can be calculated using a

trivial lower bound by dividing the total demand of all customers by the capacity of the vehicles. Alternatively, the optimum number of vehicles can be found by solving the Bin Packing Problem (BPP), which can accommodate up to hundreds of customers to optimality. The aim of the VRP is to find k simple circuits, each corresponding to one vehicle, which will service the customers in optimal sequences and return to the depot at minimum cost. The classical features to the VRP are in place for almost all variants of the problem and are adopted from Toth and Vigo (2002). Figure 2.2 also shows a small graphical solution of the VRP.

- ✓ Each circuit starts and ends at the depot;
- ✓ The total demand of the customers on the route does not exceed the vehicle capacity C;
- ✓ Each customer vertex is visited exactly once and there are no subtours (excluding the final tour starting and ending at the depot);

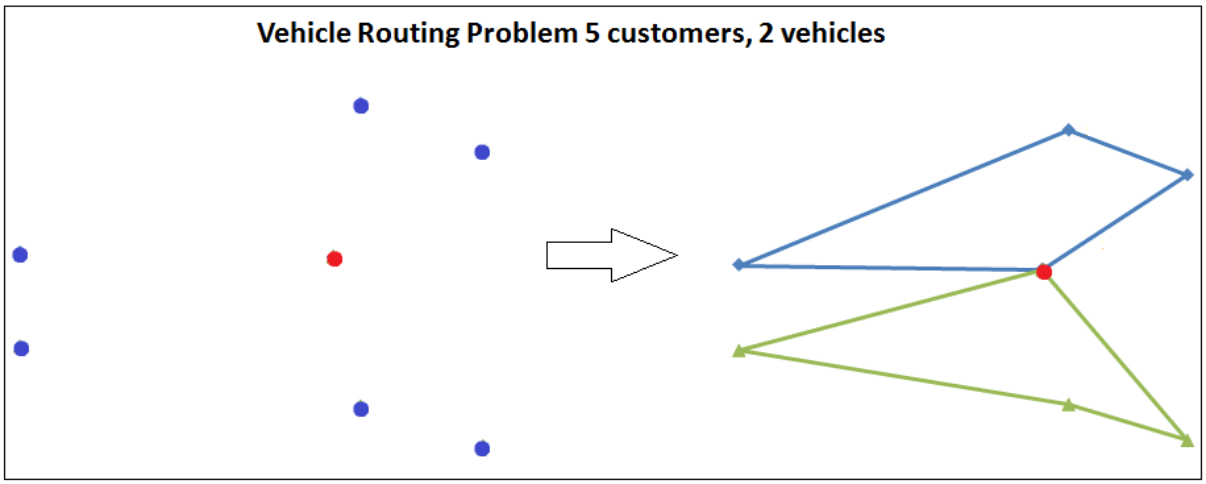

Figure 2.2: Possible solution of the CVRP

The VRP can be formulated in many ways, depending on the solution approach adopted. One can adopt a commodity flow model, a set-partitioning model, and a dynamic programming model. For further reading on the VRP formulations the reader is referred to a recent review by Laporte (2009) and Toth and Vigo (2002). The commodity flow formulation is as follows:

$$\text{Minimize } Z = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}c_{ij} \qquad (1)$$

Subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j=0,\ldots,n) \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i=0,\ldots,n) \qquad (3)$$

$$\sum_{i=1}^{n} x_{ip} - \sum_{j=1}^{n} x_{pj} = 0 \qquad (p=1,\ldots,n) \qquad (4)$$

$$\sum_{j=1}^{n} y_{ij} - \sum_{j=1}^{n} y_{ji} = \sum_{j=1}^{n} x_{ij} q_i \qquad (i=1,\ldots,n) \qquad (5)$$

$$y_{ij} \leq Q x_{ij} \qquad (i \neq j = 1,\ldots,n) \qquad (6)$$

$$y_{ij} \geq 0; \qquad (i \neq j = 1,\ldots,n); \qquad (7)$$

$$x_{ij} = \{0,1\} \qquad (i \neq j = 1,\ldots,n) \qquad (8)$$

The Objective function (1) minimizes the total cost of travel by the fleet as well as minimises the number of vehicles used. Constraints (2) and (3) are called *indegree* and *outdegree*, which impose that each customer is visited and left exactly once. Constraint (4) ensures that there are no subtours where constraints (5) and (6) ensure the capacity restriction is not violated. Constraint (7) specifies that the load should be a positive real number and constraint (8) the binary nature of the x_{ij} variable. This Mixed Integer Programming (MIP) model is commonly used as a base model for any VRP variant and any additional features, constraints or objective function components are added to the model or adjusted to the nature of the problem.

2.2 Variants of the Classical VRP

The literature is increasingly focusing on real world problems, inspired from real organisational operations within the transportation and retail sector. In reality vehicle routing is not as simple as the classical VRP problem. There are many other restrictions that apply to organisational operations. For instance, the vehicle fleet can have heterogeneous size and capacity, there may be a need not only for delivery but also for pickup of goods at customer points, the demands may be unknown, and customers may be visited only within a certain time window. Moreover, there is the human factor as well, where drivers must have

compulsory breaks, may have overtime, there may be a vehicle breakdown etc. Optimising transportation is a priority not only for academia and economy but for governments as well, and further advances in the area are highly valuable. Real routing practices inspire the research on VRP and also are the triggers for creating more complex variants of the classical VRP. This section gives some more insight into some of the most researched VRP variants and provides relevant formulations.

The Vehicle Routing Problem with Time Windows (VRPTW)

The time window is described as a window of opportunity for delivery, which has earliest allowed time for start of delivery (arrival time, a) and latest allowed time (closing time, b). Building up from the classical VRP, the VRPTW has the same constraints with an additional requirement that each customer i can be visited within a time interval $[a_i, b_i]$. Therefore, each vehicle has to deliver goods to customers within a specified service time s_i and in case of early arrival it has to wait until s_i begins. The VRPTW is usually modelled as asymmetric, because the time window and any waiting time imply that the distance between i and j may be different from j to i (Toth and Vigo, 2002). It can also be modelled as symmetric depending on the objective function and constraints. Typically there is cost c_{ij} and time t_{ij} associated with each arc. This variant is one of the most researched in the literature and there are other extended variants derived from it. There is no universally accepted way for modelling the VRPTW, however what needs to be considered in the time window restriction in addition to the CVRP. Some formulations include time windows for the depot, which govern the length of a working shift. Others also take account of service time lengths or costs. In addition to the minimum cost routing schedule, the output of the VRPTW can also include the respective timing schedule with arrival times at each customer and back at the depot. This would require some extra constraints to keep track of arrival times and maximum allowable time for the tours. Here we offer a mixed integer formulation of the VRPTW, with hard time windows where the objective function is to minimize the cost of travel and number of vehicles used, and ensure the time windows are not violated for each vehicle $k = (1,...,K)$. In addition to the Classical VRP formulation the VRPTW has an extra decision variable, which is the service time at which a vehicle begins to serve customer i and it is denoted with s_{ik}.

Minimize Z = $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K} x_{ijk} c_{ij}$ (1)

Subject to:

$\sum_{i=1}^{n}\sum_{k=1}^{K} x_{ijk} = 1$ *(j=0…n);* (2)

$\sum_{j=1}^{n}\sum_{k=1}^{K} x_{ijk} = 1$ *(i =0…n);* (3)

$\sum_{i=1}^{n} x_{ipk} - \sum_{j=1}^{n} x_{pjk} = 0$ *(k =1…K), (p =0…n);* (4)

$\sum_{i=1}^{n} y_{ij} - q_i = \sum_{i=1}^{n} y_{ji}$ *(j=1…n);* (5)

$\sum_{k=1}^{K} y_{ijk} \le Q_k \sum_{k=1}^{K} x_{ijk}$ *(i ≠ j =1…n);* (7)

$s_{ik} + t_{ij} - M(1 - x_{ijk}) \le s_{jk}$ *(i ≠ j =1…n), (k =1…K);* (8)

$a_i \le s_{ik} \le b_i$ *(i =1…n), (k =1…K);* (9)

$y_{ij}, s_{ik} \ge 0;$ *(i ≠ j =1…n);* (10)

$x_{ijk} = \{0,1\}$ *(i ≠ j =1…n);* (11)

The extra constraints to the classical VRP are (8) and (9), which make sure that the service of customer i can only start if the time window is open. There are two types of time windows, soft and hard. Soft time windows can be violated at a cost or with associated penalty, whereas hard windows cannot be violated and a vehicle can only deliver within the appropriate time. On arrival to the customer location, if the time window has not started the vehicle has to wait until it can service the client. The waiting time w_i can be calculated and taken into account in the model by the formula $w_i = \max\{0;\ a_i - e_i\}$, where e_i denotes the actual arrival time. Exact methods can solve 45 out of 56 benchmark instances introduced by Solomon (1987), including instances of 200, 400 and 1000 customers. One of the most efficient methods so far in the literature for solving the VRPTW is the memetic algorithm of Nagata et al. (2010).

The Vehicle Routing Problem with Pickup and Delivery (VRPPD)

This variant is inspired by reverse logistics. Logistics usually concerns deliveries from the manufacturer or distributor to the consumer. Reverse logistics are concerned with backward distribution at least one step back in the supply chain. This means that not only goods have to be delivered to consumers, but also picked up at consumer points and brought back to the manufacturer or depot. Many companies actually utilize reverse logistics, which makes the VRPPD a very practical problem and one of the most widely researched. The formulation derives straight from the classical VRP with some additional features. Each customer has a demand amount d_i to be delivered and amount p_i to be picked up (Nagy and Salhi, 2005). The reason why each customer has both demand and an amount for pickup is because customers may not wish to be serviced separately as it requires extra effort and inconvenience (Subramanian, 2009). This leads to the main characteristic of this variant, namely the mixed load on a given vehicle route and it is considered practical both for suppliers and consumers (Montane, 2006). In the literature there are three main types of VRPPD shown in Figure 2.3.

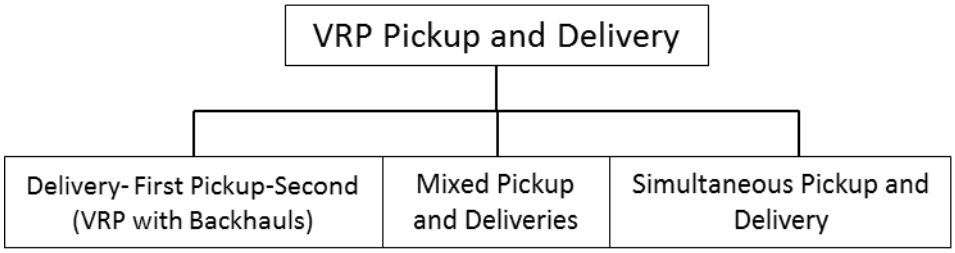

Figure 2.3: Vehicle Routing Problem with Pickup and Delivery Typology

The case of delivery-first pickup-second is also known as VRP with Backhauls (VRPB), where the deliveries must occur before the pickups. The mixed pickup and delivery VRP (MVRPPD) is when there is no order of the pickups and deliveries and they can occur at any time. However, customers have either an amount to be delivered or picked up. The simultaneous pickup and delivery also has mixed load, but customers may have both goods to be delivered and pickups. Pick up feasibility, delivery feasibility and load feasibility constraints have to be added to the classical VRP in order to model the VRPPD (Toth and Vigo, 2002). An interesting addition to the literature is the notion of strong and weak feasibility of VRPPD routes introduced by Nagy and Salhi (2005) based on load capacities and the reader is referred to the article for further information. For more comprehensive review and taxonomy on VRPPD the reader is referred to Wassan and Nagy (2014). Here a MIP formulation is presented for the VRPB. The set of

customers for the VRPB is divided into two subsets. Subset $L = (1,...,l)$ refers to the linehaul customers, which are those customers that have a given demand for goods to be delivered to them, where subset $B = (l+1,...,n)$ refers to the backhaul customers, which have a given amount of goods to be picked up.

$$\text{Minimize Z} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}c_{ij} \quad (1)$$

Subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j=0...n); \quad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i=0...n); \quad (3)$$

$$\sum_{i=1}^{n} x_{ip} - \sum_{j=1}^{n} x_{pj} = 0 \qquad (p=0...n); \quad (4)$$

$$y_{ij} \le Q x_{ij} \qquad (i \ne j = 1...n); \quad (5)$$

$$\sum_{i=0}^{L} y_{ij} = \sum_{l=0}^{n} y_{jl} + d_j \qquad (j=1...l); \quad (6)$$

$$\sum_{l=L+1}^{n} y_{jl} + y_{j0} = d_j + \sum_{i=1}^{n} y_{ij} \qquad (j=l+1...n); \quad (7)$$

$$\sum_{i=L+1}^{n} y_{i0} = \sum_{i=L+1}^{n} d_i \quad (8)$$

$$\sum_{j=1}^{L} y_{0j} = \sum_{j=1}^{L} d_j \quad (9)$$

$$y_{ij} = 0 \qquad (i=1...l),(j=0,l+1...n); \quad (10)$$

$$x_{0j} = 0 \qquad (j=l+1...n); \quad (11)$$

$$y_{ij} \ge 0; \quad (12)$$

$$x_{ij} = \{0,1\} \quad (13)$$

The extra constraints added to the CVRP (6)-(9), which govern the precedence of linehaul customers to backhaul, also make sure that the total demand for pickup and delivery is satisfied. Constraint (11) ensures that no backhaul customers can be serviced at the beginning of the delivery schedule, immediately after the depot.

The Fleet Size and Mix Vehicle routing Problem (FSMVRP)

This variant of the VRP is also very practical and common in the industry. Many companies have a heterogeneous fleet, or in other words some of the vehicles have different capacities and different associated costs (Brandao, 2011). Hoff et al. (2010) even argues that even if the vehicle fleet is homogeneous, it can become heterogeneous over time, as each vehicle may have different characteristics, depreciation, cost requirements etc. This is one of the most commonly researched variants of the VRP, as well as a part of other extended variants or rich variants proposed in the literature. Therefore, some more details of the types of FSMVRP are given in Table 2.1. Also the acronyms used to describe this variant of the VRP are not very consistent, therefore the table provides a brief description on the nature of the respective problem. The problem introduced in this thesis is also characterized with heterogeneous fleet, hence the more extensive literature review.

The formulation of the FSMVRP is the same as for the VRP with the addition of mixed fleet constraints. There are k different types of vehicles with $k = (1,...,K)$. For every $k \in K$, there are m_k available vehicles and each has capacity Q_k. Moreover, given the different size of vehicles each vehicle type has an associated fixed cost f_k. Similar to the VRP each arc has a cost of travel c_{ij}, which consists of the distance d_{ij} multiplied by the variable cost of travel a_k , for each vehicle $k = (1,...,K)$. The objective function is to minimize the cost of travel, as well as select the optimal vehicle fleet. The capacities of the corresponding vehicles must not be exceeded. Typically, when considering FSMVRP it is implied that the available fleet is unlimited. However, there are some cases in the literature which have limited (imposed) fleet availability and only a given number of vehicles from each type can be used in the optimal solution. Some of the most commonly used benchmark instances for FSMVRP are those by Golden et al. (1984), where most instances with up to 100 customers are solved to optimality

and the most commonly used instances for HFVRP with imposed fleet are introduced by Taillard (1999).

Table 2.1: Types of FSMVRP

Description	Author	Acronym	Solution Method
VRP with different vehicle capacity and variable cost, unlimited fleet	Salhi (1992)	VFM	Constructive Heuristic
	Choi and Tcha (2007)	VFM	Column Generation and Branch and Bound
	Subramanian et al. (2012)	FSMV	Iterated Local Search, Variable Neighbourhood Descent
VRP with different vehicle capacity, fixed cost and variable cost, unlimited fleet	Golden et al. (1984)	FSMVRP	Savings Heuristic
	Desrochers (1991)	FSMVRP	Savings Heuristic with route fusion
	Gendreau (1999)	HVRP	Genius, Tabu Search embedded in Adaptive Memory Procedure
	Renaud (2002)	FSMVRP	Sweep-based algorithm
	Choi and Tcha (2007)	HVRP	LP relaxation solved by column generation
	Liu (2009)	FSMVRP	Genetic Algorithm
	Brandao (2011)	FSMVRP	Tabu Search based algorithm
	Imran et al. (2009)	HFVRP	Variable Neighbourhood based Heuristic
	Subramanian et al. (2012)	FSMVF	Iterated Local Search, Variable Neighbourhood Descent
VRP with different vehicle capacity, fixed cost and unlimited fleet (without variable cost)	Subramanian et al. (2012)	FSMF	Iterated Local Search, Variable Neighbourhood Descent
VRP with limited fleet, with fixed and variable cost	Taillard (1999)	VRPHE	Adaptive Memory Procedure and Heuristic Column Generation
	Li et al. (2007)	HVRP	Record-to-Record Travel algorithm
	Subramanian et al. (2012)	HVRPFV	Iterated Local Search, Variable Neighbourhood Descent
VRP with limited fleet with variable cost, but without fixed cost	Tarantilis et al. (2003)	HFFVRP	List Based threshold accepting Heuristic
	Subramanian et al. (2012)	HVRPV	Iterated Local Search, Variable Neighbourhood Descent

A Mixed Integer formulation for the FSMVRP with fixed and variable cost is given below. It is similar to the formulation of the CVRP, but the decision variables are three-index, so as to accommodate for the fact that the different vehicles k have different characteristics.

$$\text{Minimize Z} = \sum_{k=1}^{K} f_k \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K} x_{ijk} d_{ij} a_k \qquad (1)$$

Subject to:

$$\sum_{i=1}^{n}\sum_{k=1}^{K} x_{ijk} = 1 \qquad (j=0\ldots n); \qquad (2)$$

$$\sum_{j=1}^{n}\sum_{k=1}^{K} x_{ijk} = 1 \qquad (i=0\ldots n); \qquad (3)$$

$$\sum_{i=1}^{n} x_{ipk} - \sum_{j=1}^{n} x_{pjk} = 0 \qquad (k=1\ldots K),\ (p=0\ldots n); \qquad (4)$$

$$\sum_{i=1}^{n} y_{ij} - q_i = \sum_{i=1}^{n} y_{ji} \qquad (j=1\ldots n); \qquad (5)$$

$$\sum_{k=1}^{K} y_{ijk} \le Q_k \sum_{k=1}^{K} x_{ijk} \qquad (i \ne j=1\ldots n); \qquad (7)$$

$$y_{ijk} \ge 0; \qquad (i \ne j=1\ldots n); \qquad (8)$$

$$x_{ijk} = \{0,1\} \qquad (i \ne j=1\ldots n); \qquad (9)$$

The School Bus Routing Problem (SBRP)

SBRP also falls into the larger class of VRP. It was firstly introduced by Newton and Thomas (1969) but it has received much less attention in the literature than the VRP. The SBRP is similar to the real life vehicle routing problems, because there is no general or standard approach to solving the SBRP. Most of the publications in the literature are based around the different versions of the SBRP which arise as a result of various real life constraints and assumptions. There is no dominant approach to studying the SBRP and most of the solution methods proposed are very problem specific (Li and Fu, 2002). The SBRP is somehow all-encompassing, containing smaller sub-problems. Typically in the literature the SBRP is not

addressed as an all-encompassing problem, but researchers focus on one of its sub-problems alone or in combination (Park and Kim, 2010). According to the decomposition of Desrosiers et al. (1980) the SBRP has 5 sub-problems.

✓ *Data Preparation* consists of preparing the data set, specifying the Origin-Destination (OD) matrix, student homes, bus stops, depot and school locations.

✓ *Bus stop selection* consists of deciding where to locate the bus stops and assigning students to bus stops. This step can be solved on its own as a location problem. Usually a maximum walking distance constraint for each student to a bus stop is imposed. An interesting version of the SBRP is when the walking distance is larger and students can walk to more than one stop. Then the decision on which student should walk to which stop gets incorporated into the SBRP model.

✓ *Bus Route Generation* is the step which is most similar to the VRP. It consists of generating the vehicle routes from the point of origin (depot) to the school, visiting all bus stops. This step can be solved as a VRP, or in combination with the previous step as a location-routing problem.

✓ *School Bell Adjustment* and *Route Scheduling* sub-problems arise when multiple schools are considered. The buses used to transport the students to schools are not typically owned by the schools, but operated by regional board of education. Therefore, it is common that multiple schools use the same bus fleet, hence the need for better coordination and scheduling with regards to the school opening and closing hours. This step has many real-life considerations and different SBRPs can arise based on the different assumptions and constraint imposed.

Because of the different approaches one can take when solving a SBRP, there is no standard definition of the problem. However, a VRP oriented definition can be offered for the SBRP. The SBRP consists of planning an efficient routing schedule for a fleet of buses, where each bus leaves the point of origin visits all bus stops and collect all students, ensuring the maximum bus capacity is not violated, and delivers the students to the designated school (Park and Kim, 2010). Additional constraints such as maximum riding time, maximum walking

distance and school time window can apply. The objective function of the SBRP can be minimizing distance, travel time, student riding time, numbers of buses used or it can be multi-objective. There is no common formulation of the SBRP, but we refer to Shittekat et al. (2013) for a formulation representing 2 of the SBRP sub-problems, namely student assignment to stops and route generation.

Figure 2.4 shows an example of the SBRP, where students are assigned to buses according to a maximum walking distance constraint. This means that there is a reasonable walking time for each student to reach a bus stop. The figure shows a simple assignment of students (the small black dots on the figure) to stops and the generated route for the School Bus after the assignment has taken place. The demand at each stop equals the number of students assigned to that stop, where the Bus capacity is 25. In the case portrayed in Figure 2.4 one bus can accommodate the entire demand, hence the single route.

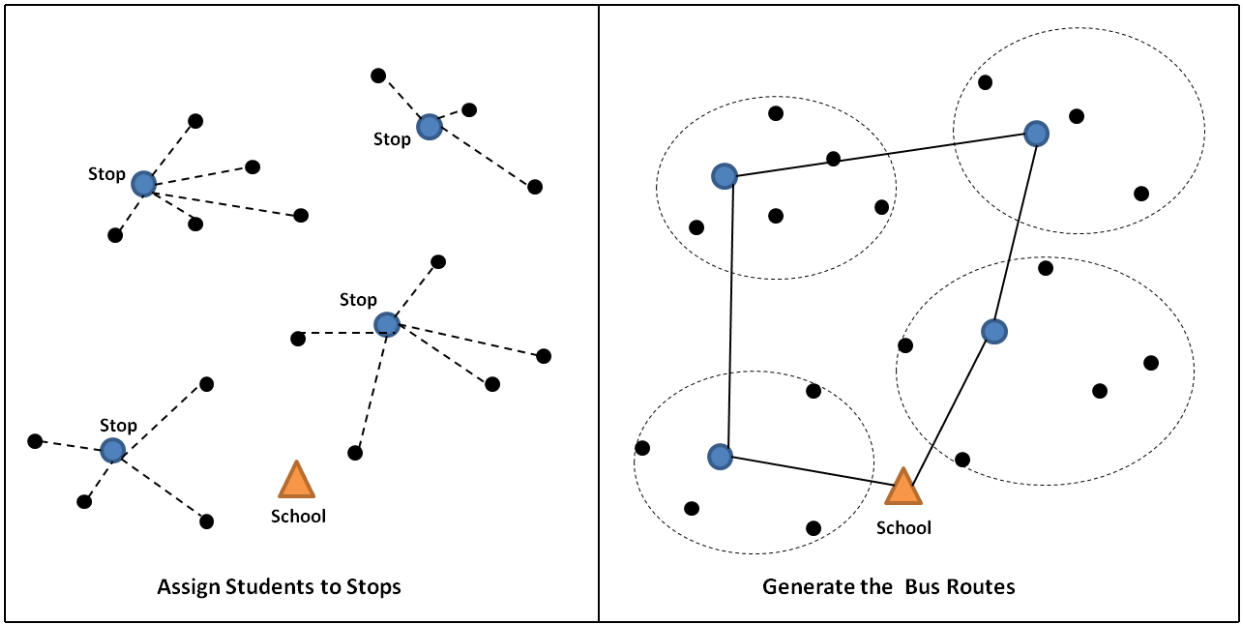

Figure 2.4: The School Bus Routing Problem

The problem becomes more interesting and arguably 'heterogeneous', when the walking distance for students to reach stops is larger. This means that a student can walk to more than one of the available stops, and a decision needs to be made which is the best stop for the student to walk to in order to minimize cost. This results in an overlapping assignment and it is portrayed in Figure 2.5. In this case, the students marked in red are those students who can be assigned to multiple stops. This makes the SBRP more interesting and difficult to solve,

especially if more stops are available (a larger datasets). We address this problem in Chapter 7 and test our proposed methodology on both overlapping and non-overlapping versions of the SBRP.

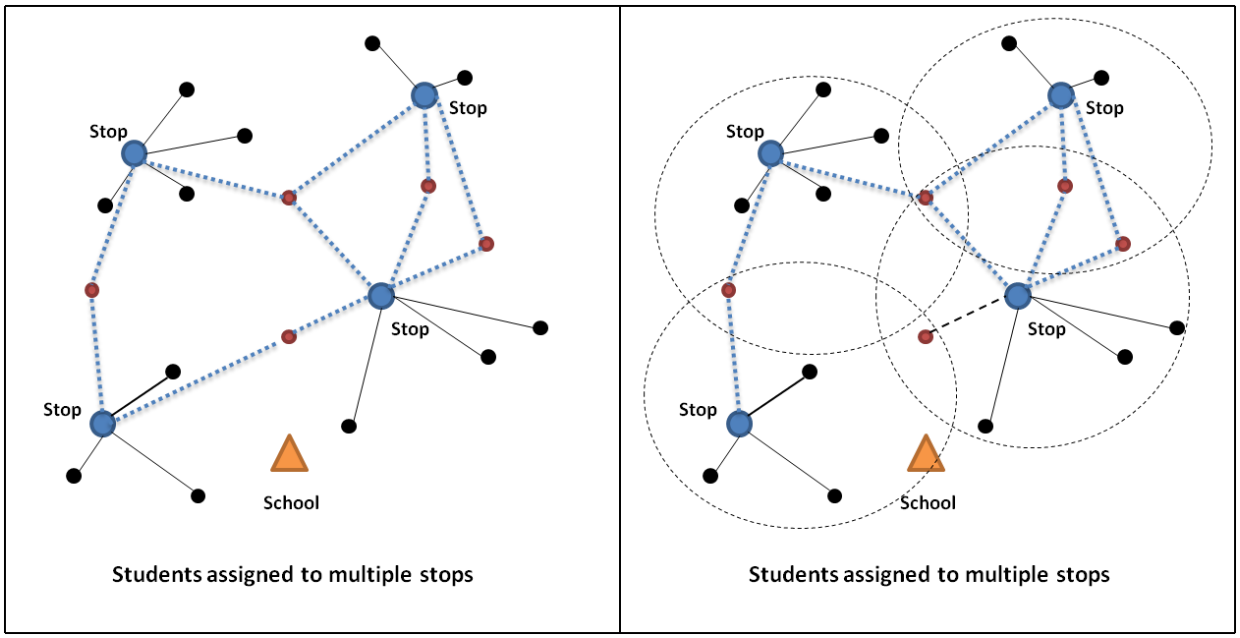

Figure 2.5: SBRP with overlapping radius of Bus Stops

The Multi Depot Vehicle Routing Problem (MDVRP)

The MDVRP is also an extension of the VRP and it is inspired by various industry cases, where companies own more than one depot, which cover different geographical area with separate vehicle fleet (Renaud and Laporte, 1996). It is very common in a real business environment that large organisations have more than one depot, such as Tesco's, Sainsbury's, Calor Gas etc. It is believed that determining more effective and cost efficient ways to assign vehicles to depots can result in great savings for companies. The problem can be represented as either a directed or an undirected graph, where $G=(V,E)$, where V is partitioned into two subsets, one for the set of customers $V_c=(v_1,...,v_n)$ and $V_d=(v_{n+1},...,v_{n+m})$ for the set of depots where not all depots are necessarily used. All vehicles must return to the depot they were based at and a constraint must be present, which does not allow a depot to be treated as a customer vertex and be part of the route of vehicles based at different depots. The MDVRP can be solved in two stages. Firstly, all vehicles are assigned to depots, according to predefined criteria and then each one is solved as a separate VRP. Alternatively, an algorithm can be used

to both assign vehicles to depots and customers to vehicles. Optimal solutions are reported for instances with up to 151 customers and 12 depots (Contardo and Martinelli, 2014). The MDVRP and the periodic VRP (PVRP) are closely related and some solution methods are formulated that are applicable across the two variants. PVRP arises when customers require service on multiple days within a time period (Gulczynski, 2011). MDVRP reduces to PVRP, when there is only one depot and PVRP reduces to MDVRP when customers require only one visit (Vidal, 2011).

Vehicle Routing Problems with Probabilistic Elements

The Stochastic VRP (SVRP) arises when there is a stochastic element in the VRP formulation. SVRP is very difficult to solve to optimality even for small instances of up to 30 customers. The most common versions of SVRP are static and stochastic, where some elements of the VRP problem are random variables, such as customer demand, service time, travel time etc. The most researched stochastic element in the literature is demand, which becomes known once the vehicle has arrived at the customer point. This type of SVRP is solved to optimality for 70 customers (Gendreau, Jabali and Rei, 2014). Gendreau (1996) found that some fundamental properties of the VRP and TSP do not hold for SVRP. Firstly, the shortest path which covers all vertices never intersects itself (it is a simple polygon), secondly customers are visited in the same order as they appear in the convex hull and finally any segment of the optimal tour is also optimal. For further reading on the SVRP a book chapter by Gendreau, Jabali and Rei in Toth and Vigo (2014) is recommended. There are also dynamic VRPs (DVRP) and stochastic VRPs, where information becomes available dynamically after the routing schedule has started. This leads to the need of route re-optimisation techniques. The two most successful methods for solving a dynamic VRP, which account for re-optimisation in different ways, are Chance Constrained Stochastic Programming and Stochastic Programming with Recourse. The integer L-shaped method and Branch and Cut are also commonly used in the literature to solve SVRP (Dror, 1989).

Other Variants

The VRP literature is very rich and there are many different variants that are proposed by academics each year, some based on extensions of the main variants or creating new ones. Some of those are worth reviewing, because they are applicable in practice even though they

are less researched. The Two Echelon Vehicle routing problem (2E_VRP) considers a multilevel distribution system inspired by city logistics, where vehicles start at a depot, go to the nearest intermediate facility (satellite) and from there are routed to various customer locations. The purpose is to minimize pollution and congestion in big cities and avoid sending large trucks into the city. Instances with 21 customers are solved to optimality so far (Hemmelmayr, 2012). The Cumulative Vehicle Routing Problem (CumVRP) is motivated by customer satisfaction and relations. Its objective function is to minimize arrival times as opposed to cost. This is perhaps the most consumer centric variant in the VRP family, which incorporates issues like just in time service, equity and fairness (Ngueven, 2010). The Multi-compartment Vehicle Routing Problem (MCVRP) arises when *m* products must be delivered to customers by *k* vehicles, which all have different compartments for each product. This variant considers the benefits of co-transportation as opposed to un-partitioned trucks and independent distribution. Some authors found that co-distribution leads to shorter routes (Muyldermas, 2010). The Open Vehicle Routing Problem (OVRP) occurs when given fleet of vehicles does not have to end the tour at the depot, but at the last customer. This variant is highly applicable for leased vehicles or any fleet that is not an asset of the company. The solution to the OVRP then becomes a Hamiltonian path, not a cycle. Last, but not least is the Truck and Trailer Vehicle Routing Problem (TTRP) introduced by Chao (2002). It is inspired by the ability to access customer locations in difficult areas. TTRP consist of finding shortest routes to serve set of customers either by the full vehicle (truck and trailer) or truck only.

Mixed Variants

It is rare in the industry that a fleet optimisation will be only constrained by time window alone, or heterogeneous fleet alone. Usually a combination of requirements and constraints is present, which need to be considered simultaneously. Therefore, academics become more creative and practical and introduce what can be referred to as '*mixed*' or '*extended*' variants to the literature. An example of a mixed variant is a combination between VRPTW and VRPPD researched by Bent (2006). Thanghai (1996) aimed to solve the VRPBTW, which includes time window and backhauling. Salhi et al. (2014) addresses a mixed variant between MDVRP and FSMVRP. There are many other mixed variants, which exist in the literature and cannot be exhaustively listed but are following the same principle. Extensions to the main variants are also common in the literature such as the VRPmiTW, which is VRPTW with multiple

independent time windows (Doerner, 2008), and the DVRP which is VRP with time dependent travel times (Haghani, 2005). Following an in-depth literature search, it can be stated that the VRPTW VRPPD and FSMVRP are the most common problems that have been mixed with other variants or extended with additional requirements and constraints. Introducing a mixed or extended variant of the VRP can also aid algorithmic advancements. One of the trends in the literature is to design more generalizable algorithms, which can be applied to a range of VRPs. Having a mixed or extended variant gives the opportunity to design a method which is applicable to the variants separately and it could be very beneficial to test the results of those algorithms on the literature benchmarks of the separate problems, or introduce new instances with the combined characteristics. Figure 2.6 shows some of the main elements based on which the VRP variants are differentiated. It is not an exhaustive list, but provides an idea of the different broad categories of VRP problems, which arise according to the elements present in the problem. The inspiration of this classification is the real life VRPs, which are differentiable based on their elements. A similar rationale can be adopted for the problems of the VRP family.

Real life (Rich) variants

The notion for real life problems is not new, because as early as 1993 there are papers in the literature, which are based on case studies or explicitly state that they are researching a real life problem. For example Semet and Taillard (1993) tackle a real-life VRP inspired by the grocery industry in Switzerland. Real life operations have been an inspiration for modelling in OR for the past 20 years. However, it was not until 2006 when real life problems were introduced as a class of the VRP. After the introduction of the Livestock Collection problem (LCP) (Gribkovskaia et al., 2006), the authors formally categorize real life problems under the term ‘rich’ Vehicle Routing Problems and provide a loose definition of what a rich VRP is. It cannot be stated with certainty that the OR society has accepted the term as descriptive of all VRP problems which have real life constraints and features. However, there are some definitions in the literature of what constitutes a rich VRP. Hastle et al. (2006) states that a rich VRP includes aspects that are essential to the routing practice in real life and the richness of the problems can stem from many elements of the routing practice such as drivers, fleet, order types, depots, tours etc.

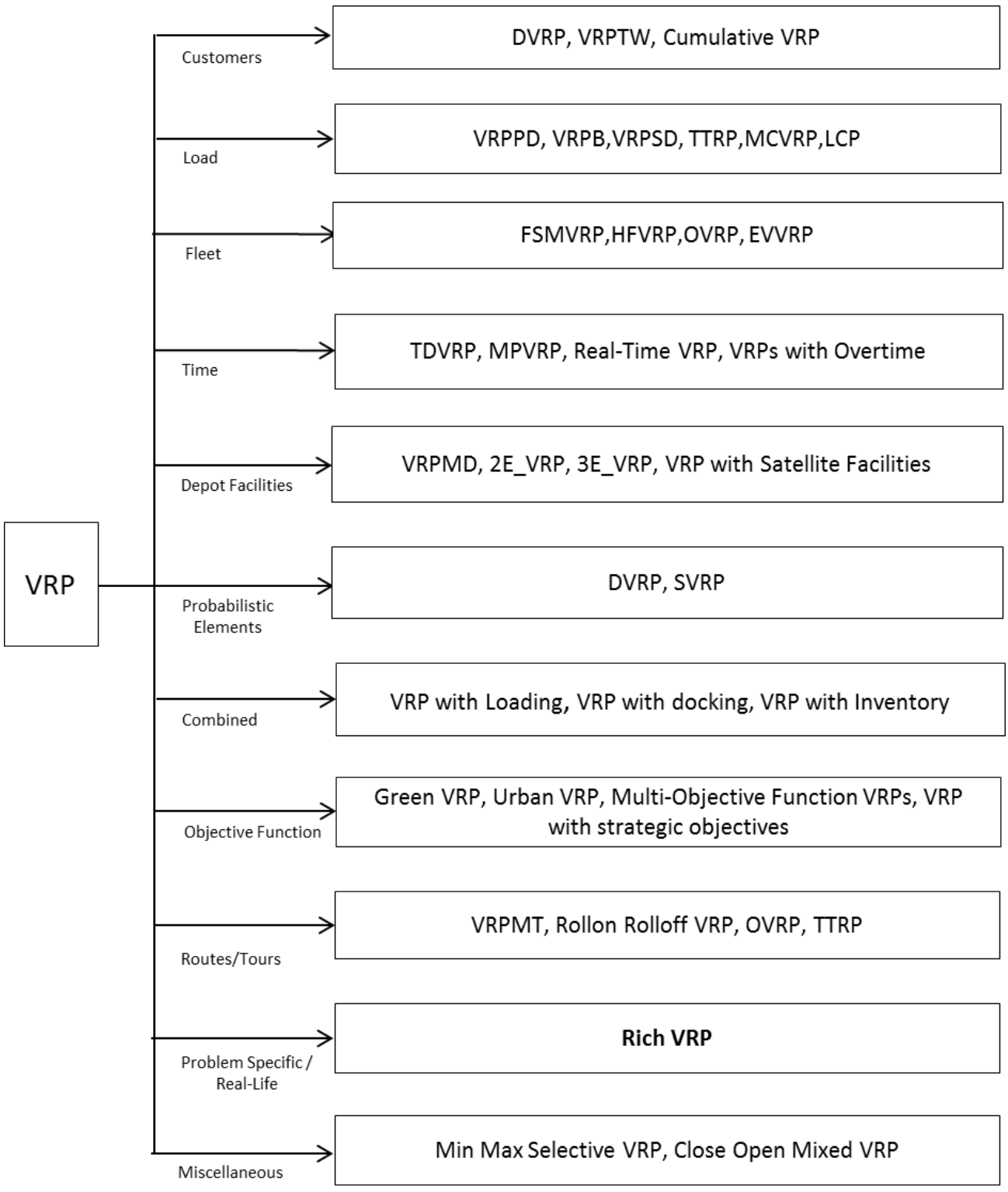

Figure 2.6: Classification of VRPs based on their elements

In addition, Rieck (2010) states that rich VRPs include different constraints or different objective functions. Vidal et al. (2013, 2014) referred to the rich VRPs as Multi-Attribute VRPs (MAVRPs), which typically arise from real life situations. However, in their classification, any deviation from the Classical VRP is considered to be MAVRP and the proposed algorithm is an all-encompassing rich solver, rather than a single algorithm applicable across variants. Goel and Gruhn (2008) refer to the real life problem they propose as General VRP, whereas other authors do not adopt any formal classification terminology when dealing with these types of problems and simply refer to them as real life routing problems.

If we look at another class of VRP problems, such as the Fleet Size and Mix VRP or the VRP with time window, we can find papers that summarize the most important aspects of the respective problems. Usually a standard formulation of the problem is provided alongside best known solutions on publically available benchmark problem instances, and the best performing methods are indicated. However, RVRPs are very diverse and such a well-rounded summary could be very difficult to achieve. One of the main reasons for this is the loose definition provided for RVRPs. Another reason is that usually a RVRP is introduced only once and it is not revisited by another author under the same form with the same characteristics. Moreover, the nature of RVRPs changed over time and some of the problems referred to as rich in the past, may not necessarily qualify as such today. RVRPs cannot be standardised, because their relevance and contribution stem from the diversity of real life routing practices. One of the first papers which provides a more comprehensive literature review of RVRPs as well as a theoretical framework and definition of RVRPs is Lahyani et al. (2015). They investigate in detail 41 publications since 2006, when the rich VRPs became a class of the VRP domain. The definition they provide is similar to the one by Hastle et al. (2006) because it expresses the nature of RVRPs in terms of the characteristics/elements they possess. A RVRP must have a sufficient number of real life routing elements in order to qualify as rich. If the RVRP is defined in terms of physical characteristics, then it should have at least 9 elements additional to the classical VRP. If the problem is defined from a strategic or a tactical view point, then it should have at least 6 additional elements. Rich VRP variants are a term that is becoming more popular in the literature, but there is no single definition of what exactly constitutes a rich VRP variant, which is accepted across the OR community. Regardless of the different acronyms used to describe RVRPs, there is some consistency grounded in the

definitions of RVRPs. A RVRP should contain relevant aspects of a real life routing problem, which can aid the decision making process in practice and be significantly differentiable from other VRPs on the merits of their real life routing characteristics. In addition, in this research it is suggested that when researching a real life VRP there should be some practical implication or recommendations for improving routing practice.

Rich problems are extremely diverse and may incorporate features of real life operations ranging from consumer requirements to driver break times. The classical VRP can be enriched and extended in many novel ways which can be valuable for academia and industry. The quest of making VRP problems more realistic and bringing them closer to industry realities provides an opportunity for being creative and adding interesting aspects of routing to the existing variants. Some of them cannot specifically be described as extensions of a certain variant, but simply reflect the nature of real life operations and the different ways a company's operations can be constrained. For instance Bortfeld (2010) includes the presence of three-dimensional loading constraints, which makes the problem a mix of routing and loading decisions. Ren (2010) discusses the multishift VRP with overtime, which emphasizes the drivers considerations and constraints, whereas Moon (2012) analyses the VRP with deadlines and travel/demand time, which is consumer centred. Battarra (2009) aims to solve the Multiple Minimum Trip VRP (MMTVRP), which is also industry inspired and assumes that one vehicle can be assigned to more than one route. Valle (2011) considers the min-max selective VRP, where not all customers need to be visited and those that are not visited must be close enough to those that are visited. All those real life aspects that can be added to a VRP problem and provide that extra richness, contribute to making real life VRP research very flexible and a fertile area for ideas and novel developments.

