Cardiff University

School of Mathematics

The Dynamic Vehicle Routing Problem: A Metaheuristics Based Investigation

By

Maxwell Wallace

B.Sc. Mathematics, Operational Research and Statistics, Cardiff University

A thesis submitted for the degree of Doctor of Philosophy

July 2007

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Acknowledgements

I would like to thank Dr. Jonathan Thompson for his guidance, support and friendship; without him this work would never have been completed. Further thanks are expressed to the staff and my fellow postgraduates of Cardiff University School of Mathematics for making my time here so enjoyable.

My thanks go to the Engineering and Physical Sciences Research Council (EPSRC) for provision of funding for this research.

Finally, I would like to thank my family and friends for their support.

Summary

The desire to optimise problems is as prevalent in today's society as it has ever been. The demand for increases in speed and efficiency is relentless and has resulted in the need for mathematical models to bear greater resemblance to real-life situations. This focus on increased realism has paved the way for new dynamic variants to classic optimisation problems.

This thesis begins by considering the Dynamic Vehicle Routing Problem. The basic premise of this routing problem is as follows; a percentage of customers are known a priori, for which routes are constructed, further customers then arrive during the course of the working day and need to be incorporated into an evolving schedule.

Literature has proposed a timeslot approach, whereby one partitions the working day into a series of smaller problems, that one is then required to solve in succession. This technique is used to produce a variety of metaheuristics based implementations, most noticeably Ant Colony Optimisation and Tabu Search.

Consideration is then given to the Dynamic Vehicle Routing Problem with Time Windows. This problem is similar to the Dynamic Vehicle Routing Problem, but requires each customer to be serviced within a predefined period of the day. A metaheuristic approach adapted from the most successful algorithm implemented on the Dynamic Vehicle Routing Problem is presented.

Finally consideration is given to a time-based decomposition technique for the Vehicle Routing Problems with Time Windows (Large-Scale instances). This work makes use of the dynamic solution technique developed in the preceding work, and is used in conjunction with an Ant Colony Optimisation algorithm and a descent algorithm.

Contents

Frequently Used Abbreviations

ACO	Ant Colony Optimisation
ACS	Ant Colony System
AS	Ant System
BI	Best Improvement
CH	Commitment Horizon
CU	Capacity Utilisation
CVRP	Capacitated Vehicle Routing Problem
DDV	Decreased Depot Visibility
DOD	Degree of Dynamicity
DVRP	Dynamic Vehicle Routing Problem
DVRPTW	Dynamic Vehicle Routing Problem with Time Windows
EAS	Elite Ant System
FI	First Improvement
GA	Genetic Algorithm
GRASP	Greedy Randomised Adaptive Search Procedure
LP	Linear Programming
MMAS	Max-Min Ant System
NN	Nearest Neighbour
PNN	Probabilistic Nearest Neighbour
RCL	Restricted Candidate List
RPR	Random Proportional Rule
SA	Simulated Annealing
TC	Tabu Characteristic/Criterion
TL	Tabu List
TS	Tabu Search
TSP	Traveling Salesman Problem
TTV	Temporary Taboo Vehicle
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

Chapter One

Introduction

1.1 Introduction

The need to produce vehicle schedules is as prevalent in today's society as it has ever been. People and business often rely on the picking-up and delivery of goods on a daily basis, and as a result of this, the costs associated with transporting goods are often significant. However, one is able to minimise these costs, if efficient schedules can be produced within a suitable timeframe.

Although the exact costs of picking-up or delivering goods are often specific to an individual person or business, they will usually relate to the number of vehicles required (and driver costs) and the distance the vehicles have to travel (e.g. petrol costs).

With very small problems, one may be able to produce high quality solutions by intuitively building solutions a customer at a time, until they are all included in the vehicle schedule. However, as the problem size increases, this becomes a more complex task; people cannot rely on their own logic to produce the schedules.

In response to this problem, mathematicians developed a family of combinatorial optimisation problems known as Vehicle Routing Problems (VRP). These different problems were able to effectively model a variety of different problem scenarios (through the introduction of various constraints and objectives) that one might be faced with in real-life. It is within one of the less studied variants (that has only recently become solvable due to increases in computing power) that this thesis begins.

Given that this chapter is simply introducing the work in this thesis, it only provides an overview of the problem and the information required to produce solutions (further details will be included within the main chapters as and when required). The remainder of this chapter is structured as follows:

Firstly, some further background information about why these types of problem exist is presented. This section will include some basic details about general routing problems as well as a brief discussion on uncertainty (an idea central to the variants that this thesis considers) and the impact that this can have on solving these problems.

Secondly, the aims of this thesis are outlined. With a piece of work as sizable as this, it is important to establish a reason for conducting the research, and what one hopes to achieve from it. Note that these aims are of a very general nature (more detailed aims are outlined at the beginning of each chapter) as one cannot pre-empt the direction in which the research is likely to progress.

Finally, details relating to the structure of the thesis are presented (including a short discussion of the contents of each chapter). These details are discussed in a generalised way, so one is not required to have read the preceding chapters to understand.

1.1.1 Background

As explained previously, this thesis is concerned with a recently introduced complex variant of the VRP. The idea of adjusting certain components of the VRP to create a new problem is a popular way of furthering research in this area; many such variants have already been proposed (see Section 2.2). As they were developed at different times, and people have varying levels of interest in certain variants, the research into

these problem variants is at different stages of advancement. Some problems (such as the traditional Capacitated VRP (CVRP), explained in Chapter Two) are beginning to become of less interest, as so much research has been conducted; producing optimal (or near-optimal) solutions is now commonplace (for many of the standard benchmark datasets).

However, there is still a wide selection of more recent (or less fashionable) problem variants that have received relatively little attention. It is within one of these variants that this thesis begins (though in Chapter Six an entirely new problem variant is considered). Before further details about the specific problem variant are presented, it seems sensible to discuss some more general details about routing problems.

Introducing Routing Problems

The VRP and its variants all share the same principal aim, that being the production of a set of vehicle routes starting and ending at the depot(s), such that all customers are serviced. Complementing this aim, are a series of constraints that must be adhered to in order for a solution to be declared valid. These constraints often try to provide increased realism (or complexity) to the mathematical problem, introducing ideas such as vehicle capacity and the maximum distance vehicles can travel.

The particular constraints being applied to a problem determine which variant of the VRP family is being considered. A more complete definition of the VRP can be seen in Chapter Two along with an overview of some of the most prominent members of the VRP family.

Although constructing a valid set of routes is often trivial (unless the constraints are particularly difficult to adhere to), it is the desire to produce these routes with a predefined objective of minimising some characteristic(s) that makes these problems so complex. There is a considerable number of possible characteristics that can constitute the objective, including minimising: distance, number of vehicles, waiting times etc. To further complicate proceedings, it is often common practice to try to produce a schedule that minimises a combination of these (in which case a weighting system or order of precedence must be established a priori).

With the basic details about VRPs presented, the idea of routing with uncertainty can now be introduced. This is the key component that distinguishes the work in this thesis, from much of the research into routing problems that preceded it.

Routing with Uncertainty

With the majority of routing problems being NP-Hard, the production of optimal solutions is a complex task. As people often require solutions within a predefined timeframe, they often have to rely on good solutions that can be identified in a considerably shorter period of time (see discussion in Section 2.2).

However, due to recent enhancements in computing power, people have been able to consider more complex variants of the VRP (whilst still solving them in an acceptable period of time). One particular class of problem variant that has only recently begun to receive attention is the idea of routing with uncertainty; whereby one does not have all of the information one requires to produce a vehicle schedule.

Although there are different types of routing problem containing uncertainty (see Section 2.9 for details), the type considered in this thesis requires one to produce an initial schedule that is then updated (while it is being traversed) to account for the uncertain component being revealed. This type of uncertainty is known as dynamicity and allows for many new variants of the VRP.

With the details of the types of problem that are to be investigated presented, consideration can be given to the aims of the thesis and what one hopes to achieve by investigating these problems.

1.2 Thesis Aims

At its most basic, the aim of this thesis is to contribute to the growing body of work on routing problems containing uncertainty. It was felt that this would be an interesting area to investigate, given that solving this type of problem has only recently become possible.

With enhancements in computational power making the solution of dynamic problems a reality, this thesis abandons the idea of theoretical discussions (favoured

by many previous publications) in favour of producing results (akin to the way one would be likely to consider a well researched problem variant such as the CVRP). Note that as these problems are so recent, the scope for this work is also much wider reaching than it would have been for a problem that has received greater levels of attention.

In Section 2.9 there is a discussion on the different forms of uncertainty that could have been investigated, and the reasons presented as to why the Dynamic VRP (DVRP) with Dynamic Customers was decided upon. This problem is similar to the CVRP (with an objective of minimising distance) but one is only aware of a predefined proportion of customers in advance; others become known as the vehicles are traversing their routes (i.e. new customers need to be incorporated into an evolving schedule). When considering which solution techniques (i.e. metaheuristics) to utilise, it seemed beneficial to consider a wide selection and continue with the best performing. The results of this analysis suggested that the ACO metaheuristic was particularly well suited to this type of problem, and thus constitutes the main solution technique utilised in this thesis.

1.3 Thesis Structure

When undertaking research of this nature, it is not possible to identify in advance the direction that one is likely to end up pursuing. A structure was not established in advance (and followed rigidly), the work evolved as one discovered more about the problems being investigated.

The majority of the research in this thesis was conducted in the order in which it is now presented[1]; thus enabling one to see how the research has evolved over time. Although they consider similar problems, one is able to view each chapter as an individual piece of research, which relies on information obtained from certain aspects of the work that preceded it.

This chapter now concludes with an overview of each chapter:

[1] The work in the Addendum of Chapter Five was done out of sequence.

1.3.1 Chapter Two: Literature Survey

This chapter is designed to complement the work that is in the subsequent chapters. It presents details regarding a considerable volume of literature that will support the work in this thesis and provide one with a greater understanding of the techniques involved. It should be noted that the literature is also discussed within the chapters (specifically the earlier ones), where one is provided with greater details than given in this overview.

Consideration is given to similar problems (to those tackled in this thesis) and a variety of solution methods. The amount of detail provided for each solution method is proportional to the techniques prominence in the subsequent chapters. One is encouraged to refer back to this chapter should they be unfamiliar with either the problem or solution technique that is being considered.

1.3.2 Chapter Three: Establishing a Base for Further Research

This chapter is concerned with identifying the best direction (in terms of solution methods) that this thesis should take. Given that the problem being tackled (described in Section 2.9.4) is so recent, there is no significant body of research on which to base this decision. It seems prudent to try to test a variety of techniques, with the intention of identifying whether any are likely to be particularly successful or interesting.

This chapter contains two heuristic and five metaheuristic implementations that are all assessed on the same benchmark problems (thus allowing one to make a comparison of the results produced). This chapter will consider only very basic implementations of the metaheuristics, and from this set two techniques will be carried forward for further research in Chapters Four and Five (Ant Colony Optimisation (ACO) and Tabu Search (TS) respectively).

1.3.3 Chapter Four: ACO: Enhancing its Performance

In Chapter Three the ACO metaheuristic is identified as one that is worthy of further research; this chapter details that investigation. Using an existing algorithm that had been produced for the same problem (the ACS-DVRP, see Section 2.9.5) as a base, attempts are made to improve the quality of the solutions produced by considering a series of adaptations.

This chapter will conclude with a set of results that will be directly comparable to those produced via the ACS-DVRP, thus allowing one to critique the algorithm as one would for a traditional combinatorial optimisation problem.

1.3.4 Chapter Five: TS: Enhancing its Performance

In Chapter Three the TS metaheuristic is also identified as one that is worthy of further research; this chapter details that investigation. The direction that this work takes is fairly intuitive, with consideration given to the two key components of the TS metaheuristic: the neighbourhood and the way in which one determines if a move is tabu.

Once the basic TS algorithm has been developed, consideration is given to a variety of intensification and diversification techniques, aimed at further improving the performance of the search. As with Chapter Four, this chapter concludes with a set of results that will be directly comparable to the ACS-DVRP (though one will also be able to compare it to the algorithm developed in Chapter Four).

A short addendum is also included in this chapter. It presents the first steps of an interesting direction that one could explore were one interested in continuing work on this problem (the subsequent chapters of this thesis consider different problems). It concerns altering the amount of uncertainty and considering the impact that this has on the quality of solutions one is able to achieve.

1.3.5 Chapter Six: ACO for the DVRPTW

This chapter considers a different type of routing problem from the three chapters that preceded it. Although there remains considerable scope for further work on the previously investigated problem, it seems sensible to see if the techniques developed could be applied to a different problem.

The same type of uncertainty (i.e. dynamic customers) is introduced into a popular routing problem, which requires customers to be serviced during a predefined period of the day (known as a time window). Although the introduction of time windows significantly alters the problem with regards to producing solutions, one also has to compensate for a change in the objective function.

Instead of minimising distance, the VRP with Time Windows (VRPTW) (and hence the Dynamic VRPTW (DVRPTW)) is primarily concerned with minimising the number of vehicles deployed (though distance is retained as a secondary objective). With the existing ACO algorithm developed for the DVRP, this chapter is concerned with tailoring this algorithm to the new constraints and this new objective function.

Results for a newly created set of benchmark problems are presented, enabling one to assess the quality of the algorithm developed.

1.3.6 Chapter Seven: Problem Decomposition for the VRPTW

This chapter presents an interesting by-product of the work that was produced in Chapter Six. It considers making the VRPTW artificially dynamic, and by employing the techniques developed previously, enabling one to decompose large problems into a series of smaller sub-problems.

Given that the VRPTW is NP-Hard, there are obvious time benefits associated with solving smaller problems. If one can identify a suitable way of decomposing the problem (and then bringing it back together) one may be able to produce better solutions than would be possible when considering the problem as a whole. Problem decomposition is a well-known technique for the CVRP, but has received little attention with regards to the VRPTW.

1.3.7 Chapter Eight: Conclusions and Future Work

The key findings of each chapter will be summarised, along with details regarding future work that could be conducted (were one to wish to continue to build on the work in this thesis). Note that as each chapter contains its own conclusions and discussion on where further work may be of particular interest, many of these details will already have been presented.

As it is hoped that research will continue beyond the problems and techniques developed in this thesis, some potential further work of a more diverse nature (than that presented within the chapters) will also be discussed.

Note that this chapter also includes a comparison of the results produced by the algorithms developed in Chapters Four and Five, with results that have recently become publicly available.

