Using Metaheuristics to solve Dynamic Vehicle Routing Problems

Author:
AbdAllah, AbdelMonaem

Publication Date:
2013

DOI:
https://doi.org/10.26190/unsworks/16262

Using Metaheuristics to solve Dynamic Vehicle Routing Problems

AbdelMonaem Fouad Moustafa AbdAllah

B.Sc. in Information Systems and Technology

M.Sc. in Information Systems

Faculty of Computer Science and Informatics, Zagazig University, Egypt

UNSW

THE UNIVERSITY OF NEW SOUTH WALES

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science (Computer Science) from the School of Engineering and Information Technology (SEIT), the University of New South Wales (UNSW) @ Canberra.

March 2013

PLEASE TYPE

THE UNIVERSITY OF NEW SOUTH WALES
Thesis/Dissertation Sheet

Surname or Family name **AbdAllah**

First name: **AbdelMonaem** Other name/s: **Fouad Moustafa**

Abbreviation for degree as given in the University calendar:
Master (Research)

School: **SEIT** Faculty: **Engineering**

Title: **Using Metaheuristics to solve Dynamic Vehicle Routing Problems**

The Vehicle Routing Problem (VRP) is considered a complex, high-level set of various routing problems. One of its important variants is the Dynamic Vehicle Routing Problem (DVRP) in which not all customers are known in advance but are revealed as the system progresses. Consequently, DVRP applications are seen to operate on a dynamic basis in various real-world systems. Like the classical VRP, as DVRP is NP-hard optimization problems, so optimization techniques that have the capability to produce high quality solutions under time limitation i.e. meta-heuristics, are the most suitable and applicable techniques to be used to solve them.

This thesis aims to solve DVRP using the Genetic Algorithm (GA) enhanced by five proposed modifications, including to the initial population for the first time slice and/or other time slices, the selection process, the swap mutation and detection and management processes of the Local Optimal Condition (LOC). Through experiments, it is clear that these improvements enhance the GA's ability to solve DVRP. Also, based on the quality of its generated solutions regarding its best and average results, the enhanced GA is competitive and out-performs previously published DVRP systems.

To date, a time-based evaluation approach has been used to evaluate DVRP systems. However as, in it, all DVRP systems are run for a specified amount of time for each time slice, another objective of this research is to propose a fairer evaluation approach whereby four evaluation approaches include generations, raw fitness, weighted fitness and distance calculations, are tested as alternatives. Of these, as the weighted fitness evaluation technique has the lowest standard deviation, it is the most stable for use regardless of a running system's CPU specifications and power requirements. Overall, based on its results from both the time-based and weighted fitness evaluation approaches, the modified GA is better than previously published algorithms.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

THIS SHEET IS TO BE GLUED TO THE INSIDE FRONT COVER OF THE THESIS

COPYRIGHT STATEMENT

Signed ..AbdelMonaem AbdAllah.....

Date .....10-07-2013.....

AUTHENTICITY STATEMENT

Signed ....AbdelMonaem AbdAllah.....

Date .....10-07-2013.....

ORIGINALITY STATEMENT

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed .....AbdelMonaem AbdAllah.....

Date .....10-07-2013.....

Abstract

The Vehicle Routing Problem (VRP) is considered to be a complex and high-level set of various routing problems. One of its important variants is the Dynamic Vehicle Routing Problem (DVRP), in which not all customers are known in advance, but are revealed as the system progresses. Consequently, DVRP applications are seen to operate on a dynamic basis in various real-life systems. Like the classical VRP, as DVRP is an NP-hard optimization problem, so optimization techniques that have the capability to produce high quality solutions under time limitations, i.e. metaheuristics, are the most suitable and applicable techniques to be used to find good solutions for them.

This thesis aims to find good solutions for DVRP by using a Genetic Algorithm (GA) enhanced by five proposed modifications, including the initial population for the first time slice and/or other time slices, the selection process, the swap mutation and the detection and management processes of the Local Optimal Condition (LOC). Through experiments, it is clear that these improvements enhance the GA's ability to solve DVRP. Also, based on the quality of its generated solutions, with regard to its best and average results, the enhanced GA is competitive and out-performs previously published DVRP systems.

To date, a time-based evaluation approach has been used to evaluate DVRP systems. However, as all DVRP systems are run for a specified amount of time for each time slice, another objective of this research is to propose a fair evaluation approach whereby four evaluation approaches, including generations, raw fitness, weighted fitness and distance calculations, are tested as alternatives. Of these, as the weighted fitness evaluation technique has the lowest standard deviation, it is the most stable for use, regardless of a running system's specifications and power requirements. Overall, based on its results from both the time-based and weighted fitness evaluation approaches, the modified GA is better than previously published algorithms.

Keywords

Vehicle Routing Problem, Dynamic Vehicle Routing, Metaheuristics, Modified Genetic Algorithm, Local Optimal.

Acknowledgements

In the name of ALLAH, the Most Gracious, the Most Merciful.

Alhamdullilah, all praises to ALLAH for his strength and blessing in completing this thesis. I also thank ALLAH for providing me with supportive supervisors, family and friends during my master study.

I would like to express my appreciation to all the individuals without whom the completion of this thesis would not be possible. First of all, my special thanks to my supervisor Dr. Daryl Essam, for generously providing guidance on the technical aspects of this thesis, for continuously encouraging me and pushing me to my limit to complete my thesis, and for all of the patience and support he gave me and accepting me as a student. My appreciation likewise extends to my co-supervisor Associate Prof Ruhul Sarker, for his useful discussions, suggestions, comments and his invaluable assistance.

To my ever supportive partner in life, my wife (Maiooom), thank you for helping me in my life, patience and support. Thank you for your love and for believing in me that I can finish this thesis.

My deepest gratitude goes to my mother, father, mother in law, father in law, brothers Wael, Moustafa and Sameh and their children Zeinab (Zoba), Fouad (Fo2sh) and AbdelMonaem (elBal6agy) and sisters in law (Mahytab, Maram and Menna) for their unflagging love and support throughout my life.

Individual acknowledgement to my friends for sharing their joyful moments with me and their moral support; Ahmed Fathi, Saber ElSayed, Ibrahim Hamid, Ahmed Mostafa, Ahmed Samir, Amr Ghoneim, Ayman Ghoneim, Osama Hassanein, Mohamed Abdella, Essam Soliman and their wives and children. Also to my research mates Eman Samir, Sohel Rana, Abdul Barik, Sajal Kumar Das and Humyun Fuad Rahman.

I would also like to thank the examiners of my thesis Prof. Graham Kendall and Dr. Phil Kilby.

Last but not lease, I would like to thank all the staff of the University of New South Wales (UNSW) at Canberra and the UNSW for providing me with a scholarship to study at this institution.

List of Publications

- AbdelMonaem F. M. AbdAllah, Daryl L. Essam, Ruhul A. Sarker “USING A DIVERSIFIED GENETIC ALGORITHM TO SOLVE DYNAMIC VEHICLE ROUTING PROBLEMS”, Proceedings of the IASTED International Conference Artificial Intelligence and Applications (AIA 2013), pp. 9-14 February 11-13, 2013 Innsbruck, Austria.
- AbdelMonaem F. M. AbdAllah, Daryl L. Essam, Ruhul A. Sarker “On using Genetic Algorithm to analyse and solve Dynamic Vehicle Routing Problems”, is submitted to “Computers & Operations Research” journal, Elsevier.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

TSP	Travelling Salesman Problem
NP-hard	non-deterministic polynomial hard
VRP	Vehicle Routing Problem
DVRP	Dynamic Vehicle Routing Problem
STSP	Symmetric TSP
ATSP	Asymmetric TSP
m-TSP	Multiple TSP
CVRP	Capacitated Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
SVRP	Stochastic Vehicle Routing Problem
VRPPD	Vehicle Routing Problem with Pickup and Delivery
LIFO	Last-In, First-Out
VRPB	Vehicle Routing Problem with Backhauls
MDVRP	Multiple Depot Vehicle Routing Problem
dod	Degree of dynamism
B&B	Branch and Bound
B&C	Branch and Cut
NNI	Nearest Neighbour Insertion
CWS	Clarke and Wright's Savings algorithm
ILK	Iterated Lin-Kernighan
GRASP	Greedy Randomized Adaptive Search Procedure
SA	Simulated Annealing
TS	Tabu Search
VNS	Variable Neighbourhood Search
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
RWS	Roulette Wheel Selection
PMX	Partially-Mapped Crossover
BCRC	Best Cost Route Crossover

Chapter 1
Introduction

The Dynamic Vehicle Routing Problem (DVRP) is one of the important variants of the Vehicle Routing Problem (VRP), which is a complex type of routing problem that involves determining an efficient set of multiple routes for a fleet of vehicles that start and end at a central depot, so as to service a given set of customers. The purpose of this thesis is to focus on solving the DVRP using an enhanced Genetic Algorithm (GA). This chapter introduces the research motivations, provides a short background to the research topic and then describes the problem, research objectives and contributions. Finally, an overview of the thesis organization and its contents are given.

1.1 Background and Motivations

In the last few decades, peoples' lives have been greatly improved due to advances in transportation and logistics, which are important because their costs constitute a significant percentage of the total cost of any product. In fact, a company usually spends more than 20% of a product's value on transportation and logistics (Hoff et al., 2008). Furthermore, the transportation system itself is a sector worthy of consideration, because its volume and impact on society continue to increase daily (Hosny, 2010). Nevertheless, it also produces negative impacts, such as congestion, noise, pollution and/or accidents (Hosny, 2010). Efficient automated vehicle routing systems using computerized optimization tools can help to minimize these negative impacts, as well as costs, because by reducing mileage they can improve the efficiency of both a driver and vehicle. In addition, they can improve customer service, reduce carbon emissions, and decrease administrative costs (Hosny, 2010). As a result, the study of routing problems, which are complex problems relating to real-life systems, such as feeder, courier and medical emergency services, has increased tremendously during the last few decades (Laporte, 2009).

One of the simplest and most common routing problems is the Travelling Salesman Problem (TSP), which is an old and well-studied problem in computer science

(Applegate et al., 2006). In it, given a set of cities, a salesman must visit each city once and return to the starting city. Its objective is to minimize the total distance travelled in a single trip. It is a non-deterministic polynomial hard (NP-hard) combinatorial optimization problem (Gutin et al., 2002). Thus, the running time required for any algorithm to solve the TSP increases exponentially with the number of cities.

The VRP is a more complex and higher-level type of routing problem than the TSP (Dantzig and Ramser, 1959). However, the two problems are closely related because the basic VRP involves determining an efficient set of multiple TSP routes for a fleet of vehicles, that all start and end at a central depot, to serve a given set of customers. It is an NP-hard combinatorial optimization problem which has been studied for over fifty years due to its importance and complexity (Laporte, 2009). There are various techniques for solving it, which can be categorized into three main types: exact, heuristics and metaheuristics.

Actually, the VRP is considered to be a broad class of routing problems, as it has several variants, such as the Capacitated VRP (CVRP), VRP with Time Windows (VRPTW), Stochastic VRP (SVRP), VRP with Pickup and Delivery (VRPPD), Multiple Depot VRP (MDVRP) and DVRP (Laporte, 2009).

1.2 Problem Description

One of the most important variants of the VRP is the DVRP. Unlike the static VRP, in the DVRP not all customers are known in advance, but are revealed as the system progresses (Montemanni et al., 2005). Thus, DVRP applications are frequently seen in various real-life systems that operate on a dynamic basis, such as supply and distribution companies, and taxi cabs, emergency and courier services.

Similar to the VRP, the DVRP is an NP-hard optimization problem, so optimization techniques that are capable of producing high-quality solutions within a limited time, i.e., metaheuristics, are very useful in finding a solution for them. Such techniques include the Ant Colony System (ACS) (Montemanni et al., 2005), GA (Hanshar and Ombuki-Berman, 2007), Tabu Search (TS) (Hanshar and Ombuki-Berman, 2007), Variable Neighbour Search (VNS) (Khouadjiaa et al., 2012) and Particle Swarm (PS) (Khouadjiaa et al., 2012).

1.3 Research Objectives and Contributions

The main objective of this research is to find good solutions for the DVRP using an enhanced GA. To enhance the GA, modifications to its processes for determining the initial population for the first time slice and/or other time slices, selection, swap mutation and the detection and management of the Local Optimal Condition (LOC) are proposed.

To date, a time-based approach, in which a DVRP system is run for a specified amount of time for each time slice to solve the problem, has been used to evaluate the system (Montemanni et al., 2005; Hanshar and Ombuki-Berman, 2007; Runka, 2008). Another objective of this research is to propose a new, fair and non-system-dependent approach for evaluating DVRP systems, in which four alternatives include generations, raw fitness, weighted fitness, and distance calculations, are tested.

1.4 Computational Environment

All algorithmic implementations presented in this thesis are programmed in C++ using Microsoft Visual Studio 2008, and all computational experiments were carried out using an Intel (R) Core i7 CPU, 2.80 GHz machine with 4 GB RAM and under a Windows 7 64-bit operating system. The developed algorithms were tested on published DVRP benchmark data.

1.5 Thesis Organization

This thesis is organized in the following 5 chapters.

- Chapter 1: Introduction
- Chapter 2: Background Study
- Chapter 3: System Design, Implementation and Preliminary Experiments
- Chapter 4: Advanced Experimental Studies
- Chapter 5: Conclusions and Future Research Directions

Chapter 1 provides a brief background to the research topic, describes the problem, and presents the research objectives, contributions and organization of this thesis.

Chapter 2 consists of two parts. The first describes the VRP, starting with the TSP and followed by VRP and its variants, while focusing on the DVRP, and the second introduces some of the solution techniques proposed for the VRP and DVRP.

Chapter 3 describes how the DVRP is handled in this thesis and how it is solved using the enhanced GA. Also, the preliminary experimental results are presented and a time-based comparison with the previously published algorithms is also provided.

Chapter 4 provides a detailed discussion of the effects of the proposed modifications to the GA. It also demonstrates the effects of the proposed GA modifications for improving and increasing diversity and escaping from local optima. Then it presents four proposed approaches for evaluating DVRP systems, from which a new evaluation approach is determined. A comparison with previously published algorithms, based on the new evaluation approach, is also provided.

Chapter 5 summarizes the main research findings and conclusions, together with possible future research directions.

Chapter 2
Background Study

The purpose of this research is to focus on solving the Dynamic Vehicle Routing Problem (DVRP), which is a member of the large Vehicle Routing Problem (VRP) family, by using an enhanced Genetic Algorithm (GA). The first part of this chapter describes the Travelling Salesman Problem (TSP), and the VRP and its variants, focusing on the DVRP. Then, some of the solution techniques that have been proposed for VRPs, including the DVRP, are mentioned.

2.1 Routing Problems

Routing problems are complex problems that relate to transportation networks, including such real-life systems as feeder, courier and medical emergency services. Their importance comes from the fact that any product's transportation cost constitutes a significant percentage of its total cost. In fact, a company usually spends more than 20% of a product's value on transportation and logistics (Hoff et al., 2008). In addition, a transportation network is itself a significant sector, as its volume and impact on society continue to increase daily (Hosny, 2010).

In the subsequent parts of this section, several routing problems are discussed.

2.1.1 Travelling Salesman Problem (TSP)

The TSP is an old and well-studied problem in computer science, in which a salesman must visit each of a given number of cities once, and then return to the city from which he/she started (Applegate et al., 2006). Its objective is to minimize the total travel cost incurred by specifying a single route that determines the most efficient order to achieve this task, while using only one vehicle with no further capacity restriction.

More formally, a routing problem such as the TSP can be structured as a complete graph, $G(V, E)$, with a cost matrix, C (Cormen et al., 2001; Hahsler and Hornik, 2007; Montemanni et al., 2005), given that:

- $V = \{v_0, v_1, v_2, ..., v_n\}$ is a set of vertices (nodes) that represents the starting location (0) and cities to be visited (1, . . . , n);
- $E = \{(v_i, v_j) \mid 0 \leq i, j \leq n, i \neq j, v_i, v_j \in V\}$ is a set of edges (arcs) between the vertices called the roads where each edge is associated with a cost (time and/or distance); and
- $C = (c_{ij})$ is a cost matrix, where if $i = j$, $c_{ij} = 0$, and if the problem is symmetrical, $c_{ij} = c_{ji}$. Also, the triangle inequality is generally assumed to hold, that is, $c_{ij} \leq (c_{ik} + c_{kj})$ $(0 \leq i,j,k \leq n)$.

For the TSP, a route could be defined as a vector, $R = (v_0, v_1, ..., v_{k+1})$, where $v_0 = v_{k+1}$ and $0 \leq k \leq n$. The first restriction, $v_0 = v_{k+1}$, ensures that the route starts and ends at the same location (Applegate et al., 2006) and the goal is to minimize $\sum_{j=0}^{k} c_{j,j+1}$.

A TSP may be symmetric or asymmetric. In most cases, as the distance between two nodes (for example, *A* to *B* or *B* to *A)* in a TSP network is the same in both directions, it is called a symmetric TSP (STSP), but otherwise, an asymmetric TSP (ATSP) (Cormen et al., 2001; Gutin et al., 2002; Glover et al., 2001; Gutin and Punnen, 2004; Hahsler and Hornik, 2007). Figure 2.1 shows an example of a TSP.

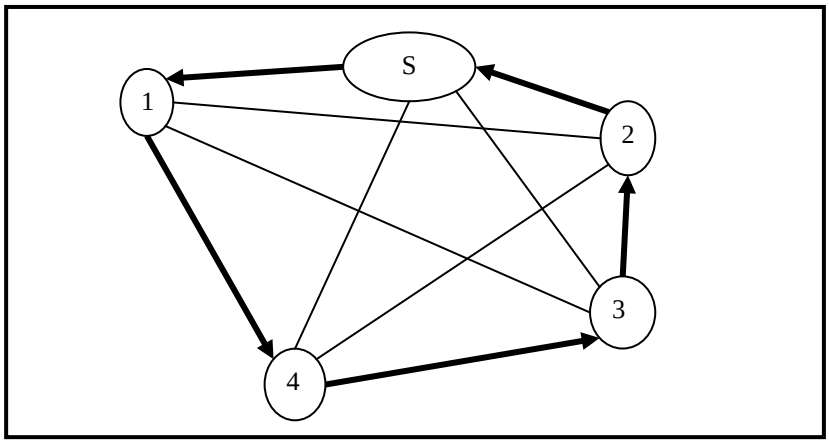

Figure 2.1: Example of a TSP (S: Source)

As the TSP is a non-deterministic polynomial hard (NP-hard) combinatorial optimization problem, it is likely that the worst-case of running time for any algorithm for solving it increases exponentially with the number of cities involved (Grover, 1992; Dorigo and Gambardella, 1997; Gutin et al., 2002; Punnen et al., 2003; Gutin and Punnen, 2004).

2.1.2 Vehicle Routing Problem (VRP)

The VRP is a generalization of the TSP, that may also be called a multiple TSP (m-TSP), which has more than one salesman, where m ≥ 1, defines the number of vehicles or fleet size (Bektas, 2006).

It is a more complex and higher-level type of routing problem than the TSP, although the two are closely related, and is also one of the most important combinatorial optimization problems. The basic VRP involves determining an efficient set of multiple TSP routes, that all start and end at a central depot, for a fleet of vehicles that intends to serve a given set of customers (Dantzig and Ramser, 1959). All customers should be visited once by only one vehicle, and typically, as a single route may exceed a given length or travel time, there is therefore a need for multiple routes (Cordeau et al., 2005; Hinton, 2010; Gendreau et al., 2002; Hosny, 2010; Laporte, 2009).

The objective of the VRP, is to minimize the route length, service cost, travel time, number of vehicles and/or a combination of these depending on the particular application (Dantzig and Ramser, 1959). It aims to serve a set of customers with specific demands using a fleet of vehicles while minimizing the times taken and/or distances travelled. A VRP solution has some specific constraints, which may include that every customer must be served by only one vehicle (Gendreau et al., 2002; Cordeau et al., 2005). As the VRP is an NP-hard problem which is both important and complex, during the last few decades, researchers have been paying more attention to solve it (Cordeau et al., 2005; Hinton, 2010; Gendreau et al., 2002; Hosny, 2010; Laporte, 2009).

The VRP can be modelled in the same way as the TSP in the previous section. The quantity of goods, q_i, requested by each customer, i (where $i > 0$), is associated with its corresponding vertex (1, . . . , n), while the starting location of the vehicles is node 0 (the depot) (Montemanni et al., 2005). If the set of vehicle routes = {VR_1, . . . , VR_m} is a partition of VR into m vehicle routes that serve all customers, the cost/distance traveled for a given vehicle route, $VR_i = \{v_0, v_1, \ldots, v_{k+1}\}$, where $v_j \in V$ and the depot $v_{k+1} = v_0$, is given by:

$$Cost\ (VR_i) = \sum_{j=0}^{k+1} c_{j,j+1}, \tag{2.1}$$

and the overall solution cost, S, is

$$S = \sum_{i=1}^{m} Cost(VR_i). \quad (2.2)$$

In general, the VRP aims to minimize the overall cost of the solution using m vehicles under the following constraints:

- each vehicle starts from and returns to the depot; and
- each customer is served once by only one vehicle.

Figure 2.2 shows a small example of a VRP.

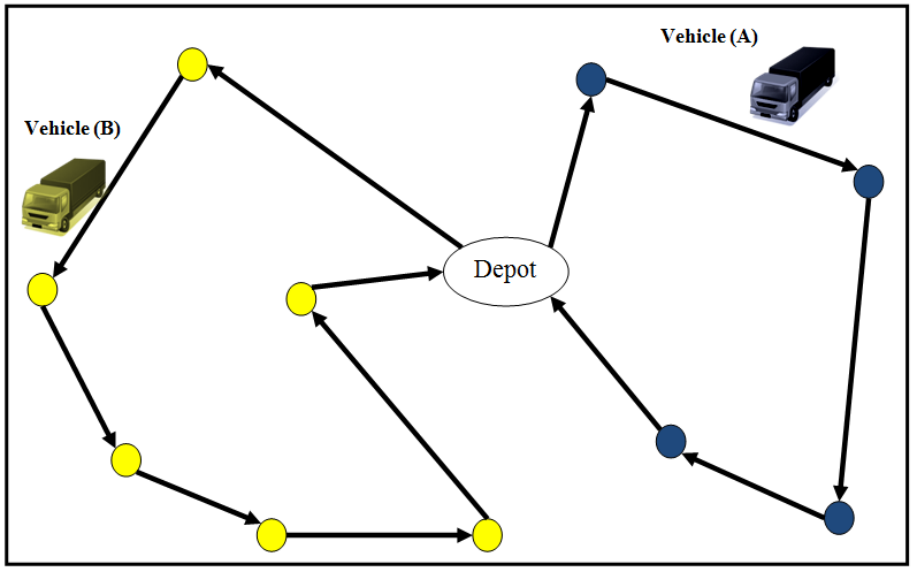

Figure 2.2: Example of a VRP with 10 customers and 2 vehicles

2.1.2.1 VRP Variants

The VRP is actually considered to be a broad class of routing problems and is composed of many variants that may require modifications of the definition given in the previous section. This section provides an overview of the most common variants. It is also possible that two or more can be combined to form more complex VRP variants.

a) Capacitated VRP (CVRP)

The CVRP is both the most common and basic variant of the VRP. It has the additional constraint that each vehicle within the fleet has a specific carrying capacity for a commodity, and thus the sum of the customers' demands that it serves along any route assigned to it must be less than, or equal to, its capacity, Q (Takes, 2010; Ralphs et al., 2003).