Most of the published papers emphasize the real life constraints of the RVRPs and somehow place the emphasis on the problem nature. This is where the proposed definitions of RVRP and taxonomy are a useful guide for future research. However, there is an issue which has not been much addressed in the literature, which relates to the solution methodologies. The main contributions to the literature of VRP come from the nature of the problem or from the proposed methodology. Typically, a contribution to the literature can be made either by introducing a new interesting problem, which is different from previous research or by introducing a new methodology, which is powerful and relevant in terms of performance or

novelty. When it comes to RVRPs, however, given the fact that they are so different from other variants and from each other, it poses a challenge for proving algorithmic efficiency compared to other methods in the literature. There are some common practices that authors use when dealing with RVRPs. An exact method can be used for RVRPs, which guarantees optimal solutions. For instance Dayarian et al. (2015) use a Column Generation (CG) method for a real life case inspired by milk collection, whereas Oppen (2010) applied CG to the livestock collection problem. However, heuristic methodologies are more common, because they can tackle larger sized problems. Using a heuristic method, there is no guarantee that the obtained solutions are in fact optimal or of a good enough quality. Some papers on RVRPs are case studies, based on real company data. Using a real dataset, which is also used by the company in question, is a good way of showing algorithmic efficiency by directly comparing the results from the study to the actual practices of the company. This way the impact of the study can be measured and recommendations can be made on how to improve the routing practice. In other cases, the datasets used for the RVRPs can be either randomly generated or adapted from literature benchmarks. In these cases the issue of comparability is more significant. Therefore, it is important that some form of comparison or test of algorithmic merits has to be adopted. It has to be noted here, that some papers propose algorithms for RVRPs which aim to be all-encompassing and qualify as rich solvers, rather than single methodologies for one particular problem. An example of this is the Genetic Algorithm based rich solver proposed by Vidal et al. (2014). In their paper many aspects of the VRP, including rich elements can be addressed by the proposed methodology, hence the results can be tested on various literature benchmarks. However, the purpose of the method is to be all-encompassing and able to accommodate VRPs across the different variant classes. This means that in the cases where algorithms have a degree of generalizability the issue of comparison can be overcome. This research proposes two ways one can address the issue of algorithmic comparison and efficiency, especially when heuristic methods are used to address a RVRP.

First, it is beneficial if a RVRP is solved to optimality (alongside a heuristic solution), where optimal solutions or lower/upper bounds are reported. This could aid the discussion of algorithmic efficiency and be a guideline for the performance of the heuristic algorithm. There are some very powerful exact methods for solving larger instances to optimality, such as CG. However, if the methodological effort of the research in question is on designing powerful

heuristic algorithms, and not exact, it could be useful to propose a Mixed Integer formulation (where feasible) and solve as many instances to optimality using solvers such as CPLEX. The new version 12.4 is more powerful and employs Branch and Bound (BB), as well as valid cuts and it is able to achieve some good lower/upper bounds, on larger problem sizes, previously not so well accommodated.

Second, a specific approach with dealing with RVRPs can also be introduced, where algorithmic efficiency can be tested on publically available benchmarks. This is not to state that a method which is created to address a RVRP must be able to outperform or match the results of methods specifically created for a main variant of the VRP. However, the smaller the gap the stronger the evidence that the proposed method for RVRP has the potential for generating good solutions.

The approach we propose for dealing with RVRPs is to design them in a way that the RVRP can be reduced to a well-researched variant of the VRP. We refer to this approach as 'Standardise-First Customise-Second'. It can be argued that having a main variant of the VRP, such as the VRPTW or FSMVRP as a base of a RVRP is not an unreasonable assumption. The most researched variants of VRP are so popular, because they occur most frequently in industry settings. Even if the proposed RVRP is very different from other existing research and cannot be generalized to a main variant, it should typically be generalizable to the classical VRP. Some exceptions may be problems which maximize the objective function, where the aim is to maximize customer satisfaction (i.e. Ambulances, Red Cross Coverage of territory) or some other strategic or operational objective. Embedding a well-researched variant of the VRP into RVRPs is the Standardizing Stage of the approach. The reason for this is that the solution methodology designed for the RVRP can also be tested on literature benchmarks for a main variant of the VRP. This could provide a better perspective on how powerful and adaptive the solution method is and be directly comparable with other methods in the literature. The stage of customization will provide the problem with the richness a RVRP needs to possess in order to qualify as rich. The customization stage would build up on the standardization stage, which will be tailored to reflect the specifics of the problem that is being researched and portray a relevant picture of distribution practices in a real industry setting.

The standardization stage is useful for testing against benchmarks and making sure that the algorithms are adequate and have a good level of performance (if not superior), where the customisation stage provides the "richness" of the problem and makes it adjusted to real life operations. The rich variants should provide that notion of customisation, because each and every company has different considerations when designing distribution routes, based on its strategic, marketing and corporate goals. Moreover, the nature of some distribution systems may require special considerations that are unusual and occur less frequently such as distribution after natural disasters.

Evaluation of the literature on the VRP Variants

There is no doubt that the literature on VRP is very rich and covers many aspects of business operations across different industries. The advances made in the areas of VRPTW, VRPPD and the other main variants is highly valuable and can result in savings for companies and minimizing costs. One of the main trends in the literature of VRP is the quest for more realistic modelling. Having this in mind, this section provides a real routing inspired critical perspective on some of the modelling practices of VRP problems. This is not done with the aim of criticising the existing research on the problems considered, but to offer a real life motivated perspective on some issues which became apparent throughout the literature search.

The VRPTW usually implies that every customer has requested a time window for delivery, which is either hard or soft. In reality, large companies not only do not accept such request often, but they are the once that provide the time windows (i.e. Furniture deliveries). Real operations are influenced by companies bargaining power, whether they are Business-to-Customer (B2C) or Business-to-Business (B2B) and the construction of routes is much more flexible because consumers have low bargaining power to specify time request. SDVRP is mostly feasible when the demand quantity is very large (Fizzell and Griffin, 1995) and the product being delivered can actually be split. OVRP does not take account of other costs for leasing vehicles and transportation costs to the depot.

The fact that most problems use distance, time and cost interchangeably in the objective function (Toth and Vigo, 2002) is inaccurate, since factors like traffic, congestion and driver break times are not included. It could even be argued that using distance in the objective function is much more accurate than cost, because cost is very complex to calculate and

things like *opportunity cost* are very difficult to incorporate. Many formulations also impose maximum time or length constraints for each route, which can be misleading since using staff to do overtime and extra deliveries happens on a daily basis. It is quite common in the literature that the objective function also decides the optimal number of vehicles. This is very useful when a company start-up is considered, downsizing or restructuring, but it is not greatly relevant to fully functioning company, which has its own fleet. The reason for this is that cutting on vehicle numbers also implies redundancy or use of temporary staff in peak periods and this can be damaging for the company for their long term strategic horizons and Corporate and Social Responsibility.

The RVRP class of problems are much more open to criticism, because of their diversity. Recently a more comprehensive definition for RVRP was proposed by Lahyani et al. (2015), which is a useful guide for research, but it has two aspects which could be argued to be problematic. Firstly, the definition proposed is retrospective. This means the definition is not generic, but formulated on the basis of what already exists. It could be argued that this could limit creativity and place taxonomical boundaries on future research. Secondly, the number of elements the authors propose to be sufficient for a RVRP are extracted from pure VRP problems and combined VRP problems. What is meant by a combined VRP is that the problem consists of a VRP and another CO problem such as Loading or Inventory optimization. Combined problems possess more elements by definition even in their classical forms, because they combine two or more CO problems. Therefore, it can be argued that the sufficient elements for RVRP in the proposed definitions are artificially inflated, because the combined problems are considered. Perhaps, if a review of the routing problems alone was considered, the number of elements that are sufficient to make a problem rich would be different.

Designing more sophisticated problems, which are relevant in real setting, is one of the literature trends since the new millennium and any current research should offer enough real-life features in order to claim the ‘rich’ title. However, the argument on how much richness a problem should have in order to be categorised as rich can go beyond the number of features required in the nature of the problem. The fact that RVRPs are difficult to categorise and express with compact acronyms, can leave one’s interpretation of RVRPs open to criticism. At the same time however, it is also what makes the RVRP domain so interesting and such a

fruitful area for future research. It provides the opportunity to minimize the gap between academia and real life operations and introduce some novel characteristics of routing, as well as an opportunity for extending existing methodologies.

Addressing a RVRP with a heuristic method is another area that may leave a research open to criticism. Heuristic methods are generally problem specific and this is one of their main limitations. Coupled up with the problem-specific nature of the RVRPs it poses a double threat and the need for methodological justification. This is the main reason why a way to show algorithmic efficiency is needed in the RVRP domain. This would strengthen the contribution of RVRP research and overcome any perceptions that RVRP research is peripheral or weaker than the mainstream, which is generally focused around benchmarking and algorithmic superiority. This is one of the main arguments of this research, hence the propositions for overcoming the issue of methodological comparability. In fact, researching a real life variant can act as an inspiration to adapt and adjust well known methods and provide an opportunity to extend those methods to other problems and ideally make them more generalizable across VRP problems.

2.3 Solution Methods for the VRP

There are many exact and heuristic methods that can be used to solve the VRP. Each of them has its advantages and limitations. This section aims to describe the most popular and efficient solution methods for the VRP and analyse their capability to solve the VRP and its variants. The first general class of solution methods are the exact methods, which can solve VRP problems with guaranteed optimality. Exact methods are typically exhaustive search methods, where each combination of decision variables is explored before the best solution is found. They are usually very computationally expensive, because of their nature and solve relatively small problem instances to optimality. The second class of solution methods are the heuristic methods, which do not guarantee an optimal solution to the problem, but a good approximate solution within reasonable computational time. Sometimes good heuristic methods can match the optimal solution (assuming it is known). It is not possible to provide a full-rounded summary of all the solution methods in the literature. Each variant of the VRP has its own best performing methods and it cannot be stated that one single method is

superior to another. However, some of them are more powerful than others and more commonly used, and are responsible for some of the best known solutions in the VRP domain.

2.3.1 Exact methods for the VRP

As previously discussed the VRP and its variants do not have one universal way of formulating the problem. Depending on the purpose of the problem there are many different constraints that can be added to the classical VRP model and can be presented in novel ways. There is two-index vehicle flow formulation, three-index vehicle flow formulation, two-commodity flow formulation (Baldacci, 2004) and Set Partitioning formulation (Balinski, 1964). The lower bounds for the VRPs (assuming a minimization objective function) are usually computed using cutting planes, column generation and in some cases linear programming relaxations. Many valid inequalities are introduced to the literature to complement existing formulations and make them more robust for generating solutions. Extensive computational results on the same VRP instances available in the literature have been reported based on Branch and Cut (BC), Branch and Cut and Price (BCP), and Set Partitioning (SP) methodologies. Moreover, they have been proven to be most effective for generating lower bounds and exact solutions in the VRP family of problems (Baldacci, 2010). This research only offers a brief review of the exact methods for VRPs, because the main methodology here is heuristic based. For further information on mathematical formulations for the VRP readers are referred to a recent review by Baldacci (2010). However, mixed integer programming is still one of the most popular approaches to obtaining optimal solutions and it is aided by the advances and new versions of available software such as Cplex. The new version provides for the use BB and BC techniques which lead to obtaining better lower bounds, tighter gaps, and optimal solutions where possible.

- ✓ Branch and Cut

BC is widely used in the literature by many authors for obtaining exact solutions to problems of the VRP family and obtaining better lower bounds. Given an integer solution or LP relaxation, the purpose of BC methods is to design an algorithm that effectively separates a fraction of the obtained solution convex hull using valid inequalities. The underlying running algorithm is BB and the cutting planes are used to tighten the solution. After applying BB the cutting planes are used to design valid inequalities to be applied to the original formulation,

which aims to make the solution less fractional and enhance its speed and quality. The valid inequalities should be of polynomial size in order to improve the time it takes to solve a problem, since LP and BB have exponential complexity. The way the algorithm works is by starting from an optimal solution generated by LP relaxation and branching on the different variables by forcing them to take integer values and then re-solving the problem. The branching continues until all variables take integer values. The cuts that can be added during the process of branching can either be global cuts (valid for all feasible integer solutions) or local cuts (valid only for the current branch solution). The process of branching is not random; there are strategies that can be applied depending on the nature of the problem and the value of the variables. The initial choice may be to start with the most infeasible value of a variable (most fractional) or start with the variable that brings the best objective function improvement and consequently continue with depth-first, or breadth-first tree search approach. For further reading one can refer to Baldacci (2010).

✓ Branch and Cut and Price

BCP is an extension of BC which adopts similar steps and principles, but both cuts and variables are generated in a dynamic way, which allows for more flexibility and solving larger instances of a given problem. BCP is one of the most successful exact method for solving VRP and its variants. Branching cutting and pricing is combined in this method. Pricing comes from the idea of Column Generation. At the start of the algorithm through the search tree, columns are excluded from the search so as to reduce the problem size and then are added later in the search if necessary. The idea behind this method is that some variables are non-basic and have a value of 0 in the optimal solution so they can be excluded from the search process. Then a pricing problem is solved which aims to find a column with negative reduced cost. There is no need to find the most negative reduced cost, therefore simple heuristics can be used for this purpose. If no columns can be added to the solution then branch and cut takes place. There is available software which uses BCP for solving CO problems, such as ABACUS, CPLEX, CONCORDE, SYMPHONY (branchandcut.org).

✓ Set Partitioning

Based on recently reported results for exact method performance on common benchmarks, Baldacci (2010) reported that SP methods outperform BCP and BC on the classical VRP,

especially when combined with column generation and additional cuts. SP methods are quite general and flexible in terms of their ability to incorporate further constraints such as time windows and fleet heterogeneity. Therefore it is also one of the most useful exact methods for solving VRP variants and potentially for addressing rich VRPs. The objective function is to cover each customer exactly once by a given vehicle route, where the original classical VRP constraints apply.

2.3.2. Heuristic Methods for the VRP

Heuristic methods date back to 1950 and they have been evolving ever since to become more sophisticated and powerful. Heuristics seek good and fast solutions by intelligently exploiting the structure of the given problem and performing simple intuitive steps in order to achieve an approximate solution without a guarantee for optimality. The main limitation of the heuristic methods is that they do not give optimal solutions or any indications of how far a solution is from optimality. The main strength of these methods is that they can tackle very complex problems where exact methods fail and a good and fast solution is considered sufficient enough for decision making. Some CO problems such as the VRP cannot be solved optimally for the large instances. This is the reason why heuristic methods have been in the focus of academic research in the recent years. There are four main characteristics of heuristics that need to be in place, in order to be considered efficient and useful. Heuristics have to be accurate, fast, robust, simple and easy to implement and modify (Cordeau, 2002). Heuristic methods can be grouped into six classes. Class A are constructions heuristics, Class B are Improvement heuristics, Class C Mathematical Programming, Class D Partitioning Heuristics, Class E Relaxations and Reductions, Class F Composite heuristics and Class G is Metaheuristics. Very often in the literature authors associate NP-Hard problems such as VRP with a necessity to use heuristic methods.

Construction Heuristics build a solution by making series of decisions one by one, at each step. For instance a vehicle is leaving from the depot and the first customer it serves is the nearest to the depot (or any other predefined choice criteria). It uses the greedy criterion, because at each step through the graph a decision is made that seems best at the time and it is usually not changed later. Improvement Heuristics have the purpose of improving a given initial solution which can be obtained by the use of construction heuristic, or be randomly generated or based on some predefined criteria as in the case of compulsory vertices for the attractive

TSP (Erdogan, 2010). This class of heuristic methods is also called local search, because the solution space is explored by means of various local search operators.

Metaheuristic methods are believed to be amongst the most popular solution methods in the VRP domain and are responsible for some of the best known solutions found in the literature. Figure 2.7 shows the main types of metaheuristic methods. The remainder of the chapter covers the local search operators and the different types of metaheuristic methods which are most commonly used in the literature, and those relevant to this research.

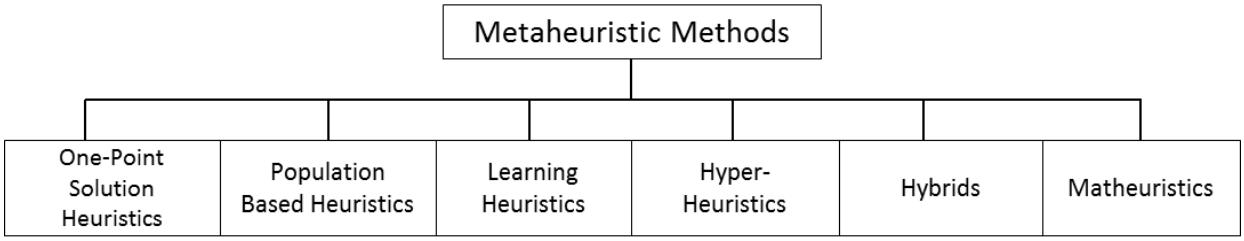

Figure 2.7: Metaheuristic Methods Typology

Construction Heuristics

Construction heuristics are typically used for the generation of initial solutions. They are not powerful enough to reach good heuristic solutions, but are a common way for generating a starting point for further metaheuristic search. There are many construction heuristics which are used in the literature, with different rationale. For instance there are greedy heuristics, cluster-first route-second heuristics and petal heuristics. There are also composite construction heuristics, which combine a construction and an improvement heuristic, such as the GENI-US. It has a construction stage, namely the Generalized Insertion Procedure (GENI), followed by an improvement phase, namely Unstringing and Stringing (US). Moreover, there are different ways each construction heuristic can be executed, either in its classical form or with some relevant adaptation to the researched problem. Some of the most widely used construction heuristics are described in this section.

✓ *The Nearest Neighbour (NN)*

The NN is a very simple greedy heuristic method which has the following steps.

Step 1: **Choose** a random node i as a starting point, **Mark** i as visited*;*
Step 2: **Find** the nearest unvisited node j to node i , where $(i \neq j)$*;*
Step 3: **Mark** node j as visited*;* ***Set*** $i = j$;
Step 4:
If all nodes are visited ***end****;*
Else Go to step 2;

There can be many variations and adaptations of the NN, such as having different starting points or additional criteria by which nodes are added to routes. In this research we use an adaptation of the NN for initial solution generation.

✓ *The Savings Heuristic*

The savings heuristic was first introduced by Clarke and Wright (1964) and there are also many adaptations to it introduced in the literature. Some of the most used are the parallel version and the sequential version of the heuristic. The idea of the method is to find arcs between nodes, which will result in greatest cost savings if the nodes are to be joined together. The cost savings for each i , j are calculated using the following formula:

$S_{ij} = c_{0i} + c_{0j} - c_{ij}$, where c_{ij} denotes the cost or the arc between nodes i and j , and 0 is the depot. The steps for the classical savings heuristics are given below:

Step 1: Calculate all the saving S_{ij}
Step 2: Sort the pairs i , j in descending order according to their saving;
Step 3:
Do
Add New Route
Add an unvisited pair (i, j) to a route with largest saving;
Mark i and j as visited
Do
Find the largest saving from pair (i, j) to an unvisited node p , S_{ip} or S_{jp}
Add customer p to the route, mark it as visited
Until capacity is full
Until all nodes are visited

✓ *The Sweep Method*

The Sweep method was first introduced by Gillet and Miller (1974). It is based on the idea, that nodes are added to routes based on their adjacency in terms of their location

in the plane. Figure 2.8 shows a graphical representation of the Sweep Method. The black arrow represents a *ray* which sweeps the plane anticlockwise. The customers are added to the routes when the ray 'hits' them, while sweeping the plane. In Figure 2.8, customer 1, which has the smallest angle, will be first added to the tour, where number 6 will be added last.

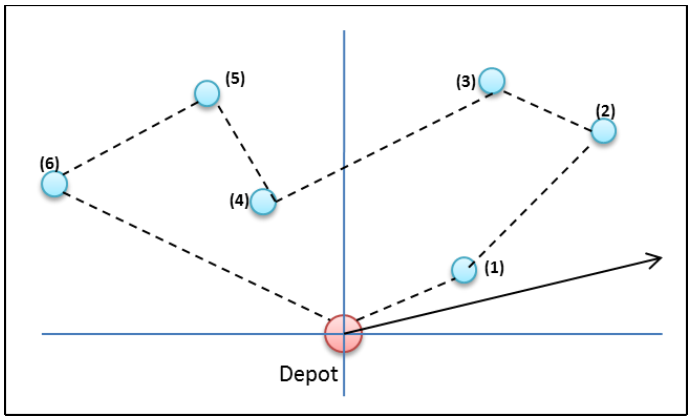

Figure 2.8: The Sweep Method

A simple pseudo code is given below and further detail on the Sweep method is given in Chapter 4.

Set depot as a centre of two-dimensional place
Calculate the angle of each customer node with respect to depot
Sort the customers in ascending order according to their angle
  Do
  Add New route
    Do
     Add customers from sorted list to the route
    Until capacity is full
  Until all nodes are visited

Local Search Operators

Local search operators are perhaps the most commonly used solution search techniques and are present in a great part of the research on VRP. They can be embedded in almost any metaheuristic method such as VNS, TS, Memetic Algorithm (MA), GA, SA. They can also be used as a post-optimization of Construction Heuristics before a more sophisticated method is employed for further solution quality improvement. This is the reason why we dedicate a separate paragraph for them. The local search operators can be used either until first improvement of the objective function occurs or by exhausting all possible moves and select

the best improving move. It has to be noted however, that researchers usually employ different neighbourhood reduction techniques, in order to minimize the admissible moves from the operators. This is usually done for the purpose of saving computational time. One example of this is the tabu tenure during TS, where some moves are not allowed to be performed for a certain number of iterations. The most commonly used local search operators are as follows.

✓ One customer Moves

Figure 2.9 shows an example of a 1 customer Shift and Swap. The 1-0 Shift operator involves shifting one customer from one route into another route in a systematic manner. Typically each customer is shifted from a route into every other route at every available position. The 1-1 Swap operator involves swapping one customer from one route with a customer from a different route. This is also done systematically where each customer is swapped with every other available customer in all routes.

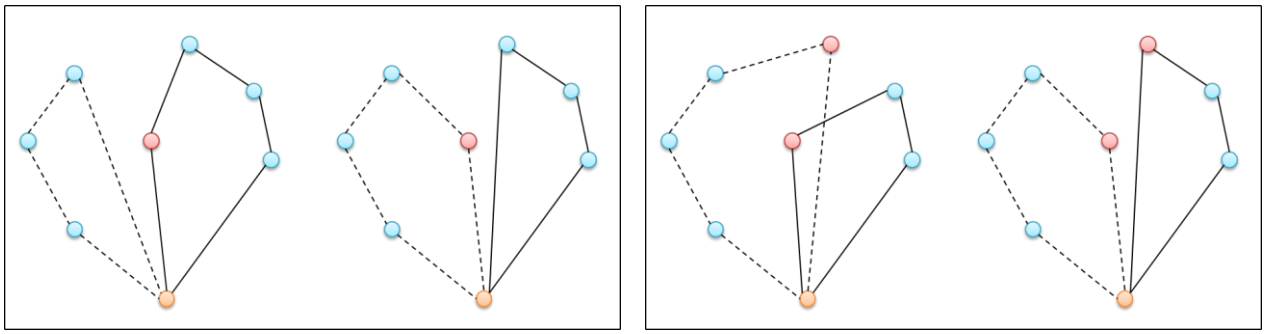

Figure 2.9: 1-0 Inter-route Shift and 1-1 Inter-route Swap

Figure 2.10 shows 1-0 Intra-Route shift, where one customer is removed from its current position and repositioned at another, within the same route.

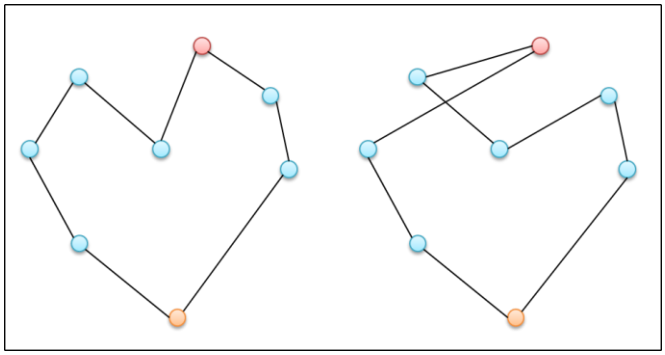

Figure 2.10: 1-1 Intra-route shift

✓ Two customer moves

The operators involving the move of two customers are 2-0 inter-route shift, 2-1 inter-route swap and 2-2 inter-route swap. Figure 2.11 illustrates these moves.

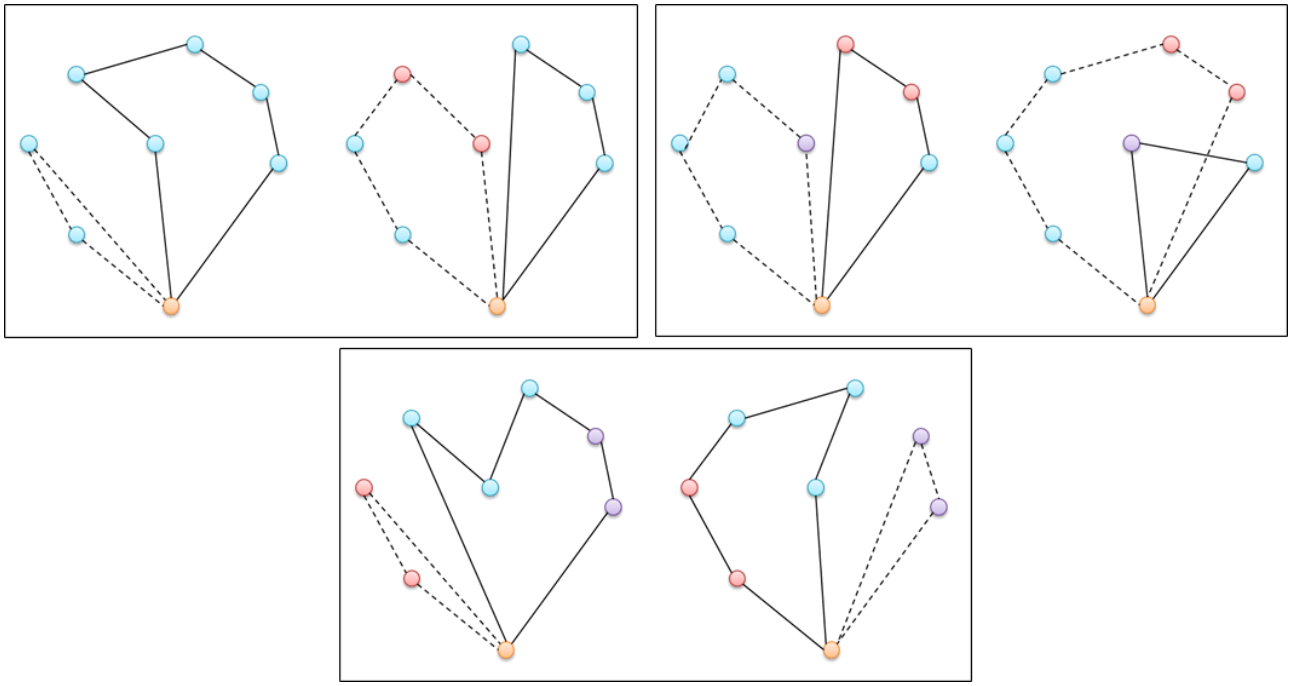

Figure 2.11: 2-0 Inter-route Shift, 2-1 and 2-2 Inter-route Swap

Another famous 2 customer move is 2-opt (Lin, 1973). It involves removing 2 arcs from the current route composition and reversing the order of customers between the deleted arcs. This is an Intra-route operator and it is less computationally expensive than 1-0 Shift. There is also 3-opt which involves reversing the order of 3 consecutive customers, however it is not as commonly used as 2-opt. 2-opt is portrayed in Figure 2.12.

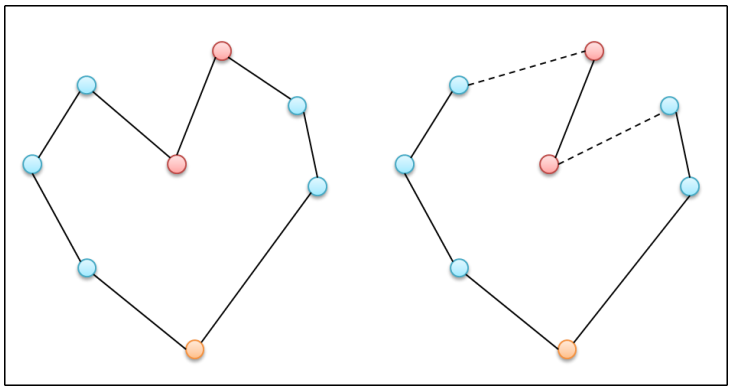

Figure 2.12: 2-opt

Metaheuristic Methods

Metaheuristics belong to Class G heuristics methods and are designed to solve very difficult optimisation problems, where classical heuristics and exact methods are not suitable for providing good solutions. Typically they are used for larger sized problems. As discussed in the previous section, the complexity of VRP increases with the size of the problem. There are exact algorithms which provide optimal solutions with up to 130 customers, however this applies to the classical VRP. There are some variants like the stochastic VRP, which are only solved to optimality for 70 customers (in the case of stochastic demand with one stochastic element), MDVRP for up to 151 customers and dynamic VRP with only up to 30 customers. In reality vehicle fleets are much larger and a company can serve hundreds of customers on a daily basis. This is where metaheuristic methods are most useful. For smaller instances metaheuristics are very powerful and often reach the known optimal solution. However, as the problem becomes larger there are many benchmark instances in the literature which allow for showing efficiency and effectiveness of the methods against other similar methods or best known solutions found so far. Metaheuristics make very few assumptions about a certain problem and therefore they are very flexible and highly applicable. There are some metaheuristics which require good knowledge of the research problem and can be somehow problem specific such as Simulated Annealing (Henderson, 2006; Eles, 2010) but by and large they are quite generalizable and adaptable to different problems and only an initial solution is required. Metaheuristics are improvement heuristics, but much more powerful and structured than Class B Heuristics. Some utilize memory structures in order to better explore the solution space. Moreover, most of the metaheuristic methods allow for degradation of the objective function and have various hill-climbing mechanisms, so as to escape getting trapped in local minima, which is a major weakness of construction and improvement heuristics. Intensification and Diversification are two very important aspects of metaheuristic methods. Intensification refers to mechanisms which aims to explore better regions of a given solution neighbourhood and depending on the adopted solution method there are various strategies that can be found in the literature for instance using local search operators or probabilistic rules on the incumbent solution. Diversification refers to exploring further topography of the solution space to the incumbent, in order to get a better coverage of the solution space, escape from local optima and ideally find the global optima for the problem. An example of

diversification strategy is Random Restart from a random point or a new solution structure to the incumbent.

There are three types of metaheuristics.

(i) Local search methods or one-point solution methods. Most of the methods belong to the first category and include Tabu Search (TS), Simulated Annealing (SA), Variable Neighbourhood Search (VNS), Adaptive Neighbourhood Search (ANS), Large Neighbourhood Search (LNS). They are iterative methods and explore the solution space by performing various structured inter-route and intra-route moves according to some predefined criteria (Laporte, 2009). They typically need one initial solution to use as a starting point and employ shift and swap moves, as well as other principles until the best solution is found or a stopping criteria is applied.

(ii) Population based methods include Genetic Algorithms (GA), Memetic Algorithms (MA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO). They involve a population of initial solutions, generated according to different principles and work with more than one solution structure at a time.

(iii) Learning methods are the third broad type, where one of the most famous solution methods, based explicitly on learning is Adaptive Memory Programming (AMP), but there are other methods which make use of memory and learning such as TS, Path Relinking (PR) and Scatter Search (SS). There are not many methods which solely rely on learning. It mostly occurs during a local search or population generation. Learning is a key aspect in the intuitive design of heuristic methods and can act as an enhancement strategy of any other method, or learning principles can be used in a hybrid manner. The next paragraphs offer a description on some of the most widely used metaheuristic methods and those we use for this research.

- ✓ Variable Neighbourhood Search

Variable Neighbourhood Search (VNS) was introduced by Mladenovic and Hansen (1997). It is a very simple and powerful metaheuristic, which is its classical forms aims to provide structure and guidelines for the local search operators in a systematic fashion. The structure provided to the local search operators gives an opportunity for intensification and better exploring the current neighbourhood structures to the incumbent solution.

The search space refers to the space of all possible solutions that can be visited during the search process via selected local search operators such the Inter-Route and Intra-Route Shift and Swap. The neighbourhood structure is closely linked to the definition of the search space. A candidate solution S has a set of neighbouring solutions N(S) which can be reached by one local transformation. For instance if one customer is moved to a position in another route (the 1-0 Inter-Route Shift) will make the current solution go to a new neighbourhood. There are many possible neighbourhoods of a current solution S and some of them may be more attractive than the current neighbourhood structure. In some cases a single move or shift can result in a neighbourhood very close to the topography of the current solution S, but in other cases the solution structure may be significantly different. An example of this is when a heterogeneous fleet is present, one swap or shift can lead to change in the fleet composition, as well as the solution sequence. VNS uses multiple neighbourhood structures N_k, as opposed to one, which is typically the case with local search operators, N_k *for k = (1,…,*k_{max}*)* and $\mathrm{N_k}(x)$ the set of solutions in the k^{th} neighbourhood *of x.* VNS has only a few parameters that need to be considered by the researcher in advance. This includes the value of k_{max}, the order in which the neighbourhoods are visited and the strategy for changing from one neighbourhood to another.

The methodological steps of the VNS are as follows:

Step 1: Find Initial Solution *x*, using some construction heuristic or at random;

Step 2: Select the set of neighborhood structures N_k *for* $k = (1,...,k_{\max})$ that will be used in the search;

Step 3: Set $k = 1$, until $k = k_{\max}$, repeat the following steps:

(3a) Generate a point x' at random from the k^{th} neighborhood of $x(x' \in N_k(x))$;

(3b) Apply a local search method with x' as the initial solution; denote with x'' the obtained local optimum;

(3c) If the solution obtained is better than the incumbent, move there $(x := x'')$ and continue the search with N_1 $(k := 1)$, Else next k ;

The way one can go from one neighbourhood to another it is up to the researcher. Shift and Swap moves can be used or some probabilistic rule. VNS typically has a shake stage, which aims to diversify the search process. The shake stage is implemented if stagnation of the objective function occurs. It acts as a diversification strategy and brings the search process to distant regions of the current search space. This provides for better coverage of the solution topography. The shake stage can be done either by some perturbation mechanism or more commonly by a probabilistic rule, which involves random shifting of customers. Some authors use Random Restart strategy for diversification where a previously found good neighbourhood is further explored and all steps of the VNS are repeated from the beginning.

Large Neighbourhood Search (LNS) is similar to VNS but has construction and destruction mechanisms to identify the best neighbourhood. It involves exploring large parts of one neighbourhood, where a proportion of customers is removed from the solution sequence and then re-built typically using some construction heuristic. Adaptive Large Neighbourhood Search (ALNS) adopts several insertion and removal heuristics, by giving them proportional weight depending on their success in generating improvements in the current solution. LNS was firstly introduced by (Shaw, 1997) which is then extended to ALNS by (Ropke and Pisinger, 2006).

Another popular type of VNS is the Variable Neighbourhood Descent (VND). It is proposed by Mladenovic and Hansen (2001) as an extension of the classical VNS. The motivation behind is very simple. When working on a given candidate solution S, which has a set of neighbourhoods N(S), one cannot assume that the local optima for one neighbourhood will be the same as the local optima of another. Therefore, VND works systematically on finding the descent of each neighbourhood using local search operators and once the descent of the neighbourhood is found the search moves with another local search operator. VND is a probabilistic version of VNS, where Neighbourhood descent can be found either by first improvement strategy or best improvement. Usually best improvement is more computationally expensive. It is common that local optima, with respect to one or several neighbourhoods are very close to each other in the solution topography. This is because neighbourhoods of solution N(S) are nested, which means that each neighbourhood constrains the previous. In these cases VND and its deterministic nature may have the

problem escaping from those regions. Reduced VNS (RVND) is a stochastic VNS, with pre-selected neighbourhood structures chosen at random. There are many ways one can amend a given method to give it more intensification or diversification as suitable for the researched problem. This research makes use of a population VNS (PVNS), where neighbourhoods of multiple solutions are further explored, which acts as a diversification strategy.

Another method which is worth mentioning is Simulated Annealing (SA). It is a probabilistic local search method with a key function of hill-climbing. A parameter t (temperature) is introduced, which is the probability of accepting a non-improvement move. At the beginning of the algorithm this parameter is usually very high and it reduces with the number of iterations until it reaches zero or a certain stopping criteria. There are some generic decisions, which need to be set prior to the execution of the algorithm such as cooling schedule (choice of t, length of t and rate of decreasing of t) and stopping criteria. There are also some problem specific decisions such as the neighbourhood structure, the size of the solution space and the representation of the objective function (Eles, 2010).

- ✓ Tabu Search

Tabu Search (TS) is a very sophisticate metaheuristic method, which was originally introduced by Fred Glover in 1989. The word *'tabu'* originates from the Tongan language and means that something cannot be touched, because it is sacred. TS is a one-point solution method, however, because it makes use of memory structures it can also be classified as a learning algorithm. Similar to VNS, TS has an iterative search approach to finding good solutions through the well-known local search operators. However, the search process is strategically guided through forms of memory. TS has a few parameters which need to be considered in advance, or by computational experience. Firstly is the *tabu tenure*, which is part of the short-term memory of TS. Tabu tenure is one of the most important elements of TS and it has to do with prevention of short-term cycling over the current neighbourhood structure, or cycling over neighbourhoods which have already been visited. For instance if a local search method is adopted, which shifts one customer to a different position in another route without improvement of the objective function, the tabu tenure ensures that this move will not be performed again during the search process for a given number of iterations. Hence this move will become tabu, and it will be added to the tabu list. The reason why the move will be tabu only for a few iterations is the key to good intensification. Once a move becomes tabu, it is

not permitted again in the short term, therefore other moves will be applied and ideally lead to a further reduction of the objective function (assuming minimization). Once a better region is found from a few consecutive iterations, the moves that were tabu in the past iterations become admissible again. This is done because when further improving moves are found the solution goes to a new neighbourhood and when a tabu move is freed and applied to the new neighbouring solution it may result in further improvement. During the search process the tabu list is updated, by adding moves which are recently declared as tabu, and freeing those which have been tabu for a while. The tabu list can either by of dynamic length or fixed length, depending on the preference of the researcher. One of the key parametric considerations that have to be made is for how many iterations the tabu tenure is valid. This is typically done by trial and error, by running the algorithm with different values of tabu tenure and fine tuning the parameters based on experience and the results obtained. The tabu tenure value would also be dependent on the nature of the problem. Another benefit of having tabu tenure is that when a move is inadmissible it is not performed, which means that the computational time speeds up. Hence the tabu list acts as a neighbourhood reduction technique. There is trade-off when deciding the value for tabu tenure. If it is set too high, it will restrict the search process and cause stagnation, if it set too low, it will result in cycling.

The tabu status can be given to different types of moves which occurred in past iterations. Some authors give tabu status to improving moves, which means that the arc connecting the improving move cannot be broken. Others give tabu status to dropped arcs, or in other words the inverse of the improving, which is the strategy we use in this research. Tabu status can also be given to non-improving moves. For instance if a move degrades the objective function significantly, relative to a given threshold can receive a tabu status to that it does get revisited again, because it is not a promising move.

Another key parameter is the *aspiration criteria* which to an extent, addresses the problem with the tabu tenure trade-off. Having tabu moves is very powerful, but sometimes during the search process a move that is currently on the tabu list may lead to a better solution that the best found so far. This is because the structure of the solution changes over the runtime of the method and one cannot guarantee that all moves in the tabu list will be non-improving for a number of iterations. Therefore, in cases like this, the aspiration criteria can be applied, which means that the tabu status of a certain move on the list can be overridden and become

admissible again. Typically the tabu tenure forces the search into good regions of the search space by prohibiting worse quality moves, but in some cases it can lead to stagnation of the solution quality and getting trapped in local optima. Therefore, the aspiration criteria allows for some flexibility, which could intensify the search in the specific region. The termination criterion governs when the algorithm should terminate. There are a few typical ways of deciding when to terminate the tabu search algorithm. One can terminate the algorithm after a pre-specified number of iterations, or pre-specified number of iterations without improvement of the objective function. Another way is to terminate when the objective function reaches a certain value or when a maximum time limit of the runtime is imposed.

TS has been a focus of research in the VRP domain and responsible for some of the best known solutions in the literature. Therefore, many versions of the TS were introduced over the years. Battiti and Tecchiolli (1994) proposed a Reactive Tabu Search (RTS), which was later adapted by Wassan (2007). It adopts new powerful mechanisms to escape from local optima. One of the main differences between TS and RTS is how the tabu tenure value is decided. In the case of RTS it is decided dynamically, based on the reaction to repetitions of moves which occur during the search process. There is also an escape diversification strategy which moves the current solution away from its current topography if the neighbourhoods and structures of the solution are extensively repeated / cycled over. Some authors use randomized strategies, where others use a more guided approach.

Granular TS is another version of TS which is successfully implemented over a number of VRPs. It was proposed by Toth and Vigo (2002). What is different here to the classical TS is that moves which involve long distance arcs (customers too far from eachother) are not visited. This is not to state that a long arc is not permitted in a solution sequence, but that a sequence of only long arcs is not permitted. An arc is defined to be long if it is over the minimum granularity threshold defined as

$$\vartheta = \beta \frac{z}{(n+K)},$$

Where ϑ is the granular threshold, β is a scaling parameter, *n* is the size of the problem and K is the number of vehicles. The scaling parameter value is decided by trial and error. There

are other types of TS, which include probabilistic rules, Greedy Randomised Tabu Search (GRTS) proposed by Resende and Ribeiro (2003) and many more hybrid applications.

✓ Genetic Algorithms

Genetic Algorithms (GA) are informed by the metaphor of natural selection and genetics. They build on the principle that certain genes (solution sequences) have better characteristics than others, and those individuals (candidate solution) who have those 'good' genes are the fittest, hence they survive. When applied to CO, this means that the aim of the algorithm is to create parent population of solutions with different chromosomes (solution sequences), which are evaluated based on their fitness (value of objective function). Those parents who have better fitness are mated to create children solution offspring, which contains genes from both parents. Typically two parent solutions are combined to create a child individual. Parents can be generated randomly, or using some construction heuristic. The method which is used for combining the parent solutions is *crossover*. There are one point and two-point crossovers and some further variants proposed in the literature. A *mutation* operator is also used to slightly amend the inherited genes from the parents, hence the unique nature of the child. Usually GA and other population based methods are not very powerful on their own when dealing with complex VRPs. They are often applied together with local search methods (Laporte, 2009) and this forms a typical hybrid method of two metaheuristics. They include Ant Colony Optimisation (ACO) and Neural Networks (NN).