Chapter Two

Literature Review

2.1 Introduction

This chapter is intended to act as a support document to the work contained in the main body of this thesis. One can choose to read it in advance (if one is unfamiliar with combinatorial optimisation problems and metaheuristic solution techniques), or refer back to it as and when required.

Although a considerable volume of literature is also included within the earlier chapters to support the work being undertaken, it seemed sensible to include the discussion of the less prominently involved publications within a separate chapter. This chapter presents details of existing research that has been conducted in areas related to either the problems considered or the solution methods explored. It is structured as follows:

It begins with consideration being given to the most prominent (and relevant) routing problems (TSP, CVRP and VRPTW). This section includes details on how and why these problems were first developed, and a discussion on the different ways people have solved them (exact and approximation techniques). Details for further reading on

these problems are also included, though any information required to understand the work in this thesis will have been presented.

A short introduction into the development of metaheuristics is then presented. This is followed by more detailed discussions (Sections 2.4-2.6) into the most relevant solution techniques. These methods are discussed in a very generalised way as they can be applied to a variety of problems (TSP implementations are presented where applicable). Given that one is simply intending to convey the theory of the techniques, no further reading is suggested.

Once the solution techniques have been suitably introduced, implementations of these for the CVRP and the VRPTW are presented (Section 2.7 and 2.8). These will include details for further reading should one be interested in learning more about a particular solution technique (for one of these two problems).

Finally, consideration is given to the idea of dynamic vehicle routing problems, what the term can mean, and what it means in this thesis. This section will include a discussion on the first set of benchmarks to be used in this thesis (Chapters Three through Five) and the only existing algorithm that has been implemented upon them.

2.2 The Evolution of Vehicle Routing Problems

The term VRP has come to encompass a great number of routing problems, each with their own different properties. As these varied problems were established over a period of time (spanning many years), it is useful to see how the problem evolved into the dynamic problems that are considered in this thesis.

There are three key routing problems (TSP, CVRP and VRPTW) that support the work in this thesis, and each of them is to be considered individually. The same basic structure will be used to present the prominent literature on each of the problems:

- Problem Introduction/Definition
- Background Information
- Route Construction Techniques

- Neighbourhoods for Route Improvement.

The considerable volume of research that has been conducted in this area makes it impossible to present all of the details and literature that has been produced, and as such, this literature review is simply an introduction to the work. Readers will be directed to further workings should they desire a greater knowledge in problems being discussed.

As is the case throughout this chapter, these details are being provided to support the work in this thesis. One will find further information (often in greater depth) when the particular techniques (or variants of them) are first implemented on the problems tackled in this thesis.

2.2.1 The Traveling Salesman Problem

Perceived by most as the single most important routing problem in combinatorial optimisation, the TSP is the most sensible place at which to begin this journey through the family of routing problems. It acted as a precursor to the many more complex routing problems (including the two of particular interest, the CVRP and VRPTW) and was the environment in which many of the solution techniques were developed.

There exist multiple ways in which one can explain the TSP, with many authors choosing to reword the explanation, presenting the problem in their own way. However, with little to be gained from a further definition being proposed, an existing explanation is presented. Taken from the book 'The Traveling Salesman Problem' Hoffman and Wolfe (1985), it not only offers up a concise description of the problem, but offers an insight into its complexity.

'If a salesman, starting from his home city, is to visit exactly once each city on a given list and then return home, it is plausible for him to select the order in which he visits the cities so that the total of the distances traveled in his tour is as small as possible. Let us assume he knows, for each pair of cities, the distance from one to another. Then he has all the data necessary to find the minimum, but it is by no means obvious how to use these data in order to get the answer'.

While it is simple to describe the problem, the history behind the problem is somewhat complex. It does not appear that there is a definitive answer as to who first considered the problem, but some of the early important workings are now discussed.

Background Information

The origins of the problem are still debated today, though it is believed that the current earliest example of a similar problem can be found in German texts dating back to the early 1800s. However, even this is debated with some believing the works of the mathematicians Sir William Rowan Hamilton and Thomas Penyngton Kirkman should (also) be credited.

The TSP is one of the earliest (and is easily the most famous) routing problems ever studied. Many believe that its popularity stems from the contrast between the simplicity of its description and the complexity of solving it. It was a problem that non-mathematicians could identify with, and it had an obvious real-life application.

Regardless of who was responsible for its inception, what is known, is the person who brought the problem to the operational research and wider mathematical community, Merrill Flood. In 1940s America, whilst working at the RAND Corporation, Flood described the problem to colleagues and the challenge of producing techniques to solve this problem was born.

To consider why some mathematical algorithm is needed, it is necessary to look at how one would solve the problem otherwise. To solve the problem one would have to evaluate all possible solutions, from which the best could be chosen (this technique is known as complete enumeration). However, for all but the smallest problems this is not a feasible idea. The number of possible solutions (each of which would need to be evaluated) to the TSP is shown in Equation 2.1, where n represents the number of cities (known as customers in the VRP) to be visited.

$$Number\ of\ possible\ tours = \frac{(n-1)!}{2} \tag{2.1}$$

The factorial element of this equation increases at such a rate that even for small problems, this soon becomes too large to contemplate complete enumeration (e.g. for n=10, there exist 181440 different solutions).

While one may be able to compute this number of tours on the computers of today, Flood and his colleagues did not have access to such technology. Furthermore, the size of problems being tackled now is far higher than even modern computers can deal with e.g. a popular current TSP instance, known as the Sweden Tour contains 24978 cities.

Solving problems of this size by compete enumeration (even on the most powerful computers) is completely infeasible, hence the need for a more mathematical approach.

The first published (with regards to modern mathematical journals) work, containing the moniker TSP was a report by Robinson (1949) from the RAND Corporation entitled 'On the Hamiltonian game (a travelling salesman problem)'. This work does not contain a working TSP solution technique (it solved a variant of the problem), but it discusses how their work came about via an unsuccessful attempt to solve the TSP.

The first breakthrough on the TSP came in 1954 when an optimal solution for a n=49 TSP was presented in Dantzig et al. (1954). It utilised a linear programming method, and contained many of the ideas still prominent in modern linear programming techniques. Interestingly, the title of this work made reference to it being a solution technique for a large-scale TSP problem, and while it would not be considered such now, n=49 was sizable for the time.

TSP: Route Construction Techniques

Two years after the breakthrough of Dantzig et al. (1954), the first heuristic approach was presented in Flood (1956). This work was responsible for the development of the nearest neighbour heuristic that is still central to many of the modern day techniques.

Flood (1956) implemented this method on the same n=49 problem that was solved to optimality by Dantzig et al. (1954) and produced a solution in 904 units (an increase

of nearly 30% to the optimal of 699). Regardless of this poor performance, the idea of approximation techniques for TSPs was born; an area of research that was to provide many challenges for years to come.

The seminal nature of this paper did not stop with the creation of the nearest neighbour technique, the first TSP improvement heuristic was developed and the two were used together in a composite procedure. Given that there is a discussion on improvement techniques in the next sub-section, this part of the algorithm is not discussed here.

There exist many other TSP construction techniques, though one does not need to focus too heavily on their descriptions, as they are not used in this thesis. Two examples of these are outlined below to demonstrate how the techniques can vary:

- Furthest Insertion

  One begins with a tour consisting of the two customers furthest apart; the next furthest customer (r) is added to the route in the place at which it will cause the smallest increase to the distance travelled, see Equation 2.2.

  $$c_{ir}+c_{rj}-c_{ij} \tag{2.2}$$

  where i and j are the customers which r is to be placed between and c is the distance.

  This process is repeated until all customers have been included in the route, see, Rosenkrantz et al. (1977).

- Convex Hull

  One begins with the convex hull of the customers (the smallest route that encloses all of the customers); the best place to insert each of the remaining customers (r) is then identified according to Equation 2.2. The insertion that has the smallest impact on the distance is made, and the process is repeated, see Golden et al. (1980).

Although these are only outlines, one can see how varied the construction techniques can be. Given that this thesis is to focus on the use of metaheuristics, it seems sensible to not dwell on the simple construction techniques. Further details of these and other construction techniques can be found in Golden et al. (1980).

TSP: Neighbourhoods for Route Improvement

Once a feasible tour has been identified via a construction heuristic (e.g. through one of the methods explained previously), one might wish to try to improve that solution. A number of methods exist for this, and are grouped under the heading, improvement heuristics. Many metaheuristics have a similar objective, but they will be discussed in a later section.

The general technique is known as descent and although the implementation details are not discussed here (they appear in Chapter Three when one of these improvement heuristics is first implemented), it seems sensible to present an outline of the theory here.

The basic premise of all of these methods is to make some form of adjustment to the current solution, and assess whether the distance has been reduced. If the distance is reduced, then the move is accepted, and adjustments to this new solution are considered. If the adjustment would have increased the distance, the move is discarded and a new adjustment is considered.

The way these adjustments are identified is through the concept of a neighbourhood, which is explicitly defined within each implementation. A neighbourhood contains all of the moves that can be made to a solution under its proposed definition. The various TSP improvement heuristics are essentially a collection of different neighbourhood definitions, the most prominent of which are now discussed.

The most common neighbourhood definitions are 2-Opt and 3-Opt:

- 2-Opt (Croes (1958), though the move was suggested in Flood (1956))
  This edge exchange neighbourhood removes two edges from the tour and reconnects the route in the only alternative way possible. This will result in the

latter part of the tour being traversed in the opposite direction to that which it was previously. This is not a problem for the TSP, but one should be aware if the problem is asymmetrical this is unlikely to be a productive neighbourhood.

- 3-Opt (Lin (1965))
  This works in a similar way to the 2-Opt neighbourhood but removes three edges from the tour. This means that there are two ways of reconnecting the resulting three paths into a valid tour (excluding 2-Opt moves) and both must be assessed.

The processing of these moves continues until one reaches a solution where no further moves can be made that reduce the cost (within the neighbourhood definition). Once this has been reached, the solution is said to be either 2-Optimal or 3-Optimal (depending which neighbourhood was used).

Perttunen (1994) states that '*a 3-optimal tour is always 2-optimal*', and as such, one can be sure that 3-Opt is a better neighbourhood than 2-Opt for reducing the cost. However, given the exponential increase in time required to run the 3-Opt algorithm compared to 2-Opt, one is often unable to run the algorithm to 3-Optimality.

One area of particular interest is in finding a balance between 2-Opt and 3-Opt such that the performance is increased to a standard higher than 2-Opt without requiring the processing time of 3-Opt. These neighbourhoods are found by only allowing certain 3-Opt moves to be made, the imposed restriction on the 3-Opt neighbourhood will lower the quality of the result but speed up the algorithm.

There are two popular neighbourhoods that exist between the 2-Opt and 3-Opt algorithms:

- 2.5-Opt (Bentley (1992))
  This neighbourhood expands upon the 2-Opt neighbourhood, by including a single type of 3-Opt move. The extra move relocates a single city from its current location and repositions it between two current tour neighbours.

Johnson and McGeoch (1995) report that the inclusion of this move will result in an improvement of 0.5-1% in solution quality, though at an expense of increasing run times by 30-40%.

- Or-Opt (Or (1976), though made popular by Golden and Stewart (1985))
  This neighbourhood further expands the 2-Opt (and 2.5-Opt) neighbourhood by including the relocation of chains of cities (of length 1, 2 or 3). These chains are then repositioned between two current tour neighbours. Johnson and McGeoch (1995) suggest that '*...an implementation should be a bit faster than full 3-Opt, although a slight loss in tour quality is to be expected*'.

Although the majority of work focuses on 2-Opt, 3-Opt or somewhere in between, one need not stop at removing this restricted number of edges. This technique can be expanded to the idea of k-Opt[1], where k is a used defined parameter. 4-Opt was considered in Lin (1965), but it was shown that there was little improvement in the solution quality beyond the more traditional 3-Opt.

The lack of success provided from higher levels of k (along with the obvious increase in run time) led people away from the area of k-Opt neighbourhoods, and towards variable-Opt neighbourhoods.

The most famous variable-Opt neighbourhood, and the only one discussed in this review is the Lin-Kernighan neighbourhood (Lin and Kernighan (1973)). This is widely accepted to be one of the best performing TSP neighbourhoods and has been used in multiple algorithms since its inception.

The traditional k-Opt algorithm required that one specify a value for k in advance of the algorithm being run. Lin-Kernighan removed the need to do this, by allowing the algorithm to increase the value of k based on how the search is progressing. The Lin-Kernighan algorithm is too complex to be described in further detail here, though readers are directed to the founding paper and Johnson and McGeoch (1995) for further details.

[1] Initially known as λ-Opt, but soon assigned this new moniker.

Discussion on Other Prominent TSP Workings

The workings of Dantzig et al. (1954) and Flood (1956) could be seen as causing a split in the way in which the TSP was tackled. With the early approximation techniques having been discussed, and the metaheuristics due for a similar analysis later in the chapter, consideration is now given to the exact methods.

The identification of optimal solutions via LP methods has continued to the present day, with much success. The many workings of Applegate, Bixby, Chvátal and Cook e.g. Applegate et al. (1998) built upon the cutting plane work of Dantzig et al. (1954) and lead to successfully identifying optimal solutions for problems of size n=7397, n=13509 and n=15112. The same group of authors (though this time in conjunction with Helsgaun) also managed to find the optimal solution for the Sweden Tour (n=24978) in 2004[2].

This concludes the introduction to the heuristic and exact methods for solving the TSP. The TSP is one of the most researched combinatorial optimisation problems of all time, and as such, producing an exhaustive review in this thesis is not possible. Readers are directed to Reinelt (1994) for a more detailed discussion of the history of this problem.

2.2.2 The Capacitated Vehicle Routing Problem

As interest in the TSP continued to grow, people began to adapt the problem to fit real-life situations that they were being faced with. It was within one of these adaptations, that another of the most important and heavily researched combinatorial optimisation problems was first identified.

Dantzig and Ramser (1959) (see upcoming background information for further details) proposed generalising the TSP; instead of allowing one vehicle (then referred to as the salesman) to service all of the customers, one has a fleet of vehicles with limited capacity. If a single vehicle can service all of the customers, one is faced with a TSP, if not, one must decide which vehicles are going to service which customers (this problem is known as the CVRP).

[2] Computation took the equivalent of 84.8 CPU years on a single Intel Xeon 2.8 GHz processor.

The CVRP usually employs a single objective function, whereby one is solely concerned with the distance that the vehicles have to travel. However, other variants exist, such as minimising the number of vehicles deployed then distance, or equalling the distribution of work amongst the fleet. All CVRP (and its associated DVRP) in this thesis will use the standard distance based objective function.