It can be further classified as homogeneous and heterogeneous. In a homogeneous (uniform fleet) CVRP, each vehicle has the same capacity, Q. To ensure that all vehicles are sufficiently large; the demand of any customer is never greater than the capacity of any vehicle, that is, $q_i \leq Q$ $(1 \leq i \leq n)$. Also, the total demands of all customers cannot be greater than the total capacities of all vehicles, that is, $(\sum_{i=1}^{n} q_i) \leq m * Q$ (Toth and Vigo, 2001b). An example of this type of system is shown in Figure 2.3.

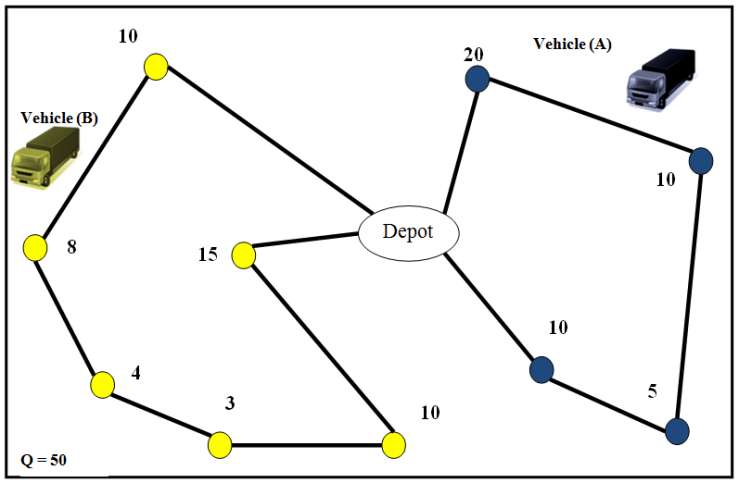

Figure 2.3: Example of homogeneous CVRP

The heterogeneous (mixed fleet) CVRP is composed of different vehicle types, each with its own capacity, and the demand of any customer is never greater than the capacity of the largest vehicle, i.e., $q_i \leq Q_{max}$ $(1 \leq i \leq n)$. Restrictions similar to the ones defined above for the homogeneous CVRP, are also applied to the heterogeneous CVRP (Toth and Vigo, 2001b). An example is shown in Figure 2.4.

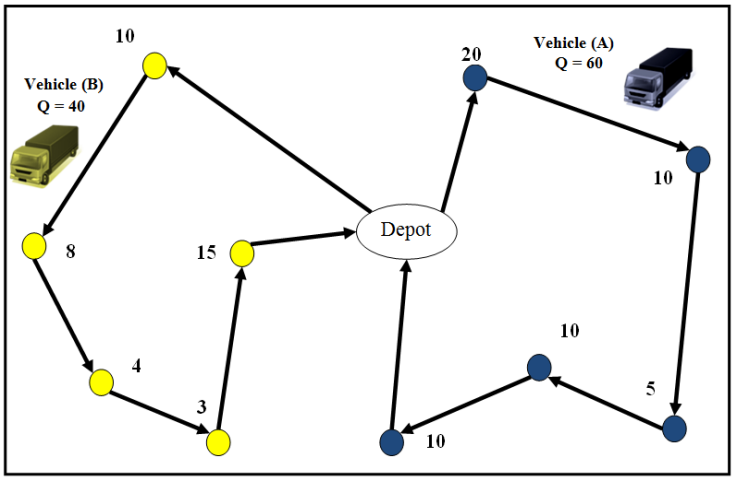

Figure 2.4: Example of heterogeneous CVRP

b) VRP with Time Windows (VRPTW)

The VRPTW is an extension of the traditional VRP, in which there is the additional restriction that customer *i* has to be served within a scheduling horizon, called a time window, $[s_i, e_i]$, where s_i is the start time and e_i the end time for serving customer i, for which definitions of the feasibility constraints of the VRP are extended (Cordeau et al., 2001; Bräysy, 2003).

c) Stochastic VRP (SVRP)

The SVRP is an extension of the traditional VRP, in which some or all of the variables or properties of the VRP are random (Takes, 2010). Examples for the SVRP include (Bianchi et al., 2006; Van Woensel et al., 2008; Secomandi and Margot, 2009):

- a customer may be present only with a certain probability;
- a customer has a certain random demand; and
- a random factor can be incorporated into the customer's demand or service time.

d) VRP with Pickup and Delivery (VRPPD)

The VRPPD is an extension of the traditional VRP in which goods are transported from pickup to delivery points. Such a problem needs to be solved in real-life situations and is usually much more complicated than the classical VRP; for example, in practice goods not only need to be taken from the depot to the customers, but are also picked up from a set of customers and are brought back to the depot (Takes, 2010).

A similar variant is the VRP with Backhauls (VRPB) which is, in essence, the same as the VRPPD except that it has the critical restriction that all goods must be delivered before any goods can be picked up, because the vehicle is full, that is, 'Last-In, First-Out' (LIFO) (Dethloff, 2001; Crispim and Brandao, 2001).

e) Multiple Depot VRP (MDVRP)

The MDVRP is an extension of the traditional VRP in which there is more than one depot from which vehicles can start and terminate their routes. For instance, as large companies such as those in the gas industry and emergency services, have more than one depot, it is common that their goal is to minimize the total travel cost of their vehicles (Ombuki-Berman and Hanshar, 2009).

The MDVRP can further be classified as a fixed or non-fixed destination problem. In the former, each vehicle starts and terminates its route at the same depot, whereas in the latter, vehicles start and terminate at different depots (Ho et al., 2008; Surekha and Sumathi, 2011), as shown in Figure 2.5.

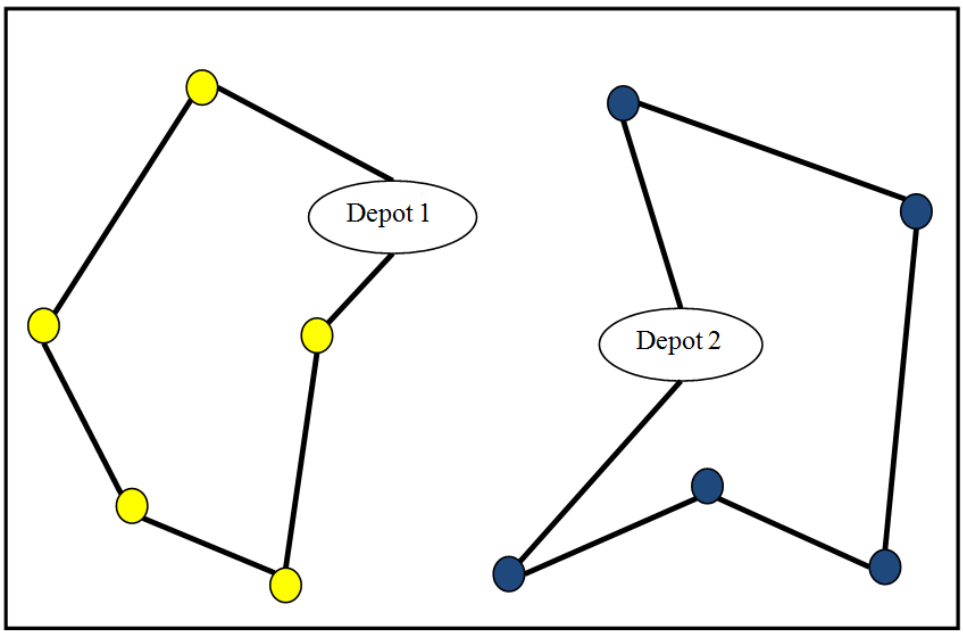

Figure 2.5: Example of MDVRP

2.1.2.2 Dynamic Vehicle Routing Problem (DVRP)

As this thesis focuses on solving the DVRP, it is considered in more detail in this section.

The DVRP is an extension of the traditional VRP, in which not all the customers are known in advance, but are revealed as the system progresses. Thus, system routes must be changed to consider new customers (Kilby et al., 1998).

Consequently, the DVRP is applicable to many real-life problems that operate on a dynamic basis, whereby vehicles leave their depots before all customers are known. A short review of the main applications that motivate research in the field of the real-time/dynamic VRP are (Montemanni et al., 2005; Hanshar and Ombuki-Berman, 2007; Cre´Put et al., 2011; Larsen, 2000; Ghiani et al., 2003; Gendreau and Potvin, 1998; Pillac et al., 2011):

- Supply and distribution companies: in seller-managed systems, as the actual demand quantity may be uncertain, some customers might run out of stock and have to be served urgently.
- Taxi cab services: their percentage of dynamic customers is very high because only a few are known before each taxi cab leaves its station to begin its duty.
- Emergency services: these include police, fire fighting and ambulance services, for which the percentage of dynamic customers is also very high, and sometimes the problem is purely dynamic because all the customers are unknown and arrive in real time.
- Courier services: as national/international express mail services offer to pick up packages and/or mail at certain locations and deliver them safely to other locations, they must respond to customer requests in real time.
- Rescue and repair service companies: these usually involve utility firms which deal with problems such as car breakdowns and electricity, gas and water faults, and must respond to customer requests for maintenance and/or repair of vehicles/facilities in a short time.
- Dynamic dial-a-ride systems: these deal with customers and/or goods that must be picked-up at one location and taken to another; for example, the transportation of elderly and disabled people.

The objective to be optimized is often a combination of different measures which depend on the nature of the system. Also, the DVRP inherits the classical objectives defined in the conventional VRP (Larsen, 2000; Larsen et al., 2008; Pillac et al., 2011).

Figure 2.6 shows an example of the DVRP, where bold arrows show completed route segments (that contain already committed customers), normal arrows represent planned routes (that contain serviceable customers), dashed arrows represent new route segments, circles represent known customers' orders, and rectangles represent new customers' orders.

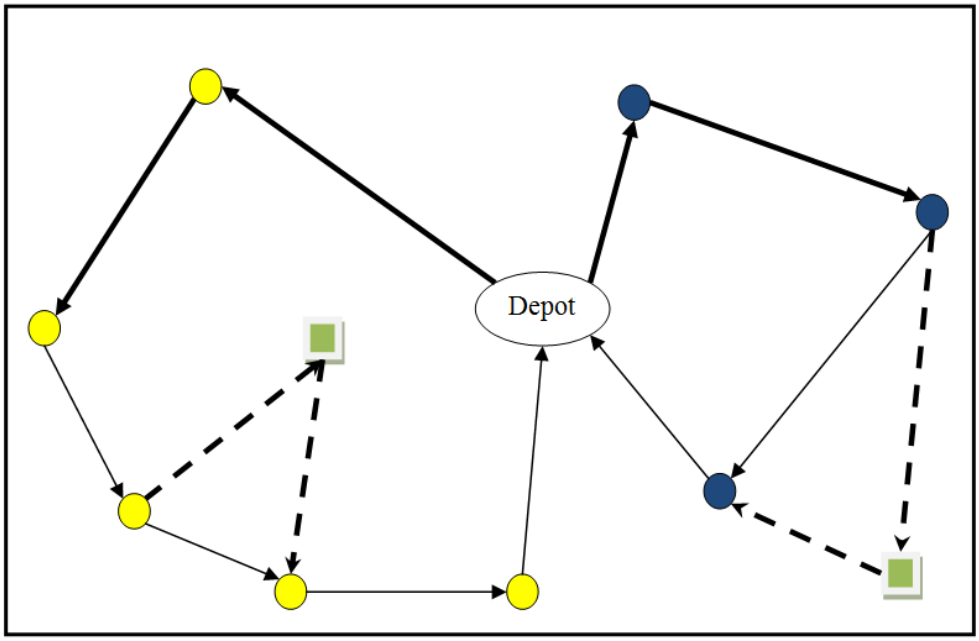

Figure 2.6: Example of DVRP with two vehicles, 8 known customers' orders and 2 new customers' orders

The main difference between static and dynamic VRP, is that as in the latter new orders arrive when the working day has already started, it is thus necessary to dynamically change the optimization problem (Montemanni et al., 2005). Therefore, the DVRP can be modelled as a sequence of static VRP instances which contain all the customers known at that time but not yet served. The ratio of the known to unknown customers when the system starts is called the degree of dynamism (*dod*) (Kilby et al., 1998; Pillac et al., 2011; Larsen et al., 2002):

$$dod = \frac{number\ of\ static\ customers\ (known\ in\ advance)}{All\ customers} \tag{2.3}$$

where the *dod* $\in$ [0, 1]. If the *dod* is 0, all requests are known in advance (completely static problem) while, if it is 1, no requests are known in advance (completely dynamic problem) (Khouadjiaa et al., 2012; Kilby et al., 1998; Larsen et al., 2002; Pillac et al., 2011).

As well as the basic VRP data, the DVRP requires the following three types of data (Kilby et al., 1998; Montemanni et al., 2005):

1. the working day that determines the available time to serve all customers;
2. an available time that determines when the customer enters the system as it is assumed that nothing is known about the task until its available time comes; and

3. a duration of time that will be spent serving each customer.

Also, the vehicle must be dispatched in time to make its visits and return to the depot before the depot closes. It will wait at its last committed customer unless the 'return home' rule happens, which only occurs in the following two situations (Montemanni et al., 2005; Khouadjiaa et al., 2012):

- all customers have been served; or
- a vehicle has used all its capacity.

The DVRP variant considered in this thesis was originally proposed by Kilby et al. (1998) and was then modified by Montemanni et al. (2005). It is based on the idea of dividing the working day, *T*, into n_{ts} time slices with $\frac{T}{n_{ts}}$ length, so that any new customer's order that arrives during a time slice is postponed to the end of it (Kilby et al., 1998). During each time slice, a problem very similar to a static VRP, but with vehicles that might have different capacities and starting locations, is created and then optimization is carried out (Montemanni et al., 2005).

While solving the DVRP, two times are used. The first is the *cutoff* time, T_{co}, which is a parameter defined by the user that is expressed as a fraction of the working day *T* which used to determine the *dod*. As mentioned before, in this DVRP model, all customers have an associated 'available time'; and each customer that has an arrival time greater than the T_{co} is interpreted as being one that arrived the day before and was not serviced. (Kilby et al., 1998; Montemanni et al., 2005; Hanshar and Ombuki-Berman, 2007).

The second time is the advanced commitment time, T_{ac}, which is the commit horizon, and is also expressed as a fraction of the working day, *T* (Kilby et al., 1998; Montemanni et al., 2005; Hanshar and Ombuki-Berman, 2007). In practice, as an order has to be committed to a driver at least T_{ac} seconds prior to the planned time of departure from the last location visited, a T_{ac} of zero seconds means that all decisions are committed at the last possible moment. After each time slice, the solution is chosen and customers with processing times that start within the next $\frac{T}{n_{ts}} + T_{ac}$ seconds are committed to their respective vehicles.

Therefore, in each time slice, a given customer may be in one of the following three states (Runka, 2008; Hanshar and Ombuki-Berman, 2007):

- not serviceable, that is, not included in the routing scheme because he/she is not known a priori;
- serviceable, that is included in the routing scheme and does not have a fixed position in the routing scheme; or
- committed, that is already served, and so has a fixed position in the routing scheme.

Moreover, in contrast to the VRP, for the DVRP the solution(s) obtained from the previous time slice(s) can be used as an initial population for the next time slice(s).

2.2 Solution Techniques

There are many techniques for solving routing problems. Some of these techniques are exact methods that guarantee to find the optimal, exact and best solution given sufficient time, and others are approximation techniques (heuristics and metaheuristics) which produce good solutions in a reasonable amount of time, although there is no guarantee of achieving optimality.

These techniques can be categorized into three main types: exact, heuristics-based, and metaheuristics-based.

2.2.1 Exact Methods

Generally, exact methods attempt to obtain the exact, and thus the best solution to a given problem. As both the TSP and VRP are NP-hard problems, there is no known exact method that works well for solving every one of their instances (Takes, 2010; Hosny, 2010). In other words, they might take an arbitrarily large time to obtain an optimal solution.

The most well-known exact methods are Branch and Bound (B&B) and Branch and Cut (B&C) (Toth and Vigo, 2001a; Fukasawa et al., 2006) while many others have also been proposed in the literature (Christofides et al., 1981; Laporte and Nobert, 1987).

In general, as exact methods search among all the possible solutions to a problem, they are usually prohibitive for large problems, which limits in their applicability in comparison to heuristic-based and metaheuristic-based methods.

2.2.2 Heuristic-based Methods

Although heuristic methods do not guarantee to obtain the best solution, they are able to find a good solution in a reasonable time. Most of those used to solve the VRP are derived from the TSP (Laporte, 1992).

Heuristic methods can be classified into the following three main categories, that are based on their nature.

1) Construction algorithms are those in which the solution starts with an empty route, which is then constructed by including customers, usually one at a time, until a complete tour is developed, while trying to keep the total cost as low as possible. The most well-known algorithms in this category are Nearest Neighbour Insertion (NNI) and Clarke and Wright's Savings algorithm (CWS) (Clark and Wright, 1964; Hahsler and Hornik, 2007; Laporte, 1992; He et al., 2010).
2) Improvement algorithms are those which construct a solution and then try to improve it by searching for a more efficient one. The most well-known algorithms in this category are r-Opt local searches (2-Opt and 3-Opt) (Bräysy and Gendreau, 2005; Hahsler and Hornik, 2007) and Iterated Lin-Kernighan (ILK) (Helsgaun, 2000).
3) Two-phase heuristics are those which work to find a solution through two different phases, such as clustering and routing (Taillard et al., 1996; Shen et al., 2010).

2.2.3 Metaheuristic-based Methods

As the classical heuristic methods that were mentioned briefly in the previous section perform limited searching; they can become stuck in local optimal conditions. In contrast, metaheuristics are used to minimize the probability of this happening, and so try to reach a global optimal solution (Osman and Kelly, 1996). In this section, some metaheuristics methods that have been successfully applied to the VRP are discussed.

Metaheuristics are a set of concepts that can be used to guide other heuristics to find their way out of a local optimal solution, by continuing the search for better areas of the solution space, and thereby try to reach a global optimal solution (Osman and Laporte, 1996; Manfrin, 2004). In the literature, many criteria for classifying metaheuristics have been proposed (Stützle, 1998), with a useful one being to consider the number of solutions used at the same time. In this way, metaheuristics can be classified into 'single-point' and 'population-based'; in other words, at any particular time, whether the algorithm works on a single solution or population (Manfrin, 2004; Talbi, 2009) i.e.,

i. Single-point search Metaheuristic algorithms which work on a single solution, and which are also called trajectory methods, encompass local search-based metaheuristics, such as Tabu Search (TS) and Simulated Annealing (SA).
ii. Population-based search Metaheuristic algorithms which perform search processes by describing the evolution of a set of points in the search space at the same time, such as Ant Colony Optimization (ACO) and GA.

More details of the main metaheuristic algorithms that have been applied to the VRP are provided in the following.

a) Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP is a single-point multi-start local search metaheuristic algorithm. The GRASP methodology was proposed by Feo and Resende (1995). It is obvious from its name that it is a randomized greedy metaheuristic technique which generates different starting solutions for applying a local search (Feo and Resende, 1995; Dorigo and Stützle, 2004).

It is an iterative metaheuristic, in which each iteration consists of two main phases: construction and local search. In the construction phase, initial solutions are constructed by a construction heuristic that is based on a greedy technique, then GRASP tries to improve the obtained solutions by a local search (Dorigo and Stützle, 2004).

In general, although GRASP is based on some simple greedy techniques, it includes some randomization to diversify the search of the solution space and is applied many times to increase the likelihood of identifying a good-quality solution (Contardo et al., 2011; Talbi, 2009).

b) Simulated Annealing (SA)

SA is a single-point generic probabilistic metaheuristic algorithm, and is commonly said to be the most popular metaheuristic. It was first described by Kirkpatrick et al. (1983) and is based on the work of Metropolis et al. (1953). It is one of the first algorithms that incorporates an explicit local optimal escaping strategy (Van Laarhoven and Aarts, 1987) and has been used to solve various combinatorial optimization problems (Manfrin, 2004).

The idea behind it is the annealing process of solids, such as metal and glass, which effectively minimize the energy of a system by using slow cooling until its atoms reach a stable state. It has an annealing schedule, a critical issue for solving a problem, that includes the initial temperature and the rate at which it is reduced (Van Laarhoven and Aarts, 1987; Czech and Czarnas, 2002).

SA starts with a certain feasible solution to a problem, which it then tries to optimize using a method analogous to the annealing of solids. To do this, a neighbour of a current solution is generated using an appropriate method and its cost (fitness) calculated. If this new solution is better than the current one, it is fully accepted; otherwise, it is accepted with a certain probability (Chiang and Russell, 1996; Talbi, 2009).

While SA is running, as the temperature is gradually reduced, the probability of the acceptance of low-quality solutions becomes very small. Therefore, high temperatures allow a better exploration of the search space and lower ones allow the fine tuning of a good solution. This process is repeated until the temperature approaches zero or no further improvement can be achieved (Takes, 2010; Hanshar and Ombuki-Berman, 2007).

c) Tabu Search (TS)

TS is a single-point metaheuristic algorithm proposed by Fred Glover (1977) and is one of the most successful algorithms for solving routing problems. Its basic idea is to use a search-space history to escape from local minima, and thereby implements an exploration strategy (Manfrin, 2004).

Although it is like SA in that it can 'intelligently' explore the search space in an attempt to escape a local optima trap, it has the following three main differences (Manfrin, 2004):

1. it accepts moves that improve the objective function;
2. it searches for the best solution in the current neighbourhood before applying the replacement criterion; and
3. it uses a short-term memory, called a tabu list, to record solutions that have been visited recently during the search, which are then prohibited from re-exploring.