✓ Adaptive Memory Procedure

Adaptive Memory (AM) procedure is a term introduced by Rochat and Taillard (1995), which complements Tabu Search (TS) and refers to a special utilization of the memory during the search process. AMP can be defined as a special data structure, which initializes a set of solutions and during the search process keeps track of the "best" components of the solutions, which are later combined to build a better quality solutions (Tarantilis, 2005). Initially AM was used as a complement to TS, because one of the main characteristics of TS is the utilization of memory, where AM was used to bring in more diversification and intensification of the TS process. Most of the research on AM is still by means of TS. However, TS is not the only meta-heuristic methods which makes use of memory. There are other

methods which make use of forms of memory, such as Scatter Search (SS), Path Relinking (PR), Genetic Algorithms (GA), Memetic Algorithms (MA), and AM could be used to improve their performance. Therefore, Taillard (2001) introduced an umbrella term of Adaptive Memory Programing (AMP), which unifies all those solution methods which make strategic use of memory, where knowledge of the solution is gathered during the solution search process and it is later exploited to improve the solution. In short, we can talk about AMP when the solution method in question has underlying principles of memory already embedded in the nature of the method. However, explicitly embedding AM into meta-heuristics other than TS is not very common. There are some papers where AM was used with Particle Swarm Optimization (Yin et al., 2010) and Path Relinking (Li, 2010).

AMP has three main characteristics. Memory initialization is the initial pool of solutions. Memory updating refers to the ability of compiling knowledge about the solution space and the recognition and updating of "good" solution components. Memory exploitation is the last stage, where all the pieces of knowledge gathered are used to build an improved solution. AMP can be associated to a greater extend with longer-term memory structures, even though methods like TS also make use of short-term memory updating, namely the *tabu tenure*.

Similar to other metaheuristic methods, AMP has parameters and methodological justifications which need to be specified by the researcher. Originally AMP makes use mostly of probabilistic rules for methodological decision-making, but later on other methods have been introduced. In terms of memory initialization there are some common techniques. For instance, Tarantillis (2005) and Yin et al. (2010) use Diversification generation method, Tarantillis (2002) and Zachariadis (2010) use Paessens' construction algorithm, Tarantillis and Kiranoudis (2007) use the Generalized Route Construction algorithm (GEROCA) etc. Perhaps the most important methodological consideration regarding AMP is the way "good" solution components are extracted from the memory and how do we know exactly how good they are.

Tarantillis and Kiranoudis (2007) proposed the BoneRoute method, where good solution sequences are referred to as *bones*. A bone has two main characteristics and that is the length and frequency. A node sequence is regarded as a bone only if it is of a certain length and the bone will be regarded as good and will be saved into the memory if it has frequency higher

than a given threshold, hence it re-appears in the initial solution pool. The main rationale of this method is that good solution sequences appear in good, medium and low quality solutions, so they higher the frequency of a bone, the better the chance it is a promising solution component. Another method is the Solutions' Elite Parts Search (SEPAS) introduced by Tarantillis (2005). It is similar to the BoneRoute method, but employs more deterministic principles for initial solution generation, as well as for building the new solution out of the elite parts. Moreover, the parameters for initial solution pool, length and frequency are different, as well as the form of TS used to navigate through the search space. Zachariadis (2010) extended these notions to include bones of variable length, as well as assign different cost tags to the bones, indicating their goodness of fit. Li (2010) uses a multi-start AMP where solutions are constructed at each iteration and the survival of good solution sequences is extracted based on a specific probability.

The AM procedure has been successfully applied to different variants of the VRP and has produced competitive results. It is used for the classical VRP (Tarantillis, 2010), VRP with Heterogeneous Fleet (Li, 2010), VRP with Split Deliveries (Aleman, 2010), VRP with Simultaneous Pickup and Delivery (Zachariadis, 2010) and VRP Fleet Size and Mix with Time Windows (Repoussis, 2010), Heterogeneous Fleet Open VRP (Li, 2010), VRP with Backhauls (Wassan, 2007), VRP with Multiple Trips (Olivera, 2007). This research adopts the AM procedure in a hybridized manner with a heuristic solution method, which does not make use of memory in its classical form, namely the VNS. Instead, the proposed algorithm adopts learning mechanism from the neighbourhood search and uses AM procedure to improve the incumbent solution by preserving the elite parts. It also takes a more retrospective view on the memory initialization, through learning from past experience.

Hybrid Methods

Hybridisation occurs when two or more metaheuristic methods are combined, in order to reach better solutions. This is a very popular technique in recent years and there are many hybrid methods proposed by scholars which tackle VRP variants quite successfully. For instance Vidal et al. (2013) proposes a hybrid between GA and local search methods to solve the VRPTW which matches, and in some instances, improves current best known solutions. Also Belhaiza (2010) uses a hybrid VNS with TS and Oliveira (2010) uses an SA and

Neighbourhood Search hybrid to solve the VRPTW. There are also hybrids between metaheuristics and exact methods in the literature such as Subramanian's (2012) hybrid between Iterated local search and Set Partitioning for the VRPTW. There are many examples in the literature and exhaustive review cannot be provided. However, it is important to note that hybridisation provides more flexibility and has the ability to generate competitive results. Therefore, this research focuses on the design of hybrid metaheuristic methods in order to solve the proposed real life VRP. The main methodological drive behind creating hybrids is to successfully combine good elements from different solutions methods and aim to strike the right balance between diversification and intensification, as well as guided search and randomization. Various elements of different methods have been hybridised and some methods are more commonly used than others. It can be summarized that VNS, TS and GA are amongst the most commonly hybridized metaheuristics, where SP and BCP are amongst the most commonly hybridized exact methods.

Hyper-Heuristics

The term hyper-heuristics was introduced in 2000 by Cowling (2000), but notions of the main idea behind the method can be found as early as 1960s. The main idea underpinning hyper-heuristics as a CO solution method is to automate the design of heuristic methods and make them more generalizable to various CO problems. Typically the heuristic solution methods in the literature are very much problem specific and there is a need to amend the parameters of the search process to fit the nature of the problem or sometimes the instance in hand. Generalizability is one of the main motivations behind this method, and it fits the trend in the literature for calling for more flexible methods which can be applied across problem variants, and even across CO problems. Hyper-heuristics is not concerned with exploring the search space of solutions to a given problem; rather it explores a search space of heuristics, which can be applied to the studied problem. Hyper-heuristics can be viewed as a population or a set of easy to implement low-level heuristic methods which are the components of the hyper-heuristic, where a sequence of heuristics is chosen to address the problem at hand. The goal is not to operate on the solution space, but to find a good combination and sequence of heuristic solution methods, appropriate to address the problem. Hence a hyper-heuristic can be seen as a collection of methods which are generated in accordance with the problem. An analogy here can be made with the rich Solvers, where depending on the features of the

problem different constraints and aspects of the solution method are activated. Burke et al. (2010) proposes a classification of hyper-heuristic methods, which is shown in Figure 2.12.

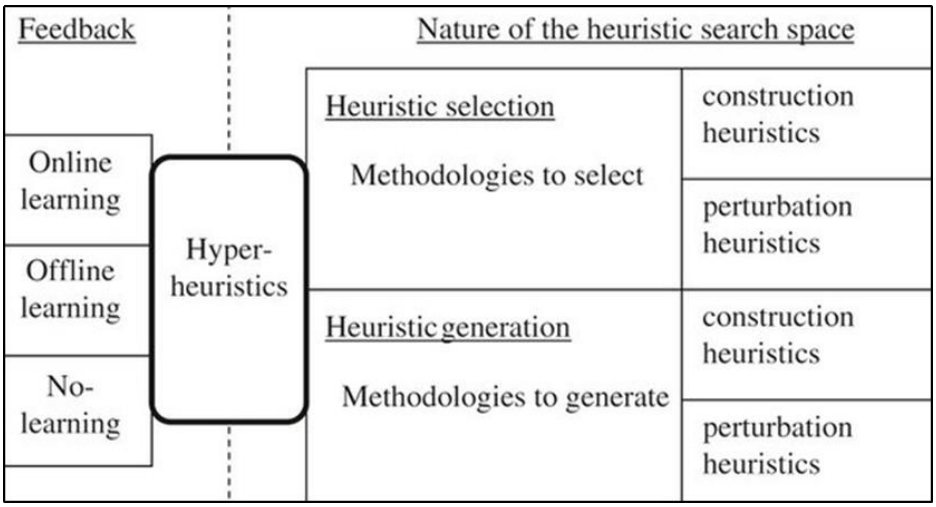

Figure 2.12: Classification of Hyper-heuristics (Burke et al., 2010)

There are two dimensions which need to be considered when classifying hyper-heuristics. One is the nature of the search space and the other is the sources of feedback. The nature of the search space has two components, namely heuristic generation and heuristic selection. Some methods from the set of methods that compose the hyper-heuristic can be directly applied to a problem. Other methods however can be generated depending on the nature of the problem, which combine features of different heuristics. This can be regarded as a form of learning, because it involves mix and match of features to best address the nature of the problem. Feedback is another learning mechanism, which can be online or offline. During online hyper-heuristics feedback, learning happens during the execution of the algorithm and the sequence of the chosen heuristic methods, where in offline hyper-heuristics learning knowledge is gathered during test instances and can be used later on instances which have not been solved before. There are types of hyper-heuristics where no learning is incorporated. There are some successful applications in VRP. For instance Garrido and Castro (2009) use hyper-heuristics to solve the CVRP incorporating constructive and perturbative heuristics and hill-climbing mechanisms. Further research into this area can have good contributions towards generalizability of algorithms, as well as learning mechanisms and ways of hybridising elements of different heuristic methods.

Matheuristics

Matheuristics are a type of hybrid metaheuristic methods, but the difference is that a heuristic method is hybridized with an exact method. Often exact methods are used as a subroutine of the metaheuristic method to solve smaller sub problems (Hanafi et al., 2010) or find an optimal assignment or partitioning. For instance Hanafi et al. (2010) uses Variable Neighbourhood Decomposition technique, where at each iteration the generation of neighbourhoods is done via solving a relaxation of the problem. The exact methods used to design matheuristics are usually set partitioning based, branching based or linear/ integer relaxations. Another popular way of hybridising exact and heuristic methods is to use candidate solutions at local optima, achieved by a given metaheuristics, as columns in a set partitioning formulation. For instance, Villegas et al. (2013) use Greedy Randomised Adaptive Search Procedure (GRASP) and Iterated Local Search (ILS) to generate the columns for the TTRP. Kramer et al. (2015) uses set covering formulation with local search for the Pollution-Routing Problem. Not all authors refer to an algorithm which combines exact and metaheuristic method as matheuristics. The terms is not universally accepted across the OR community. For instance Subramanian et al. (2012) proposes a hybrid metaheuristic algorithm which consists of Set Partitioning formulation which is executed in CPLEX solver, and calls iteratively a hybrid method based on ILS and VND. This would make the method a matheuristics since the metaheuristic is hybridized with an exact method, but it is not referred to by the authors as such. However, it has to be noted that it produces one of the best results on the VRPHF and FSMVRP benchmark instances. It shows that hybridising metaheuristics and exact methods result in one of the most competitive algorithms in terms of solution quality and it is a very fruitful area of research, because it gives an opportunity for creativity and flexibility of hybrid designs. Research efforts in this area can be very beneficial for academia and industry.

Solution Methods for RVRPs

Following from the discussion on the main VRP variants and their extensions, there is no one universal method which is used to solve RVRPs. In fact, the methods used in the literature are very diverse and cannot be all noted. There are some exact methods formulated for real life VRPs an example of which is column generation for rich VRP with inventory constraints

(Oppen, 2010) and Branch-and-Cut for Rich VRP with docking constraints (Rieck, 2010). However, some of the popular methods are TS, GA and Neighbourhood Search. The literature on rich VRP is difficult to summarize because of the loose definition of rich VRP. Some authors consider mixed variants such as VRPPDTW to be rich (Derigs, 2006), whereas the definition adopted in this research is quite different. A rich VRP is defined to consist of one or more main VRP variants and several real-world additions. Therefore, examples of solution methods will be given based on that definition. Tarantilis (2008) uses a hybrid metaheuristic method based on TS, VNS and Guided Local Search (GLS) for the VRP with intermediate replenishment facilities, Valle (2011) uses a hybrid between BC and column generation based heuristic and Greedy Randomized Adaptive Search Procedure (GRASP) for the min-max selective VRP. Some authors create new methods to fit the richness of the problem they are investigating. For instance Ren (2010) solved the VRP with multi shift and overtime and introduced a shift-dependent heuristic to tackle the problem. Benjamin (2010) proposed a waste collection VRPTW with driver's rest period and multiple disposal facilities. He develops a hybrid metaheuristic method using TS and VNS, which performed well on selected benchmark problems.

It is not common that a learning metaheuristic method is applied to a rich variant. In fact, to the best of our knowledge a learning metaheuristic has not been used to address a RVRP. Therefore, this research will attempt to propose a learning method to solve the RVRP introduced here and make it generalizable to other VR variants.

2.4. Summary

This chapter aimed to cover two areas from the literature on the Vehicle Routing Problem, namely variants of the VRP and solution methods for the VRP and any relevant VRP variants. Because of the steep growth in research in the VRP area, there are many different version of the VRP which are being addressed in the literature. They can be classified as main variants, extensions, combinations or (mixed) variants and real life (rich) variants of the VRP. In this research we introduce a real life VRP (RVRP) and we discussed the issues around researching a RVRP. We found that usually when one is addressing a RVRP there is little room for algorithmic comparability. In most cases authors focus on either designing exact methods or heuristic methods, and given the diverse nature of the RVRPs it is more challenging to

compare results against known literature benchmark instances. Therefore, we proposed to approach RVRPs in a standardize-first customize-second fashion, where a main variant of the VRP is incorporated within the RVRP, or if this does not match the real life specifications of the problem the Capacitated VRP can be used as a comparability platform. Moreover, we suggest that formulation can be provided for a RVRP and solved to optimality, on as large instances as possible, and this can act as a comparability guide for any metaheuristic method designed for the RVRP.

The literature on solution methods within the VRP domain suggests that hybrid methodologies are some of the most powerful when addressing a VRP and its variants. Combining principles from different heuristic methods can lead to enhanced performance of a method and hybrids are responsible for some of the best found solutions in the literature on different VRP variants. Therefore, we focus our methodological design on creating new hybrid metaheuristic methods which are used to solve the proposed RVRP in this research, as well as other relevant VRP problems.

CHAPTER 3

Problem Description and Formulation

This research introduces a new real life VRP variant to the body of literature on VRP. Before the problem was created, there were informal exploratory interviews with the Distribution and Logistics managers of the largest gas delivery company in the UK and market leader in gas supply with over 51% of the market share. Usually in MS/OR new variants introduced to the literature are inspired by industry. For instance the VRPPD is inspired by reverse logistics, the Roll on Roll off VRP by waste collection, the TTRP by large heavy goods vehicles etc. The rich variants are much closer to real routing practices and it can be argued that it is very important to understand the purpose of routing optimisation in the context of a real business and outline the main elements of the routing system before modelling and solving the problem.

After the exploratory interviews with the key informants of the researched company, a few important aspects of the gas delivery routing practices came to light, which will be incorporated in the modelling and problem definition.

(i) The restrictions on the length of a driver's route is either when the capacity of the vehicle is reached or and the end of the working day, which is 8 and a half hours. Overtime is also possible but at an extra cost if there is still available capacity. Allowable overtime is one of the main aspects which can be further improved. First, overtime is not considered in advance when creating the routes, but it occurs towards the end of a driver's shift, which means that if there is still capacity left in the vehicle, the deliveries made during overtime are not part of an optimized route, but only occurring where possible. Moreover, it is very common that drivers refuse overtime if it is not promptly offered and this leads to customer dissatisfaction and putting off deliveries for the next planning period. The room for improvement here can be twofold. First, in terms of cost saving, that is when overtime is considered in advance the routes can be generated to accommodate for that. Second, in terms of more qualitative gains, such as reducing drivers' resistance to overtime if they are informed about it in advance, and increasing customer satisfaction.

(ii) Gas delivery service times are proportionate to the demand. The larger the customer demand that needs to be satisfied, the longer it will take to do it. This is because the gas can be pumped into the customers' gas tanks at an average rate of 150 litres per minute. Service time is usually between 10 and 45 minutes. This leads to a special characteristic of gas delivery and that is the demand-dependent service time. This means that because of the time consuming service time, less customers can be serviced by one vehicle in one planning period, which also emphasizes the need for allowable overtime.

(iii) There is also a special requirement for light load, which commonly occurs in gas delivery or any other sector which uses heavy goods vehicles. A light load requirement is attached to the service of certain customers. For instance, if a customer lives in an area which is difficult to access, steep hills or soft grounds, then they can only be accessed when the vehicle becomes lighter. If the vehicle is full, it may be too heavy and may not be able to access that customer. This means that only once the vehicle has become lighter, a light load customer can be serviced. The maximum proportion of customers with light load requirement can be up to 20% of the total customers served, as per company's information. This is another possible area for improvement of the routing, because according to the company, the light load customers are not incorporated in the generation of the routes, but manually added at the end of the delivery period of a given vehicle. However, light load customers do not necessary have to be serviced at the very end of a shift, they can be serviced at any time a given vehicle becomes lighter. Therefore, incorporating the light load requirement can lead to cost savings for the company and a more efficient routing schedule.

(iv) The vehicle fleet is unlimited heterogeneous, with two types A and B. They have different capacities and variable costs, which are calculated based on average vehicle load and speed. The speed of the vehicles is quite low, an average of 30 mph, which leads to different travel times, which are not proportionate to the distance, but adjusted for the speed factor.

These main routing elements shape the nature of the RVRP introduced in this study, namely VRP with heterogeneous fleet, light loads, demand-dependent service times and allowable overtime. To the best of our knowledge there is no study which considers maximum overtime with unlimited vehicle fleet, which is an interesting feature to explore. It is very important that

when dealing with RVRPs practical implications are offered, in line with the main elements of the routing and the areas that can be improved as a result of optimization.

3.1 Problem Definition

The RVRP is modelled on a complete directed graph $G=(N, A)$, where *N* is the set of customers $N=\{0,1,\ldots,n\}$ with 0 being the depot, and $A=\{(i, j): i, j \in N, i \neq j\}$ is the set of arcs where each arc $(i, j) \in A$ has associated distance d_{ij} and time t_{ij}.There are *k* types of vehicles *k*, each $k=\{1,\ldots,K\}$, with a Q_k different capacities. Each vehicle is also associated with a variable cost v_k, based on how much fuel a specific vehicle consumes, given the vehicle's average speed. The number of vehicles of each type is unlimited. The distance is Euclidean and the cost is proportionate to the distance multiplied by the variable cost v_k. The time of travel is a key feature, because it takes into account the average speed of the vehicles. The average speed for heavy vehicles carrying dangerous load is approximately 50 km/h, which is quite low and has impact on the delivery schedule. Each customer $i \in N$ has a known demand q_i and known service time s_i, which is demand-dependent. Customers are divided into two types, regular *R(R ⊆ N),* which can be serviced at any time during the delivery period, and light load *L (L ⊆N, R ∩ L = {}, R∪ L = N).*If a customer is considered to be light load $(i \in L)$, it means that it can only be serviced if the remaining load in the vehicle is less than a specified threshold level, c_k for $k=\{1,\ldots,K\}$. *T* is the maximum regular time allowed for each vehicle route (7 hours and 15 min, adjusted for compulsory breaks), *O* is the maximum allowable overtime (4 hours and 30 min) and β is the cost of overtime which is 1.5 times higher than the cost of regular travel.

3.2. Formulation

The mixed integer formulation of the RVRP is presented in this section.

Decision Variables:

$x_{ijk} \in \{0,1\}$ 1 if vehicle *k* travels along arc (i, j), 0 otherwise;

y_{ijk} is a non-negative continuous variable, which denotes the remaining load on a vehicle k, travelling along the arc (i, j) before reaching customer j;

z_{ij} is a non-negative continuous variable, which keeps track of the travel time on arc (i, j);

Objective Function:

$$\text{Minimize Z} = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{K} x_{ijk} d_{ij} v_k \qquad (1)$$

Subject to:

$$\sum_{i=0}^{n}\sum_{k=1}^{K} x_{ijk} = 1; \qquad (j = 1,...,n); \qquad (2)$$

$$\sum_{j=0}^{n}\sum_{k=1}^{K} x_{ijk} = 1; \qquad (i = 1,...,n); \qquad (3)$$

$$\sum_{i=0}^{n} x_{ipk} - \sum_{j=0}^{n} x_{pjk} = 0; \qquad (k = 1,...,K), (p = 0,...,n); \qquad (4)$$

$$\sum_{i=1}^{n} y_{ijk} - q_i = \sum_{i=1}^{n} y_{jik}; \qquad (j = 0,...,n), (k = 1,...,K); \qquad (5)$$

$$\sum_{k=1}^{K} y_{ijk} \le Q_k \sum_{k=1}^{K} x_{ijk}; \qquad (i \ne j = 0,...,n); \qquad (6)$$

$$x_{ilk} \le 1 - \left((y_{ilk} - c_k)/Q_k\right); \qquad (i = 0,...,R), (l = 1,...,L), (k = 1,...,K); \qquad (7)$$

$$\sum_{i=1}^{n} z_{ij} - \sum_{i=1}^{n} z_{ji} = \sum_{i=1}^{n}\sum_{k=1}^{K} x_{ijk}\left(t_{ij} + s_i\right); \qquad (j = 0,...,n); \qquad (8)$$

$$z_{ij} \le T \sum_{k=1}^{K} x_{ijk}; \qquad (i \ne j = 0,...,n); \qquad (9)$$

$$z_{0i} = t_{0i} \sum_{k=1}^{K} x_{0ik}; \qquad (i = 1,...,n); \qquad (10)$$

$$z_{ij} \ge 0; \qquad (i \ne j = 0,...,n); \qquad (11)$$

$$y_{ijk} \ge 0; \qquad (i \ne j = 0,...,n), (k = 1,...,K); \qquad (12)$$

$$x_{ijk} = \{0,1\}; \qquad (i \ne j = 0,...,n), (k = 1,...,K); \qquad (13)$$

The Objective Function (1) aims to minimize the total cost of travel. Constraints (2)-(3) state that each vehicle arrives at a customer location and leaves that customer location exactly once. Constraint (4) ensures connectivity of the solution. Constraints (5)-(6) govern the commodity flow conservation and capacity restriction. Constraint (7) ensure that the light load customers $i \in L$, will only be serviced if the remaining load on the vehicle is less than the specified threshold c_k. Constraints (8)-(10) govern the maximum time allowed for each vehicle trip. Constraints (11)-(12) guarantee that the decision variables y_{ijk} and z_{ij} are positive, where constraint (13) specifies the binary nature of the decision variable x_{ijk}. The proposed MIP formulation has $(n+1)(2n+3)+3n+LK(R+1)$ constraints, $kn(n+1)$ binary variables and $n(n+1)+n(n-1)k$ continuous variables.

An extension of the formulation is also provided to include maximum overtime. The reason why overtime is modelled as an extension to the model without overtime (and separately solved by the metaheuristic method) is for the purpose of comparing the results and showing any possible cost savings by incorporating overtime. For the problem with overtime much smaller instances can be solved to optimality as opposed to the model without overtime, however lower/upper bounds are recorded where possible. In the case where overtime is allowed, the objective function can be modified as follows:

$$\text{Minimize Z}=\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{K} x_{ijk} d_{ij} v_k + \beta \sum_{k=1}^{K} Max\{a_k - T, 0\} \qquad (1a)$$

where a_k is a new variable denoting the arrival time at the depot for each vehicle. The additional component to the objective function ensures that upon return to the depot any time over the maximum regular *T* will be treated as overtime, multiplied by the overtime cost β and added to the total cost of travel.

The variable tracking the time z_{ij} is replaced by variable z_{ik}, which denotes the arrival time at customer i, for each vehicle $k = \{1,...,K\}$. Constraints (8)-(11) have to be replaced with the following, in order to account for any allowable overtime, where Big M is a significantly large constant.

$$z_{jk} \geq z_{ik} + t_{ij} + s_j + M\left(\sum_{k=1}^{K} x_{ijk} - 1\right); \quad (i = 0,...,n),(j = 1,...,n),(k = 1,...,K); \quad (8a)$$

$$a_k \geq z_{ik} + t_{i0} + s_i + M\left(x_{i0k} - 1\right); \quad (i = 0,...,n),(k = 1,...,K); \quad (9a)$$

$$a_k \leq T + O; \quad (k = 1,...,K); \quad (10a)$$

$$a_k \geq 0; \quad (k = 1,...,K); \quad (11a)$$

$$z_{ik} \geq 0; \quad (i \neq j = 0,...,n),(k = 1,...,K); \quad (12a)$$

Constraints (8a)-(9a) keep track of the time for each vehicle and the arrival time at the depot. Constraint (10a) ensures the maximum travel time (including regular and overtime) is not exceeded and constraints (11a)-(12a) ensure the positive nature of the corresponding variables.

The number of vehicles can also become fixed to a certain number m by adding constrain (14), but the type of vehicle chosen remains variable. Moreover, if constraints (7)-(11) in the original model are relaxed, the RVRP reduces to the VRP Fleet Size and Mix.

$$\sum_{j=1}^{n} x_{0jk} = m; \quad (k = 1,...,K); \quad (14)$$

3.3. Cplex Results for the RVRP

This section provides the problem specifications of the RVRP, as well as the results from Cplex for the RVRP with and without overtime. Testing the formulation is an important aspect of this research, since it will be used as a methodological comparability platform for the proposed metaheuristic method. Table 3.1 gives the specification of the RVRP, as well as some information on how the instances were generated.

Table 3.1: RVRP Problem Specifications

Customer Coordinates	Golden et al. (1984)
Customer Demands	Randomly Generated with Uniform Distribution [630,3950]
Vehicle Capacity	13050 litres (Type A) ,20880 litres (Type B)
Average Speed	30m/h
Service time	150 litres per minute
Variable cost per vehicle	0.36 (Type A), 0.48 (Type B)
Set of Light Load Customers $L \subseteq N$	Randomly Chosen

Table 3.2 shows the Cplex results for the RVRP without overtime, which is based on the formulation in Section 3.2. It shows optimal solutions in bold for up to 30 customers and for the other instances lower and upper bounds are provided. The time is reported in minutes and the Fleet Mix shows the fleet composition of the solution. The RVRP is solved using Cplex OPL Version 12.6, on a PC with Intel CPU 3.4G.

Table 3.2: Cplex Results for the RVRP without overtime

N	Proportion of L	LB/ Optimal	UB	Time	Fleet Mix
20	10%	**446.2**	-	4	3A 1B
20	15%	**446.9**	-	3	3A 1B
20	20%	**462.3**	-	3	3A 1B
30	10%	**560.1**	-	640	2A 3B
30	15%	**560.1**	-	640	2A 3B
30	20%	535.9	575.4	375	-
50	10%	701.1	901	1830	-
50	15%	706.8	958.2	248	-
50	20%	699.4	N/A	1109	-
75	10%	993.1	1541	971	-
75	15%	985.9	1391	1658	-
75	20%	985.9	N/A	2662	-
100	10%	1274.6	2908	1396	-
100	15%	1248.4	2844	1930	-
100	20%	1247.4	N/A	322	-

Table 3.3 shows the generated optimal solutions and lower/upper bounds for the RVRP with overtime, based on the extended formulation in Section 3.2. Only instances up to 50 customers are portrayed in Table 3.3, because for the larger instances the search tree grows to the memory limits and the program runs out of memory.

3.4. Summary

This chapter describes the RVRP proposed in this study in detail, with its different real life attributes and specifications of the dataset we used in order to create the test instances for the RVRP. Moreover, a Mixed Integer formulation of the problem with and without overtime

is presented and the results from Cplex are detailed. In the following Chapter we use the Cplex results as a comparability platform for the metaheuristic methods we designed to solve larger instances of the RVRP.

Table 3.3: Cplex Results for the RVRP with Overtime

N	Proportion of L	LB/Optimal	UB	Time	Fleet Mix
18	10%	390.3	-	4	1A 2B
20	10%	413.8	451.1	63	-
20	15%	413.8	451.1	51	-
20	20%	418.1	448.3	84	-
25	10%	474.5	511.8	22	-
30	10%	504.9	586.1	31	-
30	15%	504.9	586.1	30	-
30	20%	503.7	584.7	21	-
50	10%	699.1	-	32	-

CHAPTER 4

Initial Solution Method

Heuristic algorithms are usually constructed in a similar manner across different CO problems and their variants. Figure 4.1 gives a typical structure of the algorithmic sequence one usually follows when designing heuristic methods. It begins with the generation of an initial solution, then an execution of a main method, typically a member of the metaheuristic class or a hybrid and commonly ends with a post-optimization routine. Not all algorithm designs follow these steps, because an algorithm is also influenced by the nature of the problem, as well as the authors' preference.

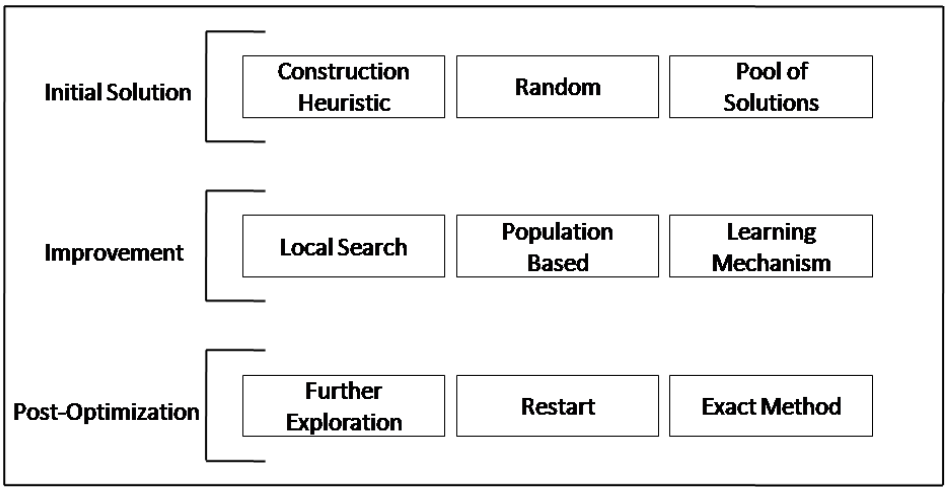

Figure 4.1: Common steps for a heuristic algorithm design

This chapter discusses in detail the generation of the initial solution for the RVRP and other VRP problems the algorithm is tested on. It is common practice in the literature for an algorithm to begin with an initial solution, generated either by a lower level heuristic of Type A or B (discussed in Section 2.3.2), a composite method as in the case of GENI-US or a randomly generated solution. There is an argument that can be made that having an initial solution generated by a given method, gives the subsequent heuristic search (main method) a good direction for exploration. Using a randomly generated solution could be more computationally expensive to transform into a better quality heuristic solution. However, a

powerful and efficient main method should be able to reach good heuristic solutions from any given starting point.

This research adopts a population-based view on the algorithm design, and this population-based nature is present throughout the algorithmic steps, including the initial solution. After an extensive literature review on the VRP domain, a few key observations came to light on the methodological side. A good algorithm requires a sufficient degree of diversification, intensification and a fine balance between randomization and guided approach. Further discussion and details on these aspects are portrayed in Chapter 6. The rationale of the algorithm proposed in this research, rests on the idea that diversification is a long term consideration throughout the algorithm, which can be achieved by changes in the solution structure in a greater scope. The diversification principle can be embedded into an algorithm from the beginning with the initial solution generation. This research uses a few methods for initial solution generation, which forms a population of initial solutions, which is referred to as the *Initial Solution Pool*. The idea is that having a pool of solutions which have different structure and sequence, could provide a coverage of a larger solution topography and explore corners which may otherwise not been explored if one initial solution (or one starting point) is used. Moreover, having a population of solutions aids the learning mechanisms which is embedded into the PVNS_AMP discussed in Chapter 5.

The initial solution used in this research is generated by using four different methods, which encompass the different ways an initial solution can be achieved. The construction heuristics used here are an Adapted version of the Sweep method (AS), an adapted version of the Nearest Neighbour method (ANN), a Parallel Clustering method and a Random initial solution. These methods are fast and can be repetitively used without too much computational effort. The reason why these methods were selected is because of their nature. The Sweep method understands proximity of customer nodes in terms of their geometric position, the Nearest Neighbour in terms of real distances regardless of their positioning on the plane, the Parallel Clustering method in terms of Least Sum of Squares, where the Random solution follows no principle. Therefore, the resulting solutions will have different composition. An example of this is portrayed in Figure 4.2.

Figure 4.2 shows two solution structures of a VRP with 4 customers (a, b, c, d) and a depot denoted with 0 using the same customer coordinates. It can be seen that the generated routes from the two methods are different in terms of solution sequence, despite having the same coordinates on the plane. The anticlockwise Sweep generates a route where customer **c** is before **b**, where the Nearest Neighbour's route visits customer **c** before **b**. Therefore, using methods with different underlying principles can lead to a diverse *Initial Solution Pool*.

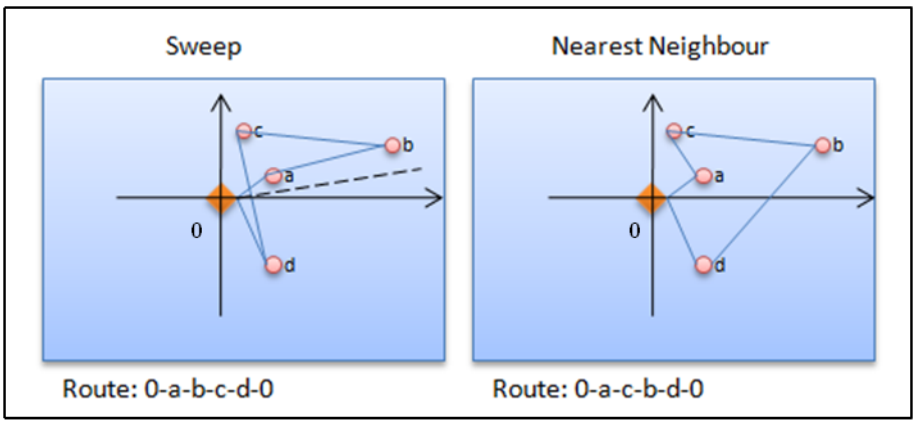

Figure 4.2: Routes generated from Anticlockwise Sweep and Nearest Neighbour

When making a methodological choice for an initial solution, there may be a relevant trade-off which is time vs. solution structure. If a random initial solution is used, it could be more computationally expensive to transform that into a better quality final heuristic solution. On the other hand, using a solution from construction heuristic can bias the search towards the solution structure of that heuristic, unless strong intensification strategies are put in place. Depending on the main method used in the algorithmic design, any initial solution strategy can be chosen. The following sections explain which methods were used in our case for initial solution generation and also provide some brief preliminary computational experience to support the methodological initial solution choices.

4.1. The Adapted Sweep Method (AS)

Initial solutions are usually adjusted to the nature of the researched problem. For instance the construction heuristics used here are influenced by the features of the RVRP, namely heterogeneous fleet, light load customers and others. Having real life characteristics in the

problem inevitably has an impact on the methods used. For instance if a construction heuristic such as the Sweep or Clarke & Wright Savings is applied to a heterogeneous fleet VRP, the question here is what fleet mix should be used. In order to overcome this drawback, here the construction heuristics are performed a fixed number of times, with different fleet compositions. The reason for this is to recognise better fleet compositions, which may lead to better solution quality. In addition, the RVRP can have allowable overtime. This further emphasizes the need for a pool of initial solutions, which can not only have different solution sequence and fleet composition, but also different allowable overtime for each route.

The use of Sweep has been adapted to fit the nature of the RVRP. The idea is motivated by the nature of the problem. Having light load requirement means that certain customers can be serviced at any time on a given route, as long as the capacity threshold has been reached.

A Giant Tour using the Anticlockwise Sweep angle sorting is first generated. The angles of the customer locations are only calculated once with respect to the depot. Once a Giant Tour based on the sorted angles is formed, the AS is performed for a fixed number of iterations (explained in a later section) from different starting nodes. After extensive computational experimentation an interesting observation was made. If the regular Anticlockwise Sweep is used, the starting point on the Giant Tour is the most negative angle, whereas the end point is the most positive angle. However, the link between the customers with most positive and most negative angles does not get explored. Therefore, we introduce a Giant Tour *Rewind* strategy, which aims to explore this link between the furthest customers in terms of angles. Therefore, we have empirically selected a Rewind section of 10 customers, regardless of the instance size. This means that the AS is performed a fixed number of times, but instead of starting from customer 1, we perform the AS for each consecutive node i as a starting point, for $i = (n-5,...,5)$.

Let us take a small example to explain this point with the following Giant Tour, consisting of 12 customers:

Depot – 1 – 3 – 5 – 2 – 4 – 7 – 9 – **10** – 8 – 6 – 11 – 12 – Depot			
Node in Rewind	5	***n-5***	n

Figure 4.3: Sample Giant Tour

Figure 4.3 shows a Giant Tour with *n = 12*. We perform the AS starting from node $i = n-5$ as a starting point which in the case of the Giant Tour is node **10**, and it is performed sequentially until $i = 5$, or in this case node **4**. This gives a better coverage of the proximity of the nodes, especially the proximity of those with the most positive and the most negative angles. The benefit of this is portrayed with a small example in Figure 4.4.

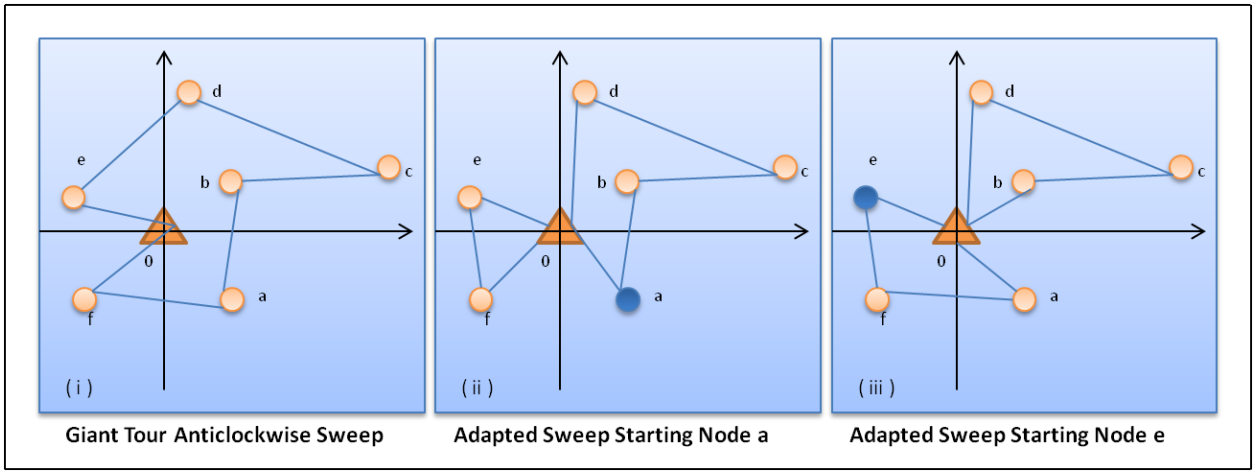

Figure 4.4: An illustrated Example of Adapted Sweep Rewind Strategy

Figure 4.4 (i) shows the Giant Tour after Anticlockwise AS. The starting point of the AS in 4.4 (ii) is node **a**, which is node number 1 on the Giant Tour. The starting point for the AS in 4.4 (iii) is node **e** on the Giant Tour, which is node $n-1$. The different starting nodes are shown in colour blue. In the instance shown in 4.4 (iii), there is a benefit in terms of distance as well as overall cost, where the customers are serviced by two smaller vehicles, rather than one small and one large as it is in the case of 4.4 (ii). This shows that having a Rewind strategy within AS can have a positive impact on the solution quality.

The number of nodes for the *Rewind* strategy was empirically tested and it was found that having 10 node *Rewind* on the Giant Tour is sufficient for achieving good quality solutions. Figure 4.5 shows the benefit of having 10 node Giant Tour *Rewind* on different sized Fleet Size and Mix benchmark instances from Golden et al. (1984). The reason why we choose to show different sized instances is for consistency purpose, since the behaviour of any method can be different when tested on different sized problems. The X axis on the figure shows the number of nodes tested for the *Rewind* Strategy for $i = (1,...,n)$, where the Y axis shows the corresponding solution quality. The line graph shows the solution quality at each node count for the four different problem instances.

It can be seen from Figure 4.5 that for all of the instances after the 10 node Giant Tour *Rewind* the solution quality does not improve, but stagnates. Therefore, 10 node rewind is performed for the final algorithm, because any additional iterations do not result in solution quality gains.

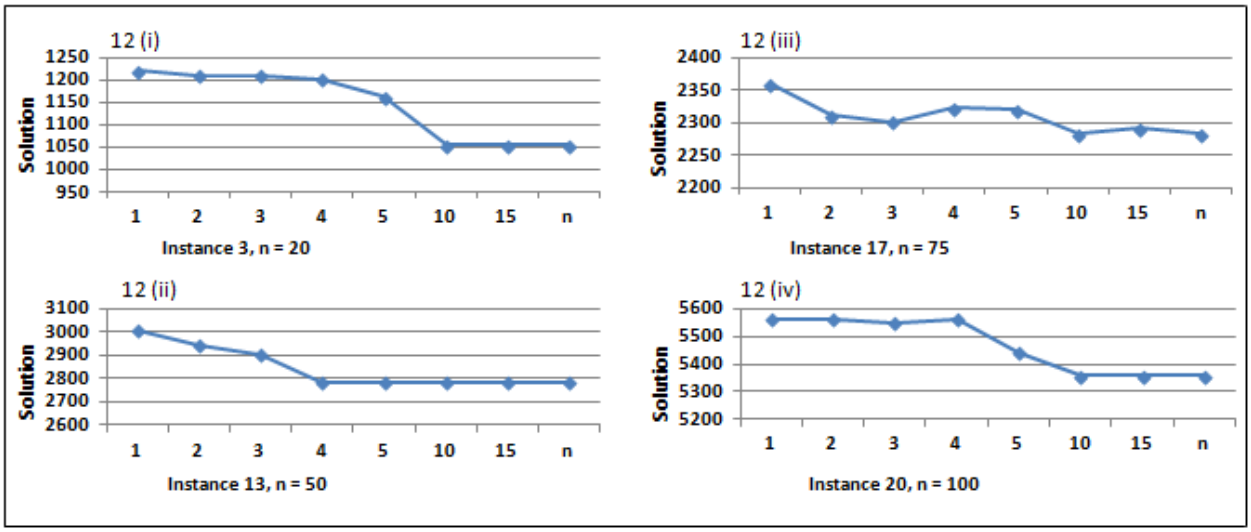

Figure 4.5: Performance of AS with different number of nodes for the *Rewind* Strategy

Another important aspect considered within the AS is the light load requirement. Having adjacent starting points for the AS means that if any customer has a light load requirement, it would be positioned at different locations along a given route and moving along the Giant Tour stepwise can ensure that at some point the light load customer will reach feasibility in terms of threshold capacity. This is shown is Figure 4.6.