Although at first glance one may feel that this is very similar to the TSP, one should not underestimate the difference it has upon producing solutions.

Background Information

Nearly twenty years after Flood (1956) had introduced the TSP to his colleagues (though only three years after his first published work), this new, more generalised variant of the problem was proposed. Dantzig and Ramser (1959) introduced the problem that has come to be known as the CVRP (though it was then referred to as the truck dispatching problem).

Concerned with '*the optimum routing of a fleet of gasoline delivery trucks between a bulk terminal and a large number of service stations*' they proposed the first mathematical programming formulation and an associated procedure for producing what they described as near optimal[3] solutions.

The results produced by Dantzig and Ramser (1959) were soon improved upon by a simple greedy heuristic known as the Clarke and Wright Savings (C&WS) algorithm (Clarke and Wright (1962), implementation details will be discussed later). In this paper they present a problem instance (n=30) and solve it via their method and that of Dantzig and Ramser (1959).

In this analysis they were purely concerned with the minimisation of distance, and reported that their method resulted in a 17% reduction. It should be noted that they also used two less vehicles, though as this is not accounted for in either technique's objective function, such a comparison has little validity.

[3] Solutions have since been shown to be far from optimal.

These two papers were responsible for interest in the CVRP 'taking-off', and subsequent research usually credits these papers with the problem's establishment.

With the first two workings differing so significantly, one is drawn to how one might best solve the CVRP. Lenstra and Rinnooy Kan (1981) have shown that the CVRP is an NP-hard problem, and as such, solving it exactly is going to be a complex and time-consuming affair. Toth and Vigo (2002) believe that there are five main solution techniques for solving the CVRP:

- Branch-and-bound algorithms
- Branch-and-cut algorithms
- Set-covering-based algorithms
- Classical heuristics
- Metaheuristics

The first three of these focus on the complex task of solving the problem exactly. As the idea is not central to the work being pursued in this thesis, an understanding of these workings is not of paramount importance. However for completeness, an indication of the work, along with further reading will be presented at the end of this section.

The final two of these techniques are approximation techniques, of which a working knowledge is required, to support the work that is presented later in this thesis. Given that the metaheuristic methods will themselves have a dedicated section in this chapter it seems sensible to delay their discussion until they have been suitably introduced.

It is therefore the heuristic methods that are the primary area of interest for the remainder of this discussion on the CVRP. The heuristic techniques can be split into two sub-groups (in much the same way as they were for the TSP), namely construction techniques and route improvement techniques.

Following a review of the prominent early work on each of these sub-groups the exact methods will be introduced along with any other literature that is of particular relevance.

CVRP: Route Construction Techniques

The most famous construction algorithm is that of Clarke and Wright (1962); developed soon after the founding work of Dantzig and Ramser (1959). It begins with each customer being serviced by an individual vehicle i.e. routes are of the form (*0-i-0*) where i=1...n and the depot=0. If two of these routes can be feasibly merged into a single route, then one can compute the saving associated (see Equation 2.3).

$$s_{ij}=c_{j0}+c_{0j}-c_{ij} \qquad (2.3)$$

The C&WS algorithm then computes every feasible merge and implements the merge that has the greatest reduction on the distance travelled. The process is then repeated until there exist no feasible merges that reduce the distance..

Over time, various enhancements were made to the algorithm. Gaskell (1967) proposed the introduction of a parameter into the savings function to place greater emphasis on the distance between the vertices being connected in the route merge, while Golden et al. (1977) focused on reducing the time taken to compute and sort the savings (needed for each iteration of the algorithm).

Alternatively, one can use one of the many sequential construction algorithms that exist; instead of calculating the savings from the merging of two routes, the solutions are constructed one customer at a time. These techniques rely on different ways of selecting the next customer to insert into a partial solution and can be implemented in either a sequential (one vehicle at a time) or parallel manner (multiple vehicles at once). Although they have been shown to be successful, they are not of paramount importance to the working of this thesis, and as such, one is directed to Mole and Jameson (1976) and Christofides et al. (1979) for further reading.

Other prominent construction algorithms such as the sweep algorithm of Gillet and Miller (1974), the petal algorithm of Ryan et al. (1993) and the matching based

savings algorithm of Altinkemer and Gavish (1991) have not been discussed as they are of little relevance to the work in this thesis.

CVRP: Neighbourhoods for Route Improvement

In much the same way as for the TSP, upon the construction of a solution, one may choose to try to improve the solution through some form of descent. As before, this requires a suitable neighbourhood in which the moves can be made. However, when improving a CVRP solution, one has to take account of the fact that there exist multiple vehicle routes. These can significantly alter the way in which the neighbourhoods can operate.

It is commonly accepted (e.g. Baker and Schaffer (1986)[4] and Laporte and Semet (2002)) that there are two distinct ways in which these improvements can take place:

- Intra Vehicle (Single Route)

  These moves do not alter the clustering that was made in the construction process and simply seek to improve the individual routes. Each of these routes is essentially a TSP, and as such, one can attempt to optimise each one separately.

- Cross Vehicle (Multiple Route)

  These moves are a much more complex affair, as they directly alter the clustering. By allowing customers to move between routes, one is expanding the size of the neighbourhood considerably; reaching a local optimum is likely to take a considerable time.

One need not give too much consideration to the intra vehicle improvements as they are essentially TSPs, and as such, all of the previously discussed neighbourhoods are appropriate. Though it should be noted that if there are not many customers assigned to each vehicle, one could choose to solve them to optimality via one of the exact methods.

[4] Although this paper tackles the VRPTW the existence of multiple vehicle routes makes this a suitable reference.

Of more interest are the cross vehicle moves, though they are a complex affair as one is faced with multiple ways in which these moves can be made. Much of the early work on cross vehicle neighbourhoods relied on the TSP *k*-Opt edge exchange strategies of Lin (1965) e.g. Stewart and Golden (1984), but with the exception of 2-Opt (which exchanges the beginning and ending portions of two routes) larger values of *k* are more complex to explain.

Cross vehicle 3-Opt procedures tend to be of a 2.5-Opt ilk, whereby a customer is removed from one route and placed between two adjacent customers on a different route. This move was given the moniker SHIFT in Salhi and Rand (1987) and expansions to this type of neighbourhood were soon considered.

The generalised version of the SHIFT neighbourhood, which includes moves similar to those in Or-Opt (though again cross route), was given the name λ-Opt by Osman (1993). Although there is some ambiguity in the term λ-Opt, with Lin (1965) and Osman (1993) offering different definitions, the definition of Lin (1965) was quickly renamed as *k*-Opt (Lin and Kernighan (1973), see TSP discussion), and as such, one can continue to use the term λ-Opt to refer to the method of Osman (1993).

Osman (1993) introduced early SA and Tabu methods for the CVRP (and will be discussed in the metaheuristics section), but for now it is purely the neighbourhood definition that is of interest. The neighbourhood defined was a cross route neighbourhood for the CVRP that relied on customer exchanges as opposed to edge exchanges.

A λ-Opt move consists of the exchange of two subsets of customers[5] (each of size $\leq\lambda$) between two vehicles. To help explain the neighbourhood, Osman (1993) proposed two examples (where, $\lambda=1$), a summary of which is now provided:

[5] The customers were not required to be consecutive (see the CROSS Neighbourhood in Section 2.2.3).

- The (1,0) Shift Process (later presented as i:1,0)
  This removes a customer from one route and places it between two successive customers in another. This is the same move as the SHIFT move of Salhi and Rand (1987).

- The (1,1) Interchange Process (later presented as x:1,1)
  This move exchanges two customers from different routes. If this move were to be explained in *k*-Opt terms it could be thought of as a special case of cross route 4-Opt.

In a similar fashion to *k*-Opt, as λ increases, so does the complexity of the neighbourhood and the time it will take to reach λ-Optimality. One should be aware that if λ is set high enough to cover the entire route, amongst the many moves assessed will be the traditional 2-Opt move.

Although many other neighbourhoods exist for the CVRP, the examples that have been detailed here will give the necessary information to understand the history of the methods employed in this thesis.

Other Prominent Early CVRP Workings

There have been many efforts made to solve the CVRP exactly, though given the NP-hard nature of the problem, their success is always going to be limited to solving problems of a restricted size. The most well known exact method is branch-and-bound, thus details are provided about this type of solution technique as opposed to the two other exact methods identified previously. References for further reading (for each of these techniques) will be provided, as the level of research in this area is far beyond the remit of this chapter.

Much of the success of the branch-and-bound techniques has been achieved on the asymmetric variant of the CVRP. However, given the symmetric problem is a special case of the asymmetric problem, one can often implement the same algorithm on both problems. Laporte et al. (1986) produced the first branch-and-bound algorithm, the results of which (for the symmetrical CVRP) are now considered.

This algorithm was able to solve instances with $n \leq 90$ provided the number of vehicles was restricted to one or two. In instances where a greater number of vehicles were required, the level of performance was significantly reduced. The computing times for the most difficult instances considered were in excess of 5000 seconds[6] for the problems solved, though no details were reported for the problems that remained unsolved.

Fisher (1994) produced a branch-and-bound algorithm that did not require any restriction on the number of vehicles, and successfully solved a problem with $n=100$ (though the computation time was in the region of 60,000 seconds[7]).

One is directed to Toth and Vigo (2002) for further reading on branch-and-bound, while details regarding branch-and-cut and set-covering-based algorithms can be found in Naddef and Rinaldi (2002) and Bramel and Simchi-Levi (2002) respectively.

This concludes the introduction to the heuristic and exact methods for solving the CVRP. Note that as the CVRP is a very well-researched problem, producing an exhaustive review in this thesis is not possible. Readers are directed to Toth and Vigo (2002) for a more detailed discussion of the history of this problem.

2.2.3 The Vehicle Routing Problem with Time Windows (VRPTW)

As before, as time passed the interest in routing problems continued to grow, and it was not long before people began to consider the possibility of evolving the problem further. One area that seemingly captured the imagination of the research community was the idea of time windows, whereby each customer had to be serviced during a particular time of the day.

The VRPTW builds upon the CVRP framework; one is still required to build a series of vehicle schedules subject to all of the constraints that were previously applicable, but accompanying this is the need to satisfy a time window constraint also. The time windows are expressed as $[e_i,l_i]$ for each customer i, whereby e_i represents the earliest

[6] On a VAX 11/780
[7] On an Apollo Domain 3000

point at which customer i can be serviced and l_i the latest. When a vehicle arrives at a customer it must remain for duration (d_i), to complete the service.

In contrast to the CVRP, one is no longer solely concerned with the distance that the vehicles are scheduled to travel. The VRPTW employs a dual objective function, primarily concerned with the number of vehicles being deployed. Solutions that are produced in the same number of vehicles are then assessed according to distance.

There are two types of time windows that can be considered:

- Soft Time Windows
  These time windows can be violated at a penalty cost (an artificial increase to the objective function), and as such, allow greater flexibility in the schedules that can be produced.

- Hard Time Windows
  These cannot be violated and as such, service must begin before l_i has passed. If a vehicle were to arrive at a customer prior to e_i it must wait (till e_i) until it can begin its service

Soft time windows, although equally valid as a problem, have not received the same level of attention as hard time windows, and as such, all references to the term time windows will relate to the hard windows unless specified otherwise. The work in the later parts of this thesis (Chapter Six) that investigates the VRPTW and the associated DVRPTW will focus on the problem with hard time windows.

Consideration is now given to how and why time windows were considered such an important and interesting variant to develop.

Background Information

The addition of time windows to instances was not simply proposed as a way of making the CVRP more complex; they appeared gradually as people encountered their existence in real-life routing problems. Obviously, it is the VRPTW that is of

paramount interest in this section, but time windows have been considered in all manner of routing problems (e.g. Traveling Salesman Problem with Time Windows and Pick-up and Delivery Problem with Time Windows), a review of them is available in Solomon and Desrosiers (1988).

There appear to be three central papers that led to the establishment of the VRPTW as it has come to be known. Given that they were real-life problems, each one was slightly different in its make up, but they were restricted variations of the more general VRPTW.

The first of these workings, Pullen and Webb (1967), concerned the delivering of mail in the Central London area. Focusing on the reduction of the idle time that was apparent in the driver's schedules when waiting to service a customer, it used a simple construction heuristic to identify which vehicles were best suited to which customers. Given the ad-hoc nature of this investigation and the lack of numerical results, critiquing the performance of this algorithm is not possible. It is likely that the performance was relatively poor in comparison to modern techniques, but the authors report that result quality was of a higher standard than the existing manually developed schedules.

The remaining two early workings: Knight and Hofer (1968) and Madsen (1976) focused on the establishment of schedules for a contract transport company and a newspaper and magazine distribution company respectively.

Knight and Hofer (1968) developed a simple heuristic for increasing vehicle utilisation (which they defined as the average number of calls per vehicle hour) while minimising the number of vehicles required. They report that their implementation was successful and *'resulted in about 15% of vehicle time being made available for additional revenue-earning work without increasing the fleet size'*.

Madsen (1976) considered a time window problem without vehicle capacities or earliest service points. He developed two algorithms: a basic insertion heuristic and one based on Monte Carlo simulation. Solutions were improved via a cross route neighbourhood.

Since these early workings, Savelsbergh (1985) has identified that the VRPTW is an NP-hard problem, and that finding a feasible solution with a fixed fleet size is itself an NP-complete problem. In addition to this, Savelsbergh (1985) and Solomon (1986) have both shown that the VRPTW is a more difficult problem than the VRP, and as a result of this, it is clear why the majority of the early work into the problem focused on the use of heuristics.

The VRPTW represented a significant step in the evolution of vehicle routing problems, as it appears to be primarily a customer-orientated enhancement. The improvement of solutions to the TSP and CVRP shortened the routes travelled by drivers, but there was less discernible difference to the individual customers. If a company were able to produce high quality solutions to the VRPTW, then it would be possible for it to allow for customers to express a preference for when they wished to be serviced.

VRPTW: Route Construction Techniques

As mentioned in the previous section, the majority of the early workings did not focus on the identification of optimal solutions. People were well aware of the complexity of the problem, and as such, tended to pursue the use of heuristic techniques that had proved successful when applied to the TSP and CVRP.

The heuristics will be discussed first, while a short discussion of the early work into producing exact solutions will be provided at the end of this section.