TS is based on a best-improvement local search and uses short term memory (tabu list) to escape from local minima and to avoid cycles (repeating the same sequence of moves). The size of the tabu list is an issue for TS, as with small tabu lists the search will concentrate on small search spaces. On the opposite hand, large tabu lists force the search process to explore larger regions, because it forbids revisiting a higher number of solutions (Manfrin, 2004; Talbi, 2009).

d) Variable Neighbourhood Search (VNS)

VNS is a single-point metaheuristic algorithm proposed by Mladenovic and Hansen (1997), which tries to escape from local optima by systematically changing neighbourhoods (Khouadjiaa et al., 2012; Talbi, 2009).

It explores a distant neighbourhood of the current solution and then tries to move from the current solution to a new one, if and only if, an improvement is available. Thus, it systematically exploits neighbourhood changes by both searching for local minima and escaping from the valleys which contain them (Khouadjiaa et al., 2012; Hansen et al., 2010; Talbi, 2009).

The VNS strategy is motivated by the following three principles (Hansen et al., 2010; Brownlee, 2011):

- the local minima of different neighbourhood structures may not be the same;
- a global minimum is a local minimum for all possible neighbourhood structures; and
- for a problem, the local minima are relatively close to the global minima.

e) Large Neighbourhood Search (LNS)

LNS is a single-based metahuristic algorithm that was proposed by Shaw (1997) and can be defined as a combination of destroy and repair methods (Shaw, 1998). A destroy method destructs part of the current solution, while a repair method rebuilds the destroyed solution (Pisinger and Ropke, 2010).

LNS takes advantage of the expressiveness of Constraint Programming (CP) and the speed of Local Search (LS); its working principle is to maintain a candidate solution through out the search, that is not violating any constraint but that may not be optimal (or not known to be) (Ropke and Pisinger, 2006).

There is an extended version of LNS that is called Adaptive LNS (ALNS), which was proposed in (Ropke and Pisinger, 2006). ALNS extends the LNS algorithm by allowing multiple destroy and repair methods to be used within the same search. Thus, ALNS investigates multiple solutions (neighbourhoods) within the same search to reach better solutions (Pisinger and Ropke, 2010).

f) Ant Colony Optimization (ACO)

ACO is a population-based nature-inspired metaheuristic algorithm that can be considered as one of the newest metaheuristics for solving routing problems. Its basic idea was introduced by Dorigo (1992) and its biological basis is the communication established by ants when they seek food (Dorigo and Gambardella, 1997).

ACO was inspired by the behaviour of real ants which are able to find the shortest paths between food sources and their nests. Initially, an ant explores the area surrounding its nest in a random manner until it reaches a food source before returning to its nest. During this process, it deposits pheromone trails which are later detected by the majority of ants which follow the paths most frequently used by others, an indirect communication which allows them to find the shortest paths between their nests and food sources (Dorigo, 1992; Gendreau et al., 2008).

In an ACO implementation, the artificial ants must retain certain properties of real ants, each has a limited memory (tabu list) in which it can store both the partial paths it has followed and the cost of the links it has traversed, thereby building solutions

to its problem without entering a loop. However, artificial ants have the following major differences compared with real (natural) ones (Dorigo and Stützle, 2004):

- they have some memory;
- they are not completely blind; and
- they live in an environment where time is discrete.

ACO has three main components which are briefly explained below (Takes, 2010; Talbi, 2009).

1) Route construction determines how ants build a solution. They work in a parallel fashion with each ant designing all vehicles' routes at the same time by choosing only one client at each iteration.
2) Transition rule determines which customer is chosen next by each ant.
3) Pheromone update is performed either during or after the building of a solution. It includes an evaporation process in which the pheromone concentrations of all the solutions, except the best, are decreased, and that of the best is increased.

g) Particle Swarm Optimization (PSO)

PSO is a population-based nature-inspired metaheuristic algorithm proposed by Kennedy and Eberhart (1995), and its biological basis is the simplification of social behaviour simulations of birds flocking and fish schooling (Kennedy and Eberhart, 1995).

In PSO, each single solution (e.g., a bird) in the search space is called a particle which has a fitness value, evaluated by the fitness function of the problem to be optimized, and a velocity which directs its flying (Talbi, 2009). Particles are initially placed at random positions in the search space and then each adjusts its strategy of flying in the search space according to both its own and its peers' flying experiences to discover better positions (Kennedy et al., 2001; Clerc, 2006; Brownlee, 2011).

2.2.3.1 Genetic Algorithm (GA)

As this thesis focuses on solving the DVRP using a GA, the GA is reviewed in more detail in this section.

The GA is a population-based, nature-inspired metaheuristic that became popular through the work of Holland in the early 1970s. Although his book "Adaptation in Natural and Artificial Systems (1975)", was built upon Fraser's work (1957), and Fraser and Burnell (1970). It is one of the most popular metaheuristics for solving various combinatorial optimization problems, in particular, NP-hard problems. The idea behind it is to combine candidate individuals from the current population by applying crossover and mutation processes to try to create new enhanced individuals (Mitchell, 1998). Figure 2.7 shows an overview of the GA processes.

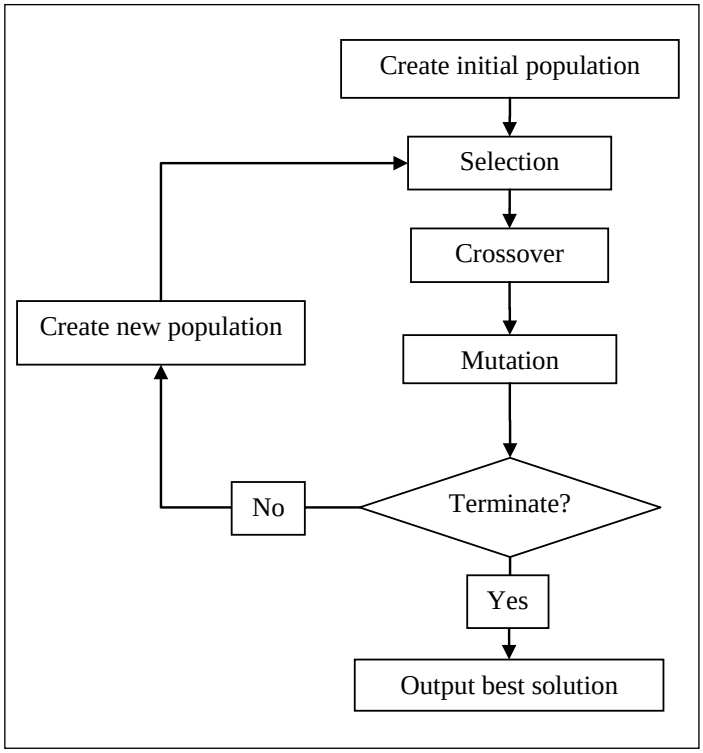

Figure 2.7: Overview of GA processes

A critical issue in the GA is encoding, which determines how to represent the problem in a chromosome form that a computer system can process to solve a given problem. There are many ways of encoding, e.g., encoding values as real or integer numbers, and the used method usually depends on the problem to be solved (Mitchell, 1998; Sivanandam and Deepa, 2008). Two different GA methods for encoding are described below.

- **Binary encoding**

Binary encoding is the most common encoding method and was first used in the GA because of its simplicity. In it, the chromosome consists of a string of bits, 0 or 1 (Sivanandam and Deepa, 2008; Mitchell, 1998), as shown in Figure 2.8.

1	0	0	1	0	1

Figure 2.8: Example of binary encoding

However, it is often not suitable, especially for problems with more than two variables, such as scheduling and routing problems.

- **Permutation encoding**

Permutation encoding is used in optimization problems which involve sorting a list or putting things in the right order, such as routing problems. In it, every chromosome consists of a string of values that represents the sequence of positions (Sivanandam and Deepa, 2008; Mitchell, 1998). Figure 2.9 shows a permutation encoding example for a TSP instance of six customers (*1*, *2*, *3*, *4*, *5* and *6*).

1	5	2	6	3	4

Figure 2.9: Example of permutation encoding

In this thesis, the DVRP, which is a routing problem, is solved, so the right order is important to get a better solution, hence the permutation encoding is used. For it, the chromosome contains all the customers to be served. It can be interpreted as the order in which a vehicle must visit all clients, assuming the same vehicle performs all trips in turn. However a split method is used on it to determine where a new vehicle must be created to serve the next customers. Prins used a splitting procedure to split optimally any sequence into trips and to thus recover the corresponding VRP solution (Prins, 2001; Prins, 2004). In this thesis, a simple splitting method is used, as it takes less time than Prins's splitting procedure. This simple splitting procedure is illustrated in Chapter 3 (Section 3.3.1).

A GA population contains a specific number of individuals (population size); each called a chromosome which consists of a string of genes. For the VRP, each gene

often represents a customer that a vehicle will visit and the chromosome determines the sequence in which that vehicle will visit all customers (Weise, 2009).

The GA begins solving a problem by creating an initial population using a random and/or heuristic construction technique. Then, each chromosome in this population is evaluated to obtain its fitness function value which is used to quantify the optimality of each solution represented (Haupt and Haupt, 2004; Sivanandam and Deepa, 2008).

When solving a problem, the GA involves three main processes (Takes, 2010; Baker and Ayechew, 2003), selection, crossover and mutation, as discussed in the following three sections.

1. **Selection** Usually, this is the first process applied on a GA population. It determines how candidate individuals in the population are chosen for reproduction (crossover and mutation) and is derived by the chromosome fitness that allows the fittest chromosomes to survive. The following are different selection methods.

- **Roulette Wheel Selection (RWS)**

RWS is used to select potentially useful solutions for recombination. In it, a chromosome's chance of being selected is proportional to its fitness value in relation to its competitors' fitness values; for example, if f_i is the fitness of individual p_i in population *P*, its probability of being selected is $p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$ (Talbi, 2009; Sivanandam and Deepa, 2008).

- **Tournament Selection**

As tournament selection is the selection method used in this thesis, it is discussed in more detail in this section. It is a chromosome selection process for recombination and closely mimics the nature of a mating competition. It randomly chooses a small subset of a certain size, *k*, of chromosomes from the mating pool (whole population), called a tournament, from which a chromosome is selected. This process is repeated for every needed chromosome (Talbi, 2009; Hanshar and Ombuki-Berman, 2007; Sivanandam and Deepa, 2008); for example, a tournament set of size two is initially randomly drawn

from population *P*. Then, a defined constant parameter, ρ, which is the selection pressure that allows some less fit solutions to be selected, is used to control how biased the selection is towards the fitter chromosomes. The steps involved in normal tournament selection are (Hanshar and Ombuki-Berman, 2007):

Step 1: randomly select a set of two from the population to fill the tournament set;

Step 2: create a random number, $r \in [0, 1]$; and

Step 3: if r <= selection threshold, select the fittest individual from the tournament to be used in reproduction

else randomly choose one of these chromosomes.

2. **Crossover** This process determines how the current population of chromosomes (parents) are combined to create new chromosomes (children). Various crossover methods are described below.

- **One-point crossover**

The one-point crossover randomly selects one crossover point and then copies every gene before this point from the first parent chromosome and every gene after this point from the second parent chromosome (Talbi, 2009; Sivanandam and Deepa, 2008), and is applied mainly on binary encoding GAs.

- **Two-point crossover**

The two-point crossover randomly selects two crossover points and then interchanges the two parent chromosomes between them to create two new chromosomes (Talbi, 2009; Sivanandam and Deepa, 2008), and is also applied mainly on binary encoding GAs.

- **Partially-Mapped Crossover (PMX)**

The PMX can be viewed as a two-point crossover extension to permutation encoding. To resolve the possible illegitimacy that may be caused by the simple two-point crossover, it uses a special mapping procedure (Talbi, 2009; Sivanandam and Deepa, 2008).

- **Best Cost Route Crossover (BCRC)**

As a modified version of the BCRC used in this thesis; it is discussed in more detail in this subsection. It was developed by Ombuki-Berman et al. (2006) and involves the following steps.

Step 1: choose the parents by using tournament selection (P_1, P_2).

Step 2: randomly select a vehicle from each parent (v_1, v_2 respectively) which must contain uncommitted customers.

Step 3: remove v_1's and v_2's customers from P_2 and P_1 respectively.

Step 4: re-insert the customers that have been removed, to locations which minimize the overall cost of the entire route. If no re-insertion location for a particular customer can be found in an existing route, create a new route (Hanshar and Ombuki-Berman, 2007) which is then added to the current parent chromosome so it can be tested for the re-insertion of upcoming customers.

The BCRC is illustrated in Figure 2.10. In step 1, after selecting two parents (P_1, P_2), a route is randomly selected from each (v_1 [*3*, *1*, *7*] and v_2 [*1*, *9*, *8*]) and that v_1 and v_2 customers are removed from P_2 and P_1 respectively. In step 2, customers *1* and *3* are tested to be inserted in the best locations in P_1 and P_2 respectively. In steps 3 and 4, customers *9* and *1*, and *8* and *7* are tested for insertion into the best locations in P_1 and P_2 respectively. However, as customer *7* has no available insertion, it is placed at the end of the chromosome. Finally, in step 5, the final newly combined vehicles are used to create new children chromosomes.

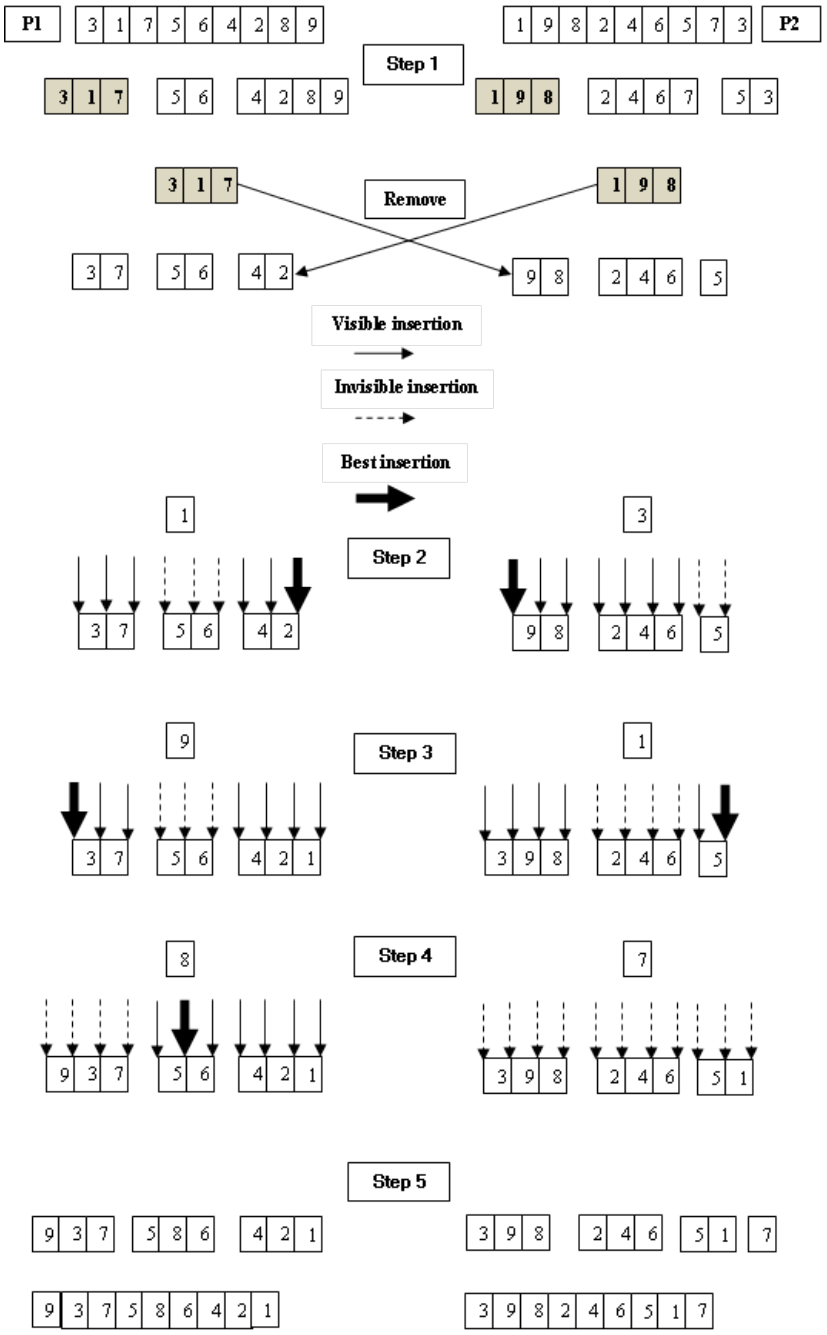

Figure 2.10: Example of BCRC

3. **<u>Mutation</u>** This process determines how new children chromosomes are self-modified through different mutation methods, such as insertion, reverse and swap.

- **Insertion**

Insertion mutation selects a gene from a random position in a chromosome and inserts it in a random position in the same chromosome (Haupt and Haupt, 2004; Sivanandam and Deepa, 2008; Mitchell, 1998), as illustrated in Figure 2.11.

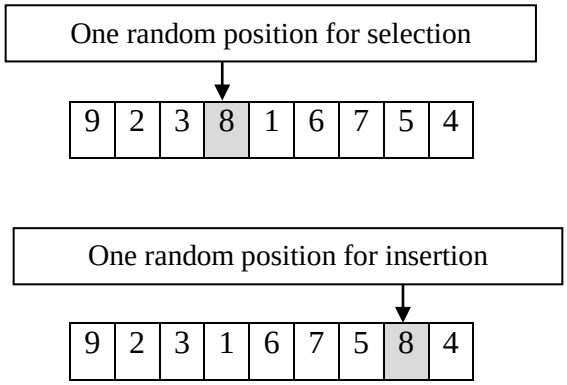

Figure 2.11: Example of insertion mutation

- **Reverse**

Reverse mutation is sometimes called inversion mutation. It selects two random positions within a chromosome and then reverses (inverts) the genes between them (Haupt and Haupt, 2004; Hanshar and Ombuki-Berman, 2007; Sivanandam and Deepa, 2008), as illustrated in Figure 2.12.

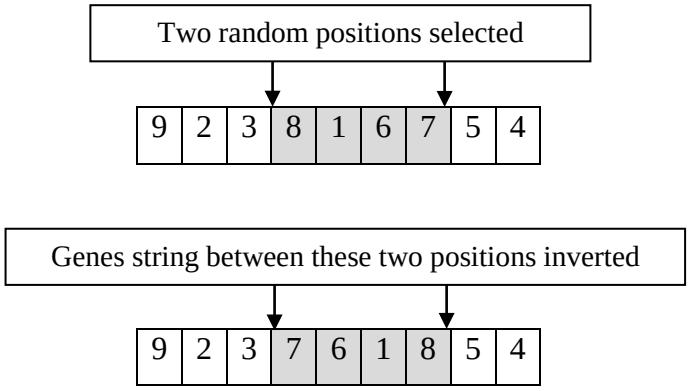

Figure 2.12: Example of reverse mutation

- **Swap**

Swap mutation selects two positions at random and then swaps their genes (Haupt and Haupt, 2004; Mitchell, 1998; Sivanandam and Deepa, 2008), as illustrated in Figure 2.13.

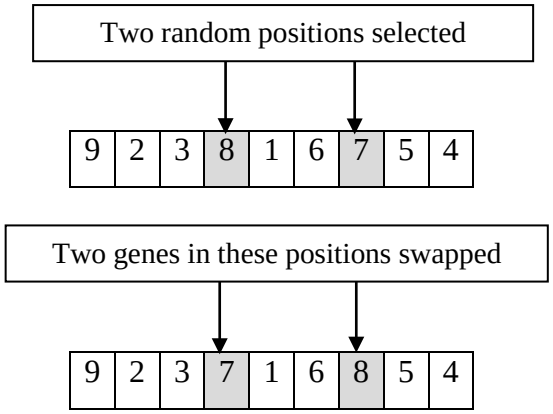

Figure 2.13: Example of swap mutation

The GA can also use an elitist strategy in which a set of the best solutions is carried from the current generation to the next one (Sivanandam and Deepa, 2008; Hanshar and Ombuki-Berman, 2007). This allows elite solutions to propagate from generation to generation and ensures that the best solution does not deteriorate over time.

2.3 Chapter Summary

In this chapter, an introduction to the TSP and the VRP and its variants, including the DVRP, as well as different techniques for solving routing problems, has been presented. Based on the literature reviewed, it is clear that the DVRP is an important routing problem applicable to many real-life problems that operate on a dynamic basis. In the next chapter, a modified GA (which aims to solve the DVRP) is proposed.

Chapter 3
System Design, Implementation and Preliminary Experiments

This thesis focuses on solving the Dynamic Vehicle Routing Problem (DVRP) using a modified Genetic Algorithm (GA). In this chapter the architecture of the adopted GA-based approach is described and solutions to DVRP benchmark problems are reported and compared with state-of-the-art algorithms.

3.1 Modified Genetic Algorithm (GA)-based DVRP

The DVRP solution approach used in this thesis comprises the two subsystems as follows:

1. The event manager subsystem controls and determines both the system and problem flows by creating a static VRP-like instance for each time slice through receiving, adding new customers and then committing the expected customers to their respective vehicles. Therefore, it manages all the inputs to and the outputs from the environment and coordinates the working of the GA.
2. The GA optimization subsystem handles the solution and optimization process. It executes the GA to solve a given static VRP-like instance generated by the event manager subsystem for each time slice. The GA tries to find the best solution for each given problem until the stopping condition is met and then reports it.

The first subsystem is the same as that used in (Montemanni et al., 2005; Hanshar and Ombuki-Berman, 2007; Runka, 2008), but the second subsystem differs as it uses a modified GA that tries to increase the diversity in an attempt to reach optimal solutions.

The proposed system structure is illustrated in Figure 3.1, and its subsystems are described in more detail in Sections 3.2 and 3.3 respectively.