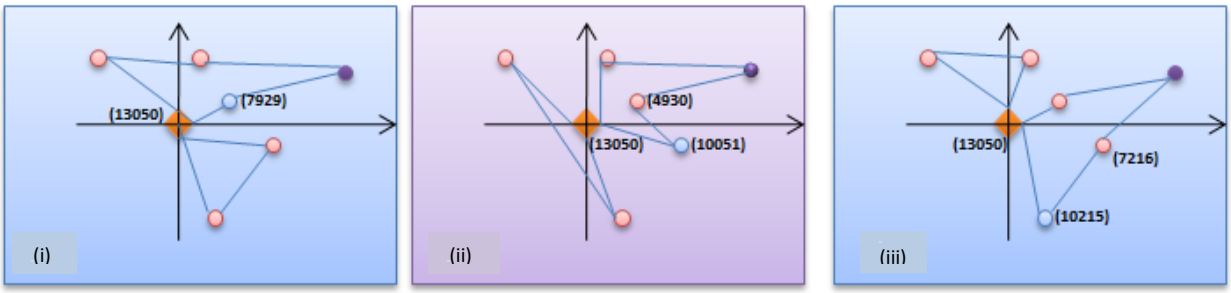

Figure 4.6: Light load customer distribution during Adapted Sweep

Figure 4.6 shows the AS with different starting points and feasibility of the tours in terms of the light load requirement. The customer which requires light load requirement is shown in purple, and the AS starting point is shown in blue. The numbers in the brackets show the load

remaining in the vehicle, where the total load leaving the depot is 13050 tonnes. In Figure 4.6 (i) the tour will be infeasible because the vehicle is not light enough to service the light load customer. The threshold for servicing light load customers is that the vehicle needs to be lighter than 5020 tonnes. Figure 4.6 (iii) is also infeasible in terms of light load requirement, even though the starting point on the Giant Tour is different. In Figure 4.6 (ii) however, the positioning of the nodes has shuffled and the light load requirement is met, hence this is the only feasible scenario. This shows that having the AS can help having a good quality initial solution as a starting point for the algorithm, which also has light load feasibility.

In case the AS cannot ensure that all nodes which require light load are in feasible positions, they have been feasibly re-assigned with a 1 – 0 Intra Route Shift (explained in Section 2.3.2, Figure 2.8) performed in a *Push_Back* fashion. Let us take a vehicle route which is formed of 6 customers after the AS has been performed. This is illustrated in Figure 4.7. The current node for the *Push Back* routine is node 2. The routine starts with the current position of the node and it re-inserts it in all other consequent positions along the route, but does not insert it in any previous positions, hence the *Push_Back* nature of the routine. This is done for the purpose of feasibly re-assigning any light load customers and also a quick post-optimization routine for the AS. The routine does not perform an exhaustive search, since only push back moves are allowed.

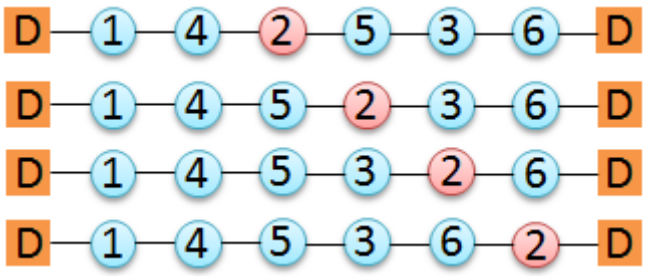

Figure 4.7: Illustration of the *Push_Back* Routine

Figure 4.8 provides a pseudo code for the AS, as well as the *Push_Back* routine applied to the routes generated from the AS. Computational experiments have been performed in order to find the best performing version of the AS. The AS has been tested for different number of iterations, different number of starting points for the Giant Tour Rewind, as well as coupled

up with the Push_Back Routine, which is the final version of the AS. These experiments show the benefit of having the different adapted features added to the Sweep and are tested on the well-known benchmark instances by Golden et al. (1984) with Heterogeneous Fleet and fixed vehicle cost. Table 4.1 shows the results of different versions of the AS which have been tested during computational experiments. AS3 refers to the final version of the method used for the PVNS_AMP and all the computational experience testing of the main metaheuristic method. A percent improvement in average solution quality between the different versions is also provided below the figure in bold.

Adapted Sweep:
Calculate customer angles
Sort Angles in Ascending Order
Generate Giant Tour for $i = (1,...,n)$;
  Do
    Set Starting node for Rewind $i = (n-5)$;
      Do
        Select vehicle type at random
        Select Overtime (in Minutes) at Random
         Assign nodes i from Giant Tour to routes
        Until capacity is full OR maximum time/overtime is reached for the route
      Save Solution S
    Next i;
  Until $i = 5$;
Select the 10 best solutions S in terms of objective function for *Push Back* routine
***Push Back* Routine:**
 For $S = (1,...,10)$
   Perform Push Back
 Next S

Figure 4.8: Adapted Sweep pseudo code

An interesting observation from Table 4.1 is that when the Giant Tour Rewind Strategy is not used, the increased number of iterations has a positive impact on the solution quality on average. However, when the strategy is used there is no impact when the AS is performed for more than 100 iterations. This means that there is a benefit in exploring different starting points from the Giant Tour and suggests a benefit of exploring the most positive-most negative angle link. When the AS is performed with the *Push_Back* routine it results in the best solution quality, therefore this is the final version of the AS used for further algorithm testing and part of the solution methodology.

Table 4.1: Computational Results for the Adapted Sweep Versions

Method						
Problem	AS1(a)	AS1(b)	AS1(c)	AS2(a)	AS2(b)	AS3
3	1221	1066	1066	1053	1053	967
4	8890	7355	7355	7355	7355	7304
5	1394	1194	1173	1170	1170	1031
6	8553	7562	7562	7562	7562	7356
13	3182	2905	2876	2869	2869	2550
14	16327	11687	11687	10797	10797	9637
15	3245	3051	3026	2998	2998	2763
16	3433	3326	3205	3195	3195	2898
17	2624	2357	2343	2323	2323	1837
18	3430	3284	3264	3163	3163	2612
19	12703	12165	12143	11095	11095	9481
20	5852	5533	5524	5358	5358	4578
IMP in Average Solution Quality:		**5.77%**	**0.16%**	**3.10%**	**0.00%**	**17.04%**

AS1(a): AS with 1 Iteration;

AS1(b): AS with 100 iterations;

AS1(c): AS with 1000 iteration;

AS2(a) AS with 100 iterations and 10 node rewind

AS2(b) AS with 1000 iterations and 10 node rewind

AS3: Final versions of AS with 100 iterations, 10 node rewind and Push_Back routine;

The results from the AS are also compared to known solutions generated by other relevant initial solution methods such as the Savings and Giant Tour based methods tested similarly to our approach. However, we have only compared our results to Initial Solution methods with no post-improvement routines. It has to be noted here that the AS makes use of a post-improvement routine, namely the *Push_Back* routine, however it is not used in an all exhaustive fashion. Table 4.2 (a) details the results obtained by other famous initial solution methods compared to the AS, together with Best Known Solutions (BKS) on the benchmark instances by Golden et al. (1984) on the Heterogeneous Fleet with Fixed Cost. The methods used for benchmarking purposes are classical the Savings heuristic (CW), Combined Savings

(SC), Combined Opportunity Savings (COS), Realistic opportunity Savings (ROS), Single Partition Giant Tour (SGT) and Multiple Partition Giant Tour (MGT). For details on those methods please refer to Golden et al. (1984).

Table 4.2 (a): Computational Results for the Adapted Sweep

Problem	Method						
	CW	CS	COS	ROS	SGT	MGT	AS
3	1119	1044	1024	1024	965	989	967
4	7822	7911	7306	7369	6918	7345	7304
5	1061	1060	1101	1052	1027	1056	1031
6	9343	7016	9401	7016	7391	7356	7356
13	2550	2650	2629	2616	2449	2494	2550
14	12000	9689	10154	9689	9637	9174	9637
15	2885	2763	2949	2763	2722	2742	2763
16	3026	2978	2982	2949	2855	2912	2898
17	1968	2043	2182	2000	1815	1837	1837
18	3447	2677	2587	2612	2479	2520	2612
19	11319	8741	10233	8741	9283	9411	9481
20	4689	4318	4921	4283	4273	4332	4578

Table 4.2 (b) shows the deviation of each Initial Solution method from the Best Known Solution (BKS). It can be seen that the AS has a good performance, where it outperforms the CW, SC, COS and ROS with up to 13.28% improvement. The AS was designed to reflect the real life nature of the RVRP, where the *Push_Back* routine was implemented in order to overcome any infeasibility in terms of the light load customers. However, it was found to perform relatively well with respect to the best performing initial solution methods SGT and MGT with a respective 2.5% and 1.1% average deviation, and a 7.8% average deviation from the BKS. All of the results have computational time of less than one second.

Table 4.2 (b): Average Deviation from BKS

Problem	BKS	Method						
		CW	CS	COS	ROS	SGT	MGT	AS
3	961.03	16.44%	8.63%	6.55%	6.55%	0.41%	2.91%	0.62%
4	6437.33	21.51%	22.89%	13.49%	14.47%	7.47%	14.10%	13.46%
5	1007.05	5.36%	5.26%	9.33%	4.46%	1.98%	4.86%	2.38%
6	6516.47	43.38%	7.67%	44.27%	7.67%	13.42%	12.88%	12.88%
13	2406.36	5.97%	10.12%	9.25%	8.71%	1.77%	3.64%	5.97%
14	9119.03	31.59%	6.25%	11.35%	6.25%	5.68%	0.60%	5.68%
15	2586.37	11.55%	6.83%	14.02%	6.83%	5.24%	6.02%	6.83%
16	2720.43	11.23%	9.47%	9.62%	8.40%	4.95%	7.04%	6.53%
17	1734.53	13.46%	17.78%	25.80%	15.31%	4.64%	5.91%	5.91%
18	2369.65	45.46%	12.97%	9.17%	10.23%	4.61%	6.34%	10.23%
19	8661.81	30.68%	0.91%	18.14%	0.91%	7.17%	8.65%	9.46%
20	4032.81	16.27%	7.07%	22.02%	6.20%	5.96%	7.42%	13.52%
Average Deviation:		**21.07%**	**9.66%**	**16.08%**	**8.00%**	**5.28%**	**6.70%**	**7.79%**

4.2. The Adapted Nearest Neighbour (ANN)

The Nearest Neighbour in its classical form is not a typical initial solution method which is used for problems with Heterogeneous fleet, and it is a relatively weaker method compared to other initial solution methods. However, the adapted version designed in this research is found to yield some good results. The testing of the ANN is very similar to the one detailed for the AS, and this section shows the benefit of having the adapted features of the NN. Figure 4.9 shows a simple pseudo code for the ANN.

The behaviour of the ANN in terms of solution quality is not similar to the AS, where after a certain number of iterations the solution quality stagnates. In ANN it was found that the more iterations and the more starting nodes on the Giant Tour are used, the better the solution quality. However, since this is only initial solution generation no more than 100 iterations per

starting node are considered, since the computational time used for initial solution generation is desired to be less than 1 second.

Calculate the Distance matrix
Do
Get the minimum distance d_{ij} from node i to unvisited node j;
Add j to the vehicle tour;
Mark j as visited;
Until All nodes are visited and Giant Tour is Formed
Do
Set $i = n - 5;$
Add nodes i from Giant Tour to form routes
Select vehicle type at random;
Select Overtime Present at Random;
Until capacity is full, maximum time/overtime is reached for the route;
$i = i + 1;$
Until $i = 5;$
Select the 10 best solutions S in terms of objective function for *Push Back* routine
***Push Back* Routine:**
For $S = (1, ..., 10)$
Perform Push Back
Next S

Figure 4.9: Adapted Nearest Neighbour pseudo code

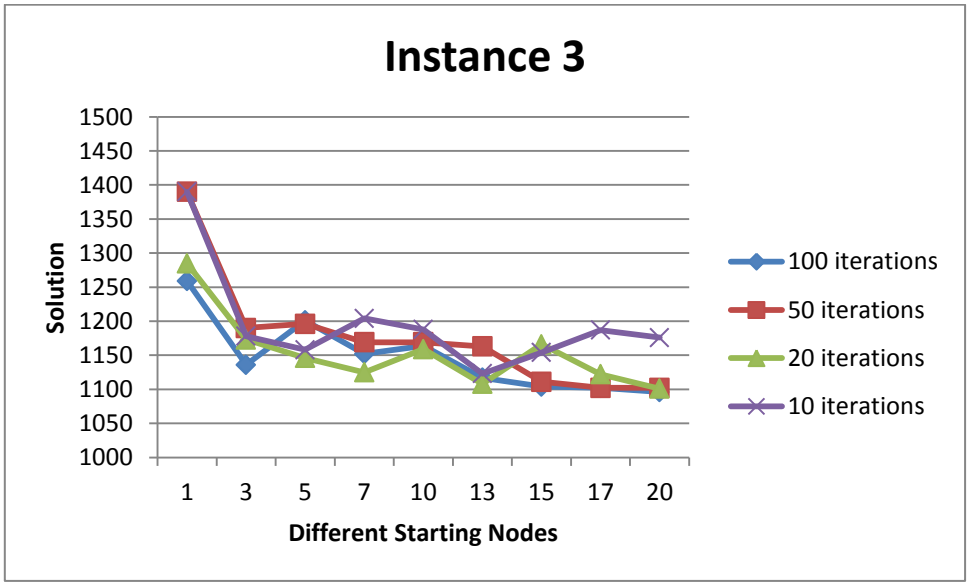

Figure 4.10 (a): Behaviour of ANN with different Iterations and Starting Nodes

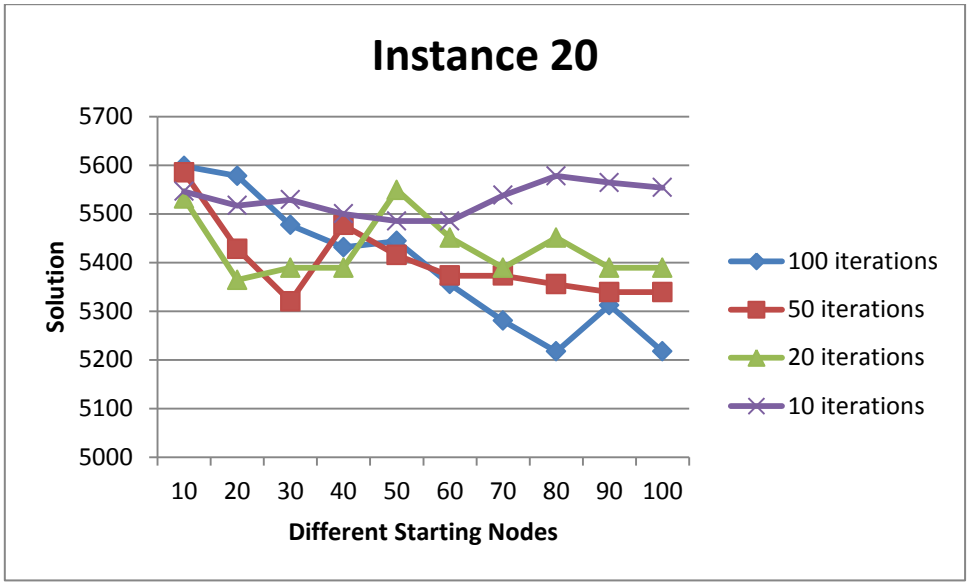

Figure 4.10 (b): Behaviour of ANN with different Iterations and Starting Nodes

Figures 4.10 (a) and (b) graphically show the behaviour of the ANN on two different sized instances when different numbers of starting nodes are used for creating the solution structure. The instances portrayed are from Golden et al. (1984) instance 3 with *n = 20* and instance 20 with *n = 100*, both with fixed vehicle cost. It can be seen that the behaviour is similar in the different sized instances. The more iterations there are the better the solution quality. Also with the increase of different starting nodes up to *n*, the solution quality improves. The trend is downward, but there are some peaks, which are due to the randomly generated fleet composition at each iteration. For the purpose of this research 100 iterations are chosen with n different starting nodes. For any instances larger than 100 customers, 100 nodes are used as a maximum number of starting nodes, in order to ensure the computational time will remain within the one second desired limit.

Table 4.3 shows the benefit of using the different versions of the ANN explored. Different number of iterations, as well as the addition of a different starting node have been tested, where the ANN4 correspond to the final version of the ANN which makes use of the *Push_Back* routine in the same fashion as for the AS. The testing of all initial solution methods is performed on the Golden et al. (1984) instances with Heterogeneous Fleet and fixed vehicle cost.

Table 4.3: Computational Results of the Adapted Nearest Neighbour Versions

Method				
Problem	ANN1	ANN2	ANN3	ANN4
3	1390	1125	1096	1024
4	8386	7956	7450	7429
5	1243	1180	1086	1034
6	8422	7499	7004	6922
13	2877	2712	2692	2608
14	14793	11633	12199	11119
15	3115	2880	2870	2815
16	3196	3196	3074	2870
17	2303	2073	1996	1901
18	3135	3112	2931	2686
19	12853	12172	12085	11280
20	5805	5457	5217	4987
IMP in Average Solution Quality		**10.69%**	**2.17%**	**5.34%**

ANN1: ANN with 1 iteration

ANN2: ANN with 1000 iterations

ANN3: ANN with 100*n iterations and different start node

ANN4: Final Version, NN with 100*n iterations, different start node and Push_Back Routine

It can be seen from Table 4.3 that there is a clear gain in the initial solution quality when the different start node strategy is used, and 100 iterations for each starting node has been found sufficient for good solution quality generation.

The obtained solutions from the ANN are also tested on the benchmark problem instances by Golden et al. (1984) and compared to other well performing initial solution methods. The results are shown in Table 4.4 (a), where Table 4.4 (b) details the average deviation from the BKS in the same manner as it was performed for the AS. The ANN outperforms the CW and the COS with up to 13% improvement, and has a 12.7% average deviation from the BKS. All of the results have computational time of less than one second.

Table 4.4 (a): Computational Results for the Adapted Nearest Neighbour

Problem	Method						
	CW	CS	COS	ROS	SGT	MGT	ANN
3	1119	1044	1024	1024	965	989	1024
4	7822	7911	7306	7369	6918	7345	7429
5	1061	1060	1101	1052	1027	1056	1034
6	9343	7016	9401	7016	7391	7356	6922
13	2550	2650	2629	2616	2449	2494	2608
14	12000	9689	10154	9689	9637	9174	11119
15	2885	2763	2949	2763	2722	2742	2815
16	3026	2978	2982	2949	2855	2912	2870
17	1968	2043	2182	2000	1815	1837	1901
18	3447	2677	2587	2612	2479	2520	2686
19	11319	8741	10233	8741	9283	9411	11280
20	4689	4318	4921	4283	4273	4332	4987

Table 4.4 (b): Average Deviation from BKS

Problem	BKS	Method						
		CW	CS	COS	ROS	SGT	MGT	ANN
3	961.03	16.44%	8.63%	6.55%	6.55%	0.41%	2.91%	6.55%
4	6437.33	21.51%	22.89%	13.49%	14.47%	7.47%	14.10%	15.40%
5	1007.05	5.36%	5.26%	9.33%	4.46%	1.98%	4.86%	2.68%
6	6516.47	43.38%	7.67%	44.27%	7.67%	13.42%	12.88%	6.22%
13	2406.36	5.97%	10.12%	9.25%	8.71%	1.77%	3.64%	8.38%
14	9119.03	31.59%	6.25%	11.35%	6.25%	5.68%	0.60%	21.93%
15	2586.37	11.55%	6.83%	14.02%	6.83%	5.24%	6.02%	8.84%
16	2720.43	11.23%	9.47%	9.62%	8.40%	4.95%	7.04%	5.50%
17	1734.53	13.46%	17.78%	25.80%	15.31%	4.64%	5.91%	9.60%
18	2369.65	45.46%	12.97%	9.17%	10.23%	4.61%	6.34%	13.35%
19	8661.81	30.68%	0.91%	18.14%	0.91%	7.17%	8.65%	30.23%
20	4032.81	16.27%	7.07%	22.02%	6.20%	5.96%	7.42%	23.66%
Average Deviation:		**21.07%**	**9.66%**	**16.08%**	**8.00%**	**5.28%**	**6.70%**	**12.70%**

One of the most important reasons why we use more than one initial solution method is for the purpose of diversification. Having initial solutions from different methods allows for diversification and creates a *pool* of initial solutions, which are further explored during the PVNS_AMP. We show the diversity of the initial solution pool with a small example portrayed in Table 4.5. Table 4.5 uses a small instance with 10 customers to show 3 Candidate Solutions which form a part of the *Initial Solution Pool*. The 3 candidate solutions have different fleet compositions, different allowable overtime and total cost. The routing schedule of each of the Candidate Solutions is given on the left hand side, whereas the characteristics of the solutions are given on the right hand side.

Table 4.5: Sample Candidate solutions from the Initial Solution Pool

Candidate Solution 1			Characteristics	
Route 1	0-1-3-7-8-4-5-10-0		**Cost:**	123.1
Route 2	0-2-6-9-0		**Overtime**:	20 min
-			**Fleet:**	1A 1B
Candidate Solution 2			Characteristics	
Route 1	0-1-3-4-5-10-0		**Cost**:	119.6
Route 2	0-2-6-9-0		**Overtime**:	0 min
Route 3	0-7-8-0		**Fleet**	2A 1B
Candidate Solution 3			Characteristics	
Route 1	0-1-3-4-0		**Cost**:	131.2
Route 2	0-2-6-9-0		**Overtime**:	0 min
Route 3	0-4-5-0		**Fleet**	4A
Route 4	0-7-8-0			

It can be seen from Table 4.5 that each of the Candidate Solutions is sufficiently different and this provides for a greater flexibility to explore different neighbourhoods of the solution space during the PVNS_AMP.

4.3. The Parallel Clustering Method (PC)

The PC used in this research is not very commonly used in the VRP domain. The clustering is based on the Ward Method which belongs to the family of Hierarchical Clustering Methods. The reason why the Ward method was chosen is because it is one of the most sophisticated

clustering methods, which uses the Least Sum of Squares criterion. Using a criterion different to those used of AS and ANN, can add to the diversification of the *Initial Solution Pool*.

There are many heuristic methods which belong to the type Cluster-First Route-Second, which are often used for initial solution generation. There are a few applications of such an approach for initial solution generation, but there are very few algorithms which hybridize clustering principles as a main ingredient to a metaheuristic method. Moreover, a clustering algorithm has never been used as means of learning. This research aims to tap into unexplored areas and research approaches which have not been used before. A Parallel Clustering algorithm is designed here, in order to aid the formation of the initial solution, but mainly it is used to aid learning as well as Neighbourhood Reduction (this is explained in more detail in Chapter 7). It is a parallel method, because the clustering and route formation happen at the same time, rather than sequentially as in the case of Cluster-First-Route-Second methods.

Based on computational experience and literature review, it is common that clustering algorithms on their own do not result in good quality solutions. The reason for this is because there are many clustering algorithms, each with their strengths and weaknesses. One of the main classes of clustering algorithms is the Hierarchical Clustering algorithms, which are based on exhaustive search and the outcome is exact in relation to the nature of the method employed. Hierarchical Clustering methods can be 2 types, namely agglomerative and divisive. Agglomerative methods start off as each data point (customer i) being its own cluster, and based on a given criteria, at each step of the clustering process cluster i is merged with cluster j. Divisive methods have a reverse approach, where all points start as being one super cluster and at each step of the clustering process the cluster is divided into smaller clusters until each point becomes a cluster on its own. They are exhaustive search methods, because at each step the proximity matrix is recalculated depending on the current number of clusters. Moreover, they can also be described as greedy, since the best move at a given time is accepted in order to create the next cluster.

A weakness of the hierarchical methods is that they cannot usually accommodate too large datasets, because the re-calculation of the proximity matrix grows with the size of the problem and can result in large computational times.

The other class of clustering algorithms is *K means,* where the search can be described as heuristic. They can handle big datasets because the number of iterations is not proportional to the size of the dataset. The rationale behind them is to be fast and easy to implement, similarly to the heuristic methods for VRP and not to hold large spaces of memory.

The reason why a clustering method is not usually used on its own or as a main ingredient for an algorithm for a VRP is threefold.

(i) One cannot make an assumption about the distribution of the customer coordinates. Usually when a clustering algorithm is applied in the field of data analysis, the behaviour of the data (linearity, spatial distribution etc.) can be checked in advance and appropriate method for clustering can be employed. When it comes to VRP, one cannot typically perform these preliminary checks. Some of the data instances which are used as literature benchmarks are random, which means that they are quite equally spread across a two dimensional plane. An example of this is shown in Figure 4.11:

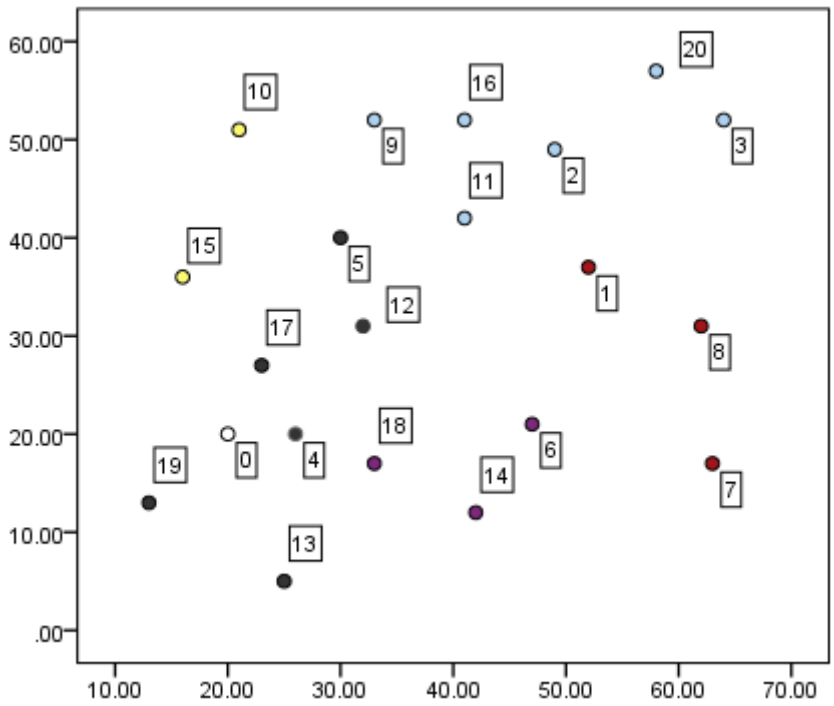

Figure 4.11: Scatter Plot of customer coordinates and Depot

This instance belongs to one the most famous benchmark instance problems by Golden et al. (1984) instance 3 of the small FSMVRP dataset. The cluster membership of the different nodes is represented with different colour, where the depot is 0. It can be seen that the customer

nodes are equally spread across the plane. In reality, this may not be the case. The datasets can have different geographical distribution and can follow naturally evolving clusters of customers to be serviced in one planning period. However, a good VRP algorithm should not be biased towards a specific dataset and ideally should perform well on any distribution. The different clustering methods tend to perform well on different datasets. For instance some methods are more prone to creating elongated clusters, where others are very sensitive to outliers. Some cannot address datasets, where there is linear or curvilinear relationship.

(ii) When creating routes it is very possible that nodes from one cluster can belong to a route which has nodes from different clusters, especially in randomly generated data. For instance, the optimal solution of the instance portrayed in Figure 4.11 contains a route which has customer 20 and customer 14 in the same route. The Ward clustering method would place them in different clusters and it is very likely that they will not be placed in the same cluster until the very end of the agglomeration schedule.

(iii) Hierarchical methods do not have means for specifying cluster number, but always end up in one giant cluster. This means that unless there is a prior knowledge on how many routes are to be formed it is difficult to decide on cluster number, as well as acceptable inter-cluster distances.

However, this does not mean that clustering algorithms cannot aid in other aspects of the VRP methodological design. This research employs the Ward method for Neighbourhood Reduction, whilst taking into account the limitations of its application and emphasizing the strengths. The Ward method has been adapted in order to fit the purpose of the RVRP introduced here.

Applying hierarchical clustering methods to the VRP have an advantage when it comes to cluster numbers. Time or capacity restrictions can act as constraints of the cluster size, that is when the capacity or maximum time is reached then the cluster is full. This allows for the parallel formation of clusters and routes. Moreover, if the fleet is fixed, the number of clusters can be easily specified. An additional benefit is the re-calculation of the agglomeration schedule and the reduction in computational time. When a capacity or time constraint is included into the agglomeration schedule, the algorithm does not end with one large cluster, but with the specified number of clusters, which leads to a complexity of $(n - c - 1)$, where c is

cluster number, instead of *(n - 1).* However, it was found that the PC has better performance if the agglomeration schedule is not constrained by capacity or time constraints, but when one Giant Tour is formed and is thereafter broken into routes. This is done in the same manner as for the ANN and the AS.

A simple pseudo code for the PC is provided in Figure 4.12.

Do
Calculate Proximity Matrix;
Merge closest clusters/nodes i and j to form a cluster;
Until all nodes are in one Giant Cluster (i.e. Giant Tour);
Do
Set $i = n - 5$;
Add nodes i from Giant Tour to form routes
Select vehicle type at random;
Select Overtime Present at Random;
Until capacity is full, maximum time/overtime is reached for the route;
$i = i + 1$;
Until $i = 5$;
Select the 10 best solutions S in terms of objective function for *Push Back* routine
***Push Back* Routine:**
For $S = (1,...,10)$
Perform Push Back
Next S

Figure 4.12: Parallel Clustering pseudo code

The results from the Parallel Clustering Method (PC) are presented in Table 4.6 and the deviation form BKS in Table 4.7. It can be seen that the method performs significantly better than the Clarke and Wright's Savings with 7.6% average improvement and 2% better than the COS. However, the method is not strong in terms of computational time. Because the Clustering method requires re-calculating of the proximity matrix each time a node is merged into a cluster, it increases the computational time significantly. Having a good initial solution method can give the heuristic search a better direction; however, this should not be at the expense of reasonable computational time. For this reason the PC was not included in the *Initial Solution Pool*, even though it results in reasonable solution quality and indeed provides diversification to the *Initial Solution Pool*.

However, the PC is used in the PVNS_AMP as a Neighbourhood Reduction technique which aids the improvement routines with probabilistic principles. This is explained further in Chapter 6.

Table 4.6: Computational Results for the Parallel Clustering

Problem	Method						
	CW	CS	COS	ROS	SGT	MGT	PC
3	1119	1044	1024	1024	965	989	1038
4	7822	7911	7306	7369	6918	7345	6945
5	1061	1060	1101	1052	1027	1056	1101
6	9343	7016	9401	7016	7391	7356	7426
13	2550	2650	2629	2616	2449	2494	2784
14	12000	9689	10154	9689	9637	9174	11807
15	2885	2763	2949	2763	2722	2742	2759
16	3026	2978	2982	2949	2855	2912	3019
17	1968	2043	2182	2000	1815	1837	2087
18	3447	2677	2587	2612	2479	2520	2895
19	11319	8741	10233	8741	9283	9411	11860
20	4689	4318	4921	4283	4273	4332	4987

The fourth type of initial solution generation is Random, both in terms of the fleet composition and solution sequence. Computational results for this will not be added since this does not add value to the computational experience.

Table 4.8 provides a summary of all 3 Initial Solution methods considered in this Chapter with the corresponding deviations from BKS.

Table 4.7: Deviation from BKS

Problem	BKS	Method						
		CW	CS	COS	ROS	SGT	MGT	PC
3	961.03	16.44%	8.63%	6.55%	6.55%	0.41%	2.91%	7.42%
4	6437.33	21.51%	22.89%	13.49%	14.47%	7.47%	14.10%	7.31%
5	1007.05	5.36%	5.26%	9.33%	4.46%	1.98%	4.86%	8.53%
6	6516.47	43.38%	7.67%	44.27%	7.67%	13.42%	12.88%	12.25%
13	2406.36	5.97%	10.12%	9.25%	8.71%	1.77%	3.64%	13.56%
14	9119.03	31.59%	6.25%	11.35%	6.25%	5.68%	0.60%	22.77%
15	2586.37	11.55%	6.83%	14.02%	6.83%	5.24%	6.02%	6.26%
16	2720.43	11.23%	9.47%	9.62%	8.40%	4.95%	7.04%	9.89%
17	1734.53	13.46%	17.78%	25.80%	15.31%	4.64%	5.91%	16.89%
18	2369.65	45.46%	12.97%	9.17%	10.23%	4.61%	6.34%	18.15%
19	8661.81	30.68%	0.91%	18.14%	0.91%	7.17%	8.65%	26.97%
20	4032.81	16.27%	7.07%	22.02%	6.20%	5.96%	7.42%	19.13%
Average Deviation:		**21.07%**	**9.66%**	**16.08%**	**8.00%**	**5.28%**	**6.70%**	**14.09%**

Table 4.8: Summary results of all initial solution methods

Problem	BKS	AS	ANN	PC
3	961.03	967	1024	1038
4	6437.33	7304	7429	6945
5	1007.05	1031	1034	1101
6	6516.47	7356	6922	7426
13	2406.36	2550	2608	2784
14	9119.03	9637	11119	11807
15	2586.37	2763	2815	2759
16	2720.43	2898	2870	3019
17	1734.53	1837	1901	2087
18	2369.65	2612	2686	2895
19	8661.81	9481	11280	11860
20	4032.81	4578	4987	4987
Average Deviation		**7.79%**	**14.09%**	**17.09%**

4.4. Summary

After performing extensive computational experiments we have designed 3 methods for initial solutions generation, namely the Adapted Sweep (AS), the Adapted Nearest Neighbour (ANN) and the Parallel Clustering (PC). Our experience suggested that all three methods have good performance compared to other initial solution generation methods, with the Adapted Sweep, being the superior method. However, we also found that Parallel Clustering consumes too much computational time and the methodological choice was to use the AS, the ANN and a randomly generated solution for final initial solution generation to be used within the metaheuristic methods designed hereafter. The Parallel Clustering will be used as a Neighbourhood Reduction Strategy.

The initial solution generation method used for the testing on every set of benchmark instances is the one detailed in this chapter. No amendments or adjustments to the initial solution method are made in any of the testing stages of the metaheuristics we designed when applied to the RVRP and the FSMVRP data instances. Small adjustments are made to the initial solution generation for testing the generalizability of the proposed metaheuristics on other VRP applications, detailed in Chapter 7.

CHAPTER 5

The PVNS_AMP Method

This chapter focuses on the developed learning-based metaheuristic method, namely the Population Variable Neighbourhood Search with Adaptive Memory Procedure (PVNS_AMP), which is applied to the RVRP of this study, as well as other VRP problems. The algorithmic steps are explained in great detail graphically and by pseudo code given in Section 5.2. Methodological justification for the algorithm design is also provided. Before the method description takes place there are some aspects and trade-offs that are important for the design of heuristic methods. To a great extent these aspect are key for the success of any given heuristic method designed to address a VRP and are briefly explained in the following section.

5.1 Strategic Choices and trade-offs of heuristic-based methods

After an extensive literature search and methodological experimentation, some aspects and trade-offs of heuristic methods' design came to light. These are explained hereafter, and the methodological design and justification of the PVNS_AMP, as well as other extensions to the algorithm take into account those aspects.

- ✓ Local Search Routines

As described in Section 2.3.2 local search routines are widely used as a part of various metaheuristic methods, such as the Shift and Swap operators. There are various operators that a researcher may choose to use in their methodological design. Moreover, there are many different variations of the standard operators that can be created by the researcher, based on randomized or systematic moves. The researcher should justify the choice of operators and conduct experiments in order to measure the impact and contribution of the operators and the necessity to use the operators chosen. Moreover, the order in which they are used needs to be considered, because each operator modifies the search space in a different way and results in different neighbourhood structures to the incumbent solution. For instance, when a heterogeneous fleet is present, one operator may not only lead to a change in the structure of the solution, but may lead to a change in the fleet composition as well.

Usually the best order of executing the operators becomes evident after extensive computational experiments. In some cases the order of execution may not have an impact on the final solution quality if the method is powerful enough to reach good heuristic solutions or best known solutions. The local search routines can be executed sequentially or the decision on which routine to choose to execute next may be as a result of a predefined criteria. For instance, if learning principles are adopted, one can select the operators based on the frequency and impact on the improvement of the solution quality. This means that an operator which is resulting in improvement of the objective function will be executed more often than those which do not result in improvement, or those which do not result in improvements as frequently. In this research, this issue is tested extensively and the local search routines are executed in a manner, which is found most suitable for the problem at hand.

✓ Intensification

When dealing with metaheuristic methods, intensification is an important aspect that needs to be incorporated into the search process. Intensification is defined as a short term anti-cycling strategy, which aims to explore close neighbourhoods of the incumbent solution in the adjacent topographies. Intensification can be described as a technique, which is usually associated with short term memory, or in other words, to prevent stagnation of the solution in the short term during the exploration of closer vicinities of the incumbent solution. An example of this is portrayed in Figure 5.1.

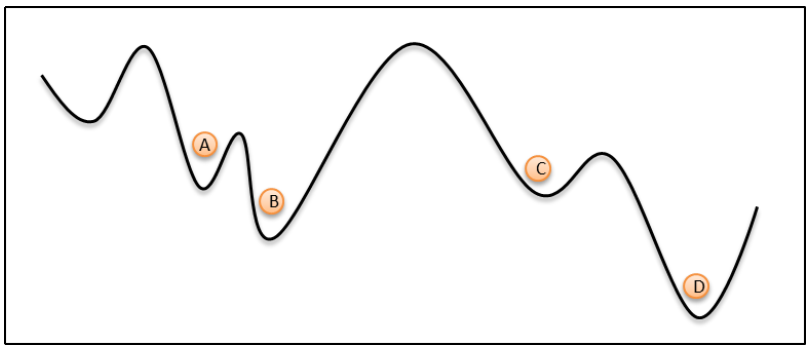

Figure 5.1: Topography of a VRP solution

At a given point of the search process, the incumbent solution finds itself at position A. A small change in the solution structure can lead the solution to drop to position B. This is an

example of intensification; exploring neighbourhoods adjacent to the incumbent via changes into the solution structure, which do not degrade the original structure too much, but enough to force the search into unexplored areas within the neighbourhood vicinity. This is an important consideration for every heuristic methodology, because the global optima can sometimes be found in areas closer to the incumbent solution. A good intensification strategy should be able to explore the adjacent topography as much in depth as possible, without resulting in cycling or stagnation of the search process. An example of intensification of the search process is the tabu tenure (a parameter of Tabu Search, described in Section 2.3.2), where cycling over regions which are already explored is prevented for a certain number of iterations. However, this can be overridden by the Aspiration criteria, where going back to a solution structure which was explored in the recent history can be accepted again, if it results in improvement of the current solution.

- ✓ Diversification

Diversification is another strategy which has to be considered when designing heuristic methods. The purpose of diversification is to avoid stagnation in the long term. Its purpose is to force the search out of a region which is already been extensively explored and bring it to a new unexplored region which can be further from the current solution structure. Sometimes intensification can only result in local optima if no diversification strategy is adopted. Therefore, in order to explore larger areas of the search space it is needed to be incorporated into the algorithmic design. Looking at Figure 5.1, an example of diversification will be when the incumbent solution is at position B, adopting diversification can bring to position C, and a subsequent intensified search of that region can lead it to position D, which is the global optimum. In many cases good diversification involves hill-climbing, when the solution has to worsen, before it improves on the current best. An example of diversification strategy is random restart, where the search begins at a new region. A perturbation or randomisation of the incumbent solution can also act as a diversification strategy, if it results in a greater change in the incumbent solution topography. An example of that can be a more vigorous Shake routine. A fine balance has to be achieved between intensification and diversification, in order to guide the search intelligently towards promising regions of the solution space.

✓ Randomisation vs. Guided approach

Randomisation is commonly used in heuristics for many purposes, mainly diversification. In some cases it can also be used for initial solution generation, or generating a pool of solutions, or Bones as in the BoneRoute method (explained in Section 2.3.2). Using randomisation is usually alternative to using some kind of systematic or guided method. For instance, an initial solution to a VRP can either be randomly obtained, or by a relevant construction heuristic. This methodological choice has to be justified, and the benefit of using one as opposed to the other should be clearly communicated. The amount of randomness in a heuristic method should also be carefully considered; usually it is done by computational experimentation. If there is too much randomisation, the search process may be corrupted and good areas which can be reached through intensification may be missed out. Moreover, if the algorithm is repeated, the best found solution may not be found again, even after a number of runs. There should be a balance of randomisation and guided approach to the search, in order to have a good degree of control over the solution search. However, too little randomisation in some cases, may not bring the desired diversification of the search.

✓ Parameter Tuning

Most of the proposed heuristic methods have a number of parameters, which need to be fine-tuned, in order to results in good solutions. An example of parameters is number of iterations, size of solution populations, stopping criteria and other parameters which are dependent on the method used. Sometimes an indication of a good value for a parameter can be found in the literature, based on recommendations of authors who have done research in a given area in the past. However, it is also very common that researchers give recommended values for algorithmic parameters based on computational experience. Different values for the parameters are explored before deciding the most appropriate one for the problem at hand. Of course, this needs to be discussed and added into the methodological justification clearly, so that further research can benefit from the findings. This is mostly necessary if a new methodology is presented, or an existing one is being modified. This research explains the benefits of using specific parameters and details the testing process.