The very earliest workings focused entirely on construction heuristics; they can be either sequential, where one vehicle is considered at a time, or parallel, where several routes are constructed simultaneously. In Solomon (1987) a comparison between seven sequential construction heuristics was conducted. With the problem still very much in its infancy, it appears that this work was conducted with the aim of establishing a base from which other work could stem.

Of the heuristics Solomon (1987) assessed, two were based on the Clarke and Wright savings algorithm (Clarke and Wright (1962)), while three were insertion heuristics that used various criteria to insert new customers into partial routes. The final two

methods considered were a time-orientated nearest neighbour heuristic and time-orientated sweep heuristic (adapted from the original sweep algorithm of Gillet and Miller (1974)).

The results of this analysis, although quite varied, suggested that one of the insertion techniques (a two-phase approach) was the most successful of the techniques when applied to a variety of problems. In the first phase, for each customer not assigned to a vehicle, the best feasible insertion position was calculated based on minimising the additional distance and time required. The second phase selected which of these customers to insert via the use of the savings concept. A parallel version of this algorithm was later developed and can be seen in Potvin and Rousseau (1993).

VRPTW: Neighbourhoods for Route Improvement

As with the TSP and the CVRP that preceded it, one is able to improve a constructed solution via some form of neighbourhood. The use of neighbourhoods in the VRPTW is a complex affair as moves must also satisfy the extra constraints that the time windows present.

However, as one will see through this discussion of the VRPTW neighbourhoods, one can select a neighbourhood that is less likely to fall foul of the time window constraints. However, careful neighbourhood selection does not make the time window constraints redundant; they still severely impact the search.

Their presence greatly decreases the number of feasible moves, and as such making improvements to a VRPTW is a more complex and time consuming affair than improving either the TSP or CVRP.

Two of the earliest published workings regarding improvement heuristics for the VRPTW are Russell (1977) and Baker and Schaffer (1986). These workings both make use of the same neighbourhood (a cross vehicle 2-Opt), though Baker and Schaffer (1986) supplement it with a further two components:

- Cross Vehicle (2-Opt)

  This is the same as the 2-Opt edge exchange procedure seen in Russell (1977) and they simply describe the method as having '*...the effect of exchanging the*

beginning and ending portions of two distinct routes'. This was a well known CVRP neighbourhood that had long since been shown to be effective.

- Cross Vehicle (3-Opt among three routes[8])
  This method is not a true 3-Opt algorithm (with regards to *k*-Opt transfers); it actually has far greater similarities to the 2-Opt algorithm described above. It has the effect of exchanging the beginning and end portions of three routes in the two possible ways. This appears to be the first appearance of this particular neighbourhood, though the technique was later generalised by Thompson and Psaraftis (1993). They gave the technique the name *b*-Cyclic *k*-Opt transfers (this particular move would be defined by setting b=3 and k=2).

- Intra Vehicle (2-Opt)
  These moves are made within a single vehicle (i.e. they are the same as the TSP), and as such, have already been discussed in some detail.

The selection of these three specific neighbourhood components was not made in an arbitrary manner; they all share one key principle that is beneficial for a VRPTW neighbourhood. Unlike the TSP and CVRP, solutions to the VRPTW have a definite orientation, and as a result of that, neighbourhoods that preserve orientation are desirable. In support of this, Baker and Schaffer (1986) carried out a series of computational experiments and reported '*...even the traversal of a short sequence of arcs in reverse order will often cause the route to become infeasible'*.

One can see that none of these three neighbourhood components will require the reversal of any portions of a route and as such, are well suited to this problem.

The desire to retain direction leads one to favour neighbourhoods of the λ-Opt ilk as reversal is common place in *k*-Opt. Unfortunately, the λ-Opt algorithm (as proposed by Osman (1993)) is somewhat unwieldy for anything other than very small values of λ

[8] Their chosen name for this technique though it differs greatly from traditional 3-Opt.

However, Taillard et al. (1997) were aware of the potential that these types of moves had and introduced a restriction that allowed one to implement a similar (but smaller) neighbourhood. Referred to as the CROSS neighbourhood, they suggested that one require the customers being moved to be consecutive. This allows cross vehicle moves of an Or-Opt ilk to be made (including cross vehicle 2.5-Opt) along with exchanging sets of customers between routes. This method still requires the selection of the parameter (though it is renamed L instead of λ), but one can use a larger value than under the original λ-Opt definition.

In the same working, they propose expanding this neighbourhood to include a similar set of intra vehicle moves, whereby a set of consecutive customers are removed from a route and replaced in the same route between two customers. Although the traditional k-Opt moves are defined for intra vehicle moves, it seems prudent to have a consistent neighbourhood definition. It should be noted that even with this restriction, the resulting neighbourhood is not small; evaluating it for anything but small values of L will be time consuming.

Other Prominent Early VRPTW Workings

Although efforts were made to solve the problem exactly, the complexity of it meant that efforts were restricted to small problem instances. Kolen et al. (1986) implemented the first exact method of solving the VRPTW, a branch-and-bound technique that borrowed heavily from the efforts of Christofides et al. (1981) for solving the CVRP. The resulting technique was able to solve an n=14 problem to optimality in 34.8 seconds of computing time[9]. Desrochers et al. (1992) state that subsequent workings (Knudsen (1989) and Madsen (1990)) managed to identify optimal solutions to 30 and 31 customer problems respectively.

In Desrochers et al. (1992) they stated that '*While heuristics have found to be very effective and efficient in solving a wide range of practical size VRPTW, optimal approaches have lagged considerably behind*'. In this paper they were attempting to readdress this unbalance, and proposed a new technique for identifying a lower bound that could be used in a branch-and-bound technique. They successfully solved a

[9] On a VAX 11/785 using VAX-Pascal compiler

number of 100 customer problems, albeit with sizable run times (up to 8939.1 seconds[10]).

This concludes the introduction to the heuristic and exact methods for solving the VRPTW. The VRPTW is a well-researched problem, and as such, producing an exhaustive review in this thesis is not possible. Readers are directed to Cordeau et al. (2002) for a more detailed discussion of the history of this problem.

2.3 The Development of Metaheuristics

In Section 2.2, a number of routing problems that were of particular interest to the workings of this thesis were presented. Accompanying each problem was a discussion relating to their complexity and how one was often reduced to using an approximation technique to identify good (but often not optimal) solutions.

The early techniques that one used to produce these solutions were known as heuristics (or simple heuristics) and took the form of a route construction or route improvement procedure (or a composite of the two). These methods were fairly simplistic in their nature (e.g. Convex Hull or the 2-Opt neighbourhood) and often did not produce results of a particularly high quality.

As people attempted to improve upon these results, the algorithms developed became more and more complex in their nature. These algorithms began to have very different properties from the heuristics that inspired them and eventually they were afforded their own title 'metaheuristics'. Due to the way in which the algorithms operate, they are likely to outperform the simple heuristics across a wide range of problems and have gone on to become one of the most popular areas of research for combinatorial optimisation problems.

This section begins the discussion into these metaheuristics, and is divided into two subsections. Firstly, consideration is given to understanding the term metaheuristic, why it was necessary to introduce the term and what it has come to mean. Secondly,

[10] On a SUN SPARK 1 using FORTRAN (F77)

some of the techniques that one would classify as metaheuristics are listed along with a discussion about how relevant they are to the workings of this thesis.

2.3.1 Understanding the Term Metaheuristic

The term metaheuristic was first proposed by Glover (1977) and is seemingly the composition of the word heuristic and the prefix meta. This discussion begins with dictionary[11] definitions for these, with a hope that an understanding of the reasons for the term's introduction can be established. This discussion will conclude with details relating as to what the term has come to encompass to the modern researcher.

- Meta (prefix) ..., beyond, ..., higher, more developed.
- Heuristics (noun) a method of solving a problem for which no formula exists, based on informal methods or experience, and employing a form of trial and error iteration.

With the more traditional term heuristic well established, one needs to consider why it was felt necessary to add the prefix meta. It seems a reasonable assumption that it was proposed as an attempt to distinguish between simple heuristics and those that intend to utilise a greater level of intelligence, to achieve (hopefully better) solutions. As the workings became more successful and complex, they no longer shared such basic concepts as the heuristics, and thus deserved to be distinguished from them.

Implementing a metaheuristic is usually a more difficult affair than the basic task of implementing a heuristic method. Recent history has shown that (for a wide range of distinct combinatorial optimisation problems) the quality of results has dramatically increased; suggesting that this added complexity is worthwhile.

Such is the development and confidence in the higher result quality provided by various metaheuristics; it could be considered unusual for one to focus on traditional heuristic techniques.

[11] Encarta World English Dictionary

With an understanding of the reasoning behind the terms creation, consideration is now given to what the term represents to the modern researcher. The term remains without a dictionary definition, thus many researchers have taken it upon themselves to propose definitions. Unfortunately given the widely accepted scope of what the term has come to encompass, none have been universally accepted.

Two interesting, yet seemingly dissimilar proposed definitions are given below:

- Osman's Definition (Osman and Hindi (2004))

'...a master process that guides and modifies the operations of subordinate heuristics to produce high quality solutions efficiently'

- The Metaheuristics Network[12] Definition

'A metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem.'

These two definitions are supposedly defining the same word, though it is clear they cover little common ground. They are inherently different as to which features they believe are central to a technique being defined as a metaheuristic. This lack of cohesion (which is further increased when one considers the multiple other definitions that exist) is likely to remain, and as such, any further discussion could be considered a somewhat fruitless task.

If one had to state a preference between the two proposed definitions; Osman and Hindi (2004) seems to have better retained the essence of why the word was required in the first place. This concise definition also hints at the processes evolution in a way in which the somewhat cumbersome Metaheuristics Network's definition does not. However, with no specific need for a single globally accepted definition, the multiple

[12] see http://www.metaheuristics.org for further details

definitions are able to happily co-exist. This has been the case for many years, and it would be reasonable to assume that this will remain so.

2.3.2 Metaheuristic Techniques

With the term suitably introduced, consideration can be given to the methods that one would classify to be metaheuristics. One must be aware that detailing a metaheuristic is a time consuming affair, they cannot be presented in the simple bullet point form that some heuristics were (in Section 2.2.1). They require a variety of principals to be explained along with a working knowledge of certain heuristics techniques that they make use of. This section does not give consideration to the workings of the algorithms, just their names and their relevancy to the work in this thesis.

A selection of some of the most popular metaheuristics can be seen below:

- Ant Colony Optimisation (ACO)
- Genetic Algorithm (GA)
- Greedy Randomised Adaptive Search Procedure (GRASP)
- Memetic Algorithm (MA)
- Simulated Annealing (SA)
- Tabu Search (TS)

These are merely a selection of some of the most prominent metaheuristics and the list is not intended to be exhaustive. However, with the workings of this thesis being restricted to a subsection of these, it demonstrates the variability of the methods that are encompassed by the term metaheuristic.

Some of these metaheuristics will receive a greater level of attention in this thesis than others. The reasoning for these selections is partly down to personal opinion but these issues along with more quantitative reasons are presented in Chapter Three. However, one must know which algorithms are to be investigated so that the relevant literature can be discussed and the reader informed on the necessary details to support the latter workings.

The most prominent metaheuristic in this thesis is ACO, and as such, the details provided regarding this metaheuristic is more in-depth than those provided for any others. Details pertaining to its development (on the TSP) can be seen in Section 2.4, while other routing problem implementations can be seen in Sections 2.7.1 and 2.8.1.

Considerable attention is also given to TS, as this is the second most utilised algorithm in this thesis (see Chapters Three and Five). Details regarding its development (on the TSP) can be seen in Section 2.5, while other routing problem implementations can be seen in Sections 2.7.2 and 2.8.2.

For completeness, two other metaheuristics are outlined: SA and GRASP. Details regarding their development (on the TSP) can be seen in Section 2.6, while other routing problem implementations can be seen in Sections 2.7.3 and 2.8.3. One should be aware that the amount of details provided for each of the metaheuristics is in correlation to the technique's prominence in this thesis.

2.4 Ant Colony Optimisation

The use of artificial ants to solve combinatorial optimisation problems is a relatively new idea; introduced in the early nineties in the thesis of Marco Dorigo (Dorigo (1992)). This application was supported by two further publications: Dorigo et al. (1991) and Colorni et al. (1992) and quickly went on to inspire many works on a broad range of problems.

This section explains the evolution of the technique, beginning with its inspiration from nature, through to the complex set of algorithms that now constitute the ACO family. All publications discussed in this section will relate to the use of ACO algorithms to solve the TSP (the problem on which the algorithms were developed).

Furthermore, given the sheer volume and diversity of the research (even on the TSP) one cannot present all of the published workings and as such, those most applicable to the work contained in this thesis are concentrated on. References for further reading will be provided.

2.4.1 Inspiration by Nature

The inspiration behind the ACO metaheuristic comes from the communication technique of the ant species Linepithema Humile. Near blind and without the ability to communicate directly, they are able to successfully work together to discover the shortest path between a food source and their colony.

Ethologists identified that the communication medium used by ants was through the depositing of pheromones (in varying quantities), which over time established a trail. Dorigo et al. (1991) used this knowledge as the basis for their artificial ants, and referenced workings such as Deneubourg et al. (1983) and Deneubourg and Goss (1989) to demonstrate how studies into the behaviour of real ants inspired their workings.

In order to understand the principal of the ACO metaheuristic, one must be familiar with the foraging activities of ants. When observing ants hunting for food, one can see that they tend to follow the shortest path between the colony and the food source. As mentioned previously this is due to the depositing of pheromones, this principal is now explained through a series of diagrams similar to those utilised by Dorigo et al. (1991).

In Figure 2.1 one can see how the ants follow the shortest path between the colony and the food source. This is because the route has been traversed frequently, leading to high levels of pheromone being deposited along it.

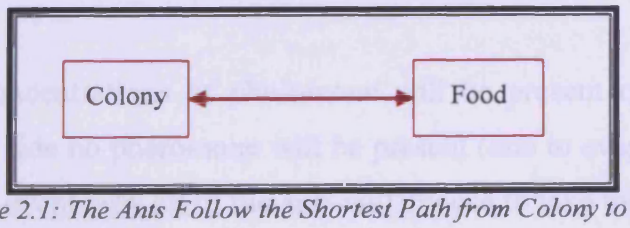

Figure 2.1: The Ants Follow the Shortest Path from Colony to Food

In Figure 2.2 an obstacle (such as a stone) has been placed somewhere along the route in an asymmetrical position. This prevents the ants from following the existing route, and as such a new route must be identified.