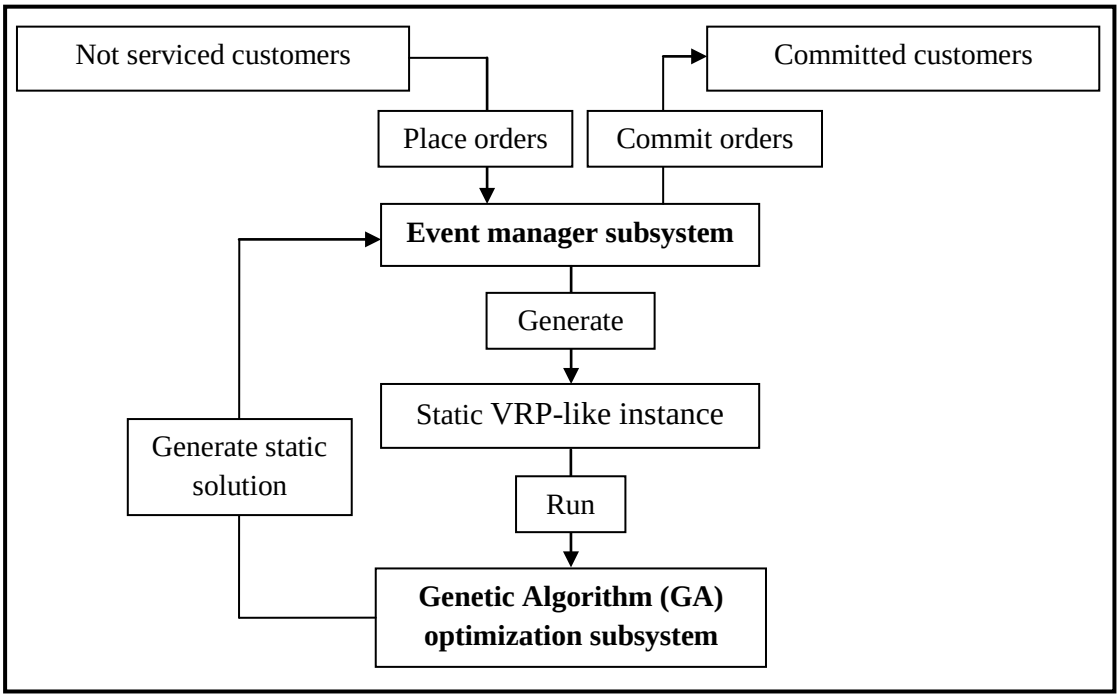

Figure 3.1: Architecture of GA-based DVRP system

3.2 Event Manager Subsystem

The event manager is a subsystem which manages the DVRP by subdividing it into a series of static VRP-like instances. Therefore, it keeps track of simulation times, customers' states and committed vehicles' routes, as well as globally static information, such as a user's parameters and GA parameters. The event manager pseudo-code is presented in Figure 3.2.

The event manager begins by solving the DVRP by initializing the problem parameters, such as: T, n_{ts}, T_{ac} and T_{co} and GA parameters such as the percentage of crossover and mutation. Then, all customers with available times greater than T_{co} are considered and assigned an initial static VRP-like instance (an initial distribution for the known customers that determines they must be visited to try to find a solution for the problem). Each instance is passed to the GA subsystem which solves the problem and returns an optimized solution. Then, the event manager uses this optimized solution to update the dynamic problem state. Each customer served in the static routing solution before the end of the next time slice is considered to be committed, that is, customers with processing times that start within the next $(\frac{T}{n_{ts}} + T_{ac})$ seconds must be committed to their respective vehicles. Therefore, this position in all subsequent static problems is fixed. Then, the event manager continues its loop to create static problems for the

remaining DVRP customers, based on the simulation time and the dynamic problem state, and continues until all customers have been committed.

Step 1: Time := 0;

Step 2: Create static VRP-like instance for initial pending customers := customers that are known from the previous day (customers with availability time > T_{co});

Step 3: StaticProb := VRP-like instance;

Step 4: Vehicles starting positions are set at the depot;

Step 5: **While** (Potentially serviceable customers still remain OR current time < T_{co})

5.1) Solution := GAsubsystem (StaticProb);

5.2) CommitCustomers (current time + T_{ts} + T_{ac} , StaticProb customers, solution);

5.3) StaticProb := RemainingCustomers in StaticProb customers ∪ customers appeared in the last $\frac{T}{n_{ts}}$ seconds;

5.4) Time := Time + T_{ts} ;

5.5) Update (Vehicles current positions, capacities, travel times);

End While

Figure 3.2: Pseudo-code for event manager subsystem

Here as an example of the event management subsystem process (Figure 3.2). It is an instance for the first three time slices in *c50*, one of the benchmark problems. For Step 1, in the first time slice, the simulation time is 0 and customers with available times of more than T * 0.5 = 351 * 0.5 = 175.5 are considered as an initial problem. Therefore, the first time-slice problem customers are *24*, *25*, *26*, *27*, *28*, *29*, *30*, *31*, *32*, *33*, *34*, *35*, *36*, *37*, *38*, *39*, *40*, *41*, *42*, *43*, *44*, *45*, 46, *47*, *48*, *49* and *50* (Step 2 and 3). Then, this problem is solved by the GA (the solution process is described in the next subsection). Note that, in the *c50* problem, all vehicles have the same capacity of 160. Now, we assume that the best chromosome for solving the first time-slice problem is [*27*, *48*, *26*, *31*, *28*, *36*, *35*, *29*, *32*, *46*, *38*, *49*, *50*, *34*, *30*, *39*, *33*, *45*, *44*, *37*, *43*, *24*, *25*, *41*, *40*, *42*, *47*]. This chromosome has three vehicles: Veh_1 [*27*, *48*, *26*, *31*, *28*, *36*, *35*, *29*, *32*, *46*] with an actual load of 110; Veh_2 [*38*, *49*, *50*, *34*, *30*, *39*, *33*, *45*, *44*, *37*] with an actual load of 160; and Veh_3 [*43*, *24*, *25*, *41*, *40*, *42*, *47*] with an actual load of 121

(Step 5.1). As the starting location of these vehicles is the depot [*0*] (initialized in Step 2), the calculated Euclidean distances from the depot to their next locations *27*, *38* and *43* for Veh_1, Veh_2 and Veh_3, are 8, 15.81 and 34.655 respectively. Note that, in the *c50* problem, each customer has time duration to be served of 15. Thus, Veh_1, Veh_2 and Veh_3 have current times after servicing customers *27*, *38* and *43* of (8 + 15 = 23), (15.81 + 15 = 30.81) and (34.655 + 15 = 49.655) respectively. To check if it is possible to commit more customers to any vehicle, the vehicles' times are compared with the current time-slice commitment time, that is, (current simulation time $+ \frac{T}{n_{ts}} + T_{ac}$) = $0 + \frac{351}{25}$ + (0.01 * *T*) = 0 + 14.04 + 0.01 * 351 = 17.55 (Step 5.2). As all the vehicles have times greater than the commitment time, it is not possible to commit more customers. Finally, Veh_1, Veh_2 and Veh_3, are assigned codes of -1, -2 and -3 respectively. Now, the system updates the Veh_1, Veh_2 and Veh_3 data so that their current locations are *27*, *38* and *43* and their updated capacities are 145, 145 and 149 respectively (Step 5.5).

In the second time slice, new customers *1*, *2*, *3*, *4* and *5* appear, as they have available times of 1, 4, 9, 12 and 12 respectively ∈ [the current simulation time, the current simulation time + $\frac{T}{n_{ts}}$] = [0, 14.04] and the simulation time is updated to 14.04 (Step 5.3). After solving the second time-slice problem, we assume that the best chromosome is [-1, *48*, *26*, *31*, *28*, *3*, *36*, *35*, *29*, *2*, *1*, *32*, -2, *50*, *34*, *30*, *39*, *33*, *45*, *49*, *5*, -3, *24*, *25*, *41*, *40*, *42*, *44*, *37*, *4*, *47*, *46*]. Again, this chromosome has three vehicles: Veh_1 [*48*, *26*, *31*, *28*, *3*, *36*, *35*, *29*, *2*, *1*, *32*] with an actual load of 143; Veh_2 [*50*, *34*, *30*, *39*, *33*, *45*, *49*, *5*] with an actual load of 141; and Veh_3 [*24*, *25*, *41*, *40*, *42*, *44*, *37*, *4*, *47*, *46*] with an actual load of 149 (Step 5.1). Firstly, the system checks if it is possible to commit more customers to the vehicles, by comparing the vehicles' times with the current time-slice commitment time updated to 31.59 (14.04 + 14.04 + 3.51) (Step 5.2). As only Veh_1 and Veh_2 have times less than the current time-slice commitment time, customers *48* and *50* will be committed to them and their times are updated to 46.60 (23 + 8.60 + 15) and 56.99 (30.81 + 11.18 + 15) respectively. After comparing the updated vehicle times, it is clear that it is not possible to commit more customers to any vehicle. Now the system updates the data of Veh_1, Veh_2 and Veh_3, with their current locations, which are *48*, *50* and *43*, and their updated capacities of 128, 135 and 149 respectively (Step 5.5).

In the third time slice, new customers *6* and *7* appear, as they have available times of 16 and 21 respectively ∈ [the current simulation time, the current simulation time + $\frac{T}{n_{ts}}$] = [14.04, 28.08] and the simulation time is updated to 28.08 (Step 5.3). After solving the third time-slice problem, the best chromosome is [-1, *26*, *31*, *28*, *3*, *36*, *35*, *29*, *2*, *1*, *32*, -2, *34*, *30*, *39*, *33*, *45*, *49*, *5*,-3, *7*, *24*, *25*, *6*, *46*, *37*, *44*, *42*, *40*, *41*, *4*, *47*] which has four vehicles: Veh_1 [*26*, *31*, *28*, *3*, *36*, *35*, *29*, *2*, *1*, *32*] with an actual load of 126; Veh_2 [*34*, *30*, *39*, *33*, *45*, *49*, *5*] with an actual load of 131; Veh_3 [*7*, *24*, *25*, *6*] with an actual load of 72; and a new vehicle Veh_4 [*46*, *37*, *44*, *42*, *40*, *41*, *4*, *47*] with an actual load of 111, which is created because Veh_3's load is too large (greater than its remaining capacity) (Step 5.1). Firstly, the new vehicle's time is assigned the current time-slice time + T_{ac} = (28.08 + 3.51) = 31.59 and its start location is the depot [0]. Then, the system checks if it is possible to commit customers to their respective vehicles by comparing the vehicles' times with the current time-slice commitment time which is updated to 45.63 (28.08 + 14.04 + 3.51). As Veh_4 has a time of less than the current commitment time, it can commit customer *46*. Therefore, vehicle Veh_4's time is updated to 48.826 (Veh_4 time = 31.59 + Euclidean distance (depot [*0*], customer *46*) = 2.236 + customer *46*'s duration = 15) and it is assigned the code -4. Then, when the system checks the vehicles' times, it is clear that it is not possible to commit more customers to them, as they all have times greater than the third time-slice commitment time. Now the system updates the Veh_1, Veh_2 and Veh_3, Veh_4 data so that their current locations are *48*, *50*, *43* and *46* and their updated capacities are 128, 135,149 and 155 respectively (Step 5.5). These processes are repeated until there is no serviceable customer remaining, or the current time is greater than T_{co}.

3.3 GA Optimization Subsystem

In this thesis, the developed GA is based on Hansher and Ombuck-Berman (2007)'s GA, but some improvements to increase both its diversity and capability to escape from local optima are proposed (these modifications are presented in bold in Figure 3.3), including:

1. modifications to the initial population for the first time slice and/or other time slices;
2. a variant GA selection process;
3. a variant swap mutation; and

4. a Local Optimal Condition (LOC) detection strategy.

The initial population for the first time slice and/or other time slices are created using both random and heuristic procedures. The proposed modification of the selection process allows the not-the-fittest chromosome to be selected to try to increase the diversity. The swap mutation modification provides more variations in the mutated chromosome which again increases diversity, while the LOC detection strategy allows the GA to try to escape from local optima by increasing diversity when such a condition is detected.

The GA optimization subsystem handles the solution and optimization processes for generating static VRP-like instances using a GA algorithm, and is repeated several times over a discrete series of time slices that are managed and coordinated by the event manager subsystem. A GA flow diagram is presented in Figure 3.3. It shows the proposed modifications in bold font. To summarise, once an initial population is created, its chromosomes are evaluated to be assigned their corresponding fitness and then a new empty population is created for the next generation. The current population is used to fill the new empty population by selecting two old chromosomes (parents), and a reproductive process (crossover and mutation) is used to create two new chromosomes (children) which are then evaluated. After evaluating the new chromosomes, the stopping condition is checked; if it is met, the GA halts and reports the best-found solution, whereas if it is not, the new population is checked. If this new population is not complete, the processes (selection, crossover and mutation) are repeated, otherwise a new empty population is created for the next generation, and to fill it, the current population is used as the new old population to generate a new new population. These processes are repeated until the stopping condition is met.

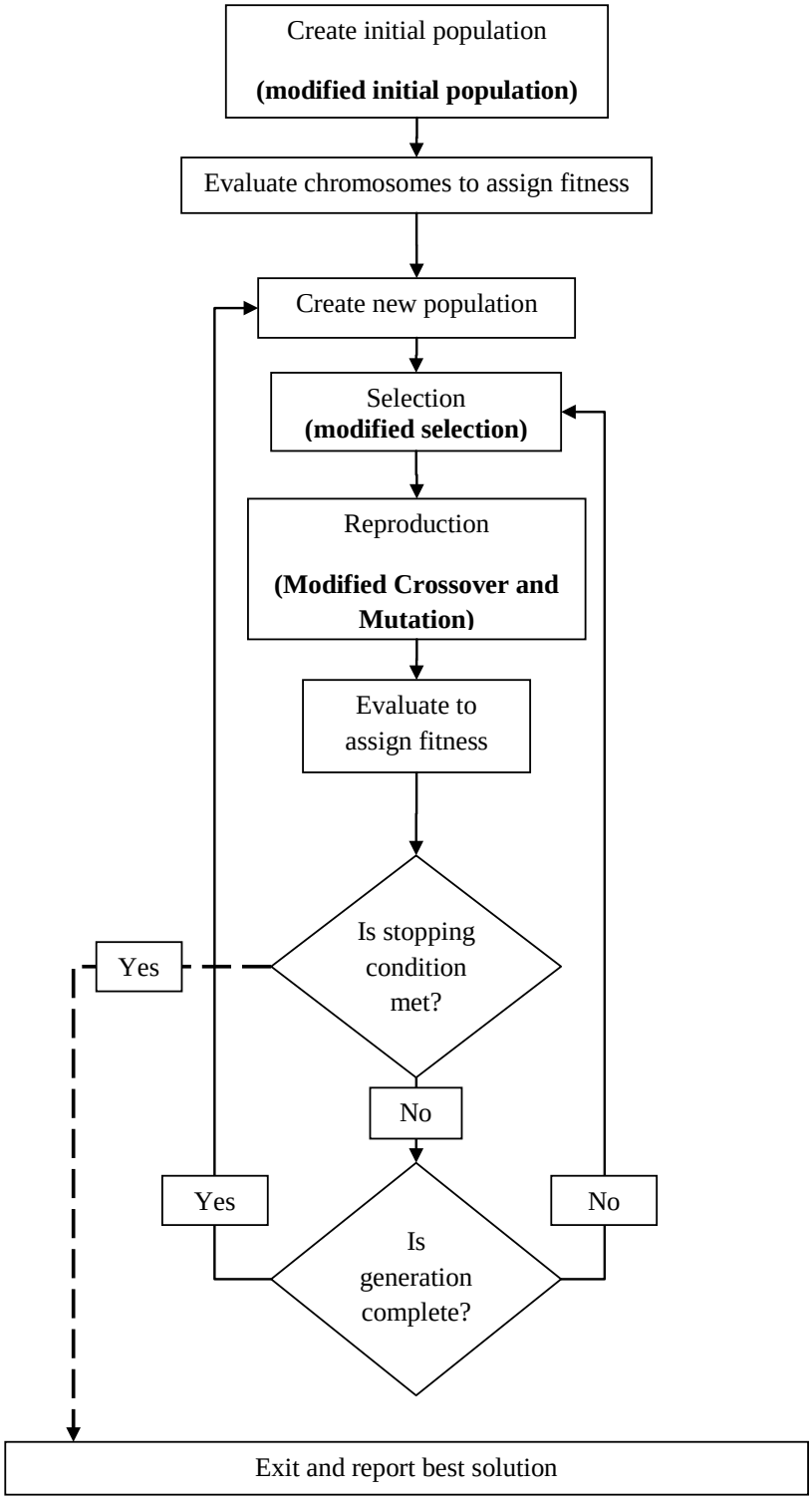

Figure 3.3: GA flow diagram (modified methods are shown in bold font)

3.3.1 GA chromosome encoding and decoding

As mentioned in the previous chapter, chromosome encoding is a critical issue in any GA implementation. Therefore, the DVRP must be represented in a chromosome to which it is possible and easy to apply genetic operations, such as crossover and/or

mutation. As the DVRP is a routing problem, and as it is necessary to have the right order to obtain better solutions, permutation encoding is used. For the DVRP, at any time, there may exist customers that have been served, others that have not yet been served (serviceable) and a remaining set that are not serviceable, but that must be inserted into the system.

In this thesis, the chromosome representation that is used consists of two types of genes: one with a positive integer value which represents a single customer (who has yet to be committed to a vehicle); and one with a negative integer value which represents a vehicle that has a set of customers that have already been served and committed to it (customers with fixed positions). This chromosome representation follows Hansher and Ombuki-Berman (2007)'s approach and an example is shown in Figure 3.4.

3	5	**-1**	7	6	**-2**	8	9	4	1

Figure 3.4: Example of chromosome representation

The chromosome representation encodes both existing vehicles' routes and new customers' orders. For this representation's decoding process, when a negative integer is encountered, it is looked up in a list of committed vehicles (vehicles that already have committed customers) which maps to a list of already committed customers and an existing vehicle (Hanshar and Ombuki-Berman, 2007). The chromosome decoding process creates new vehicles under two conditions:

1) the chromosome representation starts with a positive value; and/or

2) the current vehicle will violate its capacity constraint if a new customer is added.

Here, a chromosome decoding process is illustrated, that is based on the chromosome representation example shown in Figure 3.4. This chromosome has seven new customers' orders, represented by positive integers, and two vehicles, represented by negative integers, which already have a set of customers committed to them. The chromosome decoding process begins by encountering a positive integer, and thus, a new vehicle (Veh_A) is created (*the first condition*). This vehicle adds both customers' orders *3* and *5* to its scheduled route (assuming that this new vehicle (Veh_A) has

sufficient capacity for both) and then a negative integer is encountered (as mentioned before, this negative integer represents an existing vehicle). This means that additions to Veh_A cannot be accepted and Veh_1 (represented by the negative integer) will add customers' orders to its scheduled route if possible. As we assume that Veh_1 (represented by −1) has sufficient capacity, both customers' orders *7* and *6* are added to its scheduled route. Again, a negative integer is encountered and, this time, as we assume that Veh_2 (represented by -2) has capacity for only customers' orders *8*, *9* and *4*, only they are added to its scheduled route. Now, because customer's order *1* would violate the capacity constraints of Veh_2, it is necessary that a new vehicle, Veh_B, is created, which would then accept customer's order *1* as its only order (*the second condition*).

3.3.2 Fitness Evaluation

Each chromosome is evaluated by considering its customers' order in its chromosome representation. Therefore, the chromosome fitness function can be defined as

$$Fitness\ (DVRP)\ =\ \sum_{i=1}^{m} Cost(VR_i). \tag{3.1}$$

where *VR* is the set of routes.

3.3.3 Initial Population

Each time slice requires an initial population for the GA which can be created randomly from the problem customers' permutations or by using a greedy or heuristic technique. In this thesis, both random and heuristic procedures are used to create each time-slice initial population, as in the following subsections.

a) Initial population for the first time slice

20% of the initial population is created on a heuristic basis as follows:

a) create a set of vehicles = $ceiling\left(\frac{Sum\ of\ cutomers'\ demands}{Vehicle\ capacity}\right)$;

b) create a set of random permutations for the static VRP-like instance customers; and

c) insert the created permutations one by one in the best locations in the set of empty created vehicles and ensure that, 90% of insertions do not violate any vehicle's capacity, otherwise ignore the capacity constraint.

The remaining 80% of the initial population is created randomly.

b) Initial population for other time slices

Regarding the initial population for the other time slices, 10 % of the initial population is created in the same manner as the 20% subset for the first time slice in the previous section. Then, another 10% of the initial population is created using the best chromosome from the previous time slice as follows:

a) create a set of random permutations of the current time slice's new customers; and

b) insert the created permutations one by one in the best locations, and once again, 90% of the time consider capacities, otherwise don't.

The remaining 80% of the initial population is created randomly.

3.3.4 Selection

This process determines how candidate individuals in the population are chosen for the reproductive process (crossover and mutation). In this thesis, tournament selection is used with a modification to allow the low quality individuals. This is intended to increase the diversity in order to reach a global optimal solution. The steps involved in normal tournament selection are described in the mutation section in the previous chapter section 2.2.3.1 (Sivanandam and Deepa, 2008), and are shown in Figure 3.5(a). In the modified tournament selection, in step 3, after generating a random value (*r)*, if $r \geq$ selection threshold (ρ), the not-the-fittest chromosome is chosen. This approach is intended to increase the diversity and is illustrated in Figure 3.5(b).