- ✓ Neighbourhood Reduction vs. Cycling

There are many ways which are commonly used to reduce the complexity of the local search operators, such as Shift and Swap. For instance, one can use distances or angles of the customer nodes in order to reduce the neighbourhood search. If two nodes are far from each other, in terms of distance or another measure, then the neighbourhood moves do not explore the option of linking them with an arc, or swapping their positions, in certain conditions. The tabu tenure for example can be viewed as a reduction technique, because certain moves are tabu for a number of iterations and this reduces the number of possible moves during the local search. Like any other methodological consideration, there is a trade-off between the amount of allowable moves one can reduce and the chance for cycling. If too many customer nodes are tabu, or are characterized as restricted moves, then this could limit the search space too much, which can lead to limited exploration of the topography and possibly missing a promising solution. However, if too little restrictions apply, this could lead to cycling over the same solution structure and fail to employ successful intensification strategy, or even to diversify the solution into further regions. In this research the neighbourhood reduction technique based on Parallel Clustering, described in Section 4.3 is used with the measure of increased sum of squares, which is further discussed and illustrated in Chapter 7.

All of the trade-offs described above should be taken into consideration when designing a heuristic-based method, so as to make the search for good solutions efficient and quick.

5.2. The PVNS _AMP Method Description

This section provides a detailed description of the Population Variable Neighbourhood Search with Adaptive Memory (PVNS_AMP) method designed for the RVRP in this study. It provides methodological justification of the method, as well as extensive computational experience to show the benefit of adding learning mechanisms to the VNS. One of the main objectives of this research is to enhance a method, such as the VNS, which has no memory structures in its classical form, by the means of learning. Therefore, it is important to show what the impact on the solution quality is when learning is employed. Some interesting algorithmic behaviour is also noted, as well as any methodological trade-offs.

This research adapts the classical form of VNS, where the search is based on one-point solution search, to be population based and enhances it with learning principles of AMP. Hereafter, we refer to the proposed method as PVNS_AMP. The main idea behind VNS is to explore neighbourhoods of the incumbent solution in depth, which provides intensification of the search process. In this research a population based VNS is used, which means that more than one solution structure is kept and explored during the search (generated by the initial solution methods), which is done for the purpose of diversification and exploration of different solution structures.

The second reason for using population of the candidate solutions is for learning purposes. The idea behind this is that if same parts of the solution structure appear in different solutions, given their different solution structure, they are likely to be promising parts of that solution. For instance, if a node sequence appears in a candidate solution generated by the Adapted Sweep and it also appears in a solution generated by the Adapted Nearest Neighbour, this may suggest that this part of the solution is a promising part or a good node sequence, which can ultimately become of a part of the best heuristic solution found.

As described in section 2.3.2 AMP has 3 main considerations, namely Memory initialization, Memory Updating and Memory Exploitation. In the original AMP rationale memory initialization is done prior to the search process, memory updating is performed during the computational experience and memory exploitation is the final step where good solution structures are used to build the final solution.

In this research Memory Initialization and updating is not constructed in advance, but performed in parallel during the search in Stage 1 of the algorithm, namely the PVNS stage. Once some information on the solution structures is gathered, Memory Exploitation is performed during the Stage 2 of the algorithm, where the good solution structures found in the Stage 1 are preserved as a fixed part of the solution structure, and further exploration is only performed on the variable parts.

One of the most important considerations when adopting AMP is the way “good” solution sequences are recognised. In this research those good solution sequences are referred to as

Elite Strings. This methodological consideration is to a great extent the key feature for success of the method employed. If the criterion for good *Elite String* extraction is powerful enough, it can considerably improve the performance of the algorithm and vice versa.

In Stage 1 of the algorithm, the recognition of good node sequences depends on their length and frequency, similarly to the BoneRoute method described in Section 2.3.2 However, the length of the node sequence is variable, not fixed. Moreover, a sequence of one node is also accepted, if it is single customer route. Another significant difference with previous AMP methods is that a node is allowed to be repeated in the extracted node sequences, and the motivation behind it is that there may be more than one route composition and solution sequence, which could result in best heuristic solution. This is true especially if randomly generated instances are used for testing, or the distances between nodes are similar or identical. Adjacency to the depot is recognised as well, which means that if a customer is best suited to be serviced first after the depot the *Elite String* can include the depot node. In addition, the proposed algorithm PVNS_AMP has a few parameters, which are mostly related to memory extraction and exploitation. This is important as we aim to present a simple and intuitive method. The parameters used in this study are listed in Table 5.1. They have all been empirically tested and found most suitable for this RVRP problem.

Table 5.1: Parameters of the PVNS_AMP

Initial Solution Pool *P*	Consists of Initial Solutions $S \in P, S = (1,...,10);$
Memory Initialization Pool *M*	Consists of $x_{best} \in S^{th} neighbourhood;$ $M =$ dynamic length;
Memory Exploitation Pool with *Elite Strings E*	Consists of Solutions survived to the PVNS_AMP stage $E \subseteq M, E = (x_{best_1}(S'_1),..., x_{best_{10}}(S'_{10}));$
$iter_{max}$, for Stage 1	Dynamic, until no further improvement for 2 consecutive iterations
$iter_{max}$, for Stage 2	10
Elite Strings Recognition Criteria in *M*	Frequency >= 75%
Proportion of Elite Strings in *E*	<=30%
Elite Strings List	Dynamic Length

The PVNS_AMP has two stages. The first stage is called the PVNS stage, where information about the structure and the quality of the candidate solutions is gathered. At the end of the PVNS stage this information is used to recognize the *Elite Strings* which occur in more than 75% of the candidate solutions generated in the Initial Solution Pool and modified via the Shift and Swap operators within VNS. Stage 2 which we refer to as the PVNS_AMP stage is the memory exploitation stage, where only the best 10 solutions, in terms of solution quality survive. The *Elite strings* are encoded into the solutions which have survived from Stage 1 and are further exploited using VNS until the best solution is found. Figure 5.2 provides a simple pseudo code for our PVNS_AMP algorithm. For more information on the notation, please refer to Table 5.1.

Stage 1: PVNS (Learning) Stage
Generate P (Initial Solution Pool, explained in Chapter 4)
For Each $S \in P$
Do
Generate point x from the S^{th} Neighbourhood
$x_{best} = x, iter = 1;$
Do
Execute Neighbourhood Search Operators
Until no improvement of *x*
Add $x_{best} \in S$ to M
Shake
While no further improvement
Next S
End of Stage 1

Elite String **Recognition** in M
Select $E \subseteq M$

Stage 2: PVNS_AMP (Memory Exploitation) Stage
For Each $S^{'} \in E$
Do
$x^{"} = x_{best}, iter = 1;$
Do
Execute Neighbourhood Search Operators
Until no improvement of $x^{'}$
if $x^{"} < x_{best}$ **, update** x_{best}
Shake
While $iter < iter_{\max}$
Next $S^{'} \subseteq E$
End of Stage 2
End

Figure 5.2: PVNS_AMP pseudo code

Figure 5.3 also shows a graphical representation of the algorithm, so as to portray the sequence of the execution of the steps more clearly and visually. The PVNS stage (Stage 1) acts as the learning stage, whereas the PVNS_AMP stage (Stage 2) is the memory exploitation stage.

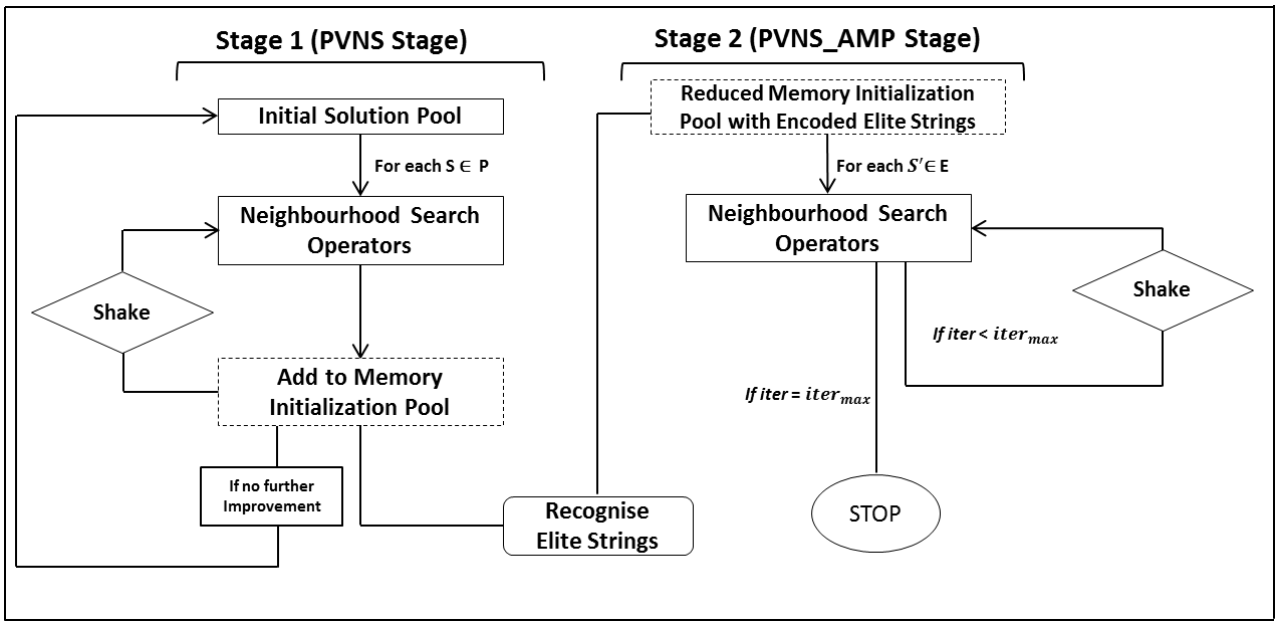

Figure 5.3: Graphical Representation of the PVNS_AMP

5.2.1 Stage 1 (PVNS Stage)

Stage 1 of the PVNS_AMP is where the experience of changes in neighbourhood structures is used to compile knowledge about good solution sequences reappearing in the diversified candidate solutions from the *Initial Solution Pool*. In the original rationale of AMP, Memory Initialization is constructed in advance. Here the Memory Initialization is done through learning from past experience. This PVNS stage is applied to the pool of generated initial solution sequentially, hence the population nature. All of the solutions explored are saved into the memory with their respective total cost. The PVNS consists of the execution of 6 neighbourhood search operators, as well as a *Shake* stage.

There are six neighbourhood search operators used to explore the solution neighbourhoods, which are explained in more detail in Section 2.3.2. The 1-1 intra-route swap, exchanges the positions of each node with all other nodes on the same route. 1-0 and 2-0 inter-route shift insert each node / each two consecutive nodes respectively, in all feasible locations on all

other routes. The 1-1 inter-route swap, exchanges the positions of one node with all other nodes from all different routes; 2-1 inter-route swap, exchanges the positions of 2 consecutive customers from one route with one customer from all other routes and 2-2 inter-route swap, exchanges the positions of two consecutive nodes from one route with two consecutive nodes from all other routes. All operators are used in a systematic exhaustive search fashion, where all feasible shifts and swaps are considered.

First-improvement moves are accepted for the purpose of "quicker learning". Therefore, first improvement strategy is used in order to find immediate good links between nodes, hence speed up the learning process for the composition of the *Elite Strings*. The operators are executed in an iterative fashion until there are no further improvements. The current best solution x_{best} of the S^{th} neighbourhood after each iteration is saved into the Memory Initialization Pool *M*. After the current best solution is saved, the *Shake* stage takes place. The *Shake* is done by probabilistic rules. In the learning stage 3 random customers from random routes are inserted into a different route at a random position. The *Shake* significantly degrades the quality of the solution. The reason for choosing a more vigorous *Shake* is because when the solution enters the neighbourhood operators at the next iteration, any good immediate links between nodes will re-appear if they were broken during the *Shake* and the frequency of the link will increase, as well as the likelihood to become a part of an *Elite String*.

The RVRP requires a very flexible methodology in order to explore greater regions of the solution space. Therefore, a special strategy for fleet diversification and allowable overtime is adopted during the execution of the neighbourhood operators, namely the *dummy route.* This means that each solution has an empty route at the end of the solution route sequence, which can be used for adding an extra route (vehicle) to the solution structure if it is feasible and results in better solution quality. This is shown in Figure 5.4.

Solution (S)			Solution (S')	
Route 1:	0-1-3-7-8-4-5-10-0	Applying 2-2 Inter-Route Shift	Route 1:	0-1-3-4-5-10-0
Route 2:	0-2-6-9-0		Route 2:	0-2-6-9-0
Dummy Route:	0-0		Route 3:	0-7-8-0
Cost:	123.1		Cost:	119.6

Figure 5.4: Implementation of the *dummy route*

Figure 5.4 shows the solution structure and objective function (with variable cost only) for a small sized instance with 10 customers. The solution on the left has an objective function of 123.1 where 1 vehicle of type B is used and 1 vehicle of type A. However, moving customers 7 and 8 to the dummy route results in the use of 3 vehicles of type A and an improvement of the objective function with 2.8%.

The reason for having a *dummy route* is twofold. First it is done for the purpose of intensification, in terms of vehicle fleet composition. The candidate solutions in the initial solution pool are diverse in terms of fleet composition, and they contain solutions with different number of vehicles. This is done in order to explore which fleet composition gives better solution quality. However, if a larger fleet is better suited for the problem at hand, it is beneficial to have the opportunity to add extra vehicles if this will result in better overall cost. The second reason is to avoid bias towards overtime.

One of the interesting features of this RVRP is the trade-off between unlimited fleet and use of allowable overtime. If there is no opportunity to explore this trade-off properly, then the analysis of results and the possibilities for savings may not be fully explored. Moreover, heuristic methods are typically problem-specific, which may lead to the biased design of the method in order to emphasize certain aspect of the problem. Adding a *dummy route* here is done for minimizing the bias towards allowable overtime, because it gives the method enough flexibility to select the most appropriate solution form. One cannot assume that having allowable overtime necessarily means that it will be beneficial, because it may in fact not be. Having population VNS here also helps increase the opportunity for finding the best conditions for the RVRP, because different solution structures are explored, which may include overtime or not, coupled with diverse vehicle fleet.

A *shrink route* strategy is also adopted, which has reverse logic to the dummy route strategy. This means that the fleet can be modified during the search process, and the number of vehicles used can be reduced, if it improves the solution quality. This is shown in Figure 5.5.

Solution (S)			Solution (S')	
Route 1:	0-1-3-7-8-4-0	Applying 2-2 Inter-Route Shift	Route 1:	0-1-3-7-8-4-0
Route 2:	0-2-6-9-0		Route 2:	0-2-6-5-10-9-0
Route 3:	0-5-10-0			
Cost:	135.7		Cost:	123.1

Figure 5.5: Implementation of the *shrink route*

Figure 5.5 shows the benefit of having variable fleet during the algorithmic runtime and the execution of the local search operators on a small instance of 10 customers with variable vehicle cost. Here the application of the 2-2 Inter-Route Shift leads to the reduction of vehicles used, from 3 vehicles of type A, to 2 vehicles of type B, with a gain of the objective function of 8.5%.

The execution of all Inter-Route Shift local search operators allow for the *shrink route* strategy and for the *dummy route* strategy. If the local search operators are performed in an exhaustive fashion it can become computationally expensive to perform many iterations, especially on the large instances. Various ways of implementing the operators have been designed and tested, in order to find a way for most efficient implementation. After an extensive computational and programing experience all of the local search operators are performed using a *gain function* γ which assists the quicker implementation of the operators, and in the later testing and extension of the algorithm (provided in Chapter 7) assists with the data structures for the Tabu Search. The gain function consists of calculating the gain in terms of cost of each move for a given local search routine and those moves which result in a negative gain are only kept in the memory and are performed in order to test the real impact of the move. The gain function γ for all local search routines is described hereafter.

✓ The 1-0 Inter-Route Shift routine involves moving one customer from one route to all positions on all other routes. The gain function for this routine is as follows, with i being the current node to be moved to route m and j is the position at which it is moved to route n and d the corresponding distance between the nodes:

$$\gamma = (d[[m][i-1]][[m][i+1]] + d[[n][j-1]][[m][i]] + d[[m][i]][[n][j]]) -$$
$$(d[[m][i-1]][m][i]] + d[[m][i]][[m][i+1]] + d[[n][j-1]][[n][j]]);$$

The first argument of the equation is the distance after the move is performed, whereas the second is the distance before the move is performed. If $\gamma < v_{\max}$, where $v_{\max}$ is the variable cost of the largest vehicle, means that the 1-0 shift may result in an improvement of the objective function, considering any possible change in the vehicle fleet as well. All the moves which have a gain function less than $v_{\max}$ are performed to see whether the move will still be improving when other constraints are not violated, such as capacity constraint and also to see the improvement on the objective function given any applicable changes in variable cost or fixed cost (if any), if the move results in change of fleet composition. If the objective function decreases the move is accepted and the search continues with the next improving move. If the problem at hand does not have heterogeneous fleet and all vehicles have identical variable cost, then all moves with $\gamma < 0$ will be improving moves. All Inter-Route Shifts and Swaps make use of the gain function.

- ✓ The gain function of the 1-1 Inter-Route swap is the following:

$$\gamma = (d[[m][i-1]][[n][j]] + d[[m][i+1]][[n][j]] + d[[m][i]][[n][j-1]] + d[[m][i]][[n][j+1]]) - (d[[m][i-1]][m][i]] + d[[m][i]][[m][i+1]] + d[[n][j-1]][[n][j]] + d[[n][j]][[n][j+1]]);$$

- ✓ The gain function of the 2-1 Inter -Route swap is the following:

$$\gamma = (d[[m][i-1]][[n][j]] + d[[m][i]][[n][j-1]] + d[[m][i+2]][[n][j]] + d[[m][i+1]][[n][j+1]]) - (d[[m][i-1]][m][i]] + d[[m][i+2]][[m][i+1]] + d[[n][j-1]][[n][j]] + d[[n][j]][[n][j+1]]);$$

- ✓ The gain function of the 2-2 Inter -Route swap is the following:

$$\gamma = (d[[m][i-1]][[n][j]] + d[[m][i+2]][[n][j+1]] + d[[m][i+1]][[n][j+2]] + d[[m][i]][[n][j-1]]) - (d[[m][i-1]][m][i]] + d[[m][i+2]][[m][i+1]] + d[[n][j-1]][[n][j]] + d[[n][j+1]][[n][j+2]]);$$

The Intra-Route Shift and Swap used in this research make use of the gain function in the same fashion, with the only difference that moves which have $\gamma < 0$ are considered improving moves. This is because when the shift is within the same route, the assumption is that the vehicle capacity will not change, therefore a negative γ will be associated with an improving move. The moves occur on the same route, therefore m refers to the route under investigation, where i and j to the different positions on that route.

✓ The gain function for the 1-0 Intra-Route shift is:

$$\gamma = (d[[m][i-1]][[m][i+1]] + d[[m][j]][[m][i]] + d[[m][i]][[m][j+1]]) - (d[[m][i-1]][m][i]] + d[[m][i]][[m][i+1]] + d[[m][j]][[m][j+1]]);$$

✓ The gain function for the 1-1 Intra-Route swap is:

$$\gamma = (d[[m][i-1]][[m][j]] + d[[m][j]][[m][i+1]] + d[[m][i]][[m][j+1]] + d[[m][i]][[m][j-1]]) - (d[[m][i-1]][m][i]] + d[[m][i]][[m][i+1]] + d[[m][j-1]][[m][j]]) + d[[m][j]][[m][j+1]];$$

Neighbourhood Reduction Strategy

All local search routines are executed in an all exhaustive fashion; however a Neighbourhood Reduction Strategy (NR) is put in place in order to decrease the size of allowable moves. Some common ways to do NR is to use the real distances between customers, or some Savings criteria. In this research a Parallel Clustering method was performed for initial solution generation. It was not used as an initial solution generation method, because of the longer computational time. However, the results from the agglomeration schedule (the Sum of Squares between customer nodes) from the clustering are used for NR. The way it works is in a probabilistic fashion. Each of the customer nodes is assigned a probability of how good a certain Shift or Swap would be with another customer node (for an exchange of a node with itself a large number is assigned). A small example with 5 customers is given in Figure 5.6.

Distances

	0	1	2	3	4	5
0	0	13.9	21	32.6	17.2	14.1
1	13.9	0	12.4	19.2	31.1	22.2
2	21	12.4	0	15.3	37	21
3	32.6	19.2	15.3	0	49.7	36.1
4	17.2	31.1	37	49.7	0	20.4
5	14.1	22.2	21	36.1	20.4	0

Neighbourhood Reduction

	0	1	2	3	4	5
0	99	0.15	0.3	0.35	0.25	0.15
1	0.15	99	0.15	0.1	0.55	0.5
2	0.3	0.15	99	0.05	0.75	0.45
3	0.35	0.1	0.05	99	1	0.9
4	0.25	0.55	0.75	1	99	0.4
5	0.15	0.5	0.45	0.9	0.4	99

Figure 5.6: Distances and Neighbourhood Reduction Probabilities

The total Sum of Squares (SST) is calculated for each node being joined with any of the other nodes. The scale of the SST has then been transformed into a probability scale from 0 to 1. The formula used to transform the SST scale into probability is given below, where the new value is computed based on the minimum and maximum values of the old range and the new range, respectively:

$$New\_Value = (((Old\_Value - Old_{\min}) * (New_{\max} - New_{\min}) / (Old_{\max} - Old_{\min})) + New_{\min};$$

Let us have look at an optimal solution generated by CPLEX. Figure 5.7 shows the optimal solution for the RVRP instance without overtime with *n = 20 L = 10%.* It can be seen that the most favourable links recognised in the partial NR schedule are reflected in the solution. Node 0 is connected to node 1, and also node 5 is connected to node 0. Moreover, the links which results in highest probability in the NR schedule are not connected, such as nodes 3 and 4, nodes 3 and 5, and nodes 4 and 2. In fact, looking at Figure 5.7 these nodes are parts of different routes.

Route 1:	0-1-8-3-2-16-11-0
Route 2:	0-12-4-19-13-18-0
Route 3:	0-14-20-7-6-0
Route 4:	0-17-15-10-9-5-0

Figure 5.7: Instance *n=20, L=10%* without overtime

As discussed in Section 4.3, sometimes the largest distances between customers can be a part of the best found solutions and even optimal solutions. Therefore, the probability of shifting customers is enforced in a diminishing fashion, similar to the cooling schedule of Simulated Annealing explained in Chapter 2. During the local search routines at the beginning of the search the probability threshold for allowable moves begin with 1 and after each iteration decreases by 0.1 until the probability of 0.3 is reached or the solution has not been improved. The NR strategy is not global, but local to the routines, which means that once the solution enters a new local search routine the probability of moves starts from 1 again and diminishes accordingly. This speeds up the computational time of the routines, since it reduces the number of feasible shifts and swaps and focuses the search to emphasize good links between customers. This strategy has been slightly amended in the extension of the PVNS_AMP where Tabu Search and short term memory is incorporated. This will be further explained in the next chapter.

The reason why we chose to diminish the NR down to 0.3 rather than 0 is portrayed in Figure 5.8. The Figure shows a small example of the behaviour of the NR during a 1-1 Inter-Route Shift. The routine is executed 10 times with different probability threshold starting from 1 down to 0.1.

Neighbourhood Reduction for 1-1 Inter-Route Shift										
NR Probability	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Feasible Moves	346	306	238	188	134	104	72	47	19	0
Promising Moves	21	21	21	20	20	20	19	16	8	-
Solution Quality	326.6	326.6	326.6	326.6	329	329	330.3	336	354.1	-

Figure 5.8: Illustrated example of the NR

Figure 5.8 shows the probability threshold at which the 1-1 Inter-Route Shift routine is performed, the total feasible moves at that threshold and the total promising moves at that threshold. Promising means those moves, which gain function γ could result in an improvement of the solution. The best solution achieved at the given conditions is also reported in the last row. It can be seen that with the decrease of NR probability, the feasible moves and the promising moves decrease, which also decreases the computational time. However, the solution quality decreases as well.

It is interesting to see from the figure that the solution quality remains unchanged until the probability reaches 0.7 and it starts to degrade as it approaches 0. Therefore, the probability starts off at 1 and diminishes until 0.3 during the iterations of the algorithm for each of the explored candidate solutions. After 0.3 the solution quality degrades significantly. During Stage 2 of the algorithm, however, the probability is fixed at 0.7 and it does not diminish. The reason for this is that the solution is already approximately 30% fixed by the *Elite Strings*, and any additional restrictions may "lock" the solution search. The idea of the diminishing parameter in Stage 1, is to begin with exploring all possible moves and gather knowledge about the good links within the routing schedule. As previously mentioned in Section 4.3 some of the longest (furthest) arcs between nodes can be part of the best known solution, so they should not be ignored. As the parameter diminishes with the local iterations it creates a "funnel" where the local routines explore more restricted set of arcs between the nodes and on the large instances it can also speed up computational time.

It is important to note that this probability is only local to the routines, and not global per candidate solution. The reason for this is that if the probability parameter diminishes with the

exploration of a candidate solution, it means that the *Elite Strings* will be biased towards those nodes with greater proximity in terms of total Sum of Squares. Therefore, in order to avoid this bias, the NR is active only locally within the search routines.

5.2.2. Stage 2 (PVNS_AMP Stage)

The PVNS_AMP stage is the Memory exploitation stage of the algorithm, where the knowledge gathered in Stage 1 is used to improve the solution quality. After the PVNS stage the *Elite Strings* are recognised, according to the pre-defined criteria. The Memory Initialization Pool is then reduced to the best 10 candidate solutions in terms of solution quality and the *Elite Strings* are encoded into them. If a candidate solution contains an *Elite String* it becomes a fixed part of the solution structure and it does not change during further neighbourhood search. The *Elite Strings List* holds the *Elite Strings* and it is of dynamic length, because in the different data instances, different number of *Elite Strings* can be recognised from the Memory Exploitation Pool *M.* Only the solution sequences, which have frequency of occurrence in the candidate solutions of 75% and higher become *Elite Strings*. When encoding the *Elite Strings* into the solutions, those with highest frequency have priority. However, the proportion of *Elite Strings* which are encoded into a solution is limited to up to 30%. The remaining nodes are the variable part of the solution, which can be further amended in Stage 2 via the local search routines. The manner in which the *Elite Strings* are encoded into the solution is further explained in Section 5.3.

The candidate solutions from the Memory Exploitation Pool *M* enter Stage 2 in a systematic fashion in ascending order in terms of their objective function. The operators and execution of the VNS search is the same as in the PVNS stage, but only the variable part of the solution is modified via the Shift and Swap operators. Having a proportion of the solution, which remains unchanged, also acts as a neighbourhood reduction technique, and speeds up the computational time of the operators. The *Elite Strings* remain fixed during the *Shake* stage as well. However, in Stage 2 of the algorithm, the *Shake* does not degrade the solution too much, where only one customer is randomly reassigned to a different route. This provides intensification of the search, but keeps the focus of the search in better regions.

The population-based nature of the VNS (the survival of a number of candidate solutions) in Stage 2 is very important for diversification. The candidate solutions which enter Stage 2 of

the algorithm are quite diverse in terms of solution structure; hence they contain different *Elite Strings* and provide for a better coverage of the solution space. The diversity of candidate solutions is desired for addressing the RVRP at hand, because different proportions of light load customers can feature in the problem. Also the opportunity to explore for allowable overtime requires more flexibility.

Another reason for working with a population of candidate solutions with *different Elite Strings* is that allowable overtime is coupled with unlimited fleet. This means that a candidate solution can either favour an extra vehicle, or allow for overtime. The variable cost of travel of the vehicles ranges from 0.36 to 0.48, which means that it is 36% or 48% of the travel cost respectively. The overtime has 1.5 times higher cost than the non-overtime travel, which means it is incurs 50% greater cost. This can make the cost of using overtime relatively higher than the cost of using an extra vehicle. Given this characteristic of the problem, it is expected that during the neighbourhood search, a first improvement move into the overtime will not typically be an improving move. Therefore, keeping a pool of solutions with different fleet composition and allowance for overtime provides for greater coverage of the search space.

There are many interesting observations made regarding the RVRP and the behaviour of the algorithm, which are detailed in Section 5.3, with extensive analysis of results and recommendations for improvement of the vehicle routing practice.

5.3. Method Justification and Parameter Testing

This section provides an overview of some of the important methodological justifications regarding the PVNS_AMP. It also shows that having a population VNS has a positive impact on the effectiveness of the algorithm. Moreover, it shows details of the *Elite Strings* encoding and the usefulness of having AMP as a learning mechanism. Each methodological choice has been tested and found best for the researched RVRP.

Elite Strings Encoding

The proportion of *Elite Strings* incorporated into the solutions in Stage 2 of the algorithm is an important methodological consideration. After the *Elite Strings* are recognized from Stage 1, the *Elite Strings* are encoded in the candidate solutions which survive to Stage 2, hence they become a fixed part of the solution. There is a clear trade-off between the proportion of the solution that is fixed via *Elite Strings*, solution quality, and computational time. If a smaller

proportion of the solution is fixed, then the solution quality may not improve in Stage 2 as it is not focused enough into better search areas. Similarly, if too much of the solution is fixed, the *Elite Strings* may not in fact be elite, which can lead the search to explore a region that is falsely recognised as good. Also, the computational time decreases as the proportion of *Elite Strings* increases in the solution.

Figure 5.9 illustrates this trade-off and shows that when the solution contains up to 30% *Elite Strings*, is sufficient for good memory exploitation. Another interesting observation is the fluctuation of the solution quality at different levels of *Elite Strings* encoding. It only fluctuates less than 4%, which suggests good quality extraction of *Elite Strings* even with up to 60% coverage of the solution. The example portrayed in Figure 5.9 is an instance with 100 customers with and without overtime. Both versions are graphically represented in order to show consistency in the algorithmic behaviour. Another observation shown in Figure 5.9 which is important is the decreasing computational time (for problems with and without overtime) as the Proportion of *Elite Strings* increases.

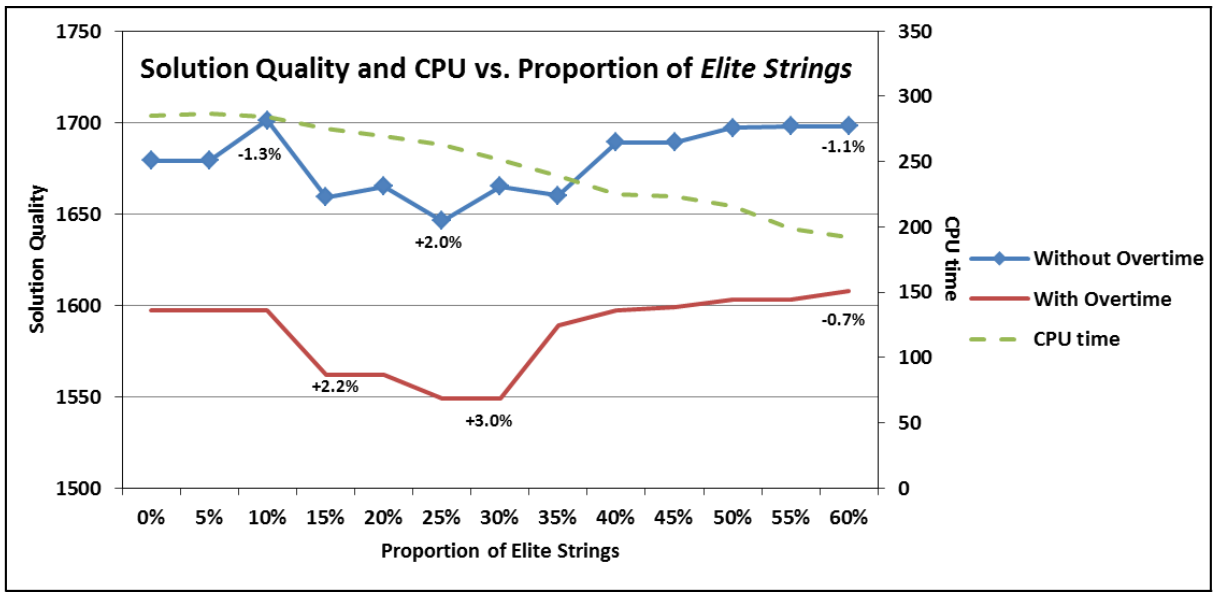

Figure 5.9: Solution Quality and CPU time vs. proportion of *Elite Strings*

The population nature of the algorithm has another benefit when it comes to *Elite Strings* encoding. It can happen that an *Elite String*, which is recognised as elite, may in fact not be an *Elite String*. This is because the PVNS_AMP is a heuristic method, not exact, which means that one cannot guarantee that the extracted *Elite Strings* are indeed a promising part of the

solution. However, working with a population of candidate solutions allows for overcoming this possible drawback.

This issue is illustrated in Figure 5.10, which shows two candidate solutions for an instance with 20 customers without overtime which survive from Stage 1 and contain encoded *Elite Strings.* The string 8-7-6 which is recognised as elite is present in the candidate solution annotated with **(b)**. However, looking at the optimal solution obtained from Cplex it is clear that it is in fact not an *Elite String*, since it is not present in the optimal solution. The candidate solution annotated with **(a)**, however contains two *Elite Strings*, which are also part of the optimal solution. When designing a metaheuristic method, one should be aware of the strengths and weaknesses of that method, and employ strategies to emphasize the strengths and overcome possible weaknesses. In this case the population nature of VNS allows to an extent for overcoming the possibility of false *Elite Strings* recognition.

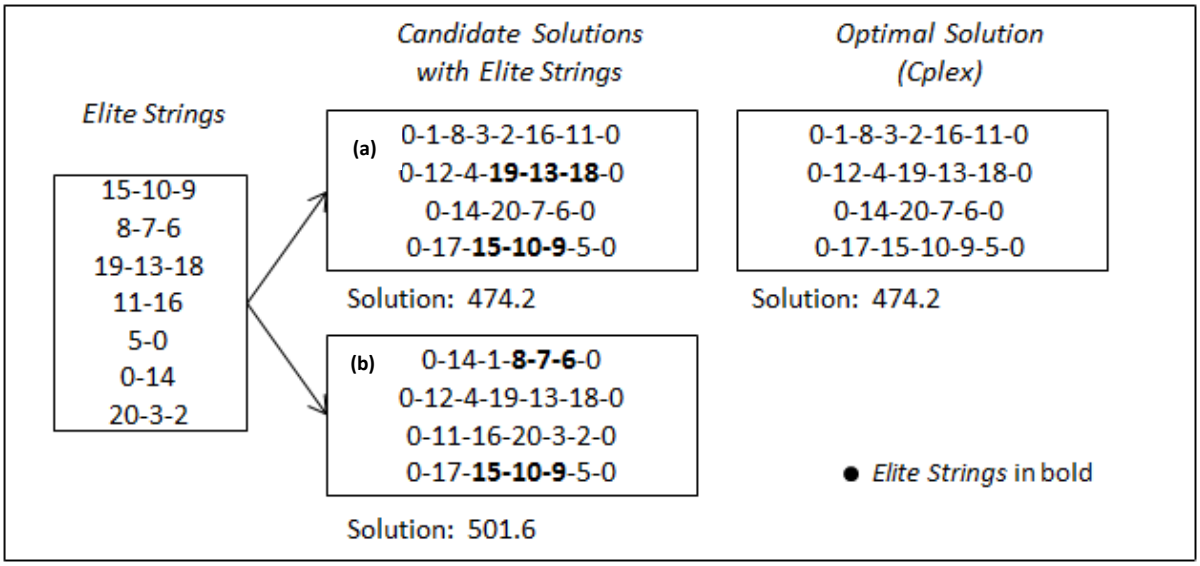

Figure 5.10: Encoding of *Elite Strings* into candidate solutions

Shake Stage

The number of customers to be re-assigned in the *Shake* stage is an important consideration for the algorithm. The method already has some diversification embedded into the pool of solutions to be explored and it is vital that the *Shake* stage has a good performance and supports the intensification of the search. Too much *Shake* can distort the solution structure beyond repair for the next iterations, and too little shake can fail to intensify the search.

Experiments were performed and it was found that re-assigning 3 customers at random in the PVNS stage (Stage 1) is sufficient for finding good quality solutions in the PVNS_AMP stage (Stage 2). Table 5.2 shows detailed computational results for the *Shake* stage on all RVRP instances without overtime. It can be seen from the table that the solution quality is quite stable for the different *Shake* For example looking at instance *N = 20, L = 10%* the solution quality is the same with 1, 2, 3 and 4 random shifts, namely 446.2 . However, having 1 or 2 shifts may not be sufficient for a good *Shake*, because it does not provide enough room for linking back good solution sequences in the subsequent search. In some of the instances the best found solutions are not reached when 1 or 2 re-assignments are performed. It is interesting to note that in most cases the best found solutions from the PVNS_AMP (detailed later in this Chapter) are reached. However, this could be due to the diverse pool of solutions, since during the computational experience it was found that the best found solutions are reached from a different candidate solution. CPU time for the shifts is not provided as all random re-assignments take fractions of a second.

Table 5.2: Experimentation of the *Shake* stage with different random re-assignments

N	L	1 random shift		2 random shifts		3 random shifts		4 random shifts	
		PVNS	PVNS_AMP	PVNS	PVNS_AMP	PVNS	PVNS_AMP	PVNS	PVNS_AMP
20	10%	446.2	446.2	446.2	446.2	446.2	446.2	446.2	446.2
20	15%	446.9	446.9	446.9	446.9	446.9	446.9	446.9	446.9
20	20%	462.3	462.3	462.3	462.3	462.3	462.3	462.3	462.3
30	10%	569.3	560.1	569.3	560.1	569.3	560.1	569.3	560.1
30	15%	569.3	560.1	569.3	560.1	569.3	560.1	569.3	560.1
30	20%	565.3	565.3	565.3	565.3	565.3	565.3	565.3	565.3
50	10%	901.2	865.3	901.2	865.3	879.1	852.2	879.1	852.2
50	15%	921.1	874.6	923.3	874.6	882.3	867.2	882.3	867.2
50	20%	903.9	877.4	903.9	877.4	903.9	877.4	903.9	877.4
75	10%	1298.3	1254.2	1301.9	1248.4	1269.5	1244.1	1269.5	1244.1
75	15%	1302.5	1265.8	1301.9	1278.5	1272.1	1254.3	1272.1	1254.3
75	20%	1311.2	1267.5	1322.6	1271.3	1292.4	1267.5	1292.4	1267.5
100	10%	1723.6	1689.2	1752.8	1702.6	1667.6	1646.4	1667.6	1646.4
100	15%	1775.9	1712.5	1762.3	1723.2	1744.1	1689.9	1744.1	1689.9
100	20%	1798.3	1721.5	1798.3	1721.5	1798.3	1705.3	1798.3	1705.3

Figure 5.11 shows some further insight into the solution quality during the *Shake* stage. We performed different number of random shifts ranging from 1 to 10. Different sized instances are used for consistency, namely with 20, 50 and 100 customers. The same behaviour was found for the problems with overtime as well. The X axis shows the number of random shits performed, whereas the Y axis shows the solution quality (objective function value) for the respective instance.

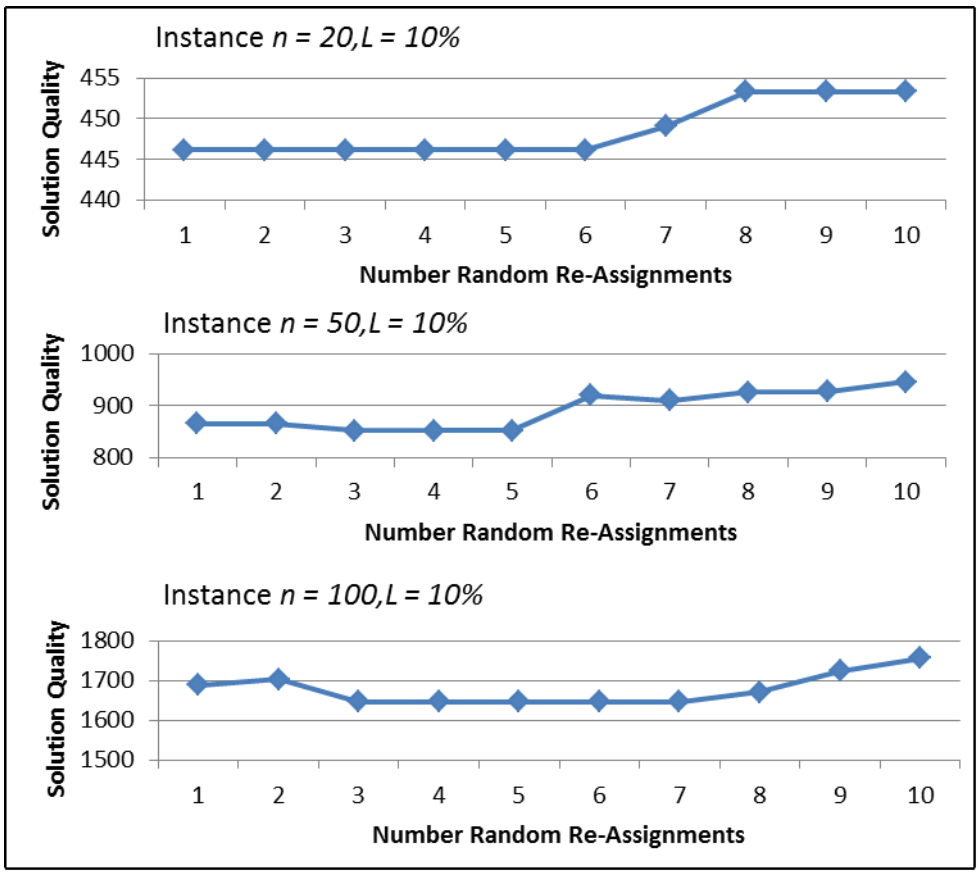

Figure 5.11: Shake Experimentation on the RVRP without overtime

Figure 5.11 shows that the solution quality has a degree of stability up to 5 random re-assignments (shifts). Moreover, it is clear from the line graphs that adopting a 3 shift re-assignment for the Shake, is sufficient for finding good quality solutions. If too many shifts are performed (more than 5) it could be difficult to re-connect all the broken links during the Shake and re-build a good solution, hence the increase of the solution quality, whereas if 1 or 2 shifts are performed it could be insufficient for good intensification of the search.

Learning Mechanism

The PVNS_AMP is mainly motivated by the idea that any solution method, which does not have memory structures in its original form, can be enhanced by learning. Therefore it is important to show the benefit of learning and memory exploitation. Figure 5.12 shows that there is a benefit from using the AMP as a learning strategy of VNS. It is clear from the figure that the solution quality from the Stage 1 fluctuates more during the runtime of the algorithm, whereas in Stage 2 it is more stable and more focused in lower topography. This is because the fixed part of the solution (i.e. the *Elite String)*, is providing for a better stability of the search.