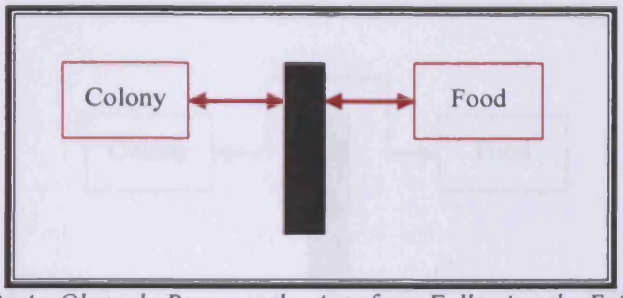

Figure 2.2: An Obstacle Prevents the Ants from Following the Existing Path

In Figure 2.3 the ants are faced with a choice (due to the absence of pheromone) as to which path they should choose. Given that they are unable to visualise the problem to determine which path they should take, they make a random choice (resulting in approximately half traversing the two possible paths).

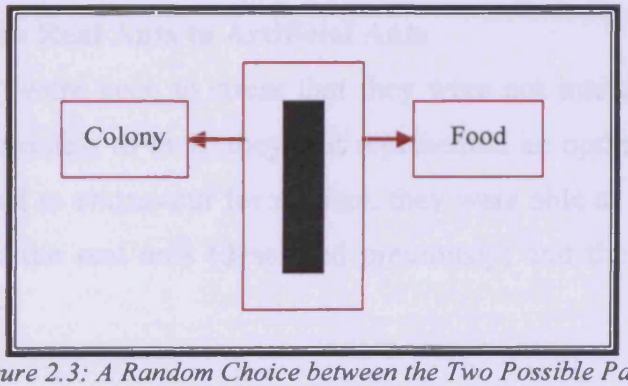

Figure 2.3: A Random Choice between the Two Possible Paths

However, there will be a quicker accumulation of pheromone on the shorter path, which in turn, will further bias the subsequent selection of that path. This starts an autocatalytic process, whereby an ant's decision to take a particular path will increase with the number of ants that have previously taken that path.

Eventually, high concentrations of pheromone will be present on the shorter side, while on the longer side no pheromone will be present (due to evaporation). In Figure 2.4 one can see that eventually all of the ants will choose to take the shorter path.

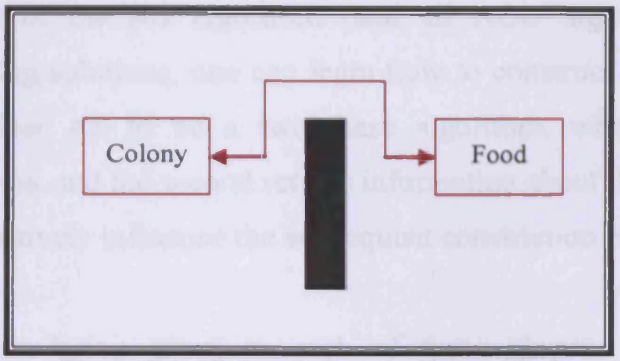

Figure 2.4: All Ants Follow the Shortest Path

With this underlying theory of the ants pheromone deposits described, one can turn their attention to how these ideas can be lifted from a real-life setting to an artificial one (for use in solving combinatorial optimisation problems).

The Transition from Real Ants to Artificial Ants

Dorigo et al. (1996) were keen to stress that they were not interested in the realistic simulation of ant colonies; to them they just represented an optimisation tool. Given that they did not need to endeavour for realism, they were able to introduce some key differences between the real ants (described previously) and the artificial ones that they intended to use:

- The artificial ants will have some memory
- They will not be completely blind.
- They will live in an environment where time is discrete.

These characteristics, coupled with the information provided about the real ants foraging activities enabled the founding ACO algorithm (Ant System) to be developed.

2.4.2 The Ant System Algorithm (Including Elitism)

The first of the ACO algorithms developed was the AS algorithm and it contains many of the features still prominent in the more complex ACO algorithms employed today. Although the majority of the ACO based work in this thesis will not make use of the AS algorithm, one is required to understand it, so as to better appreciate the algorithms which it inspired.

The basic principal of the AS algorithm (and all ACO algorithms) is that by repeatedly constructing solutions, one can learn how to construct better solutions. In fact, one can consider AS to be a two-phase algorithm, where the first phase constructs the solutions, and the second retains information about these (and previous) solutions so as to positively influence the subsequent constriction phases.

Prior to consideration being given to each of these phases, it is necessary to familiarise oneself with the way in which information learnt about previous solutions are stored.

In the real-life ant examples (Figures 2.1-2.4) the information about the quality of a decision was known (to the ant) by the amount of pheromone present. With the artificial ants in ACO algorithms (for the TSP) the pheromone values are associated with customer arcs. A matrix (referred to as the trail matrix) size $n \times n$, stores the amount of pheromone present on each customer arc. Once solutions are produced, the amount of pheromone on each arc will adjust according to the quality of the solutions that contain that arc. The pheromone values will also be adjusted to account for evaporation as would occur in the real-life environment. The trail matrix will represent the learned desirability of choosing customer j when at customer i.

The two phases of the AS algorithm are now considered.

Tour Construction

Each ant in the colony (a user controlled parameter m) independently constructs a complete solution to the TSP. Therefore, one need only detail the process for an individual ant, as all of the m ants in the AS behave in the same manner.

A solution is constructed a customer at a time, appending chosen customers to an incomplete solution until all customers are included. As one wants to ensure that only feasible solutions are constructed, one must remove customers that have already been scheduled from this selection process. The set F^k (where one is considering the k^{th} ant) represents the customers that are yet to be scheduled.

The construction process makes an informed selection from the set F^k, based on how successful that arc has been in the past, and how successful a greedy heuristic thinks it ought to be. This second component is required; as one cannot base decisions entirely on the quality of previous results (the quality may have been poor).

This second component is calculated according to some heuristic type information, similar to that which would have been used to solve the problem prior to the development of metaheuristics. Referred to as the visibility, it is user defined (an $n \times n$ matrix) and represents the a priori desirability of choosing customer j when at customer i.

An important feature of the AS construction technique is that it has to make use of the information in the trail and visibility matrices, whilst not allowing itself to follow them rigidly (and resulting in the search stagnating). One must allow the selection to be merely guided towards including previously promising arcs and those that appear they ought to be successful.

These two pieces of information are combined with an element of randomness to allow the necessary variability to occur via a probabilistic action rule known as the Random Proportional Rule (RPR), and can be seen in seen in Equation 2.4.

$$p_{ij}^k = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in F^k}\left((\tau_{il})^{\alpha}(\eta_{il})^{\beta}\right)} & \text{if } j \in F^k \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

This function makes the selection of a particular arc (i,j) in a probabilistic manner based on the trail and visibility values (τ_{ij} and η_{ij} respectively). Dorigo et al. (1996) initialised the trail values τ_{ij} to 1 and defined the visibility as the reciprocal of the distance between customers i.e. $\eta_{ij} = 1/d_{ij}$ (to favour the selection of customers in close proximity to the current vehicle location). The parameters α and β allow the user to control the bias towards each of the two components.

When each ant in the colony has produced a solution a cycle is deemed to have passed and the pheromone values in the trail matrix are updated. The AS algorithm terminates when a predetermined exit criterion (usually based on the total computing time or the number of cycles) is met.

Consideration is now given to the way in which the pheromone trails are updated.

Update of Pheromone Trails (Three Learning Phase Definitions)

After each of the m ants have constructed their solutions, the pheromone values in the trail matrix require updating (the visibility values are fixed). This updating is made according to a predefined update rule, seen in Equation 2.5.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2.5}$$

Firstly, the values of all entries in the trail matrix are reduced, simulating the evaporation of the trails in the real world. This is done via the user controlled parameter ρ, such that $(1-\rho)$ represents the evaporation of the trail during one cycle of the AS algorithm.

Secondly, the increase to the trail (the learning phase) is considered. Represented by the rather ambiguous term $\Delta\tau_{ij}^{k}$ it aims to encourage the search towards the more fruitful areas of the search space. In the founding work, three definitions of this parameter were proposed, and these are now discussed.

The various definitions for $\Delta\tau_{ij}^{k}$ (known as Ant-Density, Any-Quantity and Ant-Cycle) are all in Dorigo et al. (1991). Each of these definitions is now discussed in Equations 2.6-2.8 (note that T^k represents the tour built by the k^{th} ant, with length L_k):

- $\Delta\tau_{ij}^{k} = \begin{cases} Q & \text{if arc } (i,j) \text{ belongs to } T^k \\ 0 & \text{otherwise} \end{cases}$ (2.6)

Ant-Density, the increase in τ_{ij} when an ant travels along arc d_{ij} is independent of d_{ij} and L_k, as all increases take the form of the constant Q.

- $$\Delta\tau_{ij}^{k} = \begin{cases} Q/d_{ij} & \text{if arc } (i,j) \text{ belongs to } T^k \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

Ant-Quantity, the increase in τ_{ij} when an ant travels along arc d_{ij} is inversely proportional to the cost of that specific arc.

- $$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if arc } (i,j) \text{ belongs to } T^k \\ 0 & \text{otherwise} \end{cases} \quad (2.8)$$

Ant-Cycle, the increase in τ_{ij} when an ant travels along arc d_{ij} is inversely proportional to the total length of the tour produced by that ant.

The three methods were all analysed in Dorigo et al. (1996) on the same dataset (Oliver30, a 30 city problem first seen in Whitely et al. (1989)), thus enabling a direct comparison.

As expected (due to its use of global knowledge) the Ant-Cycle learning phase performed best. As a result of this analysis the other two definitions (Ant-Density and Ant-Quantity) appear to have been discarded (no known further research has ensued). It is now common practice to use the term AS to refer to the AS with the Ant-Cycle learning phase, and this will be adopted from here onward.

Once results were obtained via the AS, it soon became apparent that there was considerable scope for improvement, and identifying ways of achieving this became the new focus. Dorigo (1992), Dorigo et al. (1991) and Dorigo et al. (1996) were aware of this and the first improvement was actually included in these workings.

The idea proposed was that of the elitist ant, which is one of the most important improvements that has been made to the AS algorithm. Some authors consider it to be

a separate algorithm, but it is included here as it only makes a slight adjustment to the definition of the learning phase.

Elitist Ant System

The idea of elitism first appeared in De Jong (1975), where it was used as part of a GA. The idea relies on the principal that good solutions will occur close to good solutions already found. Dorigo (1992), Dorigo et al. (1991) and Dorigo et al. (1996) utilised this idea in their AS, and with it, the Elitist AS was developed.

The elitist ant was a simple intensification strategy designed to blend seamlessly within the existing AS framework. As the name suggests they do not perform like ordinary ants, in fact all elements of randomness (the fundamental principle behind ACO) are removed. These ants do not choose a route; they simply follow the 'best' existing one and hence reinforce the pheromone deposited upon it. This begins the autocatalytic process described previously, whereby as more and more ants walk a route (be they elitist or regular) they further encourage future ants to follow that same route.

It is important to realise that it only encourages the subsequent ants to select arcs belonging to this route; to force them would be counter-productive. The elitist ants simply guide the search toward the more fruitful regions of the search space; yet still allow the necessary digression.

With the theory underlined, the implementation is considered. The number of elitist ants was proposed as a user defined parameter (e), and this was used in an amended version of the update rule (Equation 2.5), and is seen below in Equation 2.9.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs} \tag{2.9}$$

The majority of this equation has remained the same, however there is an extra term added, $e\Delta\tau_{ij}^{bs}$, where e is the multiplier identified previously. The other component of

this new term is described in Equation 2.10 (where T^{bs} is the best tour found so far, and L_{bs} its cost).

$$\Delta\tau_{ij}^{bs} = \begin{cases} Q/L_{bs} & \text{if arc } (i,j) \text{ belongs to } T^{bs} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

With the definitions in place Dorigo et al. (1996) re-ran the now dubbed Elitist AS on the Oliver30 problem, and with an appropriate value of e (8 in this case), reported an improvement to the result quality along with a significant reduction in the number of cycles required to reach this best result.

With the exception of a comparison with other heuristics (TSP-tailored and general), in which it performed very well, the introduction of elitist ants ended the papers discussion of the TSP. It moved its focus onto other problems, namely the Quadratic Assignment Problem and the Job-Shop Scheduling Problem.

Since these early papers, interest in Ant Algorithms has increased and many new adaptations and enhancements have been introduced. In 1996 the term ACO was devised to cover all of these new workings. An investigation into the direct extensions to the AS or Elitist AS (the terms are often synonymous) is now presented (note that these extensions were all first implemented on the TSP).

2.4.3 The Two Direct Extensions to the AS Algorithm

Soon after the publishing of the early workings (Dorigo (1992), Dorigo et al. (1991) and Dorigo et al. (1996)), it became clear that there was scope (beyond the elitist ants) within the AS framework for further improvements. These improvements, of which there are two key ones (that do not operate in unison), use the same core features as the AS, and therefore are considered as direct extensions rather that new algorithms in their own right.

These two improvements are now discussed:

AS_{rank}

This improvement to the AS came from the University of Vienna, in the form of Bullnheimer et al. (1999a). The three authors were not involved in the earlier work (with that being based in Italy), thus highlighting how interest in the ACO field grew quickly. They too applied their algorithm to the TSP; the Oliver30 dataset was again used, this time supplemented with four further datasets (provided by Gerhard Waescher and Joerg Heuer from an industrial application).

They proposed the idea of ranking the ants, which can easily be considered as a natural extension to the idea of elitism. Instead of having only the best ant's tour being retraced by a predefined number of elitist ants, the best σ tours are retraced, and updated by a weighted system (similar to elitist ants, where the weight would have been defined as e). The technique employs a simple paradigm, whereby the better the tour the higher the weighting.

The AS_{rank} algorithm differs from AS in that this is the only form of pheromone depositing, and as such, it is usual for some ants not to deposit pheromone (it is possible if the user chooses $\sigma = m$). The AS_{rank} pheromone update rule can be seen in Equation 2.11, below.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{\mu=1}^{\sigma-1} \Delta\tau_{ij}^{\mu} + \sigma\Delta\tau_{ij}^{bs} \tag{2.11}$$

This formula still has a form similar to the Elitist AS, though the middle of the three components has changed significantly (the only other change being that the multiplier for the elitist ant has changed from e to σ, to fit with the amended middle component).