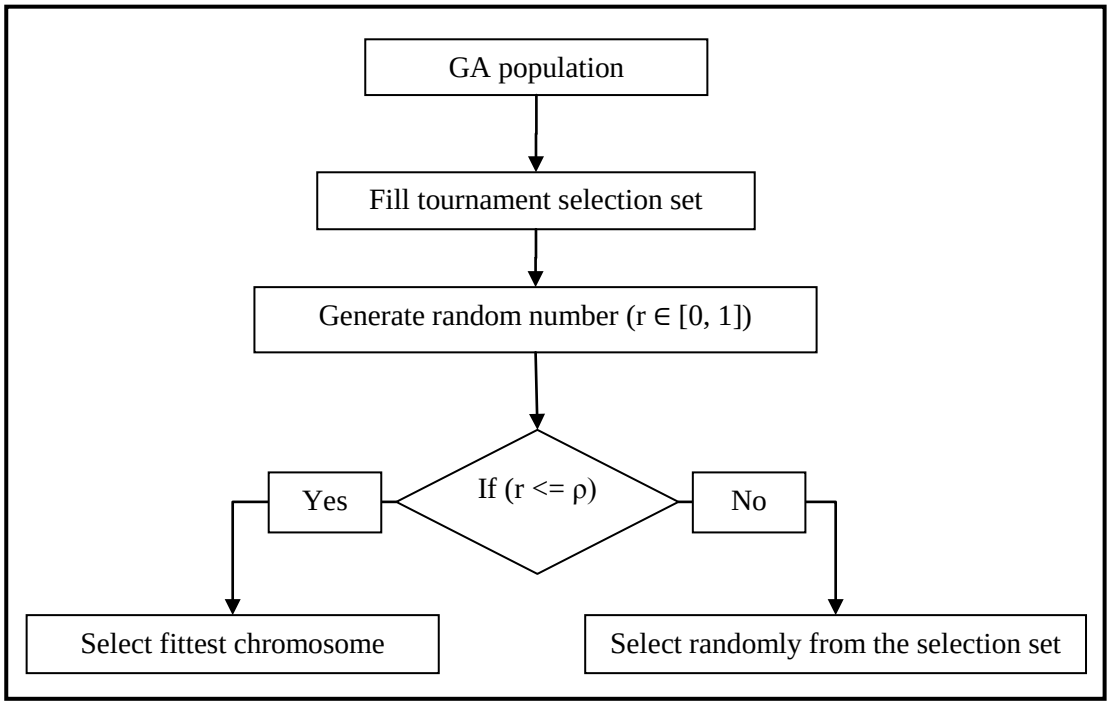

Figure 3.5(a): Original tournament selection

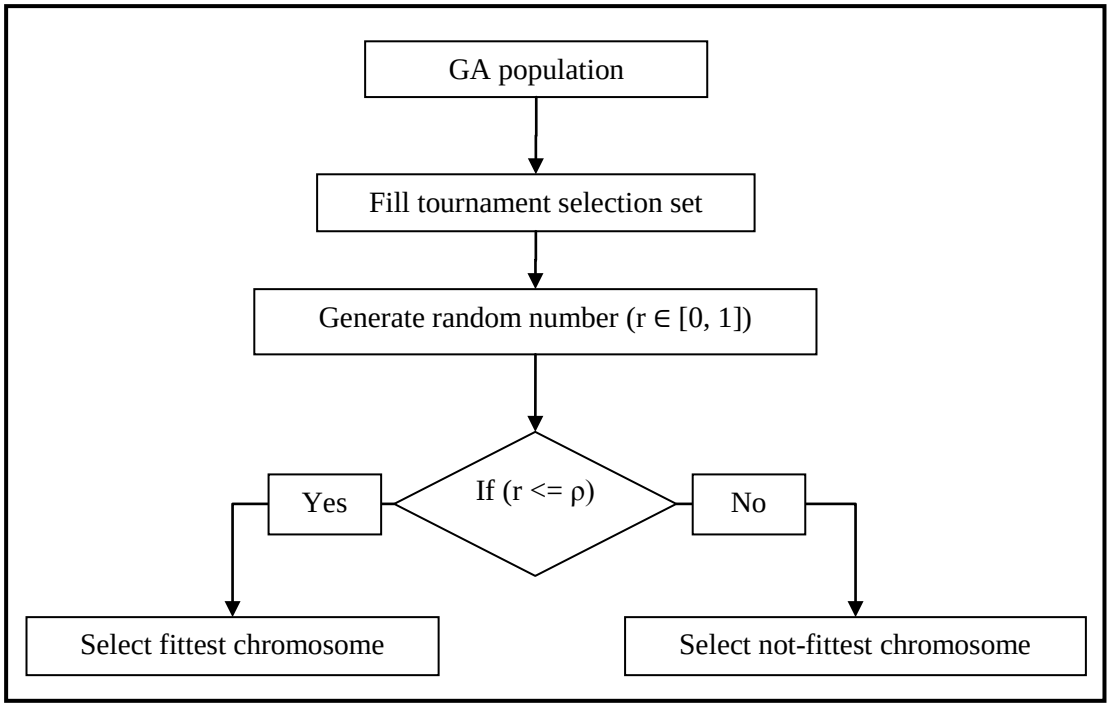

Figure 3.5(b): Modified tournament selection

3.3.5 Crossover

This process determines how the current population chromosomes, parents, are combined to create new chromosomes, children. In this thesis, a modification of the

Best Cost Route Crossover (BCRC), which was firstly developed by Ombuki-Berman et al. (2006) is used as shown in Figure 3.6(a). The steps involved in the normal BCRC are described in the crossover section in the previous chapter, while in the modified BCRC, in step 4, a random number $r \in [0, 1]$ is used. The modified BCRC is illustrated in Figure 3.6(b). The effect of this, is to sometimes generate poorer children, with once again, the aim of increasing diversity.

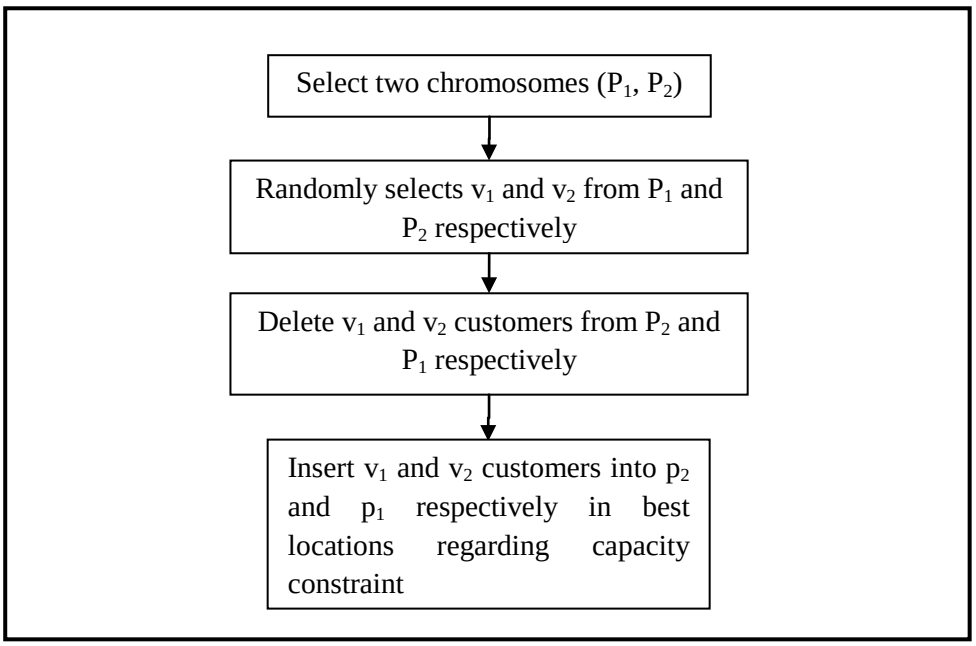

Figure 3.6(a): Original BCRC

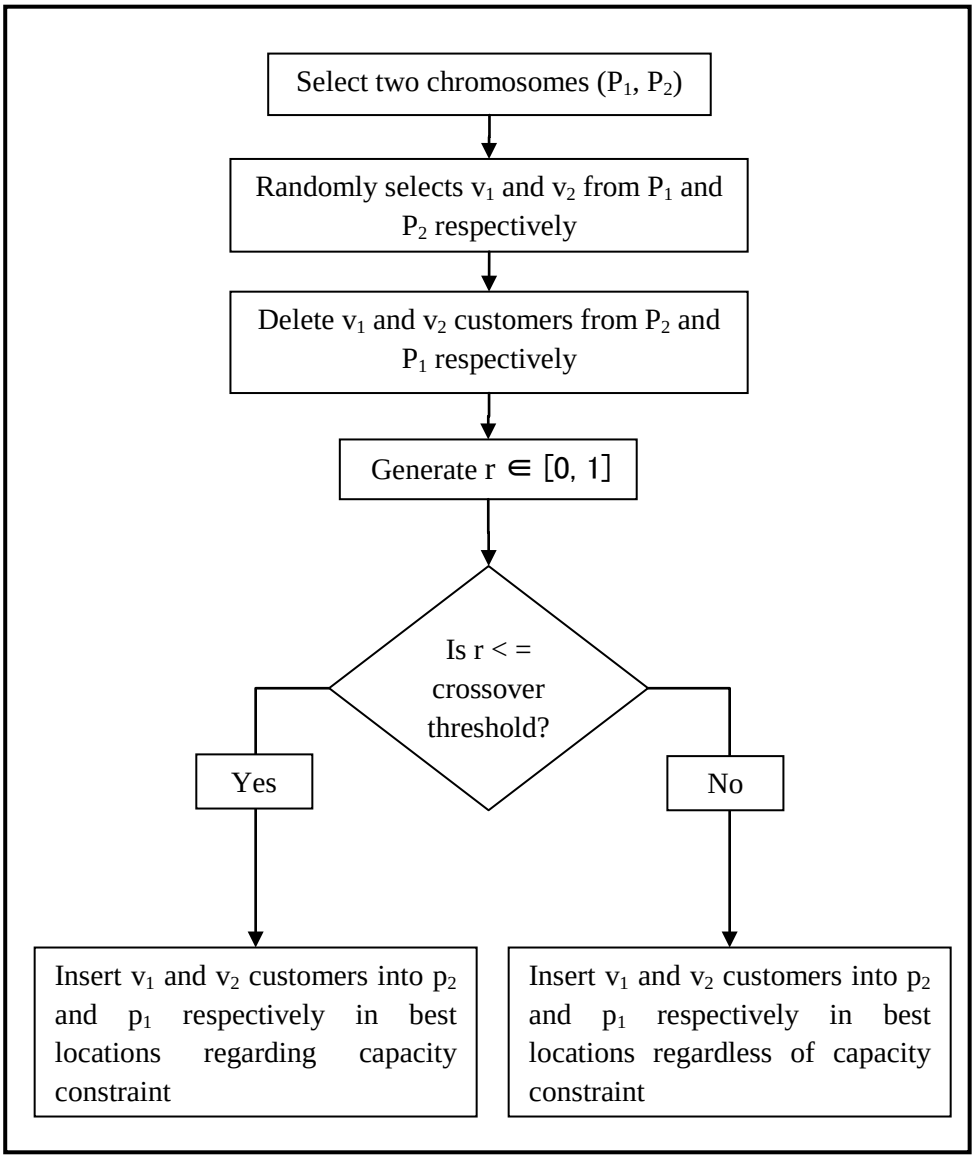

Figure 3.6(b): Modified BCRC

In this approach, the cost of inserting a customer (c) between two consecutive customers (a, b) is calculated as

$$Cost_{c,a,b} = Dist(c,a) + Dist(c,b) - Dist(a,b) \quad (3.2)$$

3.3.6 Mutation

This process determines how children are self modified in this thesis, the following two different mutations are used by the GA.

- Reverse (Inversion) (Sivanandam and Deepa, 2008) selects two cut-points along a chromosome's length and reverses the genes between them.

- Swap (Interchange) (Sivanandam and Deepa, 2008) selects two genes and swaps them. In this thesis, a modified swap is used in order to try to generate more variations and, thus, increase the GA's diversity.
    1. Step 1: creates two different random integers ($r \in$ [0, chromosome length]) and obtains the difference.
    2. Step 2: That difference is then the number of times it randomly swaps two genes, as in the original swap.

In this thesis, an elitist strategy, in which a set of the best solutions is passed over to the next generation(s), is used (Sivanandam and Deepa, 2008; Hanshar and Ombuki-Berman, 2007). It allows elite solutions to spread from one generation to the next, thereby ensuring that the best solution does not suffer over time.

3.3.7 Local Optimal Condition (LOC)

In this thesis, another strategy for enhancing the GA's performance is proposed. In it, when a LOC is detected, the GA tries to escape from it by increasing the crossover threshold, which produces more randomization; this can help the GA to reach a better and globally optimal solution. Figure 3.7 shows how randomization in LOC would help GA to escape from area (A) that contains a local optimal solution, to area (B) that contains the global one.

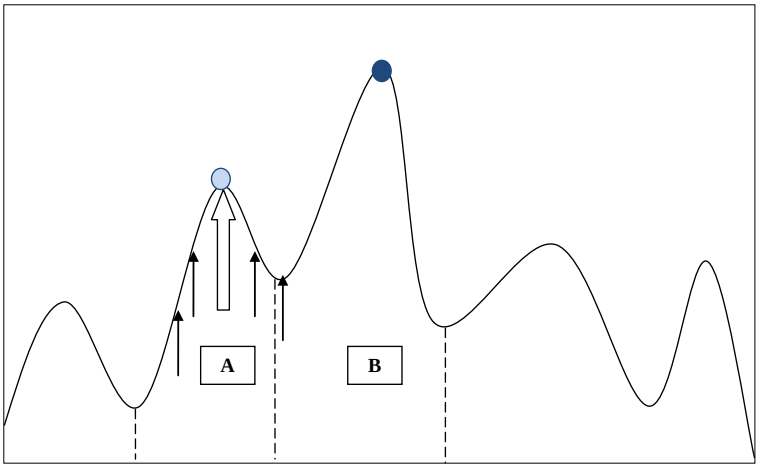

Figure 3.7: How LOC could help GA to escape from local optimal

In implementing this strategy, if the GA subsystem obtains the same best chromosome in ten consecutive generations, it is considered to be stuck in a LOC. If the

system detects this LOC, it decreases the crossover threshold by 90% and, if the crossover threshold becomes less than 0.1, it resets it to its default value.

3.4 Experimental Results and Discussion

This section discusses the results obtained from using the GA described in section 3.3 to solve well-known DVRP benchmark problems (Kilby et al., 1998; Montemanni et al., 2005; Hanshar and Ombuki-Berman, 2007) that are derived from three separate VRP sources, namely, Christophides and Beasley (Christophides and Beasley, 1984) (7 instances), Taillard (13 instances) (Taillard, 1994) and Fisher et al. (2 instances) (Fisher, 1995). The proposed system was coded in Microsoft C++ on a 2.8GHz/4GB Intel Core i7 machine.

3.4.1 GA and DVRP Parameters

The GA subsystem parameter values are shown in Table 3.1. For the time-based runs, for consistency with previous research, the maximum number of generations is set to 1000. However, if the time step does not complete 1000 generations before 30 seconds, the optimization is halted, the best solution found is reported, and the next time step begun.

Table 3.1: Values of the parameters

Parameter	Value
Heuristic initial population	20 %
Random initial population	80 %
Number of time slices (n_{ts})	25
Cutoff time (*Tco*)	(0.5) *T*
Advanced commitment time (T_{ac})	(0.01) *T*
Processing time	30 seconds
Population size	50
Tournament selection pressure (ρ)	0.80
Crossover rate	0.90
Initial crossover threshold	1.0
Mutation rate	0.1
Elitism percentage	2 %

3.4.2 Comparison of Proposed GA-based DVRP (Time-based)

A comparison of the solution quality, in terms of minimizing the travel distances of the proposed GA-based DVRP system, Montemanni et al.'s ACS (Montemanni et al., 2005), Hanshar and Ombuki-Berman's TS, GA (Hanshar and Ombuki-Berman, 2007), Khouadjiaa et al.'s Variable Neighbour Search (VNS) with a local search and Particle Swarm Optimization (PSO) with a local search (Khouadjiaa et al., 2012) was performed. Note that ACS (Montemanni et al., 2005) has been coded in ANSI C, and all the tests have been carried out on a 1.5 GHz with 256 MB Intel Pentium 4 machine, GA and TS (Hanshar and Ombuki-Berman, 2007) have been coded in Java 1.5.0, and run on an 2.8 GHz with 512 MB memory Intel Pentium 4 machine, while PSO (Khouadjiaa et al., 2012) has been run on an Intel Xeon 3 GHz with 2 GB memory and VNS (Khouadjiaa et al., 2012) has been coded in Java 1.5.0, and runs on an Intel Core 2 Quad 2.6 GHz machine with 4 GB memory.

Table 3.2 gives the best and average distances of this comparison, and the numerical results demonstrate that the proposed GA-based DVRP finds 14 out of 21 new best solutions (in this table and the following tables, the bold and shaded entries indicate the best solutions). The best solutions and averages for the proposed GA-based DVRP system are taken over 25 runs, for ACS over 5 runs (Montemanni et al., 2005), for TS and GA over 10 runs (Hanshar and Ombuki-Berman, 2007), and for PSO and VNS over 30 runs (Khouadjiaa et al., 2012).

Table 3.2: Comparison of systems

Problem	Ant system		Tabu		DVRP-GA		DAPSO		VNS		Proposed GA-based DVRP		
	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Std. Dev.
c50	631.30	681.86	603.57	627.90	570.89	**593.42**	575.89	632.38	599.53	653.84	**566.01**	597.34	15.52
c75	1009.36	1042.39	981.51	1013.82	981.57	1013.45	970.45	1031.76	981.64	1040.00	**944.46**	**990.78**	20.08
c100	973.26	1066.16	997.15	1047.60	961.10	**987.59**	988.27	1051.50	1022.92	1087.18	**943.89**	988.15	20.30
c100b	944.23	1023.60	891.42	932.14	881.92	**900.94**	924.32	964.47	**866.71**	942.81	869.41	904.03	14.18
c120	1416.45	1525.15	1331.80	1468.12	1303.59	**1390.58**	**1276.88**	1457.22	1285.21	1469.24	1288.66	1399.40	75.16
c150	1345.73	1455.50	1318.22	1401.06	1348.88	1386.93	1371.08	1470.95	1334.73	1441.37	**1273.50**	**1359.25**	34.65
c199	1771.04	1844.82	1750.09	1783.43	1654.51	1758.51	**1640.40**	1818.55	1679.65	1769.95	1646.36	**1700.54**	27.93
tai75a	1843.08	1945.20	1778.52	1883.47	1782.91	1856.66	1816.07	1935.28	1806.81	1954.25	**1744.78**	**1823.71**	53.17
tai100a	2375.92	2428.38	2208.85	2310.37	2232.71	2295.61	2249.84	2370.58	2250.50	2462.50	**2181.31**	**2290.95**	68.48
tai150a	3644.78	3840.18	3488.02	3598.69	3328.85	3501.83	3400.33	3612.79	3479.44	3680.35	**3280.79**	**3449.32**	73.12
tai75b	1535.43	1704.06	1461.37	1587.72	1464.56	1527.77	1447.39	**1484.73**	1480.70	1560.71	**1441.35**	1546.18	60.28
tai100b	2283.97	2347.90	2219.28	2330.52	2147.70	2215.39	2238.42	2385.54	2169.10	2319.72	**2119.03**	**2212.58**	44.77
tai150b	3166.88	3327.47	3109.23	3215.32	2933.40	3115.39	3013.99	3232.11	2934.86	3089.57	**2885.94**	**3073.58**	100.74
tai75c	1574.98	1653.58	**1406.27**	1527.80	1440.54	**1501.91**	1481.35	1664.40	1621.03	1746.07	1433.73	1502.56	32.68
tai100c	1562.30	1655.91	1515.10	1604.18	1541.28	1622.66	1532.56	1627.32	**1490.58**	**1557.81**	1504.63	1589.76	60.47
tai150c	2811.48	3016.14	2666.28	2913.67	2612.68	**2743.55**	2714.34	2875.93	2674.29	2928.77	**2593.78**	2759.96	108.41
tai75d	1472.35	1529.00	1430.83	1453.56	**1399.83**	**1422.27**	1414.28	1493.47	1446.50	1541.98	1408.48	1434.56	16.87
tai100d	2008.13	2060.72	1881.91	2026.76	1834.60	**1912.43**	1955.06	2123.90	1969.94	2100.38	**1793.64**	1916.03	62.95
tai150d	3058.87	3203.75	2950.83	3111.43	2950.61	3045.16	3025.43	3347.60	2954.64	3147.38	**2911.47**	**3010.34**	68.94
f71	311.18	358.69	280.23	**306.33**	301.79	309.94	**279.52**	312.35	304.32	325.18	288.30	309.49	7.73
f134	15135.51	16083.56	15717.90	16582.04	15528.81	15986.84	15875.00	16645.89	15680.05	16522.18	**14871.40**	**15789.80**	511.28
Sum	50876.23	53794.02	49988.38	52725.93	49202.73	51088.83	50190.87	53538.72	50033.15	53341.24	**47990.91**	**50648.30**	1477.69
Average	2422.68	2561.62	2380.40	2510.76	2342.99	2432.80	2390.04	2549.46	2382.53	2540.06	**2285.28**	**2411.82**	70.37

This thesis uses an evaluation technique, based on Elsayed et al. (2010), that judges systems' performances by assigning them a score for a given test problem. For it, the system which obtains the best solution is assigned a score of '1.0', and the others assigned fractional scores (between 0.0 and 1.0).

In judging system *i* in terms of its solution to test problem *j*, where *J* is a set of test problems, F_{ij} is defined as the actual fitness, and $BF_j = min\ (F_{ij})$ and $WF_j = max\ (F_{ij})$ are defined as the overall best and worst fitness values for test problem *j* respectively, and system *i*'s score for test problem *j* is

$$S_{ij} = \left(1 - \frac{|F_{ij} - BF_j|}{a\ (|BF_j - WF_j|)}\right)^p \tag{3.3}$$

where $a \geq 1$ and $p > 1$. A value of $a > 1$ will differentiate between the worst solutions by having a small positive value for S_{ij}, while a higher value of *p* will place greater emphasis on good solutions, and in this study, *a* is set to 1.1 and *p* to 2, as in (Elsayed et al., 2010).

In a similar way to how scores for averages were calculated, the final score for system *i* can be calculated from Elsayed et al. as

$$S_{ij} = \vartheta \sum_{j=1}^{J} S_{ij}^{best} + (1 - \vartheta) \sum_{j=1}^{J} S_{ij}^{average} \tag{3.4}$$

where FS_{ij} is the final score for system *i* for test problem *j*, S_{ij}^{best} is the score based on the best solutions, $S_{ij}^{average}$ is the score based on the average values and ϑ is a constant $\in$ [0, 1]. A higher value of ϑ (1 or close to 1), will place greater emphasis on the best solutions, which is appropriate when the study is interested in only the best fitness value, while a lower value of ϑ (0 or close to 0) will emphasise average solutions which is appropriate when the study is interested in a number of good alternative solutions (Elsayed et al., 2010). In this study, ϑ = (0, 0.5, and 1.0) are assessed in order to achieve different balances between the best and average results. The overall score (OS_i) for each system (i) can then be calculated by

$$OS_i = \sum_j FS_{ij} \tag{3.5}$$

Table 3.3 shows the previous comparison by applying Elsayed et al. (2010)'s.

Table 3.3: Comparison of systems (time-based) based on Elsayed et al. (2010)

ϑ	Ant System	Tabu	DVRP-GA	DAPSO	VNS	Proposed GA-based DVRP
0	1.35	8.63	12.06	8.72	8.02	**19.52**
0.5	1.40	8.55	14.30	6.79	6.59	**19.45**
1.0	1.46	8.47	16.53	4.86	5.17	**19.37**

In Table 3.3, it is clear that the proposed GA-based DVRP is the best, as it obtains the best scores for all ϑ values.

3.5 Chapter Summary

This chapter introduced and described the GA processes for handling and solving DVRPs. Also, modifications were proposed to improve its solutions by increasing diversity and escaping from local optima. From the preliminary experimental results discussed in this chapter, it is clear that the proposed modifications enhance the GA.

The next chapter examines how the GA's proposed modifications enhance its performance in solving DVRPs, and a new evaluation approach for evaluating DVRP systems is also proposed.