Figure 5.12 shows the fluctuation of the solution quality during the runtime of the algorithm and another interesting observation comes to light. Given the fact that the candidate solutions enter the VNS stage in ascending order based on their objective function quality, it can be seen that the best solution in the PVNS_AMP stage (Stage 2) was reached towards the end of the running time of the algorithm. This means that it was reached by a candidate solution from $S' \in E$ with larger objective function.

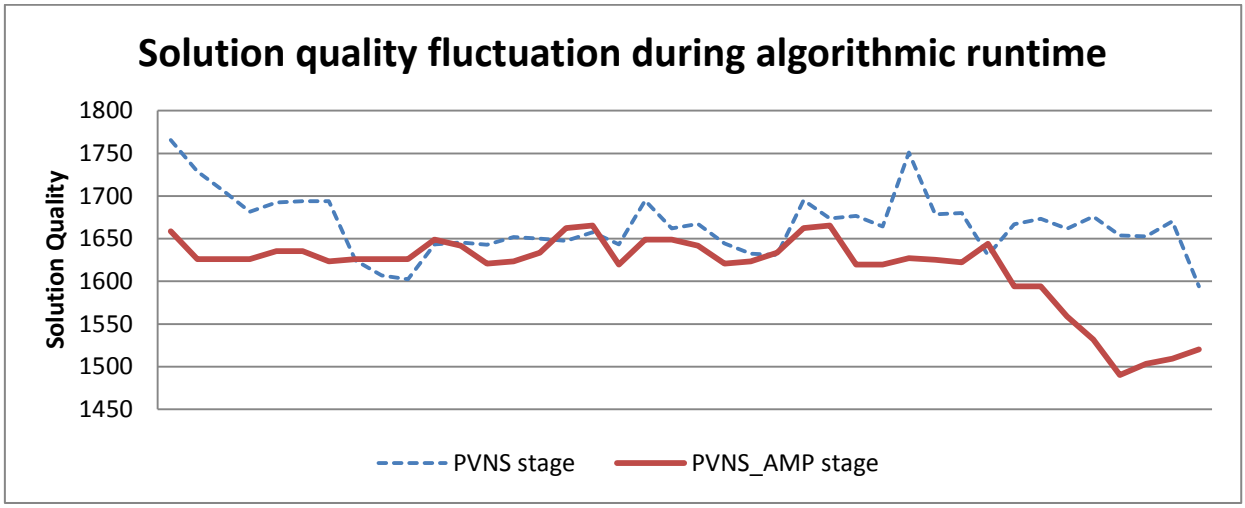

Figure 5.12*:* RVRP with Overtime *N=100, L=10%*

This is an important observation when it comes to problems with overtime. The local search routines used in the PVNS_AMP involve shifting a maximum of two customers at a time. Therefore, a shift of customers into the overtime part of the route may not result in an improvement of the cost and the move will be deemed non-improving. In the instance portrayed in Figure 5.12, the best found solution is from a candidate solution which has

allowable overtime and during further exploration via the local search operators the best heuristic solution for this instance was found.

5.4. PVNS_AMP Results on the RVRP

This section provides computational results for the PVNS_AMP applied to the RVRP with and without overtime. First of all we show the benefit of having AMP hybridized with the PVNS and how the learning mechanisms improve the solution quality. Second we show the results from the PVNS_AMP as compared to the results we achieved with CPLEX. Moreover, this section provides an analysis of the results and some interesting practical implications for the RVRP. It can be seen from the results that cost savings can be achieved, as well as a much better utilization of the vehicle capacity and the drivers' working hours.

The RVRP without Overtime

The RVRP is first solved using the MIP formulation provided in Section 3.2 in CPLEX Version 12.6. The results are then compared to those from the proposed PVNS_AMP in order to show algorithmic efficiency. Table 5.3 shows the results achieved by Cplex and those achieved by the PVNS_AMP for both stages, with the corresponding CPU time. The total CPU time (TCPU) of each stage is reported, as well as the time to the best found solution (BCPU). The last column shows the % improvement in solution quality when AMP is incorporated into the PVNS. There are a few observations that can be made from Table 5.3. First, looking at the TCPU for both stages, it can be seen that generally the higher the proportion of light load customers, the smaller the TCPU. This is a valid observation, because the higher the number of light load customers, the smaller the search space becomes, which restricts the local search allowable moves. The BCPU confirms the observation made in Figure 5.12, that some of the best solutions are found towards the end of the total runtime for both stages of the algorithm. This emphasizes the benefit of using Population VNS. Second, the objective function can either increase with the increased proportion of light load customers, or it can remain unchanged. This is an interesting observation, which is further explained in Table 5.4 using instance *n = 20,* without overtime and different light load customer composition. The light load customers are shown with underlined script.

Table 5.3: Computational Results for the RVRP without overtime

N	L	CPLEX Results				PVNS Results				PVNS_AMP Results				% IMP
		LB/Optimal	UB	Time	Fleet Mix	Solution	TCPU	BCPU	Fleet Mix	Solution	TCPU	BCPU	Fleet Mix	
20	10%	**446.2**	-	4	3A 1B	**446.2**	3	<1	3A 1B	**446.2**	5	<1	3A 1B	0.00%
20	15%	**446.9**	-	3	3A 1B	**446.9**	3	<1	3A 1B	**446.9**	5	<1	3A 1B	0.00%
20	20%	**462.3**	-	3	3A 1B	**462.3**	2	<1	3A 1B	**462.3**	4	<1	3A 1B	0.00%
30	10%	**560.1**	-	640	2A 3B	569.3	5	2	2A 3B	**560.1**	10	<1	2A 3B	1.62%
30	15%	**560.1**	-	640	2A 3B	569.3	5	2	2A 3B	**560.1**	12	<1	2A 3B	1.62%
30	20%	535.9	575.4	375	-	565.3	4	2	2A 3B	565.3	10	<1	2A 3B	0.00%
50	10%	701.1	901	1830	-	879.1	16	5	8A 2B	852.2	28	5	6A 2B	3.06%
50	15%	706.8	958.2	248	-	882.3	15	10	5A 5B	867.2	28	5	4A 5B	1.71%
50	20%	699.4	N/A	1109	-	903.9	13	7	6A 4B	877.4	26	3	6A 4B	2.93%
75	10%	993.1	1541	971	-	1269.5	36	17	2A 8B	1244.1	62	20	2A 8B	2.00%
75	15%	985.9	1391	1658	-	1272.1	33	8	7A 5B	1254.3	63	15	6A 5B	1.40%
75	20%	985.9	N/A	2662	-	1292.4	31	15	8A 6B	1267.5	58	7	8A 6B	1.93%
100	10%	1274.6	2908	1396	-	1667.6	75	20	13A 5B	1646.4	103	48	13A 5B	1.27%
100	15%	1248.4	2844	1930	-	1744.1	79	36	13A 6B	1689.9	99	86	12A 6B	3.11%
100	20%	1247.4	N/A	322	-	1798.3	62	62	9A 9B	1705.3	95	75	10A 8B	5.17%

L is Proportion of L $\subseteq N$
LB/Optimal shows Optimal Solution in bold
Time is Cplex computational time in minutes
TCPU is Total runtime of the corresponding stage in seconds
BCPU is Time to best found solution for the corresponding stage in seconds
IMP is Improvement when AMP is added to PVNS

Table 5.4: Routing Schedule for RVRP with different light load customers

Light Load Customers	None	2,11	2,10	1,10,15	1,10,15,8
Routes	0-1-8-3-2-0 0-5-15-10-9-16-11-0 0-14-20-7-6-0 0-18-13-19-4-17-12-0	0-1-8-3-2-0 0-5-15-10-9-16-11-0 0-14-20-7-6-0 0-18-13-19-4-17-12-0	0-1-8-3-2-0 0-12-15-10-9-11-16-0 0-14-20-7-6-0 0-18-13-19-4-17-5-0	0-5-11-16-2-3-1-0 0-12-17-15-10-9-0 0-14-20-7-8-6-0 0-18-13-19-4-0	0-5-11-16-2-3-1-0 0-12-17-15-10-9-0 0-14-20-7-8-6-0 0-18-13-19-4-0
Fleet Mix	3A,1B	3A,1B	3A,1B	3A,1B	3A,1B
Solution	446.16	446.16	476.04	462.32	462.32
TCPU	5 sec	5 sec	5 sec	4 sec	4 sec

The first example in the Table 5.4 is the base routing schedule if there are no light load customers. When the light load customers are chosen such that in the base schedule they are serviced after the light load capacity threshold, then the objective function and routes do not change. However, if the chosen customers are positioned before the light load threshold is reached in the base schedule, then an adjustment in the routing is necessary and the objective function increases accordingly. The fact that the base routing schedule is mostly preserved in the solutions with different light load customers means that the PVNS_AMP method can recognise good quality *Elite Strings* which preserve good solution sequences, whilst adjusting for the light load requirement. This addresses one of the important real life considerations of the RVPR, which is to incorporate light load customers into the daily routing in an efficient manner.

The RVRP with Overtime

Our computational experience suggests that the problem is computationally demanding and only very small instances are solved to optimality. The RVRP with overtime results from The PVNS_AMP are shown in Table 5.5, as compared to the results achieved by CPLEX.

The second important real life consideration for the RVRP is to show whether there can be any savings from considering allowable overtime in advance. The results from the RVRP with overtime are compared to those without overtime in Table 5.6. The total cost is provided as well as the fleet composition for each instance and how much overtime is used, if any.

Table 5.5: Results of the RVRP with overtime

N	L	CPLEX Results				PVNS_AMP Results			
		LB/Optimal	UB	Fleet Mix	Time[1]	Solution	Fleet Mix	TCPU	Overtime[2]
18	10%	**390.3**	-	1A 2B	4	**390.3**	1A 2B	3	7
20	10%	413.8	451.1	-	63	427.2	1A 2B	5	5
20	15%	413.8	451.1	-	51	427.2	1A 2B	5	5
20	20%	418.1	448.3	-	84	427.2	1A 2B	3	5
25	10%	474.5	511.8	-	22	503.1	1A 3B	5	0
30	10%	504.9	586.1	-	31	547.2	4B	7	49
30	15%	504.9	586.1	-	30	547.2	4B	7	49
30	20%	503.7	584.7	-	21	552.6	4B	7	58
50	10%	699.1	-	-	32	820.3	3A 4B	25	17

[1] Computational time in minutes
[2] Overtime present in the solution in minutes

Incorporating overtime shows the potential for cost savings up to 8% for one planning period (see Table 5.6). The saving is not only in terms of overall cost, but also in terms of fleet size. The RVRP proposed in this research has an interesting characteristic which became apparent during the computational experience. Having allowable overtime and unlimited fleet means that it is very likely that during the search process some candidate solutions could favour an extra vehicle, as opposed to allowing for overtime. Therefore, having a Population VNS allows for the exploration of solutions which favour overtime and solutions which favour extra vehicles, hence a more comprehensive picture for the possibilities for cost savings.

For the instances without overtime, the fleet is larger. This is because the allowable maximum regular time in some cases restricts the RVRP more tightly than the capacity constraint. That is, a new route is added either when the maximum time is reached or there is no more capacity left in the vehicle. This is an important aspect of the routing in the gas delivery industry, because the time it takes to service a customer (the demand-dependent nature of the service time) and the time is takes to travel between customer locations (given the lower speed of the heavy goods vehicles) is relatively large.

Table 5.6: Results on the RVRP with and without Overtime

N	L (%)	PVNS_AMP without overtime		PVNS_AMP with overtime			IMP (%)*
		Solution	Fleet Composition	Solution	Total overtime	Fleet Composition	
20	10	446.2	3A 1B	427.2	5	1A 2B	4.42
20	15	446.2	3A 1B	427.2	5	1A 2B	4.42
20	20	462.3	3A 1B	427.2	5	1A 2B	7.59
30	10	560.1	2A 3B	547.2	49	4B	2.30
30	15	560.1	2A 3B	547.2	49	4B	2.30
30	20	565.3	2A 3B	552.6	58	4B	2.25
50	10	852.2	6A 2B	820.3	27	3A 4B	3.74
50	15	867.2	4A 5B	827.1	36	3A 4B	4.62
50	20	877.4	6A 4B	842.1	46	3A 4B	4.02
75	10	1244.1	2A 8B	1230.5	19	4A 6B	1.09
75	15	1254.3	6A 5B	1241.9	7	2A 8B	0.99
75	20	1267.5	8A 6B	1253.3	62	3A 7B	1.12
100	10	1646.4	13A 5B	1549.4	25	3A 10B	5.89
100	15	1689.9	12A 6B	1579.1	29	3A 11B	6.56
100	20	1705.3	10A 8B	1592.1	38	3A 11B	6.64

*% Improvement when incorporating Overtime

Table 5.7: RVRP at a glance

	RVRP no Overtime	RVRP with Overtime		
L	**None**	**None**	**1,5,7,12,9**	**1,4,5,7,12,9,32,42,45,50**
Routing	0-6-***24-43-40-7-23-48***-0	0-1-22-28-31-26-8-48-27-0	0-6-14-***24-43-40-7-23-48***-27-0	0-6-14-***24-43-40-7-32-48***-27-0
	0-14-25-13-18-0	*0*-***32-2-20***-35-36-3-0	0-8-26-31-28-1-0	0-8-26-31-28-1-0
	0-22-28-31-26-8-27-0	0-6-*23-7*-***40-43-24***-25-14-0	0-22-3-36-35-***20-2-32***-46-0	0-22-3-36-35-***20-2-32***-0
	0-11-38-46-0	0-***15-45-33-39-10***-49-5-46-0	0-11-16-29-21-34-50-9-49-5-12-0	0-11-16-29-21-50-34-30-9-49-38-0
	0-1-3-36-35-***20-2-32***-0	0-11-16-29-21-50-34-30-9-38-0	0-38-30-***10-39-33-45-15***-0	0-***10-39-33-45-15***-5-0
	0-9-30-34-50-21-29-16-0	0-37-44-42-19-41-13-18-0	46-37-17-***4-47-0***	0-17-37-44-42-19-41-4-12-0
	0-5-49-***10-39-33-45-15***-37-0	***0-47-4***-17-12-0	44-42-19-41-13-25-18-0	0-47-18-13-25-0
	0-12-17-44-42-19-41-***4-47-0***	-	-	-
Fleet Mix	**5A, 3B**	**1A, 6B**	**3A, 4B**	**3A, 4B**
AT	**360.2**	**379.6**	**398.3**	**410.1**
AVC	**0.31%**	**0.31%**	**0.31%**	**0.31%**
AL	**14116**	**15149**	**15828**	**16132**
Overtime	**0**	**0**	**27**	**46**
Solution	**848.3**	**825.6**	**820.3**	**842.1**

Underlined nodes are light load customers, nodes in bold are *Elite Strings*
AT is Average travel time per vehicle in minutes
AVC is Average variable cost per vehicle as a proportion of total cost
AL is Average load per vehicle
Overtime used in minutes in the solution

Table 5.7 provides an overview at a glance for the key aspects of the RVRP and their combined effect. It portrays instance *N = 50* with and without overtime and with different proportions of light load customers. The key observations are summarized below.

(i) It can be seen that the route composition is mostly preserved regardless of the overtime and the light load composition, as well as the *Elite Strings*. This means that the PVNS_AMP is flexible enough to identify good solution sequences for the different versions of the RVRP and the memory procedure has preserved those good sequences in a consistent manner.

(ii) The examples with overtime of the RVRP tend to favour the larger vehicle type B, which results in a smaller fleet. Here an interesting observation is that even though the fleet mix is composed of more vehicles of type B, the average variable cost of the vehicles remains unchanged. Additionally, when considering overtime, the vehicle capacity is 12.5% better utilized, because the average load carried by the fleet is greater when overtime is considered in advance. Moreover, the working time is 12% better utilized, as the average travel time of the fleet is higher and much closer to the maximum allowable regular time. This is an important practical aspect, in relation to drivers working hours' directive and management of human resources.

(iii) Another interesting observation is the combined effect of having light load customers and allowable overtime. It can be seen that for the RVRP with overtime and no light load customers, the objective function is 825.6, with 6 vehicles of type B and only 1 of type A. In contrast, having *L=10%* with overtime has an objective function of 820.3. This means that having light load customers can actually improve the efficiency of the routing when overtime is allowed. It provides an opportunity for servicing more light load customers on a given route after the maximum regular time, when there is still capacity left; rather than placing them on a different route.

The allowable overtime in the RVRP is coupled with unlimited fleet. This means that an extra vehicle could be favoured in the solution structure, as opposed to having allowable overtime. This trade-off is an interesting feature of the problem and it has not been explored before. It can be seen from Table 5.6 that in the instances with allowable overtime the fleet is actually smaller than the instances without overtime, in terms of size. For the different instances the fleet size of the instances with overtime is up to 42% smaller. However, the larger vehicle

type B has been favoured. In the instances without overtime the fleet composition is almost balanced, with 56% of the fleet on average being vehicles of type A, and 44% being vehicles of type B. In contrast looking at the instances with overtime, the fleet composition is 26% of the vehicles are of type A, and 74% of type B. However, it can be noted from Table 5.7, that even if the fleet favours large vehicles, it does not comes at an increased variable cost on average. In fact, it prevents the formation of short routes, which only include a few or one customer at a vehicle route. This is important in real life routing because the full time drivers should utilize their working hours, in order for the company to make the most of their regular paid time. Moreover, if the drivers are on temporary basis or agency based, they may not agree to do short shifts, since it is not feasible in terms of monetary reward.

5.5. PVNS_AMP Testing on Standard Benchmark Instances

RVRPs are diverse and characterized with many real life routing elements which are typical in a given industry setting. Usually the RVRPs proposed in the literature are tested on specially designed instances, adapted literature benchmark instances or real datasets. Similar to most heuristic methods, the solution methods designed to solve RVRPs are problem specific. Moreover, they are not tested on well-known literature benchmark instances, because one cannot directly compare methods designed for different problems. However, we believe that it is important for researchers who address real life problems to be able to find a platform for comparability and show that the solution methods have a degree of generalizability. In our case one way of showing that the PVNS_AMP method can successfully address the RVRP is to compare the results to the optimal solutions or lower/upper bounds found by solving the MIP formulation. Moreover, we test the PVNS_AMP on the well-known literature benchmark instances by Golden et al. (1984), both with variable cost and fixed cost, even though the researched RVRP here does not consider the fixed cost of the vehicles. The results from the PVNS_AMP are shown in Table 5.8. They are compared to the Best Known Solutions (BKS) found in the literature. Some of the tables with computational experiments show an improvement from one version of the PVNS_AMP to another denoted by IMP, whereas the term Gap is used to show the deviation from any BKS.

Table 5.8 shows the results on the FSMVRP instances with variable cost and fixed cost, solved by the PVNS_AMP. Even though the method is designed for a RVRP it performs well on one of the most researched problem instances, yielding less than 1% deviation on average.

Table 5.8: PVNS_AMP Results on Golden et al. (1984)

Instance	N	FSMVRP with Variable Cost			FSMVRP with Fixed Cost		
		BKS	PVNS_AMP	Gap	BKS	PVNS_AMP	Gap
3	20	623.22	623.22	0.00%	961.03	961.03	0.00%
4	20	387.18	387.18	0.00%	6437.33	6437.33	0.00%
5	20	742.87	742.87	0.00%	1007.05	1007.05	0.00%
6	20	415.03	415.03	0.00%	6516.47	6516.47	0.00%
13	50	1491.86	1491.86	0.00%	2406.36	2406.36	0.00%
14	50	603.2	603.2	0.00%	9119.03	9119.03	0.00%
15	50	999.8	999.8	0.00%	2586.37	2612.1	0.99%
16	50	1131	1131	0.00%	2720.43	2750.1	1.09%
17	75	1038.6	1061.2	2.18%	1734.53	1758.02	1.35%
18	75	1800.8	1852.1	2.81%	2369.65	2401.43	1.34%
19	100	1105.44	1139.2	3.05%	8661.81	8709.1	0.55%
20	100	1530	1560.2	1.96%	4032.81	4087.1	1.35%
		Average Gap		**0.83**%	Average Gap		**0.56**%

In order to present more detailed results, the best solution of 10 iterations has also been recorded. There is a degree of randomisation of the PVNS_AMP and this has to be evened out by executing a number of iterations and taking the best found solution. Also average solutions are recorded.

Further experiments are performed on the benchmark instances with fixed and variable cost. The PVNS_AMP was run for 10 iterations on each instance and the results are presented in Tables 5.9 and 5.10. It can be seen from both tables that the average results are not far from the best known solutions, which suggests a relatively stable performance. There is improvement of the performance from 1 run to 10 runs with 0.68% and 0.47% for the instances with variable cost and fixed cost respectively.

An observation here can be made that the small gaps in best found solutions and the BKS suggest that the diversification element of the algorithm is sufficient for a good coverage of the solution space and the candidate solutions have enough diversity to explore solution structures with different fleets. However, the fact that 2 of the BKS are not reached in the

case of the instances with variable cost and 3 of the instances with fixed cost, means that the intensification element of the PVNS_AMP can be further improved.

Table 5.9: Results on Golden et al. (1984) with Variable Cost

Instance	N	1 iteration	10 iterations		
		PVNS_AMP	PVNS_AMP best	PVNS_AMP Average	% IMP*
3	20	623.22	623.22	623.22	0.00%
4	20	387.18	387.18	387.18	0.00%
5	20	742.87	742.87	742.87	0.00%
6	20	415.03	415.03	415.03	0.00%
13	50	1491.86	1491.86	1491.86	0.00%
14	50	603.2	603.2	603.2	0.00%
15	50	999.8	999.8	999.8	0.00%
16	50	1131	1131	1131	0.00%
17	75	1061.2	1038.6	1042.9	2.18%
18	75	1852.1	1822.1	1836.7	1.65%
19	100	1139.2	1112.25	1121.6	2.42%
20	100	1560.2	1530	1532.5	1.97%
				Average IMP	**0.68%**

*IMP is improvement from the PVNS_AMP with 1 iteration

Table 5.10: Results on Golden et al. (1984) with Fixed Cost

Instance	N	1 iteration	10 iterations		
		PVNS_AMP	PVNS_AMP best	PVNS_AMP Average	% IMP*
3	20	961.03	961.03	961.03	0.00%
4	20	6437.33	6437.33	6437.33	0.00%
5	20	1007.05	1007.05	1007.05	0.00%
6	20	6516.47	6516.47	6516.47	0.00%
13	50	2406.36	2406.36	2406.36	0.00%
14	50	9119.03	9119.03	9122.8	0.00%
15	50	2612.1	2586.37	2586.37	0.99%
16	50	2750.1	2720.43	2732.1	1.09%
17	75	1758.02	1749.53	1751.2	0.46%
18	75	2401.43	2368.65	2389.43	1.34%
19	100	8709.1	8661.8	8683.3	0.55%
20	100	4087.1	4043.12	4058.2	1.09%
				Average IMP	**0.47%**

*IMP is improvement from the PVNS_AMP with 1 iteration

Further experimentation was performed with the PVNS_AMP in order to test behaviour of the algorithm. We run the PVNS_AMP with different stopping criteria, namely maximum computational time. As previously noted, the PVNS_AMP shows a good potential for

diversification of the solution space. However, we also test the method for a longer period of time in order to see any gains in solution quality. Once Stage 1 of the algorithm is complete and the *Elite Strings* are encoded, we run Stage 2 of the PVNS_AMP for 2 hours on all benchmark instances from Golden et al. (1894). The results are shown in Tables 5.11 and 5.12, with variable and fixed cost, respectively.

Table 5.11: Results on Golden et al. (1984) with Variable Cost

Instance	N	PVNS_AMP	PVNS_AMP average of 10 iterations		PVNS_AMP 2 hrs maximum running time	
			Sol	% IMP	Sol	% IMP*
3	20	623.22	623.22	0.00%	623.22	0.00%
4	20	387.18	387.18	0.00%	387.18	0.00%
5	20	742.87	742.87	0.00%	742.87	0.00%
6	20	415.03	415.03	0.00%	415.03	0.00%
13	50	1491.86	1491.86	0.00%	1491.86	0.00%
14	50	603.2	603.2	0.00%	603.2	0.00%
15	50	999.8	999.8	0.00%	999.8	0.00%
16	50	1131	1131	0.00%	1131	0.00%
17	75	1061.2	1038.6	2.18%	1038.6	2.18%
18	75	1852.1	1822.1	1.65%	1800.8	2.81%
19	100	1139.2	1112.25	2.42%	1105.4	3.06%
20	100	1560.2	1530	1.97%	1530	1.97%
			Average IMP	**0.68%**	**Average IMP**	**0.84%**

Table 5.12: Results on Golden et al. (1984) with Fixed Cost

Instance	N	PVNS_AMP	PVNS_AMP average of 10 iterations		PVNS_AMP 2 hrs maximum running time	
			Sol	% IMP	Sol	% IMP*
3	20	961.03	961.03	0.00%	961.03	0.00%
4	20	6437.33	6437.33	0.00%	6437.33	0.00%
5	20	1007.05	1007.05	0.00%	1007.05	0.00%
6	20	6516.47	6516.47	0.00%	6516.47	0.00%
13	50	2406.36	2406.36	0.00%	2406.36	0.00%
14	50	9119.03	9119.03	0.00%	9119.03	0.00%
15	50	2612.1	2586.37	0.99%	2586.37	0.99%
16	50	2750.1	2720.43	1.09%	2720.43	1.09%
17	75	1758.02	1749.91	0.46%	1734.53	1.35%
18	75	2401.43	2369.65	1.34%	2369.65	1.34%
19	100	8709.1	8661.81	0.55%	8661.81	0.55%
20	100	4087.1	4043.2	1.09%	4043.2	1.09%
			Average IMP	**0.47%**	**Average IMP**	**0.53%**

*IMP is improvement from the PVNS_AMP

What these 2 tables show is that there is an improvement if the PVNS_AMP is run for an average of 10 iterations and when run for longer period of time, namely 2 hours. This shows a key observation regarding the PVNS_AMP, and that is that the best results are achieved when the algorithm is run for longer, which means with more in depth search of the candidate solutions. Another observation is that the results on the instances with variable cost are better on average, reaching all Best Known Solutions in the literature. This can be attributed to the fact that when fixed cost is present the shifts between fleet compositions incur greater costs and may not be further explored during the search.

5.6. Summary

The computation experience of the PVNS_AMP suggests that incorporating learning principles within a method which does not have learning mechanisms in its original form can yield some good results. The testing shows that the PVNS_AMP has a good potential of recognizing good solution sequences, which are often part of the optimal solution for a given problem instance. The population nature of the PVNS_AMP also acts as a driver for the good *Elite Sting* extraction as well as for providing diversification to the search process. The PVNS_AMP was successfully applied to the RVRP under study and showed potential for great operation savings in terms of total cost and better fleet utilization. Moreover, it was found to perform well on literature benchmark instances with less than 1% deviation from best known solutions. The PVNS_AMP is especially powerful when run for a longer period of time. 2 hours maximum running time can be considered reasonable for the generation of a routing schedule in practice.

After the extensive literature search on the VRP domain, it was found that usually AMP is used via means of Tabu Search. Therefore, the next Chapter offers an extension to the PVNS_AMP, where the main principles of Tabu Search are incorporated into the method. The idea is that by using TS, the intensification element of the algorithm will be better addressed and this could lead to an improvement in the algorithm performance.

CHAPTER 6

Tabu Search Aspect of the PVNS_AMP

The literature review conducted in this research suggests that AMP is usually used within Tabu Search, and some of the best solutions found on the VRP literature benchmarks involving AMP is via means of Tabu Search. One of the main ideas for algorithm design in this research was to implement learning principles using AMP, within a method which does not make use of memory in its original form. However, we cannot ignore the findings from the literature that AMP performs well when used with Tabu Search.

Some Tabu Search principles are incorporated within the PVNS_AMP which resulted in a new hybrid metaheuristic method, namely the TS_PVNS_AMP. We created this hybrid method in order to test for any improvements in the solution quality, as well as further the research aspect on the methodological side. AMP is used in this research as a long term learning mechanism, which was found to perform well on the RVRP, as well as on literature benchmark instances. The addition of the TS aspect is done for the purpose of adding short-term learning strategy. It is interesting to test whether there is any gain in terms of solution quality when this short-term memory aspect is added and how the TS aspect fits within the algorithmic design.

The previous chapter details the results from the test instances, where one main observation can be made regarding the performance of the PVNS_AMP algorithm. The results and testing suggest that the PVNS_AMP has a good diversification strategy, where the benefit of having population of solutions and a vigorous *Shake* stage has been described. However, there is one aspect that can be further improved, namely intensification. What we found in Chapter 6 is that the PVNS_AMP performs better when run for a longer period of time, which suggests that the performance is better with a more in-depth search of the solution space. The short-term memory aspect of the TS can be a good intensification strategy, because during the search process for better solutions, some of the moves are tabu. This means that using tabu status for certain moves can help to prevent solution cycling in the short-term. The motivation behind adding the TS aspect to the PVNS_AMP algorithm is to see whether this can enhance

the intensification strategy, hence it may lead to finding better solutions for the RVRP, as well as on literature benchmark instances. This algorithm extension is explained and tested in detail within this chapter and some interesting observations are reported. Hereafter we refer to the new extended algorithm as TS_PVNS_AMP. For some further detailed description on Tabu Search, please refer to Section 2.3.2.

6.1. The TS_PVNS_AMP Method Description

The TS aspect for the PVNS_AMP is not globally incorporated. This means that the TS aspect is only active within the local-search routines used within the PVNS_AMP. It is locally embedded in each Inter-Route routine, namely the 1-0 shift, 1-1 swap, 2-0 shift, 2-1 swap and 2-2 swap, as well as the Intra-Route routines, namely 1-1 swap and 1-0 Shift.

The global methodological steps, described in Section 5.2 remain the same; that is all details relating to the methodological Steps for Stage 1 and Stage 2 of the PVNS_AMP, as well as the *Shake* routine. Therefore, this section will only offer a description of the local TS aspect, which relates to the Shift and Swap routines. The reason why the global algorithmic steps are not changed is because the TS aspect is mostly added in order to add more intensification to the search process, which was outlined in the previous chapter as an aspect which can be further enhanced. Moreover, it is added in order to strengthen the short-term learning aspect of the method.

The Shift and Swap operators within the PVNS_AMP are used in a systematic fashion, until the best solution is reached. This aspect was changed for the TS_PVNS_AMP, in order to explore more deeply the surroundings of a given local optima. A graphical representation of the Shift and Swap operator methodology is given in Figure 6.1.

Figure 6.1 shows a description on how the moves are accepted, as well as the stopping criteria within all of the Shift and Swap operators. If the operator finds an improving move, it is accepted and the new solution updated. Data structures are used to memorize the 2 best feasible non improving moves (that is second best and third best to the current best), which is a hill-climbing mechanism. The reason why we allow hill-climbing moves is from our computational experience with the PVNS_AMP. We found in Section 5.3 (shown in Figure 5.12) that some of the best found solutions come from a candidate solution with a larger

objective function. Therefore, allowing for hill-climbing moves within the TS_PVNS_AMP can be beneficial for reaching better solutions.

The 2 best improving infeasible moves are also kept in the memory and are used to continue the search if no further improving move was found. For the purpose of the RVRP, a non-improving move is defined as a move which results in capacity or maximum allowable time violation. The light load constraint is kept feasible at all times. For the testing of the algorithm on literature benchmark instances, an infeasible move is defined as a move which results in capacity violation. Two parameters are used in Figure 6.1, which refer to the allowable non-improving and infeasible moves. *Inf* counts the number of times an infeasible move was accepted, whereas *Sec* counts the number of times a non-improving feasible move was accepted (hence the second best). *Imp* denotes an improving move. The '++' sign denotes an increase in the number of accepted moves. A simple pseudo code for the Tabu Search is also provided in Figure 6.2.

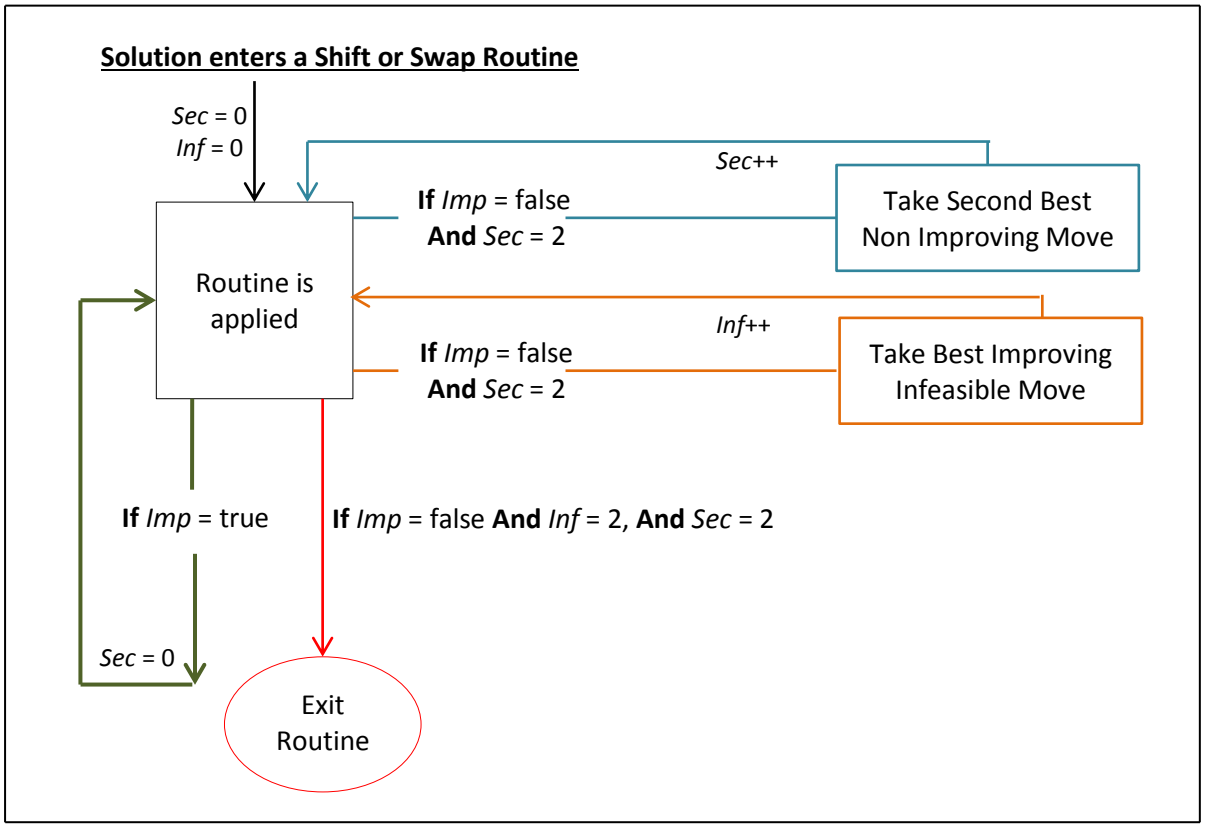

Figure 6.1: Graphical Representation of the Operators execution within TS_PVNS_AMP

✓ The improving moves are those which result in an improvement of the current best solution. Once an improvement move is made, the inverse move is set as Tabu, in order to

prevent the solution search to go back to the same structure for a number of iterations. This means that if an arc between two customers is dropped, because an improving arc was found, then the recently dropped arc does not get revisited in future iterations.

✓ The non-improving moves are those moves which result in a second best solution quality, relative to the current best found improving move. If a better second best solution is found than the one currently saved in the memory, it gets replaced. Two second best moves are memorized and used if no improving moves are found. This is motivated by the hill-climbing mechanism, where non-improving moves are accepted, in order to intensify the search and ideally force it into better regions. The number of non-improving moves allowed was empirically tested and is described later in this section.

Begin Routine
Calculate gain function γ for all moves
Save all moves with $\gamma < v_{\max}$
Initialize Tabu List, **Set** *Sec* = 0, *Inf* = 0;
Perform Move
If *Imp* = true and Move is not **Tabu** (or if **Tabu** activate **aspiration criteria**)
Accept Improving Move; **Reset** *Sec*=0;
Set inverse move as **Tabu**
Update Tabu List
Else if *Imp*=false And *Sec*<2
Accept hill-climbing move
Sec++;
Else if *Imp*=False And *Sec* = 2
Accept Infeasible Move;
Else If *Imp*=false, *Sec*=2, *Inf*=2
Save best found Solution
End If
Exit Routine

Figure 6.2: TS_PVNS_AMP pseudo code

✓ Infeasible moves are also accepted during the search process. The motivation behind it is that usually the optimal solution for Combinatorial Optimization problems lies on the convex hull (a small note here is that this is not the case for all problems), which is essentially the boundary between the feasible and the infeasible space. Therefore, allowing for infeasible moves can force the search to explore this boundary in more depth. An example of this is given in Figure 6.3. It has to be noted here that tabu status is not given to any accepted infeasible or hill-climbing moves.

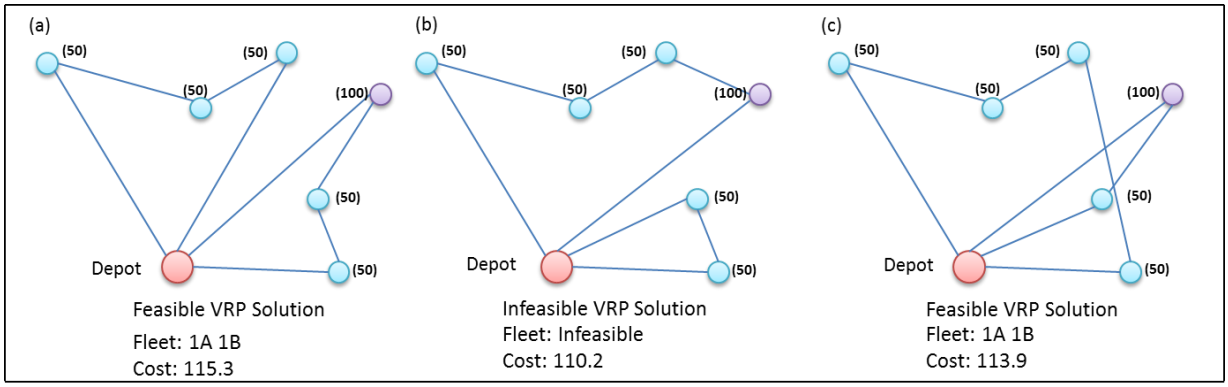

Figure 6.3: Shift from infeasible move to feasible move

Figure 6.3 shows a small sample instance with 6 customers and a depot, and two vehicle types Type A has capacity of 150 and type B has capacity of 200. The node depicted in purple is a customer with a large demand. Starting from a feasible solution (a) during the search process an infeasible solution in terms of capacity (b) with a better cost can be found and accepted, which can then be further explored to find feasible solution (c) which results in an improvement of the total cost. There is no guarantee that allowing for infeasibility or non-improving moves can results in a better overall solution quality. Therefore some results on the Shift and Swap operators are portrayed in Table 6.1, with some further explanation and observations. The table is generated based on the FSMVRP instance with fixed cost by Golden et al. (1984), instance number 3, with $n = 20$. The observations made from testing instance 3 are valid for all other instances as well. Table 6.1 shows not only the benefit of having non-improving and infeasible moves, but it also gives an idea of the execution sequence and the variable number of iterations each of the routines can have. Only the operators which are Inter-Route are shown in the table, because the Intra-Route operators do not result in infeasibility.

Similarly to the PVNS_AMP, the TS_PVNS_AMP does not have a fixed number of iterations within the local search routines, but they are executed until no further improvement is found, according to the algorithm steps listed in Figure 6.2. Therefore, it is interesting to see from Table 6.1 the manner in which the routines are executed, where the solution quality during the corresponding routine is noted per iteration.

Table 6.1: Solution Quality Change during each iteration of the operator execution

Inter-Route Shift and Swap Routines									
0-1		1-1		2-0		2-1		2-2	
Move	Solution	Move	Solution	Move	Solution	Move	Solution	Move	Solution
Imp	1068	Imp	1056	Imp	1038	Imp	1084	Imp	1061
Imp	1066	Sec	1058	Imp	1024	Imp	1075	Sec	1069
Imp	1067	Sec	1061	Sec	1026	Imp	1072	Imp	1056
Sec	1119	Inf	1048	Sec	1038	Imp	1070	Imp	1039
Imp	1028	Imp	**1042**	Inf	998	Sec	1075	Imp	**1023**
Imp	**990**	Sec	1058	Inf	980	Sec	1112	Sec	1056
Inf	998	Sec	1058	Imp	**1018**	Inf	965	Sec	1067
Inf	980	-	-	Sec	1042	Imp	1002	Inf	1044
-	-	-	-	Sec	1058	Imp	**1001**	Inf	1012
-	-	-	-	Inf	1003	Sec	1025	-	-
-	-	-	-	Inf	980	Sec	1054	-	-
-	-	-	-	-	-	Inf	1021	-	-
-	-	-	-	-	-	Inf	946	-	-
Best Solution	**990**	Best Solution	**1042**	Best Solution	**1018**	Best Solution	**1001**	Best Solution	**1023**

Imp - Improving Move

Sec – Hill Climbing Move

Inf - Infeasible Improving Move

Table 6.1 shows the solution quality of a solution which enters the corresponding routine. The solution is noted after each iteration of the routine. An iterations here refers to a full exploration of a given solution structure of all moves which result in improvement according to the gain function γ described in Section 5.2.1.

The first column shows the changes in the solution quality for the 0-1 Inter-Route Shift. It can be seen that there are 3 improving moves accepted until no further improving move was found. After accepting the second best move (hill climbing move) there was still no improvement to the best found solution, however accepting the next second best non-improving move, resulted in two consecutive improvements of the objective function. This means that accepting the non-improving move forced the search into a better area. There was no improvement of the solution after the infeasible moves were accepted; therefore the best found solution from the 6^{th} iteration was accepted as the best for that routine. In the cases of 1-1 swap and 2-0 and 2-1 shift, the best improving move was found after an infeasible move was accepted. This shows that there is a benefit for allowing infeasible moves, which can intensify the search for better solution quality. Also, the fleet composition remains variable during the search, which gives greater flexibility for the operators. This means that an infeasible move is only infeasible if the total capacity exceeds the capacity of the largest available vehicle.