Replacing the update rule that cycled through all m ants, is one where it cycles through the best σ tours (less the best, which is still expressed individually, as in the Elitist AS) and deposits variable levels of pheromone. This variability comes through the term $\Delta\tau_{ij}^{\mu}$, which is defined as follows (Equation 2.12).

$$\Delta\tau_{ij}^{\mu} = \begin{cases} (\sigma-\mu)/L_{\mu} & \text{if arc } (i,j) \text{ belongs to } T^{\mu} \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

The AS_{rank} algorithm was shown to perform significantly better than AS, but more importantly slightly better than the Elitist AS. The idea of ranking has received little further research since its inception, though the technique was used to solve the CVRP, a description of which can be seen in Section 2.2.

Max-Min AS

The aptly titled Max-Min AS (MMAS) was introduced (as a TSP solution technique) in Stützle and Hoos (1997), though this acted merely as a precursor to the much more detailed workings by the same authors; Stützle and Hoos (2000). This latter paper analysed the MMAS on the TSP and the QAP. Inspired by the successes achieved with AS by Dorigo (1992), Dorigo et al. (1991) and Dorigo et al. (1996), they hoped to improve results with an adapted version; where the adaptations focused entirely on the trail.

The trail is an obvious place to look for improvement as it represents the information that has been learnt, and is central to the ACO philosophy. The MMAS has three key differences from the standard AS.

Firstly, the update of the pheromone trails only occurs after all ants have constructed a solution (one can choose to update the trail after each ant in AS/EAS). Upon completion of the construction, only a single ant is used to modify the trail values, this is done via Equation 2.13.

$$\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}^{best} \tag{2.13}$$

where $\Delta\tau_{ij}^{best} = 1/L_{bs}$ or $\Delta\tau_{ij}^{best} = 1/L_{ib}$

The authors point out that the use of L_{bs} will result in a far greedier search than L_{ib} (cycle best); whether this is desired or not will depend on the size and type of

problem. The use of L_{bs} may cause the search to stagnate, whereas the use of L_{ib} may diversify the search too much due to variability in solutions from different cycles, thus too many arcs will receive reinforcement. In conclusion it is noted that the decision of which to use is dependent on the type and size of the problem being solved, and some mixed strategy may often represent a happy medium.

Secondly, maximum and minimum limits are imposed upon the pheromone trail (τ_{max} and τ_{min}) thus hoping to avoid stagnation. The paper includes a proof that shows that using the given update rule (Equation 2.13), the upper pheromone limit on any arc is bounded above by $1/\rho L_{opt}$, where L_{opt} is the optimal tour length. The MMAS uses an estimate of this based on L_{bs} (as L_{opt} is not known) for the upper bound τ_{max}.

$$\tau_{max} = \frac{1}{\rho L_{bs}} \tag{2.14}$$

It should be noted that each time a new best-so-far tour is constructed the value of τ_{max} is updated. The lower limit is simply set to τ_{max}/a, where a is a user defined parameter (a>1).

Finally, the trail values are initialised to the upper bound τ_{max} (or some estimate of it). Alternative trail initialisation values were considered in Stützle and Hoos (2000), but the computational results supported this selection. The high initial pheromone values coupled with a low evaporation rate will allow the search to be highly explorative in the early cycles (a conclusion that is supported in Section 4.5.2). To further increase the exploration of the algorithm, trails are reinitialised if the search appears to have stagnated or if no improvement has occurred for a predefined number of iterations.

2.4.4 The Ant Colony System (Including the Ant-Q Algorithm)

The focus now switches from the discussion of the two extensions to the original AS algorithm, to one that shares the same inspiration but can be considered as different enough to stand as an algorithm in its own right. Proposed by Dorigo and Gambardella (1997) and Gambardella and Dorigo (1996), Ant Colony System (ACS)

was applied to the TSP and achieved results of a higher standard than the other ACO algorithms.

The simplest way to explain the ACS algorithm is to consider how it differs from the basic AS algorithm:

- Construction Phase

  ACS uses a more aggressive technique to select the next customer to append to a solution. The RPR from AS is replaced by the Pseudo-RPR rule (see Equation 2.15). Using this method, one will select the best customer with probability q_0 (based on the trail and the visibility) and will use the more traditional RPR with probability (1-q_0). It should be noted that α does not appear in the RPR section of the ACS Pseudo-RPR (i.e. α=1).

$$\begin{cases} \arg\max_{l \in F^k} \left((\tau_{il})(\eta_{il})\right) & if\, q \le q_0 \\ J & otherwise \end{cases} \quad \text{wise} \tag{2.15}$$

  where q is a random variable uniformly distributed [0,1], q_0 is a user-defined parameter ($0 \le q_0 \le 1$) and J is the RPR (See Equation 2.4).

- Introduction of Local Updating

  Each time an ant traverses a particular arc (i,j) it removes some of the pheromone that has been deposited to encourage the selection of alternative customer arcs. The local update rule (see Equation 2.16) controls the amount of pheromone removed from a customer arc.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{2.16}$$

  Clearly the effect of this depends on the value chosen for $\Delta\tau_{ij}$, a variety of which are discussed following the description of the final difference between ACS and AS.

- Modified Global Updating

  Instead of all ants (or even the best σ as in AS_{rank}) depositing pheromone after each cycle, only the ant that created the best solution does. This is a very aggressive system, that works in a similar way to the elitism in AS. Furthermore, evaporation only occurs on those arcs that belong to this solution. The amount of pheromone deposited and the evaporation rate can be seen in Equation 2.17.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \frac{\rho}{L_{bs}} \tag{2.17}$$

  Tests were carried out by Dorigo and Gambardella (1997) to ascertain whether this rule operated better in its current form or with L_{bs} replaced by the best solution that cycle (L_{ib}), though results confirmed the use of L_{bs} was appropriate.

With these three differences described, consideration is given to finding a suitable value for $\Delta\tau_{ij}$. Three were analysed and are shown below:

- $\Delta\tau_{ij} = \gamma \cdot \max_{j \in F_i^k}(\tau_{ij})$

  This selection recreated some earlier work (Gambardella and Dorigo (1995) and Dorigo and Gambardella (1996)) known as Ant-Q[13].

- $\Delta\tau_{ij} = \tau_0$

  Where τ_0 is the trail initialisation value.

- $\Delta\tau_{ij} = 0$

These three definitions were tested on the Oliver30 dataset, and a further six datasets. Five were randomly generated 50-city TSPs (see Durbin and Wilshaw (1987)) and one an ATSP dataset, ry48p (see Reinelt (1994)). Results showed that setting $\Delta\tau_{ij} = \tau_0$ produced results of a noticeably higher quality than the other two definitions and was thus selected.

[13] Inspired by the paradigm of Q-learning, see Watkins (1989).

This algorithm is central to the work in this thesis and will be discussed in some detail when one considers the only existing algorithm that has been applied to the problem that forms the basis for Chapters Three through Five.

2.5 Tabu Search

The TS metaheuristic is generally attributed to Glover (1986) and is one of the most actively researched solution techniques for combinatorial optimisation problems[14]. It has reported success across a broad range of problems, and the technique has continued to evolve and improve since its inception.

This section offers an overview of the theory behind the technique, as well as details regarding the evolution of the method (including the various extensions that have been proposed since its inception). Note that the TS was not developed solely on the TSP (as was the case for the ACO metaheuristic), thus the details provided are of a more generalised ilk.

As with the ACO metaheuristic, the sheer volume and diversity of the research prevents one from presenting details regarding all of the published workings. The most applicable to the work contained in this thesis are detailed, while references for further reading will also be provided.

2.5.1 Background Principals

The TS algorithm is a local search (LS) technique that adapts the basic descent algorithm by allowing the acceptance of certain moves, even if they have a detrimental effect on the objective function. These moves, although counter-productive in the short-term, are made in the hope that they will lead to better solutions being identified at a later date.

The TS algorithm makes use of the best improvement (BI) descent strategy and selects the move regardless of whether the cost function is improved. However, it soon becomes clear that if this technique were to be run in this form, one would cycle

[14] Complementing the early works by Glover, were the independent working of another, see Hansen and Jaumard (1990).

between two solutions. To prevent this cyclic behaviour, the TS algorithm introduces a short-term memory function.

2.5.2 Short-Term Memory

The key component of the TS metaheuristic is the short-term memory function that prevents the cyclic behaviour described previously. As one cannot often store an entire solution (it would be computationally expensive) one is required to store some characteristic that enables one to identify a previously visited solution (known as the TC). However, it is often the case that the characteristic stored prevents the search from visiting previously unvisited regions of the search space (note that some authors believe that this improves the quality of the results achieved, see Glover (1989)).

As a result of this, it is necessary to remove the stored characteristics once a number of iterations have passed (ensuring the search has moved far enough away that one does not cycle through solutions). The items are stored in a tabu list (TL) and a user-defined parameter controls its length.

Glover (1977) described the technique, as a *'weak inhibition search'* because the number of moves restricted is generally small. Furthermore, as these moves that have been declared tabu lose their tabu status after a relatively short period of time, any restriction is only temporary. The reason for the terminology *'inhibition search'* comes from the classical B&B technique, which after exploitation declares certain regions are forbidden; in contrast this represents a *'strong inhibition search'*.

Reference should also made to the fact that in stark contrast to many other techniques, there are no elements of randomness involved in the search. This is made possible by two (then unique) elements of the TS algorithm, working in composition. The short-term memory (TL), and restricting the algorithms movement to the best possible (non-inhibited) move each iteration, negated the need to include any randomness. Glover (1989) summarised this new philosophy, stating *'tabu search proceeds according to the supposition that there is no value in choosing a poor move, by accident or design, except for the purpose of avoiding a path already examined'*.

2.5.3 Complementary Memory Structures

Although the details provided previously explain the basic principals behind the TS algorithm, in reality many further extensions have been proposed to increase the quality of the solutions achieved. This subsection (and the next) focus on detailing some of the more frequently included extensions.

The short-term memory will prevent the cyclic behaviour that would cause the search to become trapped in a local optimum, but beyond that, it does not allow one to exhibit any level of control over the search. In response to what could be perceived as a flaw in the technique, complementary memory structures were proposed to intensify or diversify the search, as and when required.

Note that the examples of the techniques relate to the TSP, though it would be possible to adapt the method for other combinatorial optimisation problems.

Intermediate-Term Memory

The basic premise of the intermediate-term memory is to intensify the search on the more promising regions of the search space. An example of an intermediate-term memory can be seen in Glover (1977) (though the moniker intermediate-term memory was not attributed to this technique until Glover (1986)). It operates by *'recording and comparing the features of a selected number of solutions... Features that are common to all or a compelling majority... are taken to be regional attributes of good solutions. The method then seeks new solutions that exhibit these features'*. Glover was clearly aware of the benefits of including such a complementary memory structure and continued to refine the technique in his later works (Glover (1986) and Glover (1989)).

Long-Term Memory

The basic premise of the long-term memory is to diversify the search in the hope of identifying a more promising region of the region of the search space. Glover (1989) introduces a long-term memory as a technique similar to the reverse of the intermediate-term memory. Using the TSP as an example, it is suggested that one stores the number of times each customer arc has been selected, and then penalises

those customer arcs that have been frequently chosen in the past. Further details of this technique are discussed in Chapter Five.

Another example of a long-term memory can be seen in the TS algorithm developed for the CVRP by Gendreau et al. (1994). This algorithm artificially adjusted the change in the objective function (δ) by '*adding a term proportional to the absolute frequency of movement of the vertex v*'. This results in a more even distribution of customer movement, thus diversifying the search.

2.5.4 Further Extensions to the Basic TS Philosophy

As with many of the metaheuristic techniques, it is possible to further improve the performance of a TS algorithm by including some form of extension to the basic techniques. This subsection presents two of the most popular extensions (excluding the complementary memory structures).

Aspiration Levels

These are an important feature of the TS metaheuristic, as they allow one to consider accepting high-quality solutions identified during BI neighbourhood search, regardless of the TL. This is a particularly important feature in TS algorithms where the TC does not uniquely identify a solution (i.e. one is restricting the search from areas of the search-space that have not been visited).

At their most basic (as discussed in Glover (1989)), an aspiration criterion will allow for a tabu move to be implemented if the resulting solution is of a higher quality than any previous solutions identified. However, the technique can be extended to consider accepting moves that are of a high quality and contain elements that differ from previous high quality solutions.

Strategic Oscillation

This extension allows one to relax one (or more) of the problem constraints and make a number of moves into the infeasible region. Upon completion of those moves one reintroduces the constraint and selects moves that bring the solution back into the feasible region. Glover (1991) presents a discussion regarding strategic oscillation for

the CVRP. It is suggested that one allows the capacity constraints of the CVRP to decay (for a number of iterations) before they are gradually restored. Further details regarding strategic oscillation can be seen in Glover (1989).

2.6 Other Metaheuristics of Interest

Although this thesis is primarily concerned with the ACO metaheuristic (and to a lesser extent TS), it is beneficial for one to be aware of some other metaheuristics that exist. This section selects two of the most actively studied techniques (SA and GRASP) and discusses their inception and methodology. The level of detail provided for each of these techniques will be proportional to their relevancy to the work in this thesis.

2.6.1 Simulated Annealing

The SA algorithm was proposed by two independent pieces of research: Kirkpatrick et al. (1984) and Cerny (1985). The theory was developed through a somewhat unusual analogy between solving combinatorial optimisation problems and the restructuring of solids through the annealing process.

The seminal paper with regards to the modelling of the annealing process was by Metropolis et al. (1953) (in the field of statistical mechanics). This paper introduced an algorithm, known as the Metropolis criterion, which (according to Kirkpatrick et al. (1984)) can be used to '*simulate the behaviour of a many-body system at a finite temperature*'. This algorithm can be seen in Equation 2.18, though prior to its discussion, further details of the physical annealing process are provided.

A solid material is referred to being in its 'ground state' if its atoms are arranged in such a way that they form regular patterns i.e. minimum energy state. In order to obtain a material in this state, it must be melted and then cooled in a particular way. As a material is cooled the configuration of its atoms continuously change. In order for this material to eventually reach its ground state, it is necessary to reduce the temperature in stages, and remain at each temperature until the material reaches thermal equilibrium. If the material were to be cooled too quickly, known as quenching, then defects can occur due to the atoms becoming stuck in a particular arrangement.