Chapter 4

Advanced Experimental Studies

This chapter demonstrates how the proposed modifications of the Genetic Algorithm (GA) that were described in the previous chapter, enhance its capability to solve DVRPs. Also, t-test is performed to compare the proposed modifications. To date, to evaluate a DVRP system, a time-based evaluation approach based on its specifications and power has been used. Thus a new, fair and non-system-dependent approach for evaluating DVRP systems is needed. To this end, therefore four alternatives, including generations, raw fitness, weighted fitness and distance calculations, are tested in this chapter.

4.1 Experimental Setup

To enhance the GA, modifications of the initial population for the first time slice and those of the other time slices, the GA selection process, the swap mutation and Local Optimal Condition (LOC) detection have been proposed. Therefore, six GA systems, i.e., the original system (without any modification) and the five mentioned above, have been coded and their solutions to DVRP benchmark problems compared (Kilby et al., 1998; Montemanni et al., 2005; Hanshar and Ombuki-Berman, 2007), as follows.

1) The first GA (original) is a GA-based system for which the initial populations of all time slices are randomly created, and uses a regular tournament selection process and reverse mutation. It is thus the same as Hansher and Ombuki-Berman (2007)'s GA.
2) The second GA (first pop) is the same as the first, except that the first time slice's initial population is created with the modification described in the previous chapter (Section 3.3.3 (a)).
3) The third GA (all pop) is the same as the second, except that all initial populations for all time slices, are created by the modification described in the previous chapter (Section 3.3.3 (b)).

4) The fourth GA (selection) is the same as the third, except that the modified selection process described in the previous chapter (Section 3.3.4) is added.
5) The fifth GA (mutation) is the same as the fourth, except that the modified swap mutation described in the previous chapter (Section 3.3.6) is added.
6) The sixth GA (LOC) is the same as the fifth, except that the LOC detection described in the previous chapter (Section 3.3.5) is added.

4.2 Experimental Results

In this section we discuss the results of the experiments.

4.2.1 Comparing "original" with Hanshar and Ombuki-Berman (2007)'s GA

Table 4.1 shows a comparison of the first GA and Hanshar and Ombuki-Berman (2007)'s GA for solving DVRP benchmark problems. In this table and the following tables, the bold shaded entries indicate the best results. The tables show distances best, average and standard deviation (Std. Dev.). The results are based on 25 runs for all systems.

Table 4.1: Comparison of solutions obtained by "original" system and DVRP-GA (Hanshar and Ombuki-Berman, 2007)

Problem	Random all time slices init. pop.			DVRP-GA (Hanshar and Ombuki-Berman, 2007)	
	Best	Average	Std. Dev.	Best	Average
c50	**567.61**	597.37	17.7287	570.89	**593.42**
c75	**964.80**	**1007.46**	25.0884	981.57	1013.45
c100	961.63	1016.70	28.7653	**961.10**	**987.59**
c100b	**878.02**	917.48	21.2956	881.92	**900.94**
c120	1310.99	1462.70	86.1194	**1303.59**	**1390.58**
c150	1451.66	1590.02	68.0869	**1348.88**	**1386.93**
c199	1891.38	2081.22	81.9489	**1654.51**	**1758.51**
tai75a	**1782.37**	1894.26	64.1248	1783	**1856.66**
tai100a	2249.83	2520.29	80.23	**2232.71**	**2295.61**
tai150a	3955.99	4399.98	263.406	**3328.85**	**3501.83**
tai75b	**1443.40**	1584	79.8478	1464.56	**1527.77**
tai100b	**2125.10**	2281.82	91.2819	2147.70	**2215.39**
tai150b	3415.43	3909.89	214.837	**2933.40**	**3115.39**
tai75c	1451.85	1523.94	42.2857	**1440.54**	**1501.91**
tai100c	1570.33	1653.85	57.3417	**1541.28**	**1622.66**
tai150c	2658.85	2951.87	159.028	**2612.68**	**2743.55**
tai75d	1411.02	1452.32	21.7869	**1399.83**	**1422.27**
tai100d	**1831.62**	1977.14	77.1879	1834.60	**1912.43**
tai150d	3072.24	3221.06	108.715	**2950.61**	**3045.16**
f71	**290.62**	**307.76**	6.72513	301.79	309.94
f134	15964.90	16839.50	436.628	**15528.80**	**15986.80**
Sum	51249.64	55190.63	2032.459	**49202.70**	**51088.80**
Average	2440.46	2628.13	96.783763	**2342.99**	**2432.80**

Table 4.2 shows a summary of the results obtained by the first GA-based system and Hanshar and Ombuki-Berman (2007)'s DVRP-GA. It shows the number of times each system is best in the 'Best' and 'Average' columns in Table 4.1, and the calculated percentage differences.

Table 4.2: Summary of comparison of solutions obtained by "original" system and Hanshar and Ombuki-Berman (2007)'s DVRP-GA

System	Best	Average
Random all time slices init. pop.	8	2
DVRP-GA (Hanshar and Ombuki-Berman, 2007)	13	19
% Deviation	-3.99 %	-7.4 %

Although these systems are identical, there is a small difference that could be due to random effects, differing computer power or implementation efficiencies, such as programming languages.

In the remainder of this chapter, comparisons are made between the first GA-based system and the modified systems, to demonstrate how the proposed modifications affect the performance of the GA.

4.2.2 Comparing "original" with "first pop"

Table 4.3 shows a comparison of the distances bests and averages of the first and second systems.

Table 4.3: Comparison of "original" and "first pop"

Problem	Random all time slices init. pop.			Modified init. pop.		
	Best	Average	Std. Dev.	Best	Average	Std. Dev.
c50	567.61	**597.37**	**17.73**	**559.57**	599.78	17.73
c75	**964.80**	1007.46	25.09	966.60	**1004.85**	**23.40**
c100	961.63	1016.70	**28.77**	**953.94**	**998.80**	28.93
c100b	878.02	**917.48**	**21.30**	**863.46**	923.31	45.85
c120	**1310.99**	1462.70	**86.12**	1325.72	**1441.69**	86.56
c150	1451.66	1590.02	68.09	**1425.29**	**1483.51**	**41.89**
c199	**1891.38**	**2081.22**	**81.95**	2017.79	2150.48	95.04
tai75a	1782.37	1894.26	**64.12**	**1760**	**1892.23**	69.16
tai100a	2249.83	2520.29	**80.23**	**2249.22**	**2414.71**	83.82
tai150a	3955.99	4399.98	263.41	**3835.36**	**4372.15**	**260.79**
tai75b	**1443.40**	1584	79.85	1465.58	**1562.59**	**59.94**
tai100b	**2125.10**	2281.82	91.28	2148.29	**2273.74**	**71.04**
tai150b	**3415.43**	3909.89	214.84	3510.27	**3742.42**	**155.62**
tai75c	1451.85	1523.94	42.29	**1446.06**	**1512.30**	**42.04**
tai100c	1570.33	1653.85	**57.34**	**1558.67**	**1634.30**	61.46
tai150c	**2658.85**	**2951.87**	159.03	2682.77	2965.98	**146.00**
tai75d	1411.02	1452.32	21.79	**1405.70**	**1452.24**	**19.85**
tai100d	1831.62	1977.14	77.19	**1808.95**	**1942.74**	**72.01**
tai150d	3072.24	**3221.06**	**108.72**	**3055.13**	3232.48	110.45
f71	**290.62**	307.76	**6.73**	291.90	**307.23**	7.66
f134	15964.90	16839.50	**436.63**	**15504.60**	**16740.10**	524.58
Sum	51249.64	55190.63	2032.46	50834.87	54647.63	**2023.81**
Average	2440.46	2628.13	96.78	2420.71	2602.27	**96.37**

Table 4.4 shows a summary of the comparison of the two systems' solutions which provides counts of the times they obtain the best results in the 'Best' and

'Average' columns in Table 4.3 and the calculated percentage differences between them.

Table 4.4: Summary of comparison of "original" and "first pop"

System	Best	Average	Remarks
Random all time slices init. pop.	8	5	---
Modified init. Pop.	13	16	Better
% Deviation	- 0.81 %	- 0.98 %	---

In summary, Table 4.4 shows that the second system improves the first system solutions by 0.81% and 0.98% in the best and average results respectively; in particular, it finds 13 of 21 new best solutions and 16 of 21 better averages. This slight increase could be because the second system only modifies the initial population for the first time slice.

4.2.3 Comparing "all pop" with The Previous Systems

Table 4.5 shows a comparison of the distances bests and averages for all three systems.

Table 4.5: Comparison of "all pop" with the previous systems

Problem	Random all time slices init. pop.			Modified init. pop.			Modified all init. pop.		
	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.
c50	567.61	**597.37**	**17.73**	559.57	599.78	17.73	**556.79**	609.62	23.84
c75	964.80	1007.46	25.09	966.60	1004.85	23.40	**961.70**	**993.42**	**18.98**
c100	961.63	1016.70	28.77	953.94	998.80	28.93	**937.03**	**988.28**	**22.03**
c100b	878.02	917.48	21.30	**863.46**	923.31	45.85	870.18	**911.06**	**19.17**
c120	1310.99	1462.70	**86.12**	1325.72	**1441.69**	86.56	**1307.50**	1450.77	93.40
c150	1451.66	1590.02	68.09	1425.29	1483.51	41.89	**1320.17**	**1389.72**	**32.40**
c199	1891.38	2081.22	81.95	2017.79	2150.48	95.04	**1694.37**	**1773.86**	**41.14**
tai75a	1782.37	1894.26	64.12	**1760**	1892.23	69.16	1776	**1871.02**	**60.49**
tai100a	2249.83	2520.29	80.23	2249.22	2414.71	83.82	**2208.42**	**2318.47**	**73.19**
tai150a	3955.99	4399.98	263.41	3835.36	4372.15	260.79	**3362.79**	**3491.18**	**113.40**
tai75b	**1443.40**	1584	79.85	1465.58	**1562.59**	59.94	1462.78	1563.85	**58.52**
tai100b	**2125.10**	2281.82	91.28	2148.29	2273.74	71.04	2135.24	**2265.43**	**64.64**
tai150b	3415.43	3909.89	214.84	3510.27	3742.42	155.62	**2996.84**	**3181.82**	**75.62**
tai75c	1451.85	1523.94	42.29	**1446.06**	**1512.30**	42.04	1465.68	1512.75	**31.71**
tai100c	1570.33	1653.85	57.34	1558.67	1634.30	61.46	**1526.55**	**1597.82**	**43.23**
tai150c	**2658.85**	2951.87	159.03	2682.77	2965.98	146.00	2667.79	**2876.28**	**109.10**
tai75d	1411.02	1452.32	21.79	1405.70	1452.24	**19.85**	**1400.38**	**1443.10**	26.79
tai100d	1831.62	1977.14	77.19	1808.95	1942.74	72.01	**1741.92**	**1920.42**	**67.73**
tai150d	3072.24	3221.06	108.72	3055.13	3232.48	110.45	**2984.87**	**3143.87**	**98.58**
f71	**290.62**	307.76	**6.73**	291.90	**307.23**	7.66	291.90	307.89	7.13
f134	15964.90	16839.50	436.63	**15504.60**	16740.10	524.58	15552	**16125.90**	**323.60**
Sum	51249.64	55190.63	2032.46	50834.87	54647.63	2023.81	49220.90	51736.52	**1404.69**
Average	2440.46	2628.13	96.78	2420.71	2602.27	96.37	2343.85	2463.64	**66.89**

Table 4.6 shows a summary of the comparison of the three systems' solutions in Table 4.5.

Table 4.6: Summary of comparison of "all pop" with the previous systems

System	Best	Average	Remarks
Random all time slices init. pop.	4	1	---
Modified init. Pop.	4	4	---
Modified all init. pop.	13	16	Best
% Deviation	-3.17%	-5.33%	---

Table 4.6 shows that the third system improves the second system's solutions by 3.17% and 5.33% for the best and average results respectively and, thus, improves the first system's DVRP solutions by much more than the second system presumably because it modifies the initial populations for all time slices. Also, it finds 13 of 21 new best solutions and 16 of 21 better averages.

4.2.4 Comparing "selection" with The Previous Systems

Table 4.7 shows a comparison of the distances bests and averages for all four systems.

Table 4.7: Comparison of "selection" with the previous systems

Problem	Random all time slices init. pop.			Modified init. pop.			Modified all init. pop.			Modified selection		
	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.
c50	567.61	597.37	17.73	559.57	599.78	17.73	556.79	609.62	23.84	563.51	599.85	14.25
c75	964.80	1007.46	25.09	966.60	1004.85	23.40	961.70	993.42	18.98	957.38	995.45	18.72
c100	961.63	1016.70	28.77	953.94	998.80	28.93	937.03	988.28	22.03	935.50	978.60	21.67
c100b	878.02	917.48	21.30	863.46	923.31	45.85	870.18	911.06	19.17	875.08	908.14	16.54
c120	1310.99	1462.70	86.12	1325.72	1441.69	86.56	1307.50	1450.77	93.40	1278.23	1398.50	57.88
c150	1451.66	1590.02	68.09	1425.29	1483.51	41.89	1320.17	1389.72	32.40	1322.08	1371.06	32.73
c199	1891.38	2081.22	81.95	2017.79	2150.48	95.04	1694.37	1773.86	41.14	1672.02	1739.47	34.48
tai75a	1782.37	1894.26	64.12	1760	1892.23	69.16	1776	1871.02	60.49	1738.47	1889.96	55.92
tai100a	2249.83	2520.29	80.23	2249.22	2414.71	83.82	2208.42	2318.47	73.19	2208.33	2294.05	64.75
tai150a	3955.99	4399.98	263.41	3835.36	4372.15	260.79	3362.79	3491.18	113.40	3348.77	3479.91	71.33
tai75b	1443.40	1584	79.85	1465.58	1562.59	59.94	1462.78	1563.85	58.52	1468.95	1554.67	45.80
tai100b	2125.10	2281.82	91.28	2148.29	2273.74	71.04	2135.24	2265.43	64.64	2147.34	2270.39	65.72
tai150b	3415.43	3909.89	214.84	3510.27	3742.42	155.62	2996.84	3181.82	75.62	2999.81	3163.97	116.52
tai75c	1451.85	1523.94	42.29	1446.06	1512.30	42.04	1465.68	1512.75	31.71	1456.50	1504.46	36.88
tai100c	1570.33	1653.85	57.34	1558.67	1634.30	61.46	1526.55	1597.82	43.23	1526.19	1608.78	55.75
tai150c	2658.85	2951.87	159.03	2682.77	2965.98	146.00	2667.79	2876.28	109.10	2583.57	2892.97	129.87
tai75d	1411.02	1452.32	21.79	1405.70	1452.24	19.85	1400.38	1443.10	26.79	1400.38	1434.42	22.81
tai100d	1831.62	1977.14	77.19	1808.95	1942.74	72.01	1741.92	1920.42	67.73	1789.47	1936.21	78.43
tai150d	3072.24	3221.06	108.72	3055.13	3232.48	110.45	2984.87	3143.87	98.58	2915.69	3091.72	83.73
f71	290.62	307.76	6.73	291.90	307.23	7.66	291.90	307.89	7.13	288.30	309.43	7.68
f134	15964.90	16839.50	436.63	15504.60	16740.10	524.58	15552	16125.90	323.60	15135.80	16269.40	510.05
Sum	51249.64	55190.63	2032.46	50834.87	54647.63	2023.81	49220.90	51736.52	1404.69	48611.37	51691.41	1541.52
Average	2440.46	2628.13	96.78	2420.71	2602.27	96.37	2343.85	2463.64	66.89	2314.83	2461.50	73.41

Table 4.8 shows a summary of the comparison of the four systems' solutions given in Table 4.7.

Table 4.8: Summary of comparison of "selection" with the previous systems

System	Best	Average	Remarks
Random all time slices init. pop.	2	1	---
Modified init. Pop.	2	1	---
Modified all init. pop.	5	7	---
Modified selection	13	12	Best
% Deviation	-1.24%	-0.09%	---

Table 4.8 shows that the fourth system improves the third system's solutions by 1.24% and 0.09% for the best and average results respectively. Although these improvements are not as large as the previous ones, the best improved more than the average; it finds 13 of 21 new best solutions and 12 of 21 better averages.

4.2.5 Comparing "mutation" with The Previous Systems

Table 4.9 shows a comparison of the fifth and the previous four systems, while Table 4.10 presents a summary.

Table 4.9: Comparison of "mutation" with the previous systems

Problem	Random all time slices init. pop.			Modified init. pop.			Modified all init. pop.			Modified selection			Swap mutation		
	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.
c50	567.61	597.37	17.73	559.57	599.78	17.73	**556.79**	609.62	23.84	563.51	599.85	**14.25**	**556.79**	**595.38**	18.94
c75	964.80	1007.46	25.09	966.60	1004.85	23.40	961.70	993.42	18.98	957.38	995.45	**18.72**	**950.36**	**991.66**	19.03
c100	961.63	1016.70	28.77	953.94	998.80	28.93	937.03	988.28	22.03	**935.50**	**978.60**	21.67	953.32	988.72	**17.66**
c100b	878.02	917.48	21.30	863.46	923.31	45.85	870.18	911.06	19.17	875.08	**908.14**	**16.54**	**855.66**	914.54	22.40
c120	1310.99	1462.70	86.12	1325.72	1441.69	86.56	1307.50	1450.77	93.40	**1278.23**	1398.50	**57.88**	1278.90	**1390.17**	76.84
c150	1451.66	1590.02	68.09	1425.29	1483.51	41.89	1320.17	1389.72	32.40	1322.08	**1371.06**	32.73	**1313.68**	1371.60	**30.96**
c199	1891.38	2081.22	81.95	2017.79	2150.48	95.04	1694.37	1773.86	41.14	**1672.02**	1739.47	34.48	1682.12	**1738.49**	**28.69**
tai75a	1782.37	1894.26	64.12	1760	1892.23	69.16	1776	1871.02	60.49	1738.47	1889.96	**55.92**	**1731.69**	**1869.79**	57.89
tai100a	2249.83	2520.29	80.23	2249.22	2414.71	83.82	2208.42	2318.47	73.19	2208.33	**2294.05**	**64.75**	**2184.88**	2312.68	76.78
tai150a	3955.99	4399.98	263.41	3835.36	4372.15	260.79	3362.79	3491.18	113.40	3348.77	**3479.91**	**71.33**	**3347.78**	3480.41	84.99
tai75b	**1443.40**	1584	79.85	1465.58	1562.59	59.94	1462.78	1563.85	58.52	1468.95	**1554.67**	**45.80**	1458.59	1565.77	63.30
tai100b	**2125.10**	2281.82	91.28	2148.29	2273.74	71.04	2135.24	2265.43	**64.64**	2147.34	2270.39	65.72	2166.78	**2253.32**	66.07
tai150b	3415.43	3909.89	214.84	3510.27	3742.42	155.62	2996.84	3181.82	**75.62**	2999.81	3163.97	116.52	**2925.90**	**3123.38**	114.37
tai75c	1451.85	1523.94	42.29	**1446.06**	1512.30	42.04	1465.68	1512.75	**31.71**	1456.50	**1504.46**	36.88	1450.19	1516.66	35.80
tai100c	1570.33	1653.85	57.34	1558.67	1634.30	61.46	1526.55	1597.82	43.23	1526.19	1608.78	55.75	**1525.26**	**1596.98**	**35.72**
tai150c	2658.85	2951.87	159.03	2682.77	2965.98	146.00	2667.79	2876.28	**109.10**	**2583.57**	2892.97	129.87	2624.60	**2822.24**	119.88
tai75d	1411.02	1452.32	21.79	1405.70	1452.24	19.85	**1400.38**	1443.10	26.79	**1400.38**	1434.42	22.81	1409.18	**1431.99**	**19.23**
tai100d	1831.62	1977.14	77.19	1808.95	1942.74	72.01	**1741.92**	**1920.42**	**67.73**	1789.47	1936.21	78.43	1790.11	1947.74	91.05
tai150d	3072.24	3221.06	108.72	3055.13	3232.48	110.45	2984.87	3143.87	98.58	**2915.69**	3091.72	83.73	2952.34	**3078.55**	**71.85**
f71	290.62	307.76	**6.73**	291.90	**307.23**	7.66	291.90	307.89	7.13	**288.30**	309.43	7.68	**288.30**	308.58	7.73
f134	15964.90	16839.50	436.63	15504.60	16740.10	524.58	15552	**16125.90**	**323.60**	15135.80	16269.40	510.05	**14979.70**	16300.40	546.59
Sum	51249.64	55190.63	2032.46	50834.87	54647.63	2023.81	49220.90	51736.52	**1404.69**	48611.37	51691.41	1541.52	48426.12	51599.05	1605.76
Average	2440.46	2628.13	96.78	2420.71	2602.27	96.37	2343.85	2463.64	**66.89**	2314.83	2461.50	73.41	2306.01	2457.10	76.46

Table 4.10: Summary of comparison of "mutation" with the previous systems

System	Best	Average	Remarks
Random all time slices init. pop.	2	0	---
Modified init. Pop.	1	1	---
Modified all init. pop.	3	2	---
Modified selection	7	7	---
Swap mutation	11	11	Best
% Deviation	- 0.38 %	- 0.18 %	---

In summary, Table 4.10 shows that the fifth system only slightly improves the fourth system's solutions, by 0.38% and 0.18% for the best and average results respectively, however it obtains 11 of 21 new best solutions and 11 of 21 better averages.

4.2.6 Comparing "LOC" System with The Previous Systems

Table 4.11 shows the comparison of it and the previous five systems, while Table 4.12 presents a summary of this comparison.