Another important observation here is the number of iterations per routine. The number of iterations is not fixed, but it depends on the quality of the search process. In the PVNS_AMP the iterations within the routines are also variable, but they are as many as the improving moves found. In this case, where non-improving and infeasible moves are allowed, there are more iterations, until no further improvement is found. The stopping criteria is the following: if there is no improvement and two non-improving and two infeasible moves have already been accepted, then the routine is exited, with the best found solution or if no improvement was found the solution remains unchanged from when it entered the routine. Therefore, the minimum number of iterations is 4 (assuming we accept 2 hill-climbing moves and 2 infeasible moves with no improving move), where the maximum depends on the number of improving moves. This can pose a strain on the computational time, if too many iterations for one routine are allowed. Therefore, only 2 non-improving and infeasible moves are allowed following an improving move. We call this parameter η, which is tested, in order to analyse

any trade-off between the number of allowable non-improving feasible moves vs. solution quality. These experiments are shown in Figure 6.4.

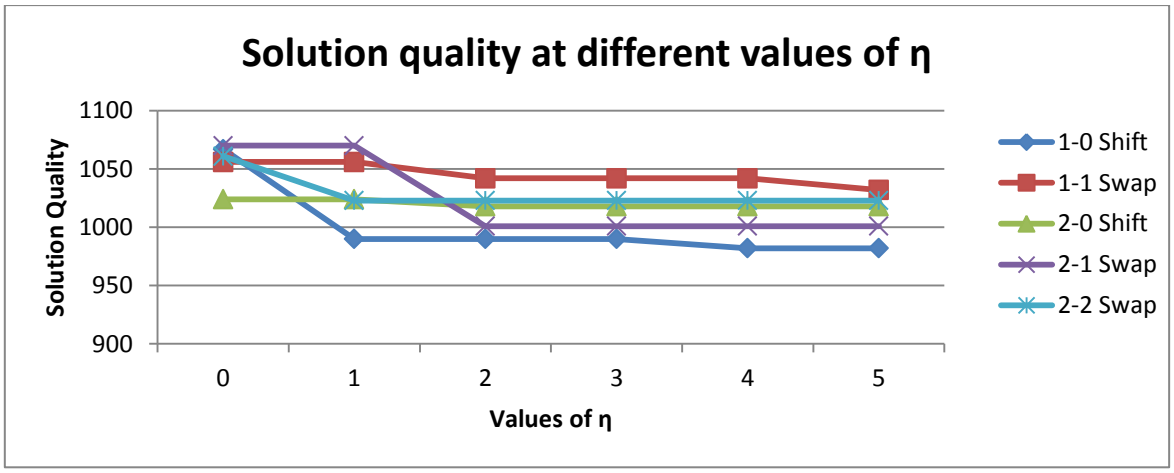

Figure 6.4: Solution Quality change during local search routines

For the experiments shown in Figure 6.4 the random seeding (generation of the initial solutions) was turned off, in order to have consistent findings and a more accurate representation of any trade-offs on solution quality relevant to the η parameter. Each of the routines was executed first, after an initial solution was generated. The reason for this is for each routine to work on the 'raw' initial solution. It can be seen from the figure that the solution quality for the 2 node shift and swaps does not improve after 2 allowable moves. However, the solution quality for the 1-0 Shift and 1-1 Swap does improve with the increase with the η parameter. However, when a large instance is used and the algorithm is run, adding more iterations to the routines increases computational time. Moreover, one cannot conclude that more allowable infeasible and non-improving moves will have a global effect on the solution quality. Indeed if the η parameter is larger, it will improve the solution quality locally for the routine. However, we need to show the effect on the entire algorithm, in order to decide on the best value for the η parameter. This is shown in Figure 6.5. Instance 20 with *n = 100* from the Golden et al. (1984) benchmark instances with variable cost is shown in figure 6.5, because the real impact of the parameter can be noted using a larger instance (the smaller instances with *n = 20* for example can be quickly reached with our method). Figure 6.5 shows the best found solution from the TS_PVNS_AMP (run of the entire algorithm) tested with different values of the η parameter.

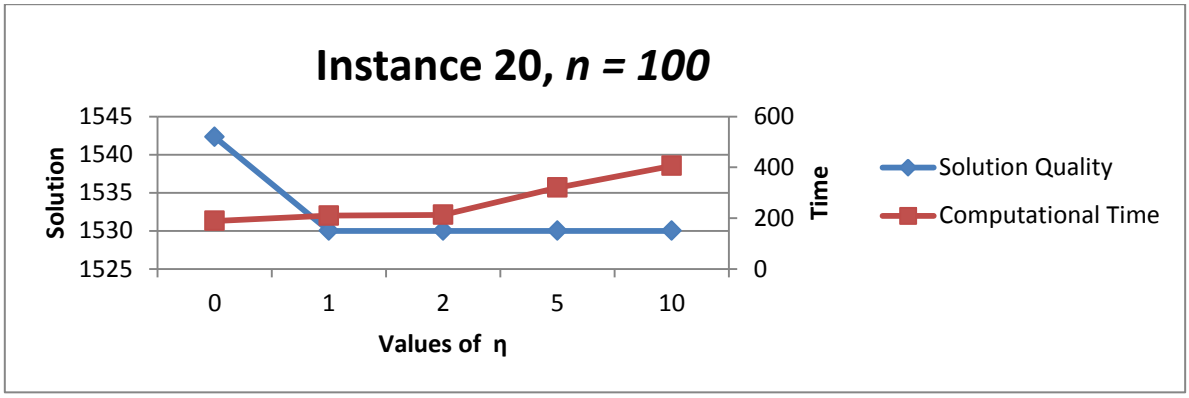

Figure 6.5: Solution quality and Computational time at different values of η

It can be seen from Figure 6.5 that a value of the parameter η = 2 is sufficient for good solution quality at a minimal increase of computational time (compared to η = 0). If the value of the parameter is increased up to 10, there is no improvement of the solution and the computational time increases significantly by 93%.

✓ The Tabu List is another important consideration. The recently visited moves are added to the Tabu List for a number of iterations, which are governed by the tabu tenure value. Here different tabu tenure values are tested for each of the routines, as well as for the entire algorithm. The reason for this is that when applied locally to the routines, different tabu tenure can be proven best for the specific routine. For instance, the 1-0 Inter-Route shift results in more improving changes to the solution structure on average compared to the 2-1 routine. The 1-0 Shift is a very powerful operator for finding better quality solutions. Therefore, larger tabu tenure may be more feasible for a routine which involves higher number of iterations. In contrast, if a routine does not have too many iterations a large tabu tenure will be 'absolute' for that routine. What this means is that having a tabu tenure = 7 means that for the next 7 iterations a given move is not allowed. However, if on average the routine does not produce more than 7 iterations, then a tabu tenure of 7 will mean that this move will never become available.

The way we chose the tabu tenure value in this research is based on the average iteration of the Shift and Swap operators. After running different instances for the RVRP and literature benchmark instances, we found that the average length of a routine (length means number of iterations within a routine) is 10. We chose the tabu tenure value to be half of the average iterations per routine, hence a value of 5. The reason for this is to allow for a move that has

been tabu to be removed from the tabu list and become available again within the life of the given routine. We use different tabu list for each routine and having a common value for the tabu tenure decreases the number of parameters used within the algorithm. Moreover, having a value based on average iterations across the routines prevents the tabu status of becoming 'absolute', as described earlier. We believe that having a different tabu list for each routine aids the intensification of the search. Let us take an example of executing two different routines with the same tabu list for all routines portrayed in Figure 6.6.

Candidate Solution		**1-0 Inter-route Shift**		**2-0 Inter-route Shift**		**1-0 Inter-route Shift**	
Route 1:	0-1-3-**5-7**-0	Route 1:	0-1-3-**5**-0	Route 1:	0-4-1-3-**5**-0	Route 1:	0-4-1-3-**5-7**-0
Route 2:	0-4-8-0	Route 2:	0-4-8-**7**-0	Route 2:	0-8-**7**-0	Route 2:	0-8-0
Route 3:	0-2-6-9-10-0	Route 3:	0-2-6-9-10-0	Route 3:	0-2-6-9-10-0	Route 3:	0-2-6-9-10-0
Route 4:	0-0	Route 4:	0-0	Route 4:	0-0	Route 4:	0-0
Total cost:	**423.1**	**Total cost:**	**412.5**	**Total cost:**	**409.7**	**Total cost:**	**401.6**

Figure 6.6: Changes in a candidate solution with Tabu Status

When the candidate solution enters the 1-0 shift the best improving move is moving customer 7 to Route 2. After the 2-0 shift is performed the solution structure changes again. If we execute the 1-0 shift again after the 2-0 shift the best move is for customer 7 to link back with customer 5. However, this was the first accepted move and it is tabu. This means that having one list for all routines will prevent that move from being accepted, unless the aspiration criterion is enforced. In this research we do make use of the aspiration criterion, though we found that in the case of our hybrid algorithm it does not have an effect on the final solution quality. Having the tabu tenure active within a routine and local to that routine, means that a smaller value for the tabu tenure can be used and often tabu moves become available again within the routine. The PVNS_AMP already has learning principles embedded within, such as the *Elite String* generation. Moreover, we work on a number of candidate solutions, therefore if one link is unexplored within one candidate solution, it could be explored in another and this will be reflected in the *Elite String* frequency. This suggests that the hybridization of the TS with a PVNS_AMP combines different aspects (strengths) of the methods in a complementary fashion, and there may not be a need to use all of the parameters a method has if used on its own.

The purpose of the aspiration criterion is to override the tabu status of a move which is tabu but results in an improvement in the solution quality. We found that there is no impact on the

final solution if the aspiration criterion is applied or not. Therefore, motivated by this rationale we use aspiration criteria for the Neighbourhood Reduction (NR) strategy. If a move is improving at any point during Stage 1 of the algorithm and it has a probability within NR which is greater than 0.7, we override that probability and reduce it to 0.3. This allows for any further routines to explore this move in more depth if it is indeed promising, and this act as the link between the learning from one routine to the next, since NR is a global strategy throughout the algorithm.

6.2. TS_PVNS_AMP Method Testing

The extension to the PVNS_AMP with Tabu Search (TS) is tested on the same problem instances as the PVNS_AMP, including the proposed RVRP and the FSMVRP instances by Golden et al. (1984). Table 6.2 and 6.3 detail the results of the RVRP with and without overtime, and tables 6.4, 6.5 and 6.6, the FSMVRP instances with variable cost and fixed cost.

Table 6.2: Results on the RVRP without Overtime

N	L	PVNS_AMP		PVNS_AMP with TS		
		Solution	CPU (sec)	Solution	CPU (sec)	IMP
20	10%	446.2	3	446.2	7	0.0%
20	15%	446.9	3	446.9	7	0.0%
20	20%	462.3	2	462.3	6	0.0%
30	10%	560.1	5	560.1	12	0.0%
30	15%	560.1	5	560.1	13	0.0%
30	20%	565.3	5	565.3	11	0.0%
50	10%	852.2	27	852.2	39	0.0%
50	15%	867.2	25	867.2	35	0.0%
50	20%	877.4	23	877.4	28	0.0%
75	10%	1244.1	58	1244.1	78	0.0%
75	15%	1254.3	58	1248.8	78	0.4%
75	20%	1267.5	56	1267.5	72	0.0%
100	10%	1646.4	120	1631.6	172	0.9%
100	15%	1689.9	120	1680.2	171	0.6%
100	20%	1705.3	118	1705.3	149	0.0%
				Average IMP		**0.13%**

*CPU time in Seconds

Table 6.2 shows a 0.13% improvement of the best found solutions by the PVNS_AMP, when TS is added. The improvements are mostly on the larger instances and are less than 1% for each of the problems. This shows that the TS can further improve the solutions, because it has more power when it comes to intensification of the search.

Table 6.3: Results for the RVRP with Overtime

N	L	PVNS_AMP			PVNS_AMP with TS			IMP
		Solution	Total overtime	Fleet Mix	Solution	Total overtime	Fleet Mix	
20	10%	427.2	5	1A 2B	427.2	5	1A 2B	0.0%
20	15%	427.2	5	1A 2B	427.2	5	1A 2B	0.0%
20	20%	427.2	5	1A 2B	427.2	5	1A 2B	0.0%
30	10%	547.2	49	4B	547.2	49	4B	0.0%
30	15%	547.2	49	4B	547.2	49	4B	0.0%
30	20%	552.6	58	4B	552.6	58	4B	0.0%
50	10%	820.3	27	3A 4B	**812.9**	10	4A 4B	0.9%
50	15%	827.1	36	3A 4B	827.1	36	3A 4B	0.0%
50	20%	842.1	46	3A 4B	842.1	46	3A 4B	0.0%
75	10%	1230.5	19	4A 6B	**1228.1**	19	4A 6B	0.2%
75	15%	1241.9	7	2A 8B	**1210.4**	32	3A7B	2.6%
75	20%	1253.3	62	3A 7B	**1238.7**	23	3A7B	1.2%
100	10%	1549.4	25	3A 10B	**1503.8**	81	5A 8B	3.0%
100	15%	1579.1	29	3A 11B	**1552.6**	46	5A 8B	1.7%
100	20%	1592.1	38	3A 11B	**1567.3**	72	4A 10B	1.6%
					Average IMP			**0.75%**

*CPU time in Seconds

The results from the problems with overtime are very interesting in terms of average improvement. It can be seen from Table 6.3 that adding the TS aspect can result in up to 3% improvement in the best found solutions by PVNS_AMP. This can be attributed to the allowable infeasibility and the greater intensification provided by TS. It was noted in Section 5.6 that the problem with overtime is very challenging when it comes to accepting improving moves which enter the allowable overtime, because usually adding an extra vehicle can be a better option in terms of overall cost. However, the TS aspect allows for non-improving moves, where more than one customer can enter the overtime without an immediate improvement of overall cost. However, when the solution is further explored, more customers are shifted to the overtime, which results in significant improvement of the overall cost. Allowing infeasible moves is very favourable when it comes to problems

with overtime, which are coupled with unlimited fleet, because it allows for a greater flexibility for exploring the solution space within the allowable overtime. Compared to the results without overtime, where only minor improvements are achieved, due to the better intensification of the TS and the short term learning, the problem with overtime shows superiority of the TS_PVNS_AMP. Moreover, looking at the Fleet Mix reported for the TS_PVNS_AMP, it can be seen that the fleet is much more balanced, where more vehicles of type A are used in the final solution. This means that capacity of the smaller vehicles is better utilized, as well as their travel time.

The methods are also tested on the Golden et al. (1984) Benchmark instances with heterogeneous fleet with fixed and variable cost against the BKS and other relevant methods. They are shown in tables 6.4, 6.5 and 6.6. It can be seen from the tables that the TS_PVNS_AMP reaches competitive results on both sets with average deviation less than 0.01% and 0.04% respectively and achieves better results than the PVNS_AMP. For the problem with fixed cost, the TS_PVNS_AMP achieved better results than the Tabu Search proposed by Brandao (2011) and the Genetic Algorithm proposed by Liu (2009) with 0.08% and 0.01% respectively. This supports the findings from the literature that AMP works well when used with Tabu Search. This also shows that the TS provides greater intensification to the search process as well as supports the AMP with the short term learning provided by the tabu tenure. When PVNS_AMP is hybridized with Tabu Search, the algorithm becomes more powerful in terms of solution quality.

Another important observation is that the computational time for the TS_PVNS_AMP is not much larger than the one of the PVNS_AMP. This can suggest an efficient programming effort and it shows that TS can be implemented successfully, without placing too much strain on the computational time. Moreover, given the fact that the TS_PVNS_AMP is a population based algorithm, the computational time is relatively faster than the GA proposed by Liu (2009), as well as the Tabu Search proposed by Brandao (2011).

6.4 Table: Results on Golden et al. (1984) FSMVRP with variable cost

Instance	N	BKS	PVNS_AMP			TS_PVNS_AMP		
			Sol	Gap*	CPU	Sol	Gap*	CPU
3	20	623.22	623.22	0.00%	10	623.22	0.00%	35
4	20	387.18	387.18	0.00%	9	387.18	0.00%	32
5	20	742.87	742.87	0.00%	8	742.87	0.00%	36
6	20	415.03	415.03	0.00%	9	415.03	0.00%	28
13	50	1491.86	1491.86	0.00%	32	1491.86	0.00%	69
14	50	603.2	603.2	0.00%	35	603.2	0.00%	58
15	50	999.8	999.8	0.00%	33	999.8	0.00%	63
16	50	1131	1131	0.00%	37	1131	0.00%	61
17	75	1038.6	1061.2	2.13%	78	1038.6	0.00%	142
18	75	1800.8	1852.1	2.77%	72	1806.3	0.31%	121
19	100	1105.44	1139.2	2.96%	121	1105.44	0.00%	201
20	100	1530	1560.2	1.94%	112	1530	0.00%	213
			Average Gap	**0.82%**		**Average Gap**	**0.04%**	

*CPU time in Seconds

Table 6.5: Results on Golden et al. (1984) FSMVRP with fixed cost against other methods

Instance	N	BKS	TSA1[1]			GA[2]			ILS-RVND-SP[3]			TS_PVNS_AMP		
			Sol	Gap	CPU	Sol	Gap	CPU	Sol	Gap	CPU	Sol	Gap	CPU
3	20	961.03	**961.03**	0.00%	21	**961.03**	0.00%	21	**961.03**	0.00%	0	**961.03**	0.00%	32
4	20	6437.33	**6437.33**	0.00%	22	**6437.33**	0.00%	18	**6437.33**	0.00%	0	**6437.33**	0.00%	29
5	20	1007.05	**1007.05**	0.00%	20	**1007.05**	0.00%	13	**1007.05**	0.00%	0	**1007.05**	0.00%	28
6	20	6516.47	**6516.47**	0.00%	25	**6516.47**	0.00%	22	**6516.47**	0.00%	0	**6516.47**	0.00%	31
13	50	2406.36	**2406.36**	0.00%	145	**2406.36**	0.00%	91	**2406.36**	0.00%	2	**2406.36**	0.00%	65
14	50	9119.03	**9119.03**	0.00%	220	**9119.03**	0.00%	42	**9119.03**	0.00%	2	**9119.03**	0.00%	56
15	50	2586.37	2586.84	0.02%	110	**2586.37**	0.00%	48	**2586.37**	0.00%	6	**2586.37**	0.00%	52
16	50	2720.43	2728.14	0.28%	111	2724.22	0.14%	107	**2720.43**	0.00%	4	**2720.43**	0.00%	55
17	75	1734.53	1736.09	0.09%	322	1734.53	0.00%	109	**1734.53**	0.00%	12	**1734.53**	0.00%	99
18	75	2369.65	2376.89	0.31%	267	**2369.65**	0.00%	197	**2369.65**	0.00%	12	**2369.65**	0.00%	124
19	100	8661.81	8667.26	0.06%	438	8662.94	0.01%	778	**8661.81**	0.00%	25	**8667.26**	0.06%	269
20	100	4032.81	4048.09	0.38%	601	4038.46	0.14%	1004	**4032.81**	0.00%	46	4034.3	0.04%	237
			Average Gap	**0.09%**		**Average Gap**	**0.02%**		**Average Gap**	**0.00%**		**Average Gap**	**0.01%**	

Brandao (2011)[1]

Liu (2009)[2]

Subramanian et al. (2012)[3]

*CPU in seconds

Table 6.6: Results on Golden et al. (1984) FSMVRP with fixed cost

Instance	N	BKS	PVNS_AMP			TS_PVNS_AMP		
			Sol	Gap	CPU	Sol	Gap	CPU
3	20	961.03	961.03	0.00%	7	961.03	0.00%	32
4	20	6437.33	6437.33	0.00%	6	6437.33	0.00%	29
5	20	1007.05	1007.05	0.00%	5	1007.05	0.00%	28
6	20	6516.47	6516.47	0.00%	6	6516.47	0.00%	31
13	50	2406.36	2406.36	0.00%	29	2406.36	0.00%	65
14	50	9119.03	9119.03	0.00%	31	9119.03	0.00%	56
15	50	2586.37	2612.1	0.99%	28	2586.37	0.00%	52
16	50	2720.43	2750.1	1.08%	28	2720.43	0.00%	55
17	75	1734.53	1758.02	1.34%	59	1734.53	0.00%	99
18	75	2369.65	2401.43	1.32%	52	2369.65	0.00%	124
19	100	8661.81	8709.1	0.54%	105	8667.26	0.06%	269
20	100	4032.81	4087.1	1.33%	98	4034.3	0.04%	237
			Average Gap	**0.55%**		**Average Gap**	**0.01%**	

*CPU in Seconds

6.3. Summary

Incorporating Tabu Search principles within the PVNS_AMP showed an improvement in the performance of the algorithm. The TS_PVNS_AMP improved the best found solutions for the PVNS_AMP for the real life VRP introduced in this research. Especially for the RVRP with overtime the TS_PVNS_AMP showed significant improvements. We attribute this to the greater degree of intensification provided to the search by adding a TS aspect, as well as to the allowance for hill climbing and infeasible moves. The problem with overtime, when coupled with unlimited heterogeneous fleet is very interesting, because there is a clear trade-off between choosing to use more overtime vs. using an extra vehicle. Allowing for infeasibility triggered a better exploration of the search into the allowable overtime and the results achieved by the TS_PVNS_AMP are superior to the PVNS_AMP.

The TS_PVNS_AMP also performs better on standard literature benchmark instances, reaching most of the best known solutions in the literature with less than half a percent on average. The next chapter focuses on the extensibility and generalizability of the TS_PVNS_AMP methodology to other VRP problems and any interesting observations in the behaviour of the algorithm are outlined.

CHAPTER 7

Generalization of the Methodology

The PVNS_AMP and the TS_PVNS_AMP were originally designed to address the real life VRP under study. One of the main ideas behind the design of these methods was the ability to work with diverse solution compositions in terms of sequence and fleet mix. The unlimited fleet is a key feature in the RVRP, which also brings the interesting trade-off of unlimited fleet vs. allowable overtime. Being able to work with a population of candidate solutions with different fleet composition is one of the main strengths of the PVNS_AMP algorithm, which also aids the learning aspect of the method. Therefore, it could be less suitable to address a VRP which has an imposed fleet, rather than assuming unlimited availability of the fleet. In this research we aim to thoroughly test our algorithm and learn from the strengths and weaknesses which become apparent during the testing. Therefore, the next paragraph will detail the experimentation on the Heterogeneous Fleet VRP with imposed fleet.

7.1. The Heterogeneous Fleet VRP with Imposed Fleet (HVRP)

The application of the PVNS_AMP and TS_PVNS_AMP raised interesting observations regarding the nature of the proposed algorithm. In order to test the algorithm on the HVRP literature benchmarks we had to do slight modifications and make small additions to the algorithm.

We believe that if one claims that a method is generalizable, this means that different versions of the VRP can be solved by the method where only minor modifications or additions are applied. We aim to test the PVNS_AMP on the instances by Taillard (1999), as well as the large instances by Li (2007), which have heterogeneous imposed fleet. During the algorithmic experimentation on the imposed fleet instances a few observations became apparent.

✓ Observation 1: This relates to the approaches to initial solution generation. The principle by which all initial solutions are generated (explained in Chapter 4) is by picking the fleet at random until all demands are satisfied. However, if the vehicle fleet is imposed, the initial solution methods may never find feasible fleet compositions which satisfy all demands, assuming no modification to the current methodology are made. The initial solution methods developed in Chapter 4 have good performance on unlimited fleet benchmark instances, but they do not find feasible solutions if the fleet is imposed. The reason for this is what can be referred to as 'unutilized free space' issue. This issue is observable for any greedy initial solution heuristic, which uses a criterion to add customers (according to a given rule) until the capacity of the vehicle is full. This is the case with all initial solution methods used in this research. An example of this is portrayed in Figure 7.1, where each box corresponds to the demand of a customer inside the vehicle.

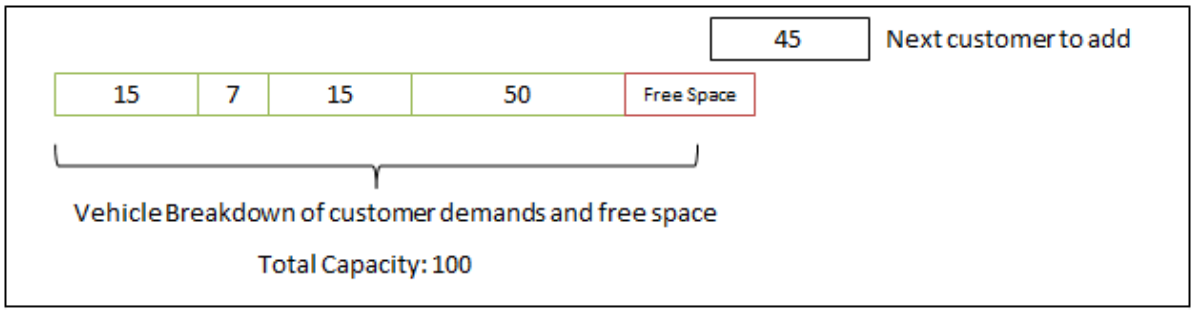

Figure 7.1: Process of adding customers (their demands) to vehicles from giant tour

Figure 7.1 shows the inside of a vehicle, where different customers have been assigned to it during an initial solution generation. If we assume that total capacity of the vehicle is 100, 87 units have been utilized by servicing 4 customers. However, if a customer which is next for insertion on the giant tour has a demand larger than the free space left, then a new vehicle has to be added to satisfy their demand. This results in all vehicles not being fully utilized. Therefore, when having an imposed fleet (maximum allowable number of vehicles) the free space factor contributes to the overall infeasibility of the solution in terms of total demand satisfaction. Furthermore, if the total available capacity of all vehicles and the sum of all demands are very close or identical, insertion methods such as the Sweep or Nearest Neighbour cannot be used to generate feasible solutions in terms of capacity (they can be used to generate infeasible solution, which is then improved by

means of Shift and Swap operators). Therefore, some amendment has to be done to the methods in order to satisfy the conditions of the VRP with imposed fleet.

✓ Observation 2: One of the main aspects of the PVNS_AMP and the TS_ PVNS_AMP is the learning aspect, which is mainly driven by the diversity of the candidate solutions and the ability to generate solutions with different fleet composition. Moreover, there are many data structures used throughout the algorithm, such as the *Shrink Route* routine and the *Dummy* Route (explained in Section 5.2.1), which encourage variable fleet composition exploration. Therefore, having an imposed fleet means very little leeway for fleet diversity and can 'paralyze' the learning process and bias it towards a given solution structure. After performing computational experimentation, two approaches were found suitable for minor modifications in the initial solution generations stage only of the PVNS_AMP and TS_PVNS_AMP in order to test them on the HFVRP instances.

7.1.1. Modification to the Initial Solution Generation

The Adapted Sweep and Adapted Nearest Neighbour developed in Chapter 4 were amended in order to accommodate the imposed fleet constraint. The only modification that was made is that instead of picking the fleet composition at random, the fleet was systematically selected starting from the largest available vehicle and inserting those customers with largest demands with a priority. The *Dummy Route* routine (please refer to Section 5.2.1 for details) was only allowed if there are available vehicles left from the imposed fleet or if the generated initial solution could not cover the total demand with the available fleet due to the issue of free space utilization. Any customers which were not covered by the available vehicles were placed in a dummy route. The *Shrink Route* routine (please refer to Section 5.2.1 for details) was fully utilized without modification, because there is only a restriction on the maximum number of available vehicles, but not all of them must be used. A simple constraint was imposed throughout the algorithm, that the number of vehicles cannot exceed the total number of available vehicles per type.

The results on the Taillard (1999) instances are shown in Table 7.1 with variable and fixed cost respectively, which compare the performance of the PVNS_AMP and TS_PVNS_AMP. It can be seen that the TS_PVNS_AMP performs better than the PVNS_AMP with an

average improvement of 0.41% for the problems with fixed cost, and 0.29% on the problems with variable cost. The reason for this is the greater intensification of the search, by means of short term memory. The PVNS_AMP makes use of long term memory which relates to the learning and *Elite Strings* encoding (partially fuelled by the diversity of the population of solutions). In contrast the TS aspect of the TS_PVNS_AMP triggers the diversity within a given route solution structure, by allowing for hill climbing and infeasible moves, as well as preventing cycling via the tabu tenure.

The TS_PVNS_AMP is also tested against the Best Known Solutions (BKS) in the literature and other relevant methods. This is shown in Tables 7.2 and 7.3, respectively. For the problem with fixed cost the TS_PVNS_AMP outperformed the HCG with 0.75% and BATA with 0.44%, and achieves all Best Known Solutions in the literature, except for one with an average gap of 0.18%. On the problem instances with variable cost the TS_PVNS_AMP outperforms the MAMP with 0.11% on average. The MAMP introduced by Li (2007) also makes use of Adaptive Memory Procedure (AMP), so this provides for an interesting comparison with the TS_PVNS_AMP, which also makes use of AMP. The computational times for both methods are also similar on average. The average gap from the Best Known Solutions is 0.08%, reaching all BKS except for instance 20.

The TS_PVNS_AMP is also tested on the large instances introduced by Li (2007) with imposed fleet. The results are shown in Table 7.4. The TS_PVNS_AMP shows some good results on the large sized instances, with 1% average deviation from the BKS.

The TS_PVNS_AMP shows some flexibility by addressing a real life VRP tested in two versions, namely with and without overtime and other real life constraints such as light load. It also shows very good results on literature benchmark instances ranging from small to large sized, with no more than 1% deviation from the Best Known solutions in the literature. Moreover, it shows good potential for accommodating problems both with unlimited and imposed fleet.

Table 7.1: Results on the Taillard (1999) instances with fixed and variable cost

HFVRP with fixed cost					HFVRP with variable cost				
Instance	**N**	**PVNS_AMP**	**TS_PVNS_AMP**		**Instance**	**N**	**PVNS_AMP**	**TS_PVNS_AMP**	
		Sol	**Sol**	**IMP***			**Sol**	**Sol**	**IMP***
13	50	3185.09	3185.09	0.00%	13	50	1517.84	1517.84	0.00%
14	50	10107.53	10107.53	0.00%	14	50	607.53	607.53	0.00%
15	50	3065.29	3065.29	0.00%	15	50	1015.29	1015.29	0.00%
16	50	3265.41	3265.41	0.00%	16	50	1144.94	1144.94	0.00%
17	75	2117.31	2076.96	1.91%	17	75	1061.96	1061.96	0.00%
18	75	3743.58	3743.58	0.00%	18	75	1823.58	1823.58	0.00%
19	100	10482.63	10420.34	0.59%	19	100	1132.54	1117.51	1.33%
20	100	4832.17	4792.87	0.81%	20	100	1572.53	1556.4	1.03%
			Average IMP	**0.41%**				**Average IMP**	**0.29%**

*IMP means improvement of TS_PVNS_AMP on PVNS_AM

Table 7.2: Results on the Taillard (1999) instances with fixed cost against Best Known Solutions (BKS) and other methods

HFVRP with fixed cost													
Instance	**N**	**BKS**	**HCG[1]**			**BATA[2]**			**ILS-RVND-SP[3]**			**TS_PVNS_AMP**		
			Sol	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**
13	50	1517.84	1518.05	0.01%	473	1519.96	0.14%	843	1517.84	0.00%	19.29	1517.84	0.00%	62
14	50	607.53	615.64	1.33%	575	611.39	0.64%	387	607.53	0.00%	11.2	607.53	0.00%	58
15	50	1015.29	1016.86	0.15%	335	1015.29	0.00%	368	1015.29	0.00%	12.56	1015.29	0.00%	61
16	50	1144.94	1154.05	0.80%	350	1145.52	0.05%	341	1144.94	0.00%	12.29	1144.94	0.00%	58
17	75	1061.96	1071.79	0.93%	2245	1071.01	0.85%	363	1061.96	0.00%	29.92	1061.96	0.00%	115
18	75	1823.58	1870.16	2.55%	2876	1846.35	1.25%	971	1823.58	0.00%	38.34	1823.58	0.00%	98
19	100	1117.51	1117.51	0.00%	5833	1123.83	0.57%	428	1120.34	0.25%	67.72	1117.51	0.00%	298
20	100	1534.17	1559.77	1.67%	3402	1556.35	1.45%	1156	1534.17	0.00%	63.77	1556.4	1.45%	305
Average Gaps from BKS				**0.93%**			**0.62%**			**0.03%**			**0.18%**	

[1] Taillard (1999)

[2] Tarantilis et al. (2009)

[3] Subramanian et al. (2012)

Table 7.3: Results on the Taillard (1999) instances with variable cost against Best Known Solutions (BKS) and other methods

HFVRP with variable cost													
Instance	**N**	**BKS**	**MAMP[1]**			**ILS_RVND[2]**			**ILS-RVND-SP[3]**			**TS_PVNS_AMP**	
			Sol	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**
13	50	3185.09	3185.09	0.00%	110	3185.09	0.00%	19.84	3185.09	0.00%	1.99	3185.09	0.00%	51
14	50	10107.53	10107.53	0.00%	34	10107.53	0.00%	11.28	10107.53	0.00%	1.29	10107.53	0.00%	42
15	50	3065.29	3065.29	0.00%	46	3065.29	0.00%	12.48	3065.29	0.00%	1.77	3065.29	0.00%	47
16	50	3265.41	3265.41	0.00%	99	3265.41	0.00%	12.22	3265.41	0.00%	1.67	3265.41	0.00%	43
17	75	2076.96	2076.96	0.00%	148	2076.96	0.00%	29.59	2076.96	0.00%	5.95	2076.96	0.00%	121
18	75	3743.58	3743.58	0.00%	119	3743.58	0.00%	36.38	3743.58	0.00%	16.47	3743.58	0.00%	109
19	100	10420.34	10420.34	0.00%	287	10420.34	0.00%	73.66	10420.34	0.00%	15.8	10420.34	0.00%	201
20	100	4761.26	4832.17	1.49%	200	4788.49	0.57%	68.46	4761.26	0.00%	16.87	4792.87	0.66%	213
Average Gaps from BKS				**0.19%**			**0.07%**			**0.00%**			**0.08%**	

[1] Li et al. (2007)

[2] Penna et al. (2011)

[3] Subramanian et al. (2012)

Table 7.4: Results on the Large Size instances by Li (2007) for HFVRP with variable cost

HFVRP Large Size Instances by Li (2007)													
Instance	**N**	**BKS**	**HRTR[1]**			**TSA[2]**			**ILS-RVND-SP[3]**			**TS_PVNS_AMP**	
			Sol	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**	**Sol**	**Gap**	**CPU**
H1	200	12050.08	12067.65	0.15%	688	12050.08	0.00%	1395	12050.08	0.00%	72.10	12050.08	0.00%	1023
H2	240	10208.32	10234.4	0.26%	995	10226.17	0.17%	3650	10329.15	1.18%	176.43	10295.36	0.85%	2698
H3	280	16223.39	16231.8	0.05%	1438	16230.21	0.04%	2822	16282.41	0.36%	259.61	16305.21	0.50%	3152
H4	320	17458.65	17576.1	0.67%	2256	17458.65	0.00%	8734	17743.68	1.63%	384.52	17761.9	1.74%	5469
H5	360	23166.56	-	-	-	23220.72	0.23%	13,321	23593.87	1.84%	621.17	23612.23	1.92%	8554
Average Gaps from BKS				**0.28%**			**0.09%**			**1.00%**			**1.00%**	

[1] Li et al. (2007)

[2] Brandao (2011)

[3] Subramanian et al. (2012)

7.1.2. Solving a Bin Packing Problem for the HFVRP with imposed fleet

We have also tested the algorithm in different settings and from a different perspective. Here we will go back to an earlier argument made in Chapter 4, which details the initial solution generation. One of the methodological choices available to researchers is whether to use an initial solution method to generate a starting point for the consequent heuristic search, or to use a randomly generated solution. We noted in Chapter 4 that a good heuristic algorithm should be able to reach good heuristic solutions from any starting point. However, there may be a trade-off between a better starting point and the computation time needed to transform this starting point into a good final heuristic solution. Here we aim to test this and we report some interesting findings.

We added an initial solution generation method for the purpose of computational experimentation, namely the Bin Packing Problem (BPP). The BPP aims to optimally pack (assign) a number of items with different weights where, w_j is the weight of item j, into a number of bins $M = (1,...,m)$ with different capacities, where c_i is the capacity of bin i, $i \in N = (1,...,n)$, such that the total weight of items does not exceed the total bin capacity. The BPP can be used not only for CO problems which aim to optimize packing, but it can also be used for the VRP with imposed fleet. A BPP was optimally solved within Cplex, where the available vehicles are the 'bins', and the 'items' to be packed are the customer's demands. By solving the BPP we guarantee a feasible vehicle mix, with minimum number of vehicles used. The mathematical formulation of the BPP is given below:

Minimize $z = \sum_{i=1}^{n} y_i$ (1)

Subject to:

$\sum_{j=1}^{n} w_j x_{ij} \le c_i y_i$ for $i \in M = (1,...,m)$; (2)

$\sum_{i=1}^{n} x_{ij} = 1;$ for $i \in N = (1,...,n)$; (3)

$x_{ij} = \{0,1\};$ 1 if item j is assigned to bin i, 0 otherwise; (4)

$y_i = \{0,1\};$ 1 if bin i is used, 0 otherwise (5)

The objective function (1) is to minimize the total number of bins used. Constraint (2) ensures that the total weight of the items do not exceed the capacity of the bins, where constraint (3) ensures that each item is packed in one bin only. Constraints (4) and (5) specify the binary nature of the variables.

Given that the BPP is solved optimally, it means that it generates only the solution which minimizes the bins used. In order to preserve the population nature of the PVNS_AMP, we have generated different possible solutions by adding constraint (6) which specifies the number of bins used to be equal to k.

$$\sum_{i=1}^{m} y_i = k; \quad \text{for } i \in M = (1,...,m); \tag{6}$$

For instance, after applying the BPP to minimize the total bins used, Cplex generates an optimal solution of k_{bpp} = 15, which means that 15 bins are the minimum number of bins (vehicles) which cover the total demand. Then the BPP is performed again with different values of k for $k = (k_{bpp},...,m)$. All the generated solutions are imported in the PVNS_AMP and form the *Initial Solution Pool*.

The interesting aspect here is not the quality of the solution from the BPP in terms of fleet, but in terms of customer sequence. The solution provided by the BPP is no better than random, when it comes to customer sequence, therefore it will be interesting to test the PVNS_AMP with a purely random starting point (in terms of customer sequence). This would further the discussion on the flexibility of the proposed methodology and how it can cope with transforming a solution of very bad quality into a good quality heuristic solution.

Tables 7.5 and 7.6 below show the results on the benchmark instances from Taillard (1999) on HFVRP with variable and fixed cost. The first column shows the objective function of the problem generated from the BPP, which gives the minimum number of bins used k_{bpp}. The second and third column show the solution generated by PVNS_AMP and TS_PVNS_AMP respectively.

Table 7.5: Results on the Taillard (1999) instances with variable cost

Instance	N	BKS	BPP	PVNS_AMP		TS_PVNS_AMP	
				Sol	Gap	Sol	Gap
13	50	1015.29	2908.34	1015.29	0.0%	1015.29	0.0%
14	50	607.53	1365.12	607.53	0.0%	607.53	0.0%
15	50	1015.29	2569.74	1015.29	0.0%	1015.29	0.0%
16	50	1144.94	2782.12	1144.94	0.0%	1144.94	0.0%
17	75	1061.96	2854.65	1089.3	2.6%	1076.3	1.4%
18	75	1823.58	3598.72	1859.92	2.0%	1823.58	0.0%
19	100	1117.51	3018.44	1146.81	2.6%	1129.57	1.1%
19	100	1559.77	3987.69	1602.35	2.7%	1587.63	1.8%
Average Gap from BKS					**1.2%**		**0.5%**

Table 7.6: Results on the Taillard (1999) instances with fixed cost

Instance	N	BKS	BPP	PVNS_AMP		TS_PVNS_AMP	
				Sol	Gap	Sol	Gap
13	50	3185.09	4388.12	3185.09	0.0%	3185.09	0.0%
14	50	10107.5	20123.52	10107.53	0.0%	10107.53	0.0%
15	50	3065.29	4236.47	3065.29	0.0%	3065.29	0.0%
16	50	3265.41	4521.83	3265.41	0.0%	3265.41	0.0%
17	75	2076.96	3784.52	2112.31	1.7%	2102.76	1.2%
18	75	3743.58	5014.75	3798.94	1.5%	3763.58	0.5%
19	100	10420.3	21125.82	10487.25	0.6%	10466.43	0.4%
19	100	4761.26	62345.97	4813.74	1.1%	4794.19	0.7%
Average Gap from BKS					**0.6%**		**0.4%**

7.1.3. Discussion

The results show a few key observations. Firstly, the quality of the solution after the BPP is very poor, which is in line with the expectation that the quality of the solution is no better than random. The PVNS_AMP and the TS_PVNS_AMP show a good capability and flexibility

to transform this random solution to a good quality heuristic solution, which in some cases matches the BKS in the literature (where in most cases the BKS have proven optimality).

An interesting observation was made during the testing stage, which can aid our future research. The learning aspect of the PVNS_AMP was not as strong when starting from a completely random solution, especially on the larger sized problems. The reason is that the learning mechanism is activated at each iteration of the algorithm (as described in Section 5.2.1). In order to reach good solutions from a very poor quality starting point, the algorithm requires more time to transform the initial solution into a better quality solution. Therefore, some of the *Elite Strings* extracted from the candidate solutions did not have a very high frequency. Since the learning process is a long term consideration in the PVNS_AMP, the learning happens at every iteration, therefore for the first iterations the extracted *Elite Strings* are not truly 'elite', since they belong to solutions structures which have not yet been transformed into good quality solutions. When testing the methodology on the solutions generated by BPP the best found solutions were reached in both stages of the algorithms, before the *Elite Strings* become fixed. In other cases, there were very few *Elite Strings* with a higher frequency than 75%, which is the threshold for a string to become a part of the Elite String List. Therefore, very small proportions of the solutions were fixed in the second stage of the algorithm. One important conclusion from here is that when using long term memory for learning purposes (specifically in our case, where AMP is used with VNS) is that the AMP learns better when applied on initial solutions generated by Initial Solution Methods, such as the Adapted Sweep or the Adapted Nearest Neighbour. It can be suggested that when using randomly generated solutions, it can be more feasible to make use of short term learning strategies, such as TS in order to focus on the intensification of the search. The reason why the learning strategy was not amended when testing the methodology on the HVRP instances is because we wanted to keep the modifications of the algorithm to a minimum, and learn from the computational experience for future research.