The algorithm introduced by Metropolis et al. (1953) (Equation 2.18) was developed with a view to simulating the change in the atom's energy, when the material was cooled. A new configuration of atoms is always accepted if the change in energy, denoted δE, is negative. If δE is positive, it is accepted with probability p, where t represents the current temperature and k is a physical constant (known as Boltzmann's Constant).

$$p(\delta E) = \exp\left(\frac{-\delta E}{kt}\right) \tag{2.18}$$

This criterion, although developed for a completely unrelated problem, is the basis for the simulated annealing combinatorial optimisation technique. As the name suggests, the method merely simulates the behaviour of the annealing process, by considering neighbourhood moves as changes in energy (i.e. different atom configurations).

The SA metaheuristic is best explained by considering how it differs from a simple descent algorithm (with the FI descent methodology). Consider the situation where the SA process has current solution x_l, from whose neighbourhood $N(x_l)$ a solution x is randomly drawn. If $f(x) \leq f(x_l)$ then the solution x_{l+1} is set equal to x (as per the descent algorithm). If this is not the case (i.e. δ>0), the following probabilities are used (Equation 2.19).

$$x_{l+1} = \begin{cases} x & \textit{with probability } p_l \\ x_l & \textit{with probability } 1 - p_l \end{cases} \tag{2.19}$$

where p_l makes use of the annealing analogy.

$p_l = \exp\left(\frac{-\delta}{t}\right)$ and t is the current temperature.

The temperature is initialised to t_i, and then is systematically reduced (via either a geometric reduction or some other predefined technique (e.g. Lundy and Mees (1986)) when some criterion is met (often based on iterations). As the temperature is

reduced, the probability of accepting moves that have a negative impact on the solution quality also reduces (though it still depends on the value of δ). This prevents the algorithm from continually selecting moves that increase the cost function, as a local optimum would never be achieved. Once the temperature reaches a suitably low value (t_l) the algorithm terminates.

Note that the SA algorithm usually makes use of an initial solution constructed via some form of simple heuristic, often random or greedy.

2.6.2 Greedy Randomised Adaptive Search Procedure (GRASP)

This metaheuristic was proposed by Feo et al. (1994) (for the set covering problem) as a means of removing the dependency of Local Search (LS) on its starting solution. Although it has many similarities to a multi-start LS algorithm, it differs in the construction, where the starting solutions are generated by a probabilistic greedy construction heuristic.

GRASP consists of a number of repetitions of the construction phase, each followed by an improvement phase. In the construction phase, a solution is built a single step at a time. At each step, the score of adding each possible element to the current (incomplete solution) is calculated and a candidate list of the best n (a user-defined parameter) is constructed. The element to be added is then selected via a roulette wheel selection i.e. each element has a probability proportional to its relative score. In the improvement phase, the constructed solution is descended to a local optimum via some predefined neighbourhood definition.

An alternative candidate list definition was proposed by Hart and Shogan (1987), whereby one considers the elements within a certain percentage (α%) of the best element on the candidate list. This idea was further investigated in Resende et al. (2000), who generated a new value for α randomly from within a predefined range for each cycle. Prais and Ribeiro (2000) proposed Reactive GRASP, whereby one initially selects α at random, but then adjusts the probability according to the quality of the solutions being produced.

Since GRASP was developed, researchers have suggested a number of ways to improve its performance. A popular technique is to incorporate feedback from previous cycles to intensify the search around good quality solutions, or to diversify the search. An example of this can be seen in Fleurent and Glover (1999), where a pool of high quality solutions are stored and further constructions are encouraged to take on the attributes of these solutions.

One can hybridise the improvement phase with a more powerful LS technique such as TS or SA (e.g. Laguna and González-Verlarde (1991)). Fleurent and Glover (1999) try to improve the incomplete solutions by periodically applying LS during the construction phase. One is directed to Pitsoulis and Resende (2001) for a summary of these and other developments on the GRASP metaheuristic.

2.7 CVRP Metaheuristic Implementations

As the basic theory behind the metaheuristics has been presented, consideration can now be given to discussing specific implementations of them. As the CVRP is directly related to the DVRP that is to be considered, it seems sensible to consider how the techniques have been applied to this problem.

This section contains three subsections which consider ACO, TS and other prominent implementations (SA and GRASP) for the CVRP. As with the metaheuristics definitions and background information, the level of detail and quantity of papers considered is proportional to the technique's prominence in this thesis.

2.7.1 ACO for CVRP

In Section 2.4 the ACO metaheuristic and the underlying principal behind the techniques (based on observing real-ants behaviour) were introduced. Given the algorithms development as a technique for solving the TSP (although it has since been applied to many other problems) its application to simple routing problems has already been discussed.

This section builds upon those discussions by considering the CVRP applications and how they differ from the TSP implementations that preceded them. The areas of particular interest are which algorithm was used (e.g. AS or ACS) and if the

implementations deviated from the details of these algorithms provided in Sections 2.4.2 and 2.4.3.

If the algorithms include any problem specific components, these will obviously be discussed, along with any form of local search (i.e. the neighbourhood) that they choose to employ. It should be noted that many of these workings were produced by a select group of authors, and as such many of the implementations are very similar.

A selection of further workings will be provided at the end of this subsection should one be interested in further details about ACO implementations for the CVRP.

Bullnheimer et al. (1999b)

This was the first ACO implementation for the CVRP, and was based on the original AS algorithm (see Section 2.7.1). It borrowed heavily from the TSP workings of Dorigo (1992) and Dorigo et al. (1996) and was primarily concerned with adapting their technique into a suitable algorithm for the CVRP. However, the algorithm did differ from the TSP implementation with regards to a number of components, some of which were required to make it applicable to the CVRP while others were problem specific improvements.

Central to this adaptation is the idea of sequential route construction, whereby one constructs the vehicle routes one at a time until there are no further customers that require scheduling. This enabled them to treat the problem like a series of smaller TSP problems, thus allowing the AS algorithm to be applied in a form similar to its current guise. The algorithm had to be altered to respect the capacity of the vehicles (Q) and the bound on the vehicle route length (L), but all other aspects remained the same.

The algorithm introduced two problem specific enhancements into the RPR as it was felt basing ones decisions purely on the trail and visibility (which was distance based) did not respect the need for the deployment of multiple vehicles. These improvements introduced the C&WS function and a capacity utilisation factor (which encouraged one to fill the vehicles), thus increasing the RPR to four components.

Once each solution has been produced, each individual route in that solution (i.e. intra route moves only) is optimised using a 2-Opt algorithm (*k*-Opt). The results were of quite a high standard (though not as good as those that had been achieved via TS), though a new best-achieved solution was produced for 1 of the 14 benchmark problems on which the algorithm was analysed.

Bullnheimer et al. (1999c)

Soon after the first ACO CVRP implementation, the same authors published a second work entitled '*an improved ant system algorithm for the vehicle routing problem*'. As the name of the work suggests, they were seeking to better their previous algorithm; something that they succeeded in doing.

This implementation made use of the AS_{rank} algorithm (see Section 2.4.3) that the same authors had developed for the TSP (see Bullnheimer et al. (1999a)). As before, the vehicle routes were constructed sequentially and they made use of the 2-Opt algorithm (*k*-Opt) to improve each of the vehicle routes.

This paper abandoned the problem specific improvements and visibility definition from the previous work and instead used a parametrical savings function. This function was developed to respect the need for distance to be minimised, while attempting to prevent the production of overlapping vehicle routes. The algorithm also made use of a Restricted Candidate List (RCL) to prevent the selection of customers that are not close to the current customer. A vehicle is scheduled to return to the depot if all of the customers on the candidate list have already been scheduled.

As before, a best solution was produced for 1 of the 14 benchmark problems. However, the results were significantly better, and when compared to a selection of other metaheuristics, one could see that ACO could be competitive and was a real alternative to TS and other LS techniques.

Reimann et al. (2002)

This paper was an extension to a previous work that considered the incorporation of a problem specific heuristic (C&WS) into an AS algorithm (Doerner et al. (2002)). The

original paper showed that the technique was workable, and this second publication was concerned with improving the quality of the results produced.

An RCL is introduced (thus limiting the selection of customers in the RPR) and unusually, it is based on the visibility and the trail values. The inclusion of the trail, will allow for greater diversification in the solutions produced (as the customers on the RCL can change). Note that this type of RCL is likely to significantly increase the computational effort required as one is required to update the RCL each time the trail values are adjusted.

The trail updating remains the same as that which was used in Bullnheimer et al. (1999c), namely the AS_{rank} technique of Bullnheimer et al. (1999a).

The authors seem to be aware that the algorithm lacked the required intensification to identify very high quality solutions (a common problem with ACO algorithms), thus they decide to expand the improvement phase neighbourhood to include some simple cross vehicle moves. The local search is a two-phase neighbourhood, that first considers swapping customers between routes (1,1 Interchange Process) and then relies on the 2-Opt (k-Opt) neighbourhood to improve each vehicle route (as the previous ACO CVRP implementations did).

Other Prominent Workings and Survey Publications

As stated previously, there were very few authors responsible for the development of ACO algorithms for the CVRP. As a result of this, the work is not as varied as one may hope, but that does not detract from the validity of their work. A selection of these workings are now presented along with the details of a survey publication:

- Prominent Workings
  Reimann et al. (2004)
  Bell and McMullen (2004)

- Survey Publications
  Dorigo and Stützle (2004)

2.7.2 Tabu Search for the CVRP

In Section 2.5 the theory behind the TS algorithm was introduced, and a discussion regarding the history of the method was presented. Although this provided the necessary details for understanding the technique, one needs to consider how it can be applied to routing problems.

This subsection presents a selection of interesting TS CVRP implementations, with particular interest paid to the TC and the neighbourhoods that they use. If any of the extensions (outlined in Sections 2.5.3 and 2.5.4) are included in the technique then these will also be discussed.

Note that given the number of TS implementations that have been made, one could just as easily selected an entirely different set of algorithms to summarise. Details regarding other prominent workings and survey publications are included at the end of this subsection, should one require further details.

Osman (1993)

This working was the first TS CVRP implementation that used a λ-Opt neighbourhood (as it was developed in this paper). This was a relatively basic TS implementation that restricted the neighbourhood to λ=2. This restriction allowed them to use a TC that stored pairs of customers that had swapped vehicles (a null vehicle was used if only a single customer was being relocated).

An interesting aspect of this work was the focus that was given to the descent strategy. Consideration was given to whether the technique should be implemented via a BI (which was referred to as best admissible) or an FI and BI hybrid that allowed FI moves if they reduced the cost but relied on the BI moves if a move was being selected that increased the cost. It was shown that the BI technique was far quicker though the solution quality was marginally worse.

The results produced via this technique are of a high standard (when compared to other metaheuristics), though subsequent (and much more complex) TS implementations have since produced better results.

Gendreau et al. (1994)

The Taburoute algorithm is a considerable step-up in terms of complexity when compared to the algorithm produced by Osman (1993). There were many innovative features in this algorithm, including many of the extensions that were discussed when the TS algorithm was first introduced.

The neighbourhood employed by Taburoute is referred to as GENI (introduced by Gendreau et al. (1992)) and is an adapted (via an RCL) λ=1 type neighbourhood. It considers removing a vertex from its current route and placing it in another route containing one of its p nearest neighbours (where p is a user controlled parameter). The solutions are then processed by a post-optimisation algorithm (known as US), which successively removes and reinserts every vertex using GENI.

The algorithm also allows for strategic oscillation to occur, by penalising the objective function based on overcapacity or exceeding the maximum route constraints (often time based). It contains a self-adjusting parameter that will encourage more feasible solutions to be built if the preceding ten solutions had a high proportion of infeasible solutions.

The TC employed by Taburoute simply prevents a customer from being moved back into a route from which it had previously been removed. This is done via a technique referred to as Tabu Tags, which allows them to choose varying lengths of time (iterations) that particular moves are deemed tabu. For reference, this is not the first example of a dynamic tabu list; the technique was first proposed by Glover and Laguna (1993).

The algorithm also contains a long-term memory to diversify the search by penalising those customers that were being moved frequently. This was done via the introduction of a penalty cost proportional to the frequency of a particular customer's movement when evaluating a neighbourhood move. The results were of a very high standard, and it has produced optimal (or best) solutions for 5 of the 14 benchmarks on which its performance was assessed.

Taillard (1994)

This algorithm contains some of the features of Taburoute, but is distinct enough to be considered as an algorithm in its own right. It uses a similar idea of random tabu durations and the same diversification strategy to penalise those customers that are being relocated frequently.

With regards to the neighbourhood, it uses a simpler technique than Taburoute, abandoning the GENI technique in favour of λ-Opt (with λ=1). The algorithm does not allow for infeasible solutions to be considered (i.e. no strategic oscillation). The US post-optimiser is also abandoned in favour of an exact TSP algorithm (Volgenant and Jonker (1983)), which is implemented periodically (on each individual vehicle).

This algorithm did contain a particularly innovative feature that is worth discussing, namely the fact that during the search, it decomposes the problem into a series of smaller sub-problems. This decomposition is made geographically (for planar problems) and each sub-problem can be solved independently. As the algorithm is run, customers periodically move to adjacent sectors, so as to allow the necessary diversification. This feature is particularly interesting as it lends itself well to the idea of parallel implementation, as each sub-problem can be considered separately.

Of the 14 benchmarks this algorithm produced optimal (or best) solutions for 12 of the benchmark datasets on which its performance was assessed. Note that the times for this algorithm were not presented, and as such, one cannot make a direct comparison with the other techniques. However, given the technique relies on the repeated solving of TSP problems (exactly), one can assume that the technique is not the quickest.

Barbarosoglu and Ozgur (1999)

In Barbarosoglu and Ozgur (1999) a relatively simple tabu search algorithm was developed to solve a real-life CVRP faced by a distribution company in Turkey. It made use of the λ-Opt type neighbourhood, though restricted it to moves between two

vehicles (i.e. no intra vehicle moves were considered)[15]. As a result of this restriction on the neighbourhood, they were able to use a very simple TC; it prevented the return of customers to the vehicles that they had previously been removed from (it was only tabu if customers were returning to both of the vehicles involved in the neighbourhood move).

The algorithm did not allow for any form of strategic oscillation to occur, though it did incorporate a type of RCL to encourage one to consider moves that favoured vertices far from the centroid of their current route and close to the centroid of another. An intermediate-type memory structure was also included to intensify the search, though no diversification strategies were included.

The results of this work seem to be of a high standard, particularly considering the relatively simple nature of this algorithm.

Other Prominent Workings and Survey Publications

As explained previously, interest in the TS metaheuristic is a particularly active area of research and the number of CVRP implementations is too numerous to present details of them all. References for a further selection of prominent implementations (excluding those that were presented in greater detail above) and a more general survey publication are now presented.