Table 4.11: Comparison of all systems

Problem	Random all time slices init. pop.			Modified init. pop.			Modified all init. pop.			Modified selection			Swap mutation			Proposed GA-based DVRP		
	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.	Best	Average	Std. Dev.
c50	567.61	597.37	17.73	559.57	599.78	17.73	**556.79**	609.62	23.84	563.51	599.85	**14.25**	**556.79**	**595.38**	18.94	566.01	597.34	15.52
c75	964.80	1007.46	25.09	966.60	1004.85	23.40	961.70	993.42	18.98	957.38	995.45	**18.72**	950.36	991.66	19.03	**944.46**	**990.78**	20.08
c100	961.63	1016.70	28.77	953.94	998.80	28.93	937.03	988.28	22.03	**935.50**	**978.60**	21.67	953.32	988.72	**17.66**	943.89	988.15	20.30
c100b	878.02	917.48	21.30	863.46	923.31	45.85	870.18	911.06	19.17	875.08	908.14	16.54	**855.66**	914.54	22.40	869.41	**904.03**	**14.18**
c120	1310.99	1462.70	86.12	1325.72	1441.69	86.56	1307.50	1450.77	93.40	**1278.23**	1398.50	**57.88**	1278.90	**1390.17**	76.84	1288.66	1399.40	75.16
c150	1451.66	1590.02	68.09	1425.29	1483.51	41.89	1320.17	1389.72	32.40	1322.08	1371.06	32.73	1313.68	1371.60	**30.96**	**1273.50**	**1359.25**	34.65
c199	1891.38	2081.22	81.95	2017.79	2150.48	95.04	1694.37	1773.86	41.14	1672.02	1739.47	34.48	1682.12	1738.49	28.69	**1646.36**	**1700.54**	**27.93**
tai75a	1782.37	1894.26	64.12	1760	1892.23	69.16	1776	1871.02	60.49	1738.47	1889.96	55.92	**1731.69**	1869.79	57.89	1744.78	**1823.71**	**53.17**
tai100a	2249.83	2520.29	80.23	2249.22	2414.71	83.82	2208.42	2318.47	73.19	2208.33	2294.05	**64.75**	2184.88	2312.68	76.78	**2181.31**	**2290.95**	68.48
tai150a	3955.99	4399.98	263.41	3835.36	4372.15	260.79	3362.79	3491.18	113.40	3348.77	3479.91	**71.33**	3347.78	3480.41	84.99	**3280.79**	**3449.32**	73.12
tai75b	1443.40	1584	79.85	1465.58	1562.59	59.94	1462.78	1563.85	58.52	1468.95	1554.67	**45.80**	1458.59	1565.77	63.30	**1441.35**	**1546.18**	60.28
tai100b	2125.10	2281.82	91.28	2148.29	2273.74	71.04	2135.24	2265.43	64.64	2147.34	2270.39	65.72	2166.78	2253.32	66.07	**2119.03**	**2212.58**	44.77
tai150b	3415.43	3909.89	214.84	3510.27	3742.42	155.62	2996.84	3181.82	**75.62**	2999.81	3163.97	116.52	2925.90	3123.38	114.37	**2885.94**	**3073.58**	100.74
tai75c	1451.85	1523.94	42.29	1446.06	1512.30	42.04	1465.68	1512.75	**31.71**	1456.50	1504.46	36.88	1450.19	1516.66	35.80	**1433.73**	**1502.56**	32.68
tai100c	1570.33	1653.85	57.34	1558.67	1634.30	61.46	1526.55	1597.82	43.23	1526.19	1608.78	55.75	1525.26	1596.98	**35.72**	**1504.63**	**1589.76**	60.47
tai150c	2658.85	2951.87	159.03	2682.77	2965.98	146.00	2667.79	2876.28	109.10	**2583.57**	2892.97	129.87	2624.60	2822.24	119.88	2593.78	**2759.96**	**108.41**
tai75d	1411.02	1452.32	21.79	1405.70	1452.24	19.85	**1400.38**	1443.10	26.79	**1400.38**	1434.42	22.81	1409.18	**1431.99**	19.23	1408.48	1434.56	**16.87**
tai100d	1831.62	1977.14	77.19	1808.95	1942.74	72.01	**1741.92**	1920.42	67.73	1789.47	1936.21	78.43	1790.11	1947.74	91.05	1793.64	**1916.03**	**62.95**
tai150d	3072.24	3221.06	108.72	3055.13	3232.48	110.45	2984.87	3143.87	98.58	2915.69	3091.72	83.73	2952.34	3078.55	71.85	**2911.47**	**3010.34**	**68.94**
f71	290.62	307.76	**6.73**	291.90	**307.23**	7.66	291.90	307.89	7.13	**288.30**	309.43	7.68	**288.30**	308.58	7.73	**288.30**	309.49	7.73
f134	15964.90	16839.50	436.63	15504.60	16740.10	524.58	15552	16125.90	**323.60**	15135.80	16269.40	510.05	14979.70	16300.40	546.59	**14871.40**	**15789.80**	511.28
Sum	51249.64	55190.63	2032.46	50834.87	54647.63	2023.81	49220.90	51736.52	**1404.69**	48611.37	51691.41	1541.52	48426.12	51599.05	1605.76	**47990.91**	**50648.30**	1477.69
Average	2440.46	2628.13	96.78	2420.71	2602.27	96.37	2343.85	2463.64	**66.89**	2314.83	2461.50	73.41	2306.01	2457.10	76.46	**2285.28**	**2411.82**	70.37

Table 4.12: Summary of comparison of all systems

System	Best	Average	Remarks
Random all time slices init. pop.	0	0	---
Modified init. Pop.	0	1	---
Modified all init. pop.	3	0	---
Modified selection	5	1	---
Swap mutation	4	3	---
Proposed GA-based DVRP	13	16	Best
% Deviation	- 0.90 %	- 1.84 %	---

Table 4.12 shows that the sixth system improves the fifth system's solutions by 0.90% and 1.84% for the best and average results respectively. Therefore, the sixth system improves the previous system more in terms of solution averages. It finds 13 of 21 new best and 16 of 21 better average solutions. It is also notable that the original system (the first system with no modifications) no longer has any best solution in the best or average results.

From the previous results, the third system, in which the initial populations for all the time slices are modified, achieves the largest improvement in solving the DVRP benchmark problems, followed by the system that modifies the selection process in the 'Best' results and the final system that uses LOC detection in the 'Average' results. This could be interpreted as being, because the selection process drives the local search more, while the LOC strategy more effectively aids exploration.

In more detail, for several problems, the third system has the greatest effect, especially for solving large uniform ones, such as c150 and c199, and large distributed cluster ones, such as tai150a and tai150b. Therefore, creating an initial population using both random and heuristic procedures enhances GA.

To confirm the above observations, Table 4.13 shows an overall comparison of these systems using the metric of Elsayed et al. (2010).

Table 4.13: Comparison of systems (time-based) based Elsayed et al. (2010)

δ	Rand all time slices init. pop.	Modified init. pop.	Modified all init. pop.	Modified selection	Swap mutation	Proposed GA-based DVRP
0	1.73	3.91	10.04	12.63	12.29	**18.66**
1	2.64	2.98	9.30	10.46	12.93	**16.13**
0.5	2.18	3.44	9.67	11.54	12.61	**17.40**

In Table 4.13, it is clear that the proposed GA-based DVRP performs best, as it obtains the best scores for all ϑ values. However the third system (that modifies the initial population for all time slices) achieves the greatest improvement in the DVRP solutions.

4.2.7 Statistical Testing

In order to be able to compare our results accurately, we have also performed statistical significance tests. Two-sided t-tests have been is performed with a confidence level of 95%. In the next two Tables (4.14 and 4.15), a cell is marked with "+" sign if a system is significantly better "-" if it is significantly worse, and is blank if there is no significant difference.

Table 4.14: t-test significance results between each system and its previous system

Problem	Sys. 1-Sys. 2	Sys. 2-Sys. 3	Sys. 3-Sys. 4	Sys. 4-Sys. 5	Sys. 5-Sys. 6
c50					
c75					
c100	+				
c100b					
c120			+		
c150	+	+	+		
c199	-	+	+		+
tai75a					+
tai100a	+	+			
tai150a		+			
tai75b					
tai100b					+
tai150b	+	+			
tai75c					
tai100c		+			
tai150c		+			
tai75d					
tai100d					
tai150d		+	+		+
f71					
f134		+			+
Sig. Diff.	+3	+9	+4	0	+5

Table 4.14 shows t-test results for comparing each system with its previous one. Here "Sig. Diff." shows the significance difference for each system in this comparison. The third system, in which the initial populations for all the time slices are modified, obtains the greater number of significance, while the fifth system does not provide any significance in this comparison.

Table 4.15: t-test significance results between each system and the first system

Problem	Sys. 1-Sys. 2	Sys. 1-Sys. 3	Sys. 1-Sys. 4	Sys.1 -Sys. 5	Sys. 1-Sys. 6
c50		-			
c75		+		+	+
c100	+	+	+	+	+
c100b					+
c120			+	+	+
c150	+	+	+	+	+
c199	-	+	+	+	+
tai75a					+
tai100a	+	+	+	+	+
tai150a		+	+	+	+
tai75b					
tai100b					+
tai150b	+	+	+	+	+
tai75c					
tai100c		+	+	+	+
tai150c				+	+
tai75d			+	+	+
tai100d		+			+
tai150d		+	+	+	+
f71					
f134		+	+	+	+
Sig. Count	3	10	11	13	17
Increase of Sig.	+3	+7	+1	+2	+4

Table 4.15 shows t-test results for comparing each system with the first one. Here “Increase of Sig.” shows how each system differs from its previous system in its comparison with the first system. The third system shows the biggest improvement over its previous system. While the sixth system (the final proposed system) obtains the greater number of significance, and the third system gets the lowest value in this comparison. It is clear that the final system improves the original GA. This is because of its mechanisms to improve the GA diversity and to escape from the local optimal solutions.

4.3 Proposed Evaluation Approach

In this research, a new evaluation approach is proposed for DVRP systems. To date, a time-based evaluation approach, in which all DVRP systems are run for a specific amount of time, such as 1 minute (Montemanni et al., 2005) or 30 seconds (Hanshar and Ombuki-Berman, 2007; Runka, 2008) for each time slice, has been used to evaluate DVRP systems. However, a time-based evaluation approach is biased as it is based on

its running system's specifications and power, whereas because the proposed evaluation approach does not, it should be fairer than the time-based evaluation approach.

4.3.1 Experiments for Proposed Evaluation Approach

As mentioned before, the new evaluation approach should be fair, as it should not depend on the running system's specifications and power. With this in mind, the following four evaluation approaches are tested.

a) Generations This approach is used in many GA implementations as a stopping condition. It counts the number of generations in each time slice, regardless of the problem's size. For its implementation, the average number of generations required to solve each time-slice problem is calculated by using a variable called *generationsSum*, that is, the total number of generations in each time slice.

b) Raw Fitness Evaluation This is the main process for evaluating the proposed solution presented by each chromosome. It counts the number of fitness evaluations in each time slice by using a variable called *rawFitnessEvalSum*, that is, the total number of fitness evaluations in each time slice, which is increased by 1 when ever chromosome fitness is evaluated.

c) Weighted Fitness Evaluation This approach counts both the number of complete fitness evaluations and a weighted fitness due to any partial fitness evaluation By using a variable called *weightedFitnessEvalSum*, that is, the total number of fitness evaluations in each time slice, which is increased by 1 when a complete chromosome fitness is evaluated. However, when a distance is calculated once in the mutation and/or crossover process, *weightedFitnessEvalSum* is increased by a fraction equal to $(\frac{1}{Fitness\ evaluation\ calculations})$. More specifically, if a chromosome of a vehicle with 2 customers, 3 distance calculations (number of customers + number of vehicles) are needed to evaluate its complete fitness, i.e., the depot to *c1*, *c1* to *c2* and *c2* to the depot (*Fitness evaluation calculations*) are equal to 3, while if a customer is tested once to be inserted in a position (i.e., in the crossover process), a fraction equal to $\frac{1}{3}$ is added to *weightedFitnessEvalSum*.

d) Distance Calculation This is a component of evaluating the proposed solutions in each chromosome. It counts the number of distance calculations performed as this reflects the problem size in each time slice; for example, if a chromosome of

a vehicle has 2 customers, 3 distance calculations are needed to assess its fitness, i.e., the depot to *c1*, *c1* to *c2*, *c2* to the depot. For implementation, the average number of distance calculations is calculated to solve each time-slice problem using a variable called *distanceCalcSum*, that is, the total number of distance calculations in each time slice.

These variants are assessed using the GA parameters values shown in Table 4.16. The DVRP model is the same as that in the previous chapter, where n_{ts} is set to 25, T_{co} to 0.5 of a working day and T_{ac} to 0.01 of the working day (Hanshar and Ombuki-Berman, 2007). The GA systems are run over 30 seconds as the time-slice processing time and, after 30 seconds, the optimization is halted and the averages of the generations, raw fitness evaluations, weighted fitness evaluations and distance calculations performed, as well as the averages for each evaluation approach are reported.

Table 4.16: GA parameters values

Parameter	Value
Processing time	30 seconds
Population size	50
Tournament selection pressure (ρ)	0.80
Crossover rate	0.90
Initial crossover threshold	1.0
Mutation rate	0.1
Elitism percentage	2 %

Table 4.17 shows the minimum, maximum, average and standard deviations for the averages of each evaluation approach with regard to the DVRP benchmark problems, after 30 seconds.

Table 4.17: Averages after 30 seconds for four proposed evaluation approaches

Problem	Generations				Raw Fitness				Weighted Fitness				Distance Calculations			
	Min	Max	Avg.	Std. Dev.	Min	Max	Avg.	Std. Dev.	Min	Max	Avg.	Std. Dev.	Min	Max	Avg.	Std. Dev.
c50	141.00	169.00	153.28	9.80	25438.90	31279.00	27492.72	1629.44	60811.90	66432.20	62962.90	1144.29	2164580.00	2353150.00	2257074.40	48435.18
c75	100.00	131.00	109.84	6.53	14389.20	16132.50	15212.51	505.93	30656.10	35787.50	32278.14	970.95	1505740.00	1853160.00	1599578.40	72291.25
c100	112.00	143.00	126.16	7.94	10930.30	16456.30	13966.68	1188.43	41617.90	45823.70	43597.99	998.01	2652920.00	3188160.00	2903769.60	120222.83
c100b	101.00	113.00	106.36	3.21	14342.70	16255.20	15359.90	424.85	38490.60	41475.40	40096.80	795.60	2071800.00	2294740.00	2186225.60	60615.90
c120	89.00	106.00	96.96	4.78	14650.40	20505.70	17919.11	2227.32	18385.10	48393.10	45260.23	5697.08	2217140.00	2781610.00	2470076.80	183391.45
c150	76.00	98.00	86.80	6.08	8327.77	11057.80	9977.87	740.63	24231.40	29136.50	26440.69	1241.24	2049680.00	2445830.00	2212200.80	125982.77
c199	58.00	73.00	64.24	3.26	7126.58	8031.08	7548.62	233.07	16617.70	18475.80	17369.29	457.05	1653020.00	1881880.00	1769890.80	56982.49
tai75a	102.00	112.00	107.84	2.91	12809.20	15733.00	14135.16	677.09	13893.20	38622.50	36171.39	4749.29	1715230.00	1988100.00	1832737.20	76190.58
tai100a	80.00	111.00	91.40	6.92	9885.48	13151.80	11504.57	967.29	32487.50	36360.50	33918.09	1134.01	1985330.00	2453430.00	2176309.60	130463.25
tai150a	63.00	74.00	68.68	3.06	7582.80	9025.96	8335.26	424.06	21984.80	25752.00	24345.76	876.68	2119420.00	2480230.00	2269838.40	104188.43
tai75b	102.00	113.00	107.24	2.93	15472.90	17066.70	16447.80	438.44	40241.80	45017.00	43007.41	1213.44	1779290.00	2020120.00	1882880.00	70607.14
tai100b	98.00	126.00	109.80	7.85	9172.60	12015.50	10575.78	880.08	31655.40	39150.60	34862.51	1785.34	2102940.00	2683890.00	2385660.80	158172.90
tai150b	57.00	78.00	69.56	3.92	7873.00	10349.90	9164.96	582.76	22643.10	27055.30	24865.87	1105.28	1945150.00	2381780.00	2140257.60	102844.98
tai75c	119.00	137.00	128.32	5.07	11284.80	17119.80	13993.99	1135.42	45776.40	50350.90	47435.06	1261.45	2172570.00	2830320.00	2437500.40	130992.78
tai100c	77.00	97.00	86.48	4.66	12276.30	15228.20	13720.04	716.79	34594.90	38278.60	36573.68	1168.76	1735260.00	2107540.00	1978185.20	97371.13
tai150c	57.00	80.00	67.72	67.72	7770.40	9951.44	8724.82	549.37	23474.40	27589.70	25289.02	1152.91	1825970.00	2354790.00	2097342.40	150378.39
tai75d	115.00	134.00	125.92	4.40	14928.80	17930.40	16387.48	903.74	41778.80	47819.60	45498.88	1436.52	1929390.00	2221670.00	2080570.40	89157.78
tai100d	85.00	109.00	98.96	5.98	8468.08	11841.80	9788.79	1087.58	32693.40	39766.60	36106.28	1953.00	2158830.00	2944080.00	2586091.60	237560.10
tai150d	68.00	92.00	77.40	5.80	7096.40	8918.16	7850.22	503.53	24190.80	28765.40	25898.74	1066.44	2094810.00	2567330.00	2334870.80	146415.29
f71	119.00	152.00	135.52	9.66	12671.60	18238.70	15839.47	1716.82	71280.80	83255.60	76865.32	3787.76	3677020.00	4492500.00	4058440.00	236609.31
f134	52.00	67.00	58.84	3.72	26417.30	28921.50	27682.05	580.67	54576.60	59867.30	57429.30	1397.89	1640400.00	1891820.00	1753380.40	65429.65
Sum	1871.00	2315.00	2077.32	176.21	258915.51	325210.44	291627.77	18113.31	722082.60	873175.80	816273.35	35392.99	43196490.00	52216130.00	47412881.20	2464303.58
Average	89.10	110.24	98.92	8.39	12329.31	15486.21	13887.04	862.54	34384.89	41579.80	38870.16	1685.38	2056975.71	2486482.38	2257756.25	117347.79

For each proposed evaluation approach, the required time for running the system, that is, 30 seconds of the average score of each evaluation approach, was calculated by multiplying each run's generations, raw fitness, weighted fitness and distance calculation values by $\frac{30}{the\ average}$. For example, each value in Generations was divided by 98.92. Table 4.18 shows the proposed four evaluation approaches using this method.

Table 4.18: Comparison of averages of four proposed evaluation approaches' running times

Problem	Generations				Raw Fitness				Weighted Fitness				Distance Calculations			
	Min	Max	Avg.	Std. Dev.	Min	Max	Avg.	Std. Dev.	Min	Max	Avg.	Std. Dev.	Min	Max	Avg.	Std. Dev.
c50	27.60	33.08	30.00	1.92	27.76	34.13	30.00	1.78	28.98	31.65	30.00	**0.55**	28.77	31.28	30.00	0.64
c75	27.31	35.78	30.00	1.78	28.38	31.81	30.00	1.00	28.49	33.26	30.00	**0.90**	28.24	34.76	30.00	1.36
c100	26.63	34.00	30.00	1.89	23.48	35.35	30.00	2.55	28.64	31.53	30.00	**0.69**	27.41	32.94	30.00	1.24
c100b	28.49	31.87	30.00	0.91	28.01	31.75	30.00	0.83	28.80	31.03	30.00	**0.60**	28.43	31.49	30.00	0.83
c120	27.54	32.80	30.00	**1.48**	24.53	34.33	30.00	3.73	12.19	32.08	30.00	3.78	26.93	33.78	30.00	2.23
c150	26.27	33.87	30.00	2.10	25.04	33.25	30.00	2.23	27.49	33.06	30.00	**1.41**	27.80	33.17	30.00	1.71
c199	27.09	34.09	30.00	1.52	28.32	31.92	30.00	0.93	28.70	31.91	30.00	**0.79**	28.02	31.90	30.00	0.97
tai75a	28.38	31.16	30.00	**0.81**	27.19	33.39	30.00	1.44	11.52	32.03	30.00	3.94	28.08	32.54	30.00	1.25
tai100a	26.26	36.43	30.00	2.27	25.78	34.30	30.00	2.52	28.73	32.16	30.00	**1.00**	27.37	33.82	30.00	1.80
tai150a	27.52	32.32	30.00	1.34	27.29	32.49	30.00	1.53	27.09	31.73	30.00	**1.08**	28.01	32.78	30.00	1.38
tai75b	28.53	31.61	30.00	0.82	28.22	31.13	30.00	**0.80**	28.07	31.40	30.00	0.85	28.35	32.19	30.00	1.12
tai100b	26.78	34.43	30.00	2.15	26.02	34.08	30.00	2.50	27.24	33.69	30.00	**1.54**	26.44	33.75	30.00	1.99
tai150b	24.58	33.64	30.00	1.69	25.77	33.88	30.00	1.91	27.32	32.64	30.00	**1.33**	27.27	33.39	30.00	1.44
tai75c	27.82	32.03	30.00	1.19	24.19	36.70	30.00	2.43	28.95	31.84	30.00	**0.80**	26.74	34.83	30.00	1.61
tai100c	26.71	33.65	30.00	1.62	26.84	33.30	30.00	1.57	28.38	31.40	30.00	**0.96**	26.32	31.96	30.00	1.48
tai150c	25.25	35.44	30.00	2.17	26.72	34.22	30.00	1.89	27.85	32.73	30.00	**1.37**	26.12	33.68	30.00	2.15
tai75d	27.40	31.93	30.00	1.05	27.33	32.82	30.00	1.65	27.55	31.53	30.00	**0.95**	27.82	32.03	30.00	1.29
tai100d	25.77	33.04	30.00	1.81	25.95	36.29	30.00	3.33	27.16	33.04	30.00	**1.62**	25.04	34.15	30.00	2.76
tai150d	26.36	35.66	30.00	2.25	27.12	34.08	30.00	1.92	28.02	33.32	30.00	**1.24**	26.92	32.99	30.00	1.88
f71	26.34	33.65	30.00	2.14	24.00	34.54	30.00	3.25	27.82	32.49	30.00	**1.48**	27.18	33.21	30.00	1.75
f134	26.51	34.16	30.00	1.89	28.63	31.34	30.00	**0.63**	28.51	31.27	30.00	0.73	28.07	32.37	30.00	1.12
Sum	565.13	704.64	630.00	34.79	556.57	705.10	630.00	40.41	557.50	675.81	630.00	**27.58**	575.31	693.01	630.00	31.98
Average	26.91	33.55	30.00	1.66	26.50	33.58	30.00	1.92	26.55	32.18	30.00	**1.31**	27.40	33.00	30.00	1.52

In Table 4.18, the standard deviations for the generations, raw fitness evaluations, weighted fitness evaluations and distance calculations are 1.66, 1.92, 1.31 and 1.52 respectively. Therefore, as the weighted fitness evaluation approach has the lowest standard deviation, it is the most stable evaluation method regardless of the running system's specifications and power.

As the average number of weighted fitness evaluations obtained in 30 seconds (38870.16) can be rounded up to the nearest 10,000, i.e., 40,000, it is proposed that DVRPs should be tested on the assumption that 40,000 weighted fitness evaluations (WFE) are calculated in each time slice. Therefore, the total number of WFEs for each problem is 40,000 (WFEs in each time slice) * 25 (time slices) = 1,000,000 WFEs.

4.3.2 Comparison of Proposed GA-based DVRP (WFE-based)

A comparison of solution quality, in terms of minimizing travel distances of the proposed GA-based DVRP system by a 40,000 WFE budget, with Montemanni et al. (2005)'s Ant Colony System (ACS), Hanshar and Ombuki-Berman (2007)'s Tabu Search (TS) and GA, Khouadjiaa et al. (2012)'s Variable Neighbour Search (VNS) with a local search and Particle Swarm Optimization (PSO) with a local search was conducted.