7.2. The Large Scale Fleet Size and Mix VRP

We believe that it is important to test any new methods introduced to the literature on common benchmarks, not only to show their efficiency, but also to further the research in a given area. The FSMVRP has a few literature benchmark instances, where the most famous

and most tested, including in this research, are the benchmarks by Golden et al. (1984). However, the largest instances for testing the VRP with unlimited heterogeneous fleet is 100 customers, which now have been solved to optimality (Subramanian et al., 2012). Therefore, we introduce some new results on the instances by Li et al. (2007), but with unlimited fleet. The customer coordinates, demands and vehicle capacities and associated costs are not changed, only the imposed fleet is removed, and all vehicle types are assumed to be unlimited. The results for the testing are shown in Table 7.7.

Table 7.7: FSMVRP Large Instances

FSMVRP on Li et.al (2007) dataset		
Instance	N	TS_PVNS_AMP
		Sol
H1	200	10153.2
H2	240	10004.3
H3	280	11428.4
H4	320	15486.9
H5	360	21376.5

Comparing these results to those reported in Table 7.4 it can be seen that there is significant difference between the overall cost of travel, which we could attribute to the difference of the fleet compositions. Future research on larger instances for the FSMVRP can provide new frontier for future methodological efforts.

7.3. The School Bus Routing Problem

The School Bus Routing Problem (SBRP) is very similar to the real life VRPs, because of the various attributes which a SBRP can have (described in more detail in section 2.2). Similarly to the RVRPs, the SBRP has no uniform definition of what attributes a problem should have in order to be classified as a SBRP. In this research we aim to show some form of comparability when it comes to real life VRPs, or those VRPs, which fall outside of the established VRP acronyms.

In this section we test our method on some SBRP instances, introduced by Schittekat et al., (2013). The reason why we have chosen to compare our results to those introduced by the authors is because of the approach they adopt to algorithmic comparability. They have solved the SBRP optimally with Mixed Integer Programming (MIP) using Cplex and also by a

metaheuristic method, namely the Greedy Randomized Adaptive Search (GRASP) with Variable Neighbourhood Descend (VND). The reported solutions compare those by the MIP to those achieved by the GRASP+VND. In this research we approached the proposed RVRP in the same manner. We have solved as large instances of the RVRP (with and without overtime) as possible to optimality, which were then compared to the solutions achieved by the PVNS_AMP and TS_PVNS_AMP. We believe that when researching a RVRP one should aim to show algorithmic comparability either by comparing their methodology to existing problems, or in the case where there are no comparable problem instances, to optimal solutions or lower /upper bounds.

The SBRP we test our methodology on consists of a depot, a set of bus stops and a set of customers. There is a maximum allowable walking distance for each student to a given bus stop. The SBRP proposed by Schittekat et al. (2013) is with homogeneous fleet. However, it can be approached as heterogeneous in the sense that students can walk to different bus stops, hence there is a choice of which stop the student can be assigned to and this will affect the load (student count) on the bus. If we refer to Table 7.8 we can see that there are different instances of the problem with different walking distance ranging from 5 to 40 minutes. This provides the interesting aspect of the SBRP with overlapping radius of the bus stops (an illustrated example is shown in Section 2.2, Figure 2.5). The TS_PVNS_AMP was not amended in order to be applied to the SBRP. The Adapted Sweep was used to assign the students to corresponding bus stops. In the cases where the radius was overlapping, the students were assigned to different stops and each of the candidate solutions in the population has a different assignment of students to stops.

The results are reported in table 7.8, which detail all 112 instances introduced by the authors. It can be seen from the results that we have achieved all the optimal solutions reported by the authors, including instance 42 and match all best found solutions by the GRASP+VND.

Table 7.8: Results on the SBRP instances by Shittekat et al. (2013)

ID	Stops	Students	Capacity	Walking Distance	Cplex Solution	VND+GRASP		TS_PVNS_AMP	
						Solution	CPU	Solution	CPU
1	5	25	25	5	141.01	141.01	0.16	141.01	<1
2	5	25	50	5	161.62	161.62	0.26	161.62	<1
3	5	25	25	10	182.14	182.14	0.39	182.14	<1
4	5	25	50	10	195.8	195.8	0.29	195.8	<1
5	5	25	25	20	111.65	111.65	0.49	111.65	<1
6	5	25	50	20	103.18	103.18	0.52	103.18	<1
7	5	25	25	40	7.63	7.63	0.29	7.63	<1
8	5	25	50	40	25.64	25.64	0.25	25.64	<1
9	5	50	25	5	286.68	286.68	0.39	286.68	<1
10	5	50	50	5	197.2	197.2	0.35	197.2	<1
11	5	50	25	10	193.55	193.55	0.43	193.55	<1
12	5	50	50	10	215.86	215.86	0.74	215.86	<1
13	5	50	25	20	130.53	130.53	1.68	130.53	<1
14	5	50	50	20	96.26	96.26	1.69	96.26	<1
15	5	50	25	40	12.89	12.89	1.38	12.89	<1
16	5	50	50	40	30.24	30.24	1.17	30.24	<1
17	5	100	25	5	360.35	360.35	1.15	360.35	<1
18	5	100	50	5	304.23	304.23	0.9	304.23	<1
19	5	100	25	10	294.21	294.21	2.08	294.21	<1
20	5	100	50	10	229.41	229.41	1.67	229.41	<1
21	5	100	25	20	134.95	134.95	2.89	134.95	<1
22	5	100	50	20	144.41	144.41	1.34	144.41	<1
23	5	100	25	40	58.95	58.95	4.24	58.95	<1
24	5	100	50	40	39.44	39.44	2.89	39.44	<1
25	10	50	25	5	242.85	242.85	1.55	242.85	<1
26	10	50	50	5	282.12	282.12	1.32	282.12	<1
27	10	50	25	10	244.54	244.54	2.45	244.54	<1
28	10	50	50	10	288.33	288.33	1.6	288.33	<1
29	10	50	25	20	108.98	108.98	2.86	108.98	<1
30	10	50	50	20	157.48	157.48	2.28	157.48	<1
31	10	50	25	40	32.25	32.25	2.84	32.25	<1
32	10	50	50	40	36.66	36.66	2.76	36.66	<1
33	10	100	25	5	403.18	403.18	0.9	403.18	<1
34	10	100	50	5	296.53	296.53	0.54	296.53	<1
35	10	100	25	10	388.87	388.87	3.82	388.87	<1
36	10	100	50	10	294.8	294.8	4.18	294.8	<1
37	10	100	25	20	178.28	178.28	5.58	178.28	<1
38	10	100	50	20	175.96	175.96	7.98	175.96	<1
39	10	100	25	40	57.5	57.5	7.38	57.5	<1
40	10	100	50	40	31.89	31.89	5.9	31.89	<1
41	10	200	25	5	735.27	735.27	8.49	735.27	<1
42	10	200	50	5	**506.06**	512.16	4.45	**506.06**	<1
43	10	200	25	10	-	513	27.17	513	<1

44	10	200	50	10	475.21	475.21	12.09	475.21	<1
45	10	200	25	20	-	347.29	25.61	347.29	<1
46	10	200	50	20	-	217.46	20.58	217.46	<1
47	10	200	25	40	-	102.93	33.35	102.93	<1
48	10	200	50	40	-	55.05	13.05	55.05	<1
49	20	100	25	5	-	520.24	8.85	520.24	4
50	20	100	50	5	-	420.64	3.4	420.64	4
51	20	100	25	10	-	422.21	7.84	422.21	4
52	20	100	50	10	-	360.86	3.88	360.86	<1
53	20	100	25	20	-	245.17	10.32	245.17	3
54	20	100	50	20	-	185.06	5.6	185.06	<1
55	20	100	25	40	-	52.52	10.4	52.52	3
56	20	100	50	40	-	19.05	23.73	19.05	2
57	20	200	25	5	-	903.84	10.6	903.84	6
58	20	200	50	5	-	485.65	29.27	485.65	4
59	20	200	25	10	-	616.93	28.85	616.93	5
60	20	200	50	10	-	462.31	18.48	462.31	3
61	20	200	25	20	-	373.21	50.39	373.21	8
62	20	200	50	20	-	250.75	26.94	250.75	6
63	20	200	25	40	-	93.01	67.73	93.01	9
64	20	200	50	40	-	45.4	33.5	45.4	4
65	20	400	25	5	-	1323.35	234.66	1323.35	13
66	20	400	50	5	-	733.54	37.64	733.54	8
67	20	400	25	10	-	975.12	139.12	975.12	10
68	20	400	50	10	-	614.67	73.23	614.67	7
69	20	400	25	20	-	763.76	132.47	763.76	9
70	20	400	50	20	-	298.47	90.54	298.47	8
71	20	400	25	40	-	239.58	307.2	239.58	11
72	20	400	50	40	-	84.49	127.08	84.49	7
73	40	200	25	5	-	831.94	60.1	831.94	21
74	40	200	50	5	-	593.35	40	593.35	15
75	40	200	25	10	-	728.44	709.28	728.44	31
76	40	200	50	10	-	481.05	91.71	481.05	19
77	40	200	25	20	-	339.75	153.04	339.75	28
78	40	200	50	20	-	273.88	53.84	273.88	17
79	40	200	25	40	-	76.77	132.52	76.77	36
80	40	200	50	40	-	58.46	77.92	58.46	23
81	40	400	25	5	-	1407.05	353.09	1407.05	31
82	40	400	50	5	-	858.8	585.98	858.8	25
83	40	400	25	10	-	891.02	496.35	891.02	33
84	40	400	50	10	-	757.42	413.29	757.42	29
85	40	400	25	20	-	586.29	739.56	586.29	37
86	40	400	50	20	-	395.95	242.91	395.95	23
87	40	400	25	40	-	195.33	1186.56	195.33	39
88	40	400	50	40	-	70.77	549.07	70.77	28
89	40	800	25	5	-	2900.14	3529.15	2900.14	41

90	40	800	50	5	-	1345.7	1257.96	1345.7	27
91	40	800	25	10	-	2200.57	3495.62	2200.57	42
92	40	800	50	10	-	1025.16	3600.03	1025.16	36
93	40	800	25	20	-	1404.16	3600.18	1404.16	42
94	40	800	50	20	-	616.58	3600.12	616.58	29
95	40	800	25	40	-	396.92	3600.81	396.92	38
96	40	800	50	40	-	200.94	3074.14	200.94	31
97	80	400	25	5	-	1546.23	958.12	1546.23	89
98	80	400	50	5	-	1048.56	471.89	1048.56	73
99	80	400	25	10	-	1216.74	1833.44	1216.74	91
100	80	400	50	10	-	760.61	576.26	760.61	69
101	80	400	25	20	-	565.49	1224.64	565.49	83
102	80	400	50	20	-	372.05	878.86	372.05	76
103	80	400	25	40	-	131.75	1116.28	131.75	84
104	80	400	50	40	-	95.84	3600.05	95.84	81
105	80	800	25	5	-	2527.96	3433.78	2527.96	101
106	80	800	50	5	-	1530.58	3600.03	1530.58	95
107	80	800	25	10	-	1809.9	3600.05	1809.9	111
108	80	800	50	10	-	1187.51	3600.04	1187.51	94
109	80	800	25	20	-	1110.44	3600.1	1110.44	116
110	80	800	50	20	-	623.03	3600.62	623.03	98
111	80	800	25	40	-	311.41	3600.21	311.41	114
112	80	800	50	40	-	126.06	3600.05	126.06	99

7.3.1. Discussion

It can be seen from Table 7.7 that the TS_PVNS_AMP has a much faster computational time than the GRASP+VND on all instances. The behaviour of both algorithms in terms of computational time is similar. It can be seen that for the instances which have the same number of students and stops the computational time can vary. This means that the impacting factors on the complexity of the problem are the walking distance and the capacity of the buses. Different combinations of these two parameters can lead to different nuances of the problem. However the computational time of the TS_PVNS_AMP is more stable, because it is mostly affected by the size of the VRP problem, rather than the size of the assignment problem. We believe that the strength of the TS_PVNS_AMP on the SBRP lies with the population of solutions, as well as the Adapted Sweep proved very efficient in the assignment of students to stops, where even the largest instances were assigned for less than 3 seconds in computational time.

There are some clear trade-offs between the features of the SBRP. It can be noted from Table 7.8 that the instances with smaller capacities and larger walking distances take more computational effort to be solved. This can be attributed to the fact that when the bus capacity is larger, fewer buses can satisfy the total demand (transport all students), hence the VRP part of the problem has less vehicles. The smaller the capacity, the more vehicles are needed and the fleet mix becomes larger. Moreover, the longer the walking distance, the greater the choice of which stop a given student can walk to, hence there are more possibilities to explore before finding a good heuristic solution.

7.4. Summary

The PVNS_AMP and the TS_PVNS_AMP have been extensively tested on the RVRP with and without overtime and other relevant VRP problems, which can be derived from the RVRP, such as the FSMVRP with fixed and variable cost. In this chapter we tested the methodology on other VRP problems such as the HFVRP with imposed fleet, with fixed and variable cost and the School Bus Routing Problem with overlapping and non-overlapping bus stop radius. Both methods show a good degree of flexibility and capability for generalization to other VRP problems different than the RVRP the method was originally designed to address.

Chapter 8

Conclusion and Future Research

The aim of this research is to introduce a real life Vehicle Routing Problem inspired by the Gas Delivery industry in the UK and to design a methodology to address this problem. After an extensive literature review was conducted a few key trends in the literature became apparent and we aimed to address all of them through our research objectives.

The first objective relates to the trend in the literature to bridge the gap between academia and real life practices. Therefore, we introduced a new real life VRP problem, which has not been researched in the past, with real life attributes such as light load requirement, demand-dependent service time, allowable overtime coupled with unlimited fleet etc. We provide a mathematical formulation and detailed description of the RVRP introduced here and we found some very interesting observations and practical implications.

The current operations of the company which motivated this research are that overtime becomes a necessity at the end of the drivers' routing schedule and the decision to go into overtime has to be taken then without considerations regarding routing efficiency. Our computational experience shows that accommodating for allowable overtime in advance and incorporating it in the routing schedule can result in up to 8% savings for one planning period. Moreover, informing the drivers in advance of any possible overtime can contribute to minimizing the resistance to accept overtime when it is offered at the end of their shifts. Accommodating for overtime into the routing schedule in advance results in 12.5% better utilization of the vehicle capacity and 12% better utilization of the working time. Another important practical implication is that by better utilizing capacity and working time, fewer vehicles can be used to satisfy customer demand.

Another interesting feature of the RVRP is the allowable overtime coupled with unlimited fleet. There is a clear trade-off between allowing overtime and using extra vehicles to satisfy the total customer demand. We found that when overtime is not used the fleet mix is larger,

whereas when overtime is considered the fleet mix is smaller where there are no short routes. This is an important consideration, because the drivers have to work up to 8 hours and 30 minutes and any unused time becomes an inefficiency for the company, both operationally and financially. The other main real life aspect is the efficient incorporation of the light load customers. We found that these customers can be incorporated into the routing schedule efficiently, at a very small extra cost compared to a base line routing schedule with no light load customers or compared to the current operations of the company which assign the light load customers manually at the end of the routing schedule, regardless of their location.

The second objective of this research relates to the design of powerful methodologies to address Vehicle Routing Problems. Firstly, we design two new initial solution methods, which are an adaptation of the Sweep and the Nearest Neighbour, namely the Adapted Sweep (AS) and the Adapted Nearest Neighbour (ANN). Both methods show good results on standard benchmark instances and show a better performance than one of the most widely used methods, namely the Savings heuristics. Incorporating the initial solution methods we designed into the main metaheuristic methods provided the search with a good starting point for the consequent heuristic search, whilst providing a degree of diversification to the method, as well as aiding its learning aspect.

We design 2 new hybrid metaheuristic methods which are extensively tested on the RVRP and other relevant VRP problems. The Population Variable Neighbourhood Search with Adaptive Memory Procedure was designed in order to test a new approach to learning mechanisms. We aim to hybridize Adaptive Memory with a method which does not use memory structures or learning in its original form, namely the VNS. To the best of our knowledge this hybridization has not been done in the past, and we have learned a lot from the computational experience. When testing the PVNS_AMP on the RVRP and literature benchmark instances, we found that it has good performance on the RVRP and reaches most Best Known Solutions in the literature with less than 1% deviation on average. We found that using AMP with a population VNS has a good potential for extracting promising parts of the solution, i.e. the *Elite Strings* and using those parts in order to build a better quality solution. The population aspect of the VNS proved to be a key for diversification of the solution search, as well as for the good extraction of the *Elite Strings*. When we compared

the extracted *Elite Strings* to the generated optimal solutions, it was found that they are indeed a part of the optimal solutions, and are encoded into the metaheuristic solution accordingly.

One of the areas of improvement of the PVNS_AMP that we found during the computational experimentation was the intensification of the search process and the possibility to incorporate short term memory strategies in order to improve the algorithm performance. Therefore, in accordance with our findings and the review from the literature we incorporated aspects from Tabu Search into the PVNS_AMP, which resulted in a new hybrid metaheuristic the TS_PVNS_AMP. When we tested the new TS_PVNS_AMP against the PVNS_AMP we found significant improvements of the total cost for the RVRP. The TS_PVNS_AMP especially proved very powerful for the RVRP with overtime, where the average improvement from the PVNS_AMP is 0.75% with a much more balanced fleet composition of the routing schedule. Moreover, the TS_PVNS_AMP proved more powerful on the literature benchmark instances with unlimited fleet, where most of the Best Known Solutions were reached, with less than half percent average deviation from the best, and at no significant increase in computational time.

Another important aspect that we addressed is the comparability of any heuristic method when it comes to real life VRPs. We proposed that when one aims to address an RVRP it is necessary to approach the problem in a standardise-first customize-second fashion. This provides the opportunity to compare the results from the RVRP on standard literature benchmarks in order to have an indication of the performance of the method. Moreover, we proposed a Mixed Integer formulation for the RVRP, which was used to solve the RVRP to optimality where possible and these results were compared to those from the PVNS_AMP and TS_PVNS_AMP.

The third objective relates to the trend in the literature to design methods which have a degree of flexibility and can be generalized to other VRP problems with minor modifications. The hybrid metaheuristic methods we designed aim to solve a real life VRP, though we tested the methodology on other relevant VRP problems. The methodology showed some good results on the heterogeneous fleet VRP with imposed fleet, on literature benchmark instances ranging from 50 to 360 customers, with less than 1% deviation on average from

the Best Known Solutions. We also applied the methodology on an interesting problem, namely the School Bus Routing Problem and our method found all optimal solutions and Best Known Solutions in the literature in a short computational time. It is important to note that on some instances the methods were tested not only in their original form, but also starting from a random solution generated by solving the Bin Packing Problem, and the TS_PVNS_AMP showed a good potential to find good heuristic solutions, not only from a good heuristic starting point (generated by the Adapted Sweep and Adapted Nearest Neighbour), but also from a random point. The PVNS_AMP proved to have good diversification and learning mechanisms, whereas the TS_PVNS_AMP added a greater intensification to the search, which resulted in a well-balanced metaheuristic hybrid which makes use of long term and short term memory structures. We believe that these findings and all lessons learned from the computational experience can be very useful for further research, especially for incorporating learning and memory principles in any existing method in order to enhance performance in an intelligent and guided manner.

The contribution of this research is twofold

(i) In terms of novelty in the body of literature on vehicle routing variants, the RVRP introduced in this research has not been addressed before. It has elements, which have not been researched, such as demand-dependent service times, requirement for light load and the simultaneous consideration of maximum allowable overtime and unlimited heterogeneous fleet. This research fits into the literature trend of minimizing the gap between optimization and real life operations, by introducing a problem, which is highly relevant in practice. Moreover, this research offers real practical implications on how to improve routing practices given the relevant routing elements. There is still a gap in the literature of RVRPs which poses an opportunity to find unexplored corners and make a fine addition to the existing literature.

(ii) In terms of methodological contribution, this research introduces two new hybrid metaheuristic methods to address the RVRP, which have a degree of generalizability and can be applied to other Vehicle Routing Problems, namely the FSMVRP, the HFVRP and the SBRP. Moreover, mathematical formulation of the problem is presented, which is used for methodological justification and a platform for comparability, since the RVRP is based on

adapted literature benchmark instances. The issue of comparability is not present in the RVRP literature. However, we believe that it is a vital part of the methodological contribution. A generalizable and efficient algorithm can contribute to the body of knowledge in the management science arena, because Combinatorial Optimization problems like the VRP are difficult to solve to optimality.

Future research

This research was a great source of learning and motivation for future research. Experiencing the design of metaheuristic methods can shape one's approach to Combinatorial Optimization problems, in this case for the Vehicle Routing Problem. From this research we learned that having a balanced design in hybridizing different methods can be very beneficial and can result in powerful heuristic methods. Here we incorporate learning principles in methods which do not make use of learning and memory in its original form and the computational experience showed that there is a potential in further research of such intuitive and intelligent methods, which can learn from past experience. Short term and long term learning aspects of the PVNS_AMP and TS_PVNS_AMP can be further tested, such as developing a learning tabu tenure, relative to each local search operator and other novel ways of extracting good *Elite Strings* from a given solution structure.

Another aspect which proved interesting and challenging, which also has a research gap in the literature, is introducing a School Bus Routing Problem with heterogeneous fleet and overlapping radius of the bus stops. This problem has a lot of potential to be developed further, as well as introducing some standard literature benchmark instances for it. Having a heterogeneous fleet and overlapping radius is a complex problem and can have real life richness and relevance not only in academia, but in practice as well.

Our experience with the Heterogeneous Fleet problem with imposed fleet (HFVRP) was also very interesting and challenging. A Bin Packing Problem (BPP) was used to address the HFVRP within the life of this research. However, it would be interesting to further this research in a more sophisticated way, by ensuring that the solution generated from the BPP is not random, but relevant for the vehicle routing problem. One aspect which could be investigated further is adding precedence constraints to the BPP where the customers are

not added to the routes at random, but according to some criteria, such as a giant tour generated by another heuristic method such as the Sweep. Moreover, we aspire to hybridize the BPP and any metaheuristic method, in a *Matheuristic* fashion, where the results from the optimal solutions and the metaheuristic can be linked together, in order to create a more powerful hybrid.

References

Aleman R., Zhang X., Hill R. (2010) An adaptive memory algorithm for the split delivery vehicle routing problem, Journal of Heuristics, vol. 16, pp. 441–473.

Baldacci R., Toth P., Vigo D. (2010) Exact algorithms for routing problems under vehicle capacity constraints, Annals of Operations Research, vol. 175(1), pp. 213-245.

Belhaiza S. (2010) Hybrid variable neighborhood—tabu search algorithm for the site dependent vehicle routing problem with time windows, Technical report, GERAD, Montreal, Canada.

Baldacci R., Mingozzi A. (2004) An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation, Operations Research, vol. 52, pp. 723–738.

Balinski M., Quandt R. (1964) An integer program for a delivery problem, Operations Research, vol. 12, pp. 300–304.

Battarra M., Monaci M., Vigo D. (2009) An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem, Computers & Operations Research, vol. 36(11), pp. 3041-3050.

Battiti R., Tecchiolli G. (1994) The Reactive Tabu Search, ORSA Journal of Computing, vol. 6(2), pp. 126-140.

Bajec P. (2011) An analysis of the logistics innovation development process at logistics service providers, Scientific Paper, accessed from http://www.upce.cz/fes/veda-vyzkum/fakultni-casopisy/scipap/archiv/e-verze-sborniku/2011/sbornik-4-2011.pdf#page=6, accessed on the 05.11.2012.

Benjamin A. (2010) Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities, Computers & Operations Research, vol. 37 pp. 2270–2280.

Bent R., van Hentenryck P. (2006) A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows, Computers & Operations Research, vol.33, pp. 875–893.

Bortfeld A. (2010) A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints, Computers & Operations Research, vol. 39, pp. 2248–2257.

Brandão J. (2011) A tabu search algorithm for the heterogeneous fixed fleet vehicle

routing problem, Computers & Operations Research, vol. 38, pp. 140–151.

Burke E., Hyde M., Kendall G., Ochoa G., Özcan E., Woodward, J. (2010) A classification of hyper-heuristic approaches: In Gendreau and J.-Y. Potvin, Handbook of metaheuristics, International Series in Operations Research & Management Science, vol. 146, pp. 449-468, Springer.

Chao I.M. (2002) A tabu search method for the truck and trailer routing problem, Computers and Operations Research, vol. 29, pp. 33–51.

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

Clarke G., Wright, J. (1964) Scheduling of vehicles from a central depot to a number of delivery points, Operations Research, vol. 12(4), pp. 568–581.

Contardo C., Martinelli R. (2014) A new exact algorithm for the multi depot vehicle routing problem under capacity and route length constraints, Discrete Optimization, vol. 12, pp. 129-146.

Cordeau JF. (2002) A guide to vehicle routing heuristics, Journal of the Operational Research Society, vol. 53, pp. 512–522.

Cowling P, Kendall G, Soubeiga E. (2000) A hyperheuristic approach for scheduling a sales summit. In: Selected Papers of the Third International Conference on the Practice and Theory of Automated Timetabling, PATAT 2000, Springer, Konstanz, Germany, Lecture Notes in Computer Science, pp. 176–190

Dantzig G, Ramser J. (1959) The truck dispatching problem, Management Science, vol. 6(1), pp. 80-91.

Dayarian I., Crainic T., Gendreau M., Rej W. (2015) A column generation approach for a milti-attribute vehicle routing problem, European Journal of Operational Research, vol. 241(3), pp. 888-906.

Derigs U. (2006) Indirect search for the vehicle routing problem with pickup and delivery and time windows, OR Spectrum, vol. 30, pp. 149-165.

Desrosiers J., Ferland J., Rousseau J., Lapalme G., Chapleau L. (1980) An overview of a School bussing system: In International Conference of Transportation, vol. 9, pp. 235-243.

Desrochers M., Verhoog T., (1991) A New Heuristic for the Fleet Size and Mix Vehicle Routing Problem, Computers & Operations Research, vol. 18, pp. 263-274.

Doerner K., Gronald M., Hartl R., Kiechle G., Reimann M. (2008) Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows, Computers & Operations Research, vol. 35, pp. 3034 – 3048.

Dror M., Laporte G., Trudeau P. (1989) Vehicle Routing with stochastic demands: Properties and Solution framework, Transportation Science, vol. 23, pp. 228-235.

Dullaert W., Janssens G.K., Sorensen K., Vernimmen, B. (2002) New Heuristic for the Fleet Size and Mix Vehicle Routing Problem with Time Windows, Journal of the Operational Research Society, vol. 53, pp. 1232-1238.

Eles P. (2010) Heuristic Algorithms for Combinatorial Optimization Problems Simulated Annealing, as appears in http://www.ida.liu.se/~petel/.

Erdogan G., Cordeau JF., Laporte G. (2010) The Attractive Traveling Salesman Problem, European Journal of Operational Research, vol. 203, pp. 59-69.

EUOR (2012) accessed from http://www.eurodecision.com, accessed on 21.11.2012.

Fields J. (2004) The travelling salesman problem and its modern applications, MTH 441, Historical Perspectives of Mathematics, April 15.

Frizzell P., Griffin J. (1995) The split delivery vehicle scheduling problem with time windows and grid network distances, Computers & Operations Research, vol. 22(6), pp. 655-667.

Gambardella L., Taillard E., Agazzi, G. (1999) A multiple ant colony system for vehicle routing problems with time windows. In: D. Corne, M. Dorigo and F. Glover (eds), New Ideas in Optimization, McGraw-Hill, London, pp. 63-76.

Gulczynski D., Golden B., Wasil E. (2011) The period vehicle routing problem: New heuristics and real-world variants, Transportation Research Part E, vol. 47, pp. 648–668.

Gambardella M. (2005) Ant Colony Optimization for rich VRP problems, Molde University College Rich Vehicle Routing Conference.

Garrido P., Castro C. (2009) Stable solving of CVRPs using hyperheuristics, Proceedings of the 11th Annual Conference on Genetic and evolutionary computation, pp. 255-262.

Gendreau M., Laporte G., Seguin R. (1996) Stochastic vehicle Routing, European Journal of Operational Research, vol. 88(6), pp. 3-12.

Gendreau M., Laporte G., Musaraganyi, C., Taillard, E. (1999) A Tabu Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem, Computers & Operations Research, vol. 26, pp. 1153-1173.

Gendreau M., Jabali O., Ray W. (2014) Stochastic VRP, In: Toth and Vigo (eds) The vehicle routing problem, Chapter 8, 2nd ed, SIAM Philadelphia.

Gillett B., Miller L. (1974) A heuristic algorithm for the vehicle-dispatch problem, Operations Research, vol. 22(2), pp. 340–349.

Glover F. (1990) Tabu Search: A tutorial, The Institute of Management Science, Programming Integer algorithms, Heuristics, July-August, pp. 74-94.

Goel A., Gruhn V. (2006) Solving a dynamic real-life vehicle routing problem, Operations Research Proceedings, pp. 367–372.

Golden B., Assad A., Levy L., Gheysens F. (1984) The fleet size and mix vehicle routing problem, Computers & Operations Research, vol. 11, pp. 49-66.

Gribkovskaia I., Gullberg B., Hovden K., Wallace S. (2006) Optimization model for a livestock collection problem, International Journal of Physical Distribution & Logistics Management, vol. 36(2), pp. 136–52.

Hanafi S., Lazic J., Mladenovic N., Wilbaut C., Crevitz I. (2010) New hybrid matheuristics for solving the multidimensional knapsack problem, Proceedings of the 7th International conference on Hybrid metaheuristics, pp.118-132, as appears in http://dl.acm.org/citation.cfm?id=1926974

Haghani A., Jung S. (2005) A dynamic vehicle routing problem with time-dependent travel times, Computers & Operations Research, vol. 32, pp. 2959–2986.

Hoff A., Andersson H., Christiansen M., Hasle G., Løkketangen A. (2010) Industrial aspects and literature survey: fleet composition and routing, Computers & Operations Research, vol. 37, pp. 2041–2061.

Hasle G., Løkketangen A., Martello S. (2006) Rich models in discrete optimization: Formulation and resolution, European Journal of Operational Research, vol. 175, pp. 1752–1753.

Hansen P., Mladenovic N. (2001) Variable neighbourhood search: Principles and Applications, European Journal of Operational Research, vol. 130, pp. 449–467.

Hemmelmayr V. (2012) An adaptive large neighbourhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics, Computers & Operations Research, vol. 39, pp. 3215–3228.

Henderson D. (2003) The theory and practice of simulated annealing, In: Glover F., Kochenberger G. (eds), Handbook of meta-heuristics, Chapter 10, Springer.

Imran A., Salhi S., Wassan N. (2009) A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem, European Journal of Operational Research, vol. 197(2), pp. 509-518.

Jobber D. (2008) Principles and practice of Marketing, 4th ed. McGraw Hill.

Kang K., Lee Y., Lee B. (2005) An exact algorithm for multi depot and multi period vehicle scheduling problem, Computational Science and its Applications, pp. 350-359.

Kirby M. (2007) Paradigm change in Operations Research: Thirty years of Debate, Operations Research, vol. 55(1), pp. 1-13.

Kramer R., Maculan N., Subramanian A., Vidal T. (2015) A speed and departure time optimization algorithm for the Pollution-Routing Problem, Working Paper - January 2015.

Laporte G. (1992) The vehicle routing problem: an overview of exact and approximate algorithms, European Journal Operational Research, vol. 59, pp. 345–58.

Laporte G. (2009) Fifty Years of Vehicle Routing, Transportation Science, Vol. 43(4), pp. 408–416.

Lahyani R., Khemakhem M., Semet F. (2015) Rich vehicle routing problems: From a taxonomy to a definition, European Journal of Operational Research, vol. 241, pp. 1–14.

Lawler E. (1976) Combinatorial Optimization: Networks and Matroids, 1st ed., Holt, Rinehart and Winston.

Li X., Leung S., Tian P. (2012) A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem, Expert Systems with Applications, vol. 39, pp. 365–374.

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

Li L., Fu Z. (2002) The school bus routing problem: a case study, Journal of the Operational Research Society, pp. 552–558.

Lee Y., Kim J., Kang K., Kim K. (2008) A heuristic for vehicle fleet mix problem using tabu search and set partitioning. Journal of the Operational Research Society, vol. 59, pp. 833–841.

Li, F., Golden, B., Wasil E. (2007) A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem, Computers & Operations Research vol. 34, pp. 2734–2742.

Liu S., Huang W., Ma H. (2009) An effective genetic algorithm for the fleet size and mix vehicle routing problems, Transportation Research Part E, vol. 45, pp. 434–445.

Lin S., Kernighan B. (1973) An Effective Heuristic Algorithm for the Traveling-Salesman Problem, Operations Research, vol. 21(2), pp. 498–516.

Marinakis Y. (2012) Heuristic Solutions of Vehicle routing problems in supply chain management, accessed from neo.lcc.uma.es/vrp/wp-content/data/articles/HeurVRP.ps accessed on the 25.11.2012.

McKendall A., Lewis A. (2006) Simulated annealing heuristics for the dynamic facility layout problem, Computers & Operations Research vol. 33 pp. 2431 – 2444.

Mladenovic N., Hansen P. (1997) Variable neighbourhood search, Computers and Operations Research, vol. 24, pp. 1097–1100.

Montane F., Galvao R. (2006) A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service, Computers & Operations Research, vol. 33, pp. 595–619.

Moon I.K., Lee J., Seong J. (2012) Vehicle routing problem with time windows considering overtime and outsourcing vehicles, Expert Systems with Applications, vol. 39, pp. 13202–13213.

Morash E. (1997) The role of transportation capabilities in international supply chain management, Transportation Journal, vol. 36(3), pp. 5-17.

Nagata Y., Braysy O., Dullaert W. (2010) A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows, Computers & Operations Research, vol. 37, pp. 724 -737.

Nagy G., Salhi S. (2005) Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries, European Journal of Operational Research, vol. 162, pp. 126–141.

Ngueven S. Prins C., Calvo R. (2010) An effective memetic algorithm for the cumulative capacitated vehicle routing problem, Computers & Operations Research, vol. 37, pp. 1877 – 1885.

Oliveira H. (2010) A hybrid search method for the vehicle routing problem with time windows, Annals of Operations Research, vol. 180, pp. 125–144.

Olivera A., Viera O. (2007) Adaptive memory programming for the vehicle routing problem with multiple trips, Computers & Operations Research, vol. 34, pp. 28-47.

Oppen J. (2010) Algorithmic and Computational Methods in Retrial Queues Solving a rich vehicle routing and inventory problem using column generation, Computers & Operations Research, vol. 37(7), pp. 1308-1317.

Office of national Statistics (2012) accessed from http://www.statistics.gov.uk/hub/index.html on the 15.11.2012.

Park J., Kim B. (2010) The school bus routing problem: a review, European Journal of Operational Research, vol. 202(2), pp. 311–319.

Ren Y., Dessouky M., Ordonez F., (2010) The multi-shift vehicle routing problem with overtime, Computers & Operations Research, vol. 37, pp. 1987–1998.

Renaud J., Laporte G., Boctor F. (1996) A tabu search heuristic for the multi-depot vehicle routing problem, Computers & Operations Research, vol. 23(3), pp. 229–235.

Renaud J., Boctor F. (2002) A Sweep-Based Algorithm for the Fleet Size and Mix Vehicle Routing Problem, European Journal of Operational Research, vol. 140, pp. 618-628.

Reimann M., Doerner K., Hartl R. (2004) D-Ants: Savings based ants divide and conquer the vehicle routing problem, Computers & Operations Research, vol. 31(4), pp. 563–591.

Repoussis P., Tarantilis C.D. (2010) Solving the Fleet Size and Mix Vehicle Routing Problem with Time Windows via Adaptive Memory Programming, Transportation Research Part C, vol. 18, pp. 695–712.

Resende M., Ribeiro, C. (2003) Greedy randomized adaptive search procedures, International series in Operations Research & Management Science, vol. 57, pp. 219–249

Rieck J., Zimmermann J. (2010) A new mixed integer linear model for a rich vehicle routing problem with docking constraints, Annals of Operations Research, vol.181, pp. 337–358.

Rochat Y., Taillard E. (1995) Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics, vol. 1(1), pp. 147–167.

Ropke S., Pisinger D. (2006) An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows, Transportation Science, vol. 40(4), pp. 455-472.

Salhi S., Sari M., Sadi D., Touati N. (1992) Adaptation of Some Vehicle Fleet Mix Heuristic, OMEGA 20, pp. 653-660.

Salhi S., Imran A., Wassan N. (2014) The Multi-Depot Vehicle routing problem with heterogeneous vehicle fleet: Formulation and a Variable Neighbourhood Search implementation, Computers & Operations Research Part B, vol. 52, pp. 315-325.

Schilde M., Doerner K. F., Hartl R. F., (2011) Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports, Computers & Operations Research, vol. 38, pp. 1719–1730.

Schittekat P., Kinable J., Sorensen K., Sevaux M., Spieksma F., Springael J. (2013) A metaheuristic for the School Bus Routing problem with Bus Stop selection, European Journal of Operational Research, vol. 229, pp. 518-528.

Semet F., Taillard E. (1993) Solving real-life vehicle routing problems efficiently using Tabu Search, Annals of Operations Research, vol. 41, pp. 469-488.

Semet F., Toth P., Vigo D. (2014) Classical exact algorithms for the capacitated vehicle routing problem, In: Toth P., Vigo D. (eds), The vehicle routing problem, Chapter 2, 2nd ed., SIAM Philadelphia.

Shaw P. (1997) A new local search algorithm providing high quality solutions to vehicle routing problems, Technical report, Department of Computer Science, University of Strathclyde, Scotland.

Solomon M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints, Operations Research, vol. 35(2), pp. 254-265.

Subramanian A., Cabral L. (2009) An efficient ILS heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery, Technical Report – RT 07/08 Instituto de Computacao.

Subramanian A., Penna P., Uchoa E., Ochi L. (2012) A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem, European Journal of Operational Research, vol. 221(2), pp. 285–295.

Taillard E. (1999) A heuristic column generation method for heterogeneous fleet. RAIRO (Recherche ope rationnelle) 33, 1–14.

Taillard E., Gambardella L., Gendreau M., Potvin J. (2001) Adaptive Memory Programming: A unified view of metaheuristics, European Journal of Operational Research, vol. 135, pp. 1–16.

Tarantilis C., Kiranoudis C. (2002) BoneRoute: An adaptive memory-based method for effective fleet management, Annals of Operations Research, vol. 115(1), pp. 227–241.

Tarantilis C., Kiranoudis C., Vassiliadis V. (2003) A List Based Threshold Accepting Metaheuristic for the Heterogeneous Fixed Fleet Vehicle Routing Problem, Journal of the Operational Research Society, vol. 54, pp. 65-71.

Tarantilis C. (2005) Solving the vehicle routing problem with adaptive memory programming methodology, Computers & Operations Research, vol. 32(9), pp. 2309–2327.

Tarantilis C., Kiranoudis C. (2007) A flexible adaptive memory-based algorithm for real-life transportation operations: Two case studies from dairy and construction sector, European Journal of Operational Research, vol. 179, pp. 806–822.

Tarantilis C. (2008) A Hybrid Guided Local Search for the Vehicle-Routing Problem with Intermediate Replenishment Facilities, INFORMS Journal on Computing, vol. 20(2) pp. 154–168.

Thangiah S., Wilson B., Pitluga A., Mennell W. (2005) School Bus Routing in Rural School Districts, Working Paper, Computer Science Department, Slippery Rock

The Logistics Report (2012) Freight Transport Association, accessed from http://www.fta.co.uk/about/logistics_report.html, accessed on the 10.11.2012.

Toth P., Vigo D. (2002) The vehicle routing problem, 1st ed., SIAM Philadelphia.

Toth P., Vigo D. (2003) The Granular Tabu Search and Its Application to the Vehicle-Routing Problem, INFORMS Journal on Computing, vol. 15(4) pp. 333-346.

Toth P., Vigo D. (2014) The vehicle routing problem, 2nd ed., SIAM Philadelphia.

Transport Statistics Great Britain (2011) Deportment of Transport, Statistics Release 15th December 2011.

Transportation Research Board (2012) accessed from http://www.trb.org/Main/Home.aspx, accessed on the 15.11.2012.

Valle C. (2011) Heuristic and exact algorithms for a min–max selective vehicle routing problem, Computers & Operations Research, vol. 38 pp. 1054–1065.

Vidal T., Crainic T., Gendreau M., Prins C. (2007) A unified solution framework for multi-attribute vehicle routing problems, European Journal of Operational Research, vol. 234, pp. 658–673.

Vidal T., Crainic, T., Gendreau M., Prins C. (2011) A unifying view on timing problems and algorithms, Technical report, CIRRELT, Montreal, QC, Canada.

Vidal T., Crainic, T., Gendreau M., Prins C. (2013) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows, Computers & Operations Research, vol. 40, pp. 475–489

Villegas J., Prins C., Prodhon C., Medaglia A., Velasco N. (2010) GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots, Engineering Applications of Artificial Intelligence, vol 23, pp. 780–794.

Wassan N. (2007) Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls, Journal of the Operational Research Society, vol. 58, pp. 1630 -1641.

Wassan N., Nagy G. (2014) Vehicle Routing Problem with Deliveries and Pickups: Modelling Issues and Meta-heuristics Solution Approaches, International Journal of Transportation vol. 2(1), pp. 95-110

Wen M., Krapper E., Larsen J., Stidsen T. (2011) A multilevel variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem, Networks, vol. 58, pp. 311-322.

www.branchandcut.org

Yeun L., Ismail W., Omar K., Zirour M. (2008) Vehicle routing problem: models and solutions, Journal of Quality Measurement and Analysis, vol. 4(1) , pp. 205-218.

Yin P., Glover F., Laguna M. (2010) Cyber Swarm Algorithms – Improving particle swarm optimization using adaptive memory strategies, European Journal of Operational Research, vol. 201, pp. 377–389.

Zachariadis E., Tarantilis C., Kiranoudi C. (2010) An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries, European Journal of Operational Research, vol. 202, pp. 401–411.