- Prominent Workings
  Xu and Kelly (1996)
  Rego and Reucairol (1996)
  Rochat and Taillard (1995)
  Toth and Vigo (1998)

- Survey Publications
  Gendreau et al. (2002)

[15] It did optimise the individual routes using a 2-Opt algorithm though this was simply in a descent algorithm and not subject to the TS methodology.

2.7.3 Other Prominent Metaheuristic Implementations for the CVRP

This section considers CVRP implementations for the techniques that were presented in Section 2.6. These details are provided for completeness, as they have little relevance to the work conducted in this thesis.

SA for the CVRP

Osman (1993) made use of an initial solution that was constructed using the savings procedure of Clarke and Wright (1962), and subsequently improved via the two neighbourhood moves attributed to Osman in Section 2.2.2 (the (1,0) Shift Process and the (1,1) Interchange Process). Information is gathered in an initial test phase (i.e. moves are not accepted) to establish suitable values for t_i and t_f. This SA differs from more traditional implementations as the neighbourhood is searched sequentially rather than at random.

Van Breedham (1995) considered a wider neighbourhood, where strings of consecutive customers were moved both within routes and between them. The parameters were established through a traditional parameter optimisation. The findings of Van Breedham (1995) agreed with Osman (1993), with both papers suggesting that TS outperforms SA with regards to both solution quality and computational time.

GRASP for the CVRP

Baker and Carreto (2003) implemented the GRASP metaheuristic within a graphical user interface to solve the CVRP. The GRASP construction phase prioritises customers that can only be placed into a single route (feasibly), and then adds customers who can only be placed into two routes into the candidate list. The next customer added to the incomplete solution is then selected probabilistically, according to their demand. The improvement phase uses the 3-Opt moves of Lin (1965) (see Section 2.2) for intra vehicle moves and the GENI procedure of Gendreau et al. (1994) for cross vehicle moves.

2.8 VRPTW Metaheuristic Implementations

In much the same way as various implementations of the metaheuristics were discussed for the CVRP, consideration needs to be given to discussing specific

VRPTW implementations. The VRPTW is directly related to the DVRPTW that is to be considered in Chapter Six, thus one needs to be aware of how the various techniques have been applied to this problem.

This section contains three subsections with consider ACO, TS and other prominent implementations (SA and GRASP) for the VRPTW. As with the metaheuristics definitions, background information and CVRP implementations, the level of detail and quantity of papers considered is proportional to the technique's prominence in this thesis.

2.8.1 ACO for the VRPTW

This subsection considers the most interesting ACO implementations for the VRPTW. The areas of particular interest are (as before) the type of ACO algorithm applied (and how it has been applied to this particular problem) and whether there are any particular components that are problem specific.

As the amount of research in this area is fairly minimal, this discussion is to focus primarily on a single publication. This particular working (the first considered) is one of the best performing algorithms for the VRPTW and it was heavily responsible for the development of the algorithm in Section 2.9.5 that is central to the workings of this thesis. A selection of various other implementations will be noted, but with regards to this thesis they offer little information of benefit.

At the time of writing, there does not appear to be any survey publications on this topic, thus one is restricted to presenting a selection of prominent workings for further reading.

Gambardella et al. (1999)

This is the key paper with regards to ACO implementations on the VRPTW. Although it was produced eight years ago, it remains one of the most competitive algorithms of any type for solving this problem. Only recently have the workings of Bräysy (2003) (non-ACO) managed to produce results of a competitive standard.

The MACS-VRPTW was based on the ACS algorithm of Gambardella and Dorigo (1996) and Dorigo and Gambardella (1997) and introduces a number of innovative components to make the algorithm applicable to the VRPTW. A number of these components are now detailed and accompanied by an explanation as to why they were included.

The M in the algorithms name stands for multiple and this is how the algorithm deals with the dual objective function (minimising the number of vehicles deployed and the distance they travel). The algorithm has two colonies that run concurrently.

The first colony (ACS-VEI) has an objective of minimising vehicles (i.e. it tries to build in one vehicle less than the lowest number of vehicles used so far). The objective function in this colony is the number of customers that were scheduled (i.e. minimising the number of unscheduled).

The second colony (ACS-TIME) is more traditional (with regards to the other ACO routing problem implementations) and simply minimises the distance of the route (though subject to the solution being produced with the number of vehicles from the best solution from ACS-VEI). If a new solution is identified in ACS-VEI, then ACS-TIME is reinitialised to account for this reduction in the number of vehicles required.

The two colonies have independent pheromone trails, though there is a certain amount of interaction with the update rules of ACS-VEI also including information from the best solution from ACS-TIME.

The two colonies share the same visibility, and it was designed specifically for the VRPTW (it is considered in great detail in Section 6.3.3). It accounts for the time window and increases the visibility of those customers whose time window is to close shortly (with respect to the vehicles current schedule). The visibility also includes a diversification component that identifies the frequency with which customer arcs have been chosen, and encourages the selection of those that have been chosen infrequently. The ACS algorithm was noted for its aggressiveness in Section 2.4.3 and this is likely to have been included to counteract this.

The two colonies both include some form of route improvement, though these differ considerably. ACS-VEI uses a route insertion technique that tentatively tries to increase the number of customers scheduled by considering each of the unscheduled customers in decreasing order of their demand. ACS-TIME uses the CROSS Neighbourhood of Taillard et al. (1997), though no mention is made as to whether any restriction (λ) is placed upon this.

The algorithm identified 13 new best solutions for the 56 benchmarks (3 were improved by reducing the number of vehicles deployed) while it often produced results equal to the best-achieved for the other datasets. The algorithm was also adapted for the CVRP and it identified 4 new best-achieved solutions (though it does not mention how many datasets were assessed).

Barán and Schaerer (2003)

This work was proposed as an extension to that of Gambardella et al. (1999) and was concerned with the use of a single colony (ACS) to solve the multi-objective VRPTW. It attempts to build a set of good solutions, where no single objective (they consider three: vehicles deployed, travel time and delivery time) has precedence. One could then choose from the pool of solutions which objective they wish to give precedence to. It should be noted that at no stage does the algorithm allow for the building of infeasible solutions in the way in which the MACS-VRPTW does.

Ellabib et al. (2002)

This is another single colony attempt at solving the VRPTW using ACS. This paper emphasises that it is working on a simple ACS (presumably to offer a contrast to the complex MACS-VRPTW).

It relies on an initial solution produced via a heuristic method (three are assessed) and then converts this into a TSP-like solution by creating numerous duplicates of the depot with distance between them set to zero (as seen in MACS-VRPTW). The ACS then attempts to improve upon the solution and minimise the number of duplicates of the depot required (i.e. the number of vehicles). Three visibility definitions are

assessed and the paper identifies which of these (along with the initial construction heuristic) is likely to perform best.

Although this paper contains some interesting ideas, they are not particularly applicable to the work in this thesis, thus no further details are presented.

Other Prominent Workings (No Known Survey Publications)

There are not a great number of other ACO VRPTW implementations that one can be directed towards for further reading. Two papers that cover similar issues are now presented (though they are not as relevant as those discussed previously).

Chen and Ting (2005) have produced what appears to be an interesting paper that combines the local search phase of ACS with SA (though the ACS component does not introduce any ideas that have not already been considered). Reimann et al. (2002) use an adapted AS to solve the VRP with Backhauls and Time Windows, though it could be interesting to see this adapted into an algorithm specifically for the VRPTW (as there does not appear to exist an AS implementation).

2.8.2 Tabu Search for the VRPTW

This subsection presents a selection of the most interesting TS VRPTW implementations, and again consideration is given to the TC and the neighbourhoods that were chosen. One is required to be familiar with the basic tabu methodology (Section 2.5) and the TS CVRP implementations (Section 2.7.2) to understand the various ideas that are used to describe these algorithms.

It should be noted that research into TS for the VRPTW is not in the same advanced state as the CVRP, and as such, there are far fewer papers that require discussing. Furthermore, one should be aware that the TS algorithm is not applied to the VRPTW (or a dynamic variant of it) within this thesis. Any details presented here are either used within a different type of algorithm (specifically the CROSS neighbourhood) or are included for completeness.

Taillard et al. (1997)

The TS implementation for the VRPTW by Taillard et al. (1997) is fairly unusual in the fact that it tackles the soft time window variant of the problem. It was mentioned previously, how work on this problem is far less frequent (than the hard window variant) and this is particularly interesting, given the fact that this paper is by far the most prominent TS implementation for the VRPTW. It should be noted that the authors do state that one would be able to adapt the algorithm developed into a suitable algorithm for the VRPTW (hard window variant) through the addition of large penalty values.

The algorithm is very much an amalgamation of a number of CVRP works that preceded it. It uses the decomposition technique of Taillard (1994) (which was discussed in Section 2.7.2) and adaptive memory of Rochat and Taillard (1995). This memory (which is an intensification technique i.e. long-term) retains a pool of high quality routes from the best quality solutions and when the algorithm is restarted, a new solution is produced through a randomised selection process of the routes that have been stored.

The neighbourhood that the algorithm uses was introduced in Section 2.2.3, when one was considering the neighbourhoods that had been developed for the VRPTW. By restricting the λ-Opt workings of Osman (1993) to consecutive customers, the CROSS Neighbourhood was developed. The TC was based on the cost of the solutions produced and required the length of the TL to be set to a high value to prevent cyclic behaviour.

The algorithm employs a post optimisation algorithm heavily based on US (Gendreau et al. (1994)). This adaptation was developed for the TSPTW by Gendreau et al. (1998), and so was easily portable to the VRPTW (in the way that US was developed for the TSP by Gendreau et al. (1992) and transferred to the CVRP).

This algorithm found new best solutions for 9 of the 29 benchmark datasets, while equalling the best produced for a further 12. Interestingly its performance was worst

on the random datasets (i.e. it behaved better on the clustered datasets) which may have been due to the geographical decomposition technique employed.

Potvin et al. (1996)

In contrast to the algorithm of Taillard et al. (1997) this TS implementation is fairly simple, and does not rely on anything other than the basic tabu methodology. It is considered here, as it was one of the first TS VRPTW (hard window variant) implementations.

Their central concern appears to be the need for the neighbourhood to maintain orientation (as in the CROSS Neighbourhood). They discuss how a neighbourhood that results in the reversal of route sections is not particularly well suited to the time window variant of the problem. In order to prevent this from occurring, the selection of the neighbourhood was restricted to the Cross Vehicle 2-Opt and Or-Opt neighbourhood that were discussed in Section 2.2.3.

The careful selection of the neighbourhood also allowed them to select a relatively simple TC that remembered customer pairs (i.e. arcs) that had been removed and did not allow from them to be reintroduced. The algorithm also contained an aspiration criterion, with them stating '*a tabu move can be overridden if an overall best solution is produced by doing so*'.

The paper discussed the potential use of strategic oscillation, but they claim that given the complexity of the VRPTW one may struggle to get back to a feasible solution if one were to consider the acceptance of infeasible moves.

Cordeau et al. (2000)

This paper considers the VRPTW along with some less studied variants (the Periodic VRPTW and Multi Depot VRPTW). Although the work on these other problems is interesting and entirely valid research, it is not particularly applicable to the work in this thesis, thus the details provided in this summary are restricted to their work on the VRPTW.

The algorithm employs a very restricted neighbourhood based on λ-Opt (cross vehicle) moves, but with λ=1 i.e. an individual customer is removed and reinserted into another route. However, to improve the quality of the neighbourhood, all possible placements for the customer being moved are considered and the best is chosen according to a least-cost insertion technique.

The work does not explicitly define its TC, though with the neighbourhood restricted to λ=1 it seems sensible to assume that they prevent a customer from being moved back into a vehicle from which it has recently been removed.

The algorithm applies two forms of strategic oscillation, firstly in the initial construction and then later in the search. The specifics of these techniques are too complex to detail here, though one should be aware that the algorithm contains a number of self-adjusting parameters that guide the search towards feasibility (separate parameters for capacity and time) should one venture too far away from the feasible region. The algorithm also contains a long-term memory that encourages diversification should it be required.

The algorithm finishes by attempting to improve each of the individual routes belonging to the final solution through the use of the same post-optimiser (adapted from the US algorithm, Gendreau et al. (1992)) as Taillard et al. (1997).

Other Prominent Workings and Survey Publications

As the VRPTW has not received anything like the amount of attention that the CVRP has with respect to the TS metaheuristic, there are not a great number of other publications that need considering. If one were to be particularly interested in finding out more information one is directed to the following publications.

- Prominent Workings
  Carlton (1995)
  Brandão (1998)

- Survey Publications[16]
  Cordeau et al. (2002)

2.8.3 Other Prominent Metaheuristic Implementations for the VRPTW

This section considers VRPTW implementations for the techniques that were presented in Section 2.6. These details are provided for completeness, as they have little relevance to the work conducted in this thesis.

SA for the VRPTW

Czech and Czarnas (2002) implemented SA on the VRPTW using parallel processing. Each processor performed a number of iterations at a specific temperature, with the best solution being shared across all of the processors. The cost function is a combination of the number of vehicles deployed, distance and the number of customers in the shortest route. The parameter t_i was a factor of the cost of the initial solution, while a geometric cooling schedule was utilised. The search terminated if a user-defined number of reductions in temperature did not lead to any improvement in solution quality. A further SA for the VRPTW (hybridised with TS) can be seen in Li and Lim (2003) where solutions are constructed via the sequential insertion heuristic of Solomon (1987) and then these were improved via the hybrid LS.

GRASP for the VRPTW

Kontoravdis and Bard (1995) use the construction phase of the GRASP metaheuristic to construct an initial solution that is then improved via a 2-Opt neighbourhood (that attempts to remove vehicle routes). They initialise the number of vehicles deployed to a given lower bound, and then allocate customers to these routes according to distance. Infeasible allocations are not considered and if a solution cannot be constructed in the given number of vehicles, a further vehicle is deployed. The LS phase is only implemented on the best solution every five cycles.

Chaovalitwongse et al. (2003) use a simple GRASP algorithm, but introduce a new neighbourhood for the local search phase. They identify the three best moves that can be made and selected randomly from them. Although these moves may worsen

[16] This is more a summary than an actual survey due to the lack of publications that have been produced thus far.

solution quality, they enable a wider search of the solution space to be conducted. A further GRASP for VRPTW implementation can be seen in Li et al. (2004) where TS is incorporated into the LS phase.

2.9 The Dynamic Vehicle Routing Problem

This is a relatively new class of problem, and as such, one cannot discuss it in the way in which other VRP