Table 4.19 shows the best and average distances of the proposed GA-based DVRP system (WFE-based), where the bold shaded entries indicate the best solutions. Table 4.19 shows the numerical results from this comparison; the proposed GA-based DVRP finds 17 of 21 new best solutions and 16 of 21 better averages. These bests and averages are taken over 25 runs, while those of the ACS are over 5 runs (Montemanni et al., 2005), of the TS and GA are over 10 runs (Hanshar and Ombuki-Berman, 2007), and of the PSO and VNS are over 30 runs (Khouadjiaa et al., 2012).

Table 4.19: Comparison of systems (WFE-based)

Problem	Ant system		Tabu		DVRP-GA		DAPSO		VNS		Proposed GA-based DVRP		
	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Std. Dev.
c50	631.30	681.86	603.57	627.90	570.89	593.42	575.89	632.38	599.53	653.84	**563.507**	**588.88**	15.5782
c75	1009.36	1042.39	981.51	1013.82	981.57	1013.45	970.45	1031.76	981.64	1040.00	**925.286**	**986.11**	22.5295
c100	973.26	1066.16	997.15	1047.60	961.10	987.59	988.27	1051.50	1022.92	1087.18	**931.409**	**972.131**	25.1234
c100b	944.23	1023.60	891.42	932.14	881.92	900.94	924.32	964.47	**866.71**	942.81	870.492	**900.918**	14.7072
c120	1416.45	1525.15	1331.80	1468.12	1303.59	1390.58	1276.88	1457.22	1285.21	1469.24	**1258.82**	**1378.73**	65.4967
c150	1345.73	1455.50	1318.22	1401.06	1348.88	1386.93	1371.08	1470.95	1334.73	1441.37	**1275.73**	**1351.3**	34.3182
c199	1771.04	1844.82	1750.09	1783.43	1654.51	1758.51	1640.40	1818.55	1679.65	1769.95	**1588.41**	**1671.76**	31.2174
tai75a	1843.08	1945.20	1778.52	1883.47	1782.91	1856.66	1816.07	1935.28	1806.81	1954.25	**1757.29**	**1848.66**	50.2529
tai100a	2375.92	2428.38	2208.85	2310.37	2232.71	2295.61	2249.84	2370.58	2250.50	2462.50	**2199.29**	**2264.76**	52.6265
tai150a	3644.78	3840.18	3488.02	3598.69	3328.85	3501.83	3400.33	3612.79	3479.44	3680.35	**3276.96**	**3411.33**	64.7037
tai75b	1535.43	1704.06	1461.37	1587.72	1464.56	1527.77	1447.39	**1484.73**	1480.70	1560.71	**1444.82**	1557.6	56.018
tai100b	2283.97	2347.90	2219.28	2330.52	2147.70	2215.39	2238.42	2385.54	2169.10	2319.72	**2087.34**	**2210.52**	62.2738
tai150b	3166.88	3327.47	3109.23	3215.32	2933.40	3115.39	3013.99	3232.11	2934.86	3089.57	**2847.78**	**3054.15**	88.7472
tai75c	1574.98	1653.58	**1406.27**	1527.80	1440.54	1501.91	1481.35	1664.40	1621.03	1746.07	1440.54	**1501.11**	40.7952
tai100c	1562.30	1655.91	1515.10	1604.18	1541.28	1622.66	1532.56	1627.32	1490.58	**1557.81**	**1490**	1571.8	49.869
tai150c	2811.48	3016.14	2666.28	2913.67	2612.68	**2743.55**	2714.34	2875.93	2674.29	2928.77	**2536.86**	2828.99	140.454
tai75d	1472.35	1529.00	1430.83	1453.56	**1399.83**	**1422.27**	1414.28	1493.47	1446.50	1541.98	1401.33	1426.54	16.1336
tai100d	2008.13	2060.72	1881.91	2026.76	1834.60	1912.43	1955.06	2123.90	1969.94	2100.38	**1777.72**	**1888.48**	73.3808
tai150d	3058.87	3203.75	2950.83	3111.43	2950.61	3045.16	3025.43	3347.60	2954.64	3147.38	**2895.01**	**2975.73**	51.4412
f71	311.18	358.69	280.23	**306.33**	301.79	309.94	**279.52**	312.35	304.32	325.18	288.298	309.729	8.24477
f134	15135.51	16083.56	15717.90	16582.04	15528.81	15986.84	15875.00	16645.89	15680.05	16522.18	**14400.8**	**15751.2**	569.87
Sum	50876.23	53794.02	49988.38	52725.93	49202.73	51088.83	50190.87	53538.72	50033.15	53341.24	**47257.69**	**50450.43**	1533.78
Average	2422.68	2561.62	2380.40	2510.76	2342.99	2432.80	2390.04	2549.46	2382.53	2540.06	**2250.37**	**2402.40**	73.04

Table 4.20 shows a comparison of the systems based on the previous metric of Elsayed et al. (2010).

Table 4.20: Comparison of systems (WFE-based) based Elsayed et al. (2010)

ϑ	Ant system	Tabu	DVRP-GA	PSO	VNS	Proposed GA-based DVRP
0	1.34	7.95	15.32	4.60	4.83	**19.58**
1	1.16	8.15	12.99	6.20	5.97	**19.87**
0.5	0.97	8.35	10.66	7.80	7.11	**20.17**

In Table 4.20, it is clear that the proposed GA-based DVRP performs best, as it obtains the best scores for all ϑ values.

4.4 Chapter Summary

In this chapter, the proposed GA modifications have been introduced and demonstrated in detail, and from the experimental results, it is clear that they enhance the GA. Also, four new approaches for evaluating DVRP systems were tested, from which one was selected as the best-performing method. This new approach, weighted fitness evaluations (WFE), is fairer than the previously used approach (time-based), as it does not depend on its running system's specifications and power.

The research conclusions, and some proposed future research work, are presented in the next chapter.

Chapter 5

Conclusions and Future Research Directions

This chapter summarizes the research carried out in this thesis, provides its conclusions, and suggests future research directions.

5.1 Research Summary

In this study, the Dynamic Vehicle Routing Problem (DVRP) was addressed by a Genetic Algorithm (GA)-based system that was enhanced by five proposed modifications: creation of the initial population for the first time slice and/or other time slices using both random and heuristic procedures; changing the selection process to allow the not-the-fittest chromosome to be selected, and thereby, increase diversity; altering the swap mutation to produce more variations in the mutated chromosome to, again, increase diversity; and, finally, a Local Optimal Condition (LOC) detection strategy, which allows the GA to try to escape local optima by increasing diversity when such a condition is detected. Subsequently, six GA-based systems were considered and compared by solving DVRP benchmark problems.

To date, a time-based evaluation approach has been used to evaluate DVRP systems. In this thesis, four new evaluation approaches, generations, raw fitness, weighted fitness and distance calculations, were proposed and tested. Also, a comparison with the previous published systems was conducted, based on the new proposed evaluation approach.

5.2 Conclusions

A comparison of the solutions obtained to the DVRP problems by the enhanced GA and previously published systems (Montemanni et al., 2005; Hanshar and Ombuki-Berman, 2007; Khouadjiaa et al., 2012) was performed (time-based). Based on the qualities of the solutions, the enhanced GA was competitive and out-performed the others regarding the best and average results, and in particular, found 14 of 21 new best solutions.

Of those proposed evaluation approaches, the weighted fitness evaluation (WFE) approach, which is based on running the DVRP system for 40,000 weighted fitness evaluations in each time slice, had the lowest standard deviation; it was thus the most stable evaluation approach regardless of the running system's specifications and power. Also, a comparison of the solutions obtained to DVRP problems by the enhanced GA and previously published systems was performed (WFE-based). Again, the enhanced GA was determined to have out-performed the other systems in terms of best and average results, as the numerical results from this comparison showed that the proposed GA-based DVRP found 17 of 21 new best solutions and 16 of 21 better averages. Also, it is reasonable that those results (WFE-based) could be compared with any proposed DVRP system in the future.

5.3 Future Research Directions

There are various possible future research directions for extending the research carried out in this thesis, including:

- parametric analysis of the system.
- test each enhancement on its own, comparing each against the same base case
- creating the initial population with some intelligent guidance with various degrees of randomness.
- using local search, like 2-Opt, as a mutation, instead of reverse and/or swapping mutations.
- solving a large set of real-world problem applications;
- considering more complex objective functions and fitness evaluation criteria;
- taking into consideration the minimizing of the number of vehicles and road balance while solving the problem; for example, considering the ratio of a vehicle's actual load to its capacity, and then balancing the ratios of all vehicles;
- considering more complex dynamic scenarios, such as introducing a customer time-window variable into the DVRP; and
- considering pairing the GA with local search techniques, such as the 2-opt local search and/or other metaheuristic techniques, such as Ant Colony Optimization (ACO).

References

Applegate, D. L., Bixby, R. M., Chvátal, V. & Cook, W. J. 2006. *The Travelling Salesman Problem: A Computational Study*, Princeton University Press.

Baker, B. & Ayechew, M. 2003. A genetic algorithm for the Vehicle Routing Problem. *Computational Operational Research,* 30**,** 787–800.

Bektas, T. 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures. *OMEGA-International Journal of Management Science,* 34**,** 209–219.

Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O. & Schiavinotto, T. 2006. Hybrid meta-heuristics for the vehicle routing problem with stochastic demands. *Math. Modelling Algorithms,* 5**,** 91-110.

Bräysy, O. 2003. A reactive variable neighborhood search for the vehicle routing problem with time windows. *INFORMS Journal on Computing,* 15**,** 347–368.

Bräysy, O. & Gendreau, M. 2005. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science,* 39**,** 104–118.

Brownlee, J. 2011. *Clever Algorithms: Nature-Inspired Programming Recipes*, Lulu Enterprises.

Chiang, W. & Russell, R. A. 1996. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research,* 63**,** 3-27.

Christofides, N., Mingozzi, A. & Toth, P. 1981. Exact algorithms for the Vehicle Routing Problem, based on spanning tree and shortest path relaxations. *Mathematical Programming,* 20**,** 255–282.

Clark, G. & Wright, J. W. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Search,* 14**,** 568-581.

Clerc, M. 2006. *Particle Swarm Optimization,* London, ISTE (International Scientific and Technical Encyclopedia).

Contardo, C., Cordeau, J.-F. & Gendron, B. 2011. A GRASP + ILP-based Metaheuristic for the Capacitated Location-Routing Problem. *In:* CIRRELT-2011-52 *Journal of Heuristics.* Université de Montréal.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., M.M., S. & Soumis, F. 2001. The VRP with Time Windows, in The Vehicle Routing Problem. *In:* TOTH, P. & VIGO, D. (eds.) *SIAM Monographs on Discrete Mathematics and Applications.*

Cordeau, J. F., Gendreau, M., Hertz, A., Laporte, G. & Sormany, J. S. 2005. New Heuristics for the Vehicle Routing Problem. *In:* LANGEVIN A., R. D. *Logistics Systems: Design and Optimization.* New York: Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. 2001. The traveling-salesman problem. *Introduction to Algorithms.* Second ed.: MIT Press and McGraw-Hill.

Cre´Put, J.-C., Hajjam, A., Koukam, A. A. & Kuhn, O. 2011. Dynamic Vehicle Routing Problem for Medical Emergency Management. *In:* Mwasiagi, J. I. *Self Organizing Maps - Applications and Novel Algorithm Design.* InTech.

Crispim, J. & Brandao, J. 2001. Reactive tabu search and variable neighborhood descent applied to the vehicle routing problem with backhauls. *MIC'2001.*

Czech, Z. J. & Czarnas, P. 2002. Parallel simulated annealing for the vehicle routing problem with time windows. *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing.* Canary Islands – Spain.

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

Dethloff, J. 2001. Vehicle routing and reverse logistics: The Vehicle routing problem with simultaneous delivery and pick-up. *OR Spectrum,* 23**,** 79–96.

Dorigo, M. 1992. *Optimization, Learning and Natural Algorithms.* PhD, Politecnico di Milano.

Dorigo, M. & Gambardella, L. M. 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation,* 1**,** 53-66.

Dorigo, M. & Stützle, T. 2004 *Ant colony optimization*, MIT Press.

Elsayed, S. M., Sarker, R. A. & Essam, D. L. 2010. A Comparative Study of Different Variants of Genetic Algorithms for Constrained Optimization. In: AL., K. D. E. SEAL, 2010. Berlin Heidelberg: Springer-Verlag, 177–186.

Feo, T. A. & Resende, M. G. C. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization,* 6**,** 109–133.

Fraser, A. 1957. Simulation of genetic systems by automatic digital computers. I. Introduction. Australian Journal of Biological Science 10, 484–491.

Fraser, A., & Burnell, D. 1970. Computer Models in Genetics. New York: McGraw-Hill. ISBN 0-07-021904-4.

Fukasawa, R., Longo, H., Lysgaard, J., Arag˜ Ao, M. P. D., Reis, M., Uchoa, E. & Werneck, R. F. 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming,* 106**,** 491–511.

Gendreau, M., Laporte, G. & Potvin, J. Y. 2002. Metaheuristics for the capacitated VRP. *In:* P. and Vigo, D. *The vehicle routing problem.* Philadelphia, PA: Society for Industrial and Applied Mathematics.

Gendreau, M. & Potvin, J.-Y. 1998. Dynamic vehicle routing and dispatching. *In:* Crainic, T. & Laporte, G. (eds.) *Fleet Management and Logistics.* London, UK: Kluwer.

Gendreau, M., Potvin, J.-Y., Bräysy, O., Hasle, G. & Løkketangen, A. 2008. Metaheuristics for the Vehicle Routing Problem and extensions: A Categorized Bibliography. *In:* Golden, B., Raghavan, S. & Wasil, E. (eds.) *The Vehicle Routing Problem: Latest Advances and New Challenges.* Springer.

Ghiani, G., Guerriero, F., Laporte, G. & Musmanno, R. 2003. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research,* 151**,** 1-11.

Glover, F. 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences,* 8**,** 156–166.

Glover, F., Gutin, G., Yeo, A. & Zverovich, A. 2001. Construction heuristics for the asymmetric traveling salesman problem. *European J. Oper. Res.,* 129**,** 555–568.

Grover, L. 1992. Local search and the local structure of NP-complete problems. *Oper. Res. Lett.,* 12**,** 235–243.

Gutin, G. & Punnen, A. 2004. *The traveling salesman problem and its variations,* Springer US.

Gutin, G., Yeo, A. & Zverovich, A. 2002. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics,* 117**,** 81-86.

Hahsler, M. & Hornik, K. 2007. TSP - Infrastructure for the traveling salesperson problem. *Journal of Statistical Software,* 23**,** 1-21.

Hansen, P., Mladenovic, N. & Perez, J. A. M. 2010. Variable neighbourhood search: methods and applications. *Annals of Operations Research,* 175**,** 367-407.

Hanshar, F. T. & Ombuki-Berman, B. M. 2007. Dynamic vehicle routing using genetic algorithms. *Applied Intelligence* 27**,** 89-99.

Haupt, R. L. & Haupt, S. E. 2004. *Practical Genetic Algorithms,* Pennsylvania, John Wiley & Sons, Inc.

He, R., Xu, W., Wang, Y. & Zhan, W. 2010. A Route-Nearest Neighbor Algorithm for Large-Scale Vehicle Routing Problem. *Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI).*

Helsgaun, K. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research,* 126**,** 106-130.

Hinton, T. G. 2010. *The Vehicle Routing Problem including a range of Novel Techniques for its Solution.* PhD, Bristol.

Ho, W., Ho, G. T. S., Ji, P. & Lau, H. C. W. 2008. A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Eng. Appl. Artif. Intell.,* 21**,** 548-557.

Hoff, A., Andersson, H., Christiansen, M., Hasle, G. & Løkketangen, A. 2008. Industrial aspects and literature survey: Fleet composition and routing. *SINTEF A7029.* SINTEF.

Holland, J. 1975. *Adaptation in natural and artificial systems*: University of Michigan press. MIT Press.

Hosny, M. I. 2010. *Investigating Heuristic and Meta-Heuristic Algorithms for Solving Pickup and Delivery Problems.* PhD, Cardiff University.

Kennedy, J. & Eberhart, R. 1995. Particle swarm optimization. *IEEE International Conference on Neural Networks.*

Kennedy, J., R.Eberhart & Shi, Y. 2001. *Swarm Intelligence,* Morgan Kaufmann Publishers.

Khouadjiaa, M. R., Sarasolab, B., Albab, E., Jourdana, L. & Talbia, E.-G. 2012. A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests. *Applied Soft Computing,* 12**,** 1426–1439.

Kilby, P., Prosser, P. & Shaw, P. 1998. Dynamic VRPs: A study of scenarios. *In:* APES-06-1998. U.K.: University of Strathclyde.

Kirkpatrick, S., Gelatt, C. D., & Vecci, M. P. 1983. Optimization by simulated annealing. Science, 220 (4598), 671–680.

Laarhoven, P. J. M. V. & Aarts, E. H. L. 1987. *Simulated Annealing: Theory and Applications,* Dordrecht, Springer.

Laporte, G. 1992. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research,* 59**,** 345-358.

Laporte, G. 2009. Fifty years of vehicle routing. *Transportation Sci.,* 43**,** 408–416.

Laporte, G. & Nobert, Y. 1987. Exact algorithms for the Vehicle Routing Problem. *Annals of Discrete Mathematics,* 31**,** 147–184.

Larsen, A. 2000. *The Dynamic Vehicle Routing Problem.* PhD, Technical University of Denmark (DTU).

Larsen, A., Madsen, O. & Solomon, M. 2002. Partially dynamic vehicle routing — models and algorithms. *Journal of the Operational Research Society,* 53**,** 637–646.

Larsen, A., Madsen, O. B. G. & Solomon, M. 2008. Recent Developments in Dynamic Vehicle Routing Systems. *The Vehicle Routing Problem: Latest Advances and New Challenges.* Springer.

Manfrin, M. 2004. *Ant Colony Optimization for the Vehicle Routing Problem.* PhD, Université Libre de Bruxelles.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. 1953. Equation of State Calculations by Fast Computing Machines. J. Chem. Phys., 21, 1087–1092.

Mitchell, M. 1998. *An introduction to genetic algorithm*, MIT press.

Mladenovic, N. & Hansen, P. 1997. Variable neighborhood search. *Computers and Operations Research,* 24**,** 1097–1100.

Montemanni, R., Gambardella, L., A., R. & Donati, A. V. 2005. A new algorithm for a dynamic vehicle routing problem based on Ant colony system. *Journal of Combinatorial Optimization,* 10**,** 327–343.

Ombuki-Berman, B. M., Ross B. J. & Hanshar, F. T. 2006. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Appl Intell,* 24**,** 17–30.

Ombuki-Berman, B. M. & Hanshar, F. T. 2009. Using genetic algorithms for multi-depot vehicle routing. *In:* PEREIRA, F. B. & J.TAVARES (eds.) *Bio-Inspired Algorithms for the Vehicle Routing Problem.* Springer

Osman, I. H. & Kelly, J. P. 1996. *Meta-Heuristics: Theory and Applications,* Norwell, MA, Kluwer Academic Publishers.

Osman, I. H. & Laporte, G. 1996. Metaheuristics: A bibliography. *Annals of Operations Research,* 63**,** 513-523.

Pillac, V., Gendreau, M., Guéret, C. & Medaglia, A. 2011. A Review of Dynamic Vehicle Routing Problems. *Industrial Engineering*.

Pisinger, D. & Ropke, S. 2010. Large Neighborhood Search. In: Gendreau, M. & Potvin, J.-Y. (eds.) Handbook of Metaheuristics. Springer US.

Prins, C. 2001. A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. MIC'2001 - 4th Metaheuristics International Conference. Porto, Portugal.

Prins, C. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research, 31, 1985–2002.

Punnen, A. P., Margot, F. & Kabadi, S. N. 2003. TSP heuristics: domination analysis and complexity. *Algorithmica,* 35**,** 111-127.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R. & Trotter, L. E. J. 2003. On the Capacitated Vehicle Routing Problem. *Mathematical Programming,* 94**,** 343-359.

Ropke, S. & Pisinger, D. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, 40, 544-472.

Runka, A. 2008. Ant Colony Optimization Algorithms with Local Search for the Dynamic Vehicle Routing. Brock: Brock University.

Secomandi, N. & Margot, F. 2009. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Oper. Res.*, 57**,** 214-230.

Shaw, P. 1997. A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems. Working Paper, University of Strathclyde, Glasgow, Scotland.

Shaw, P. 1998. Using constraint programming and local search methods to solve vehicle routing problems. in Principles and Practice of Constraint Programming - CP98, Lecture Notes in Computer Science, 417-431, M. Maher and J.-F. Puget (eds), Springer-Verlag, New York.

Shen, H., Zhu, Y., Jin, L. & Zou, W. 2010. Two-phase heuristic for Capacitated Vehicle Routing Problem. *Second World Congress on Nature and Biologically Inspired Computing (NaBIC).* Kitakyushu, Fukuoka, Japan.

Sivanandam, S. N. & Deepa, S. N. 2008. *Introduction to Genetic Algorithms,* Berlin Heidelberg, Springer-Verlag

Stützle, T. 1998. *Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications.* PhD, Technische Universität Darmstadt.

Surekha, P. & Sumathi, S. 2011. Solution To Multi-Depot Vehicle Routing Problem Using Genetic Algorithms. *World Applied Programming,* 1**,** 118-131.

Taillard, É. D., Laporte, G. & Gendreau, M. 1996. Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society,* 47**,** 1065–1070.

Takes, F. 2010. *Applying Monte Carlo Techniques to the Capacitated Vehicle Routing Problem.* Master, Leiden University.

Talbi, E.-G. (2009), in Metaheuristics: From Design to Implementation, John Wiley & Sons, Inc., Hoboken, NJ, USA. doi: 10.1002/9780470496916.refs

Toth, P. & Vigo, D. 2001a. *The Vehicle Routing Problem,* Philadelphia, SIAM.

Toth, P. & Vigo, D. 2001b. Branch-and-bound algorithms for the capacitated VRP. *In:* Toth, P., Vigo, D. *The Vehicle Routing Problem* Philadelphia: SIAM.

Weise, T. 2009. *Global Optimization Algorithms - Theory and Application* -, Self-Published.

Woensel, J. V., Kerbache, L., Peremans, H. & Vandaele, N. 2008. Vehicle routing with dynamic travel times: A queueing approach. *European Journal of Operational Research,* 186**,** 990–1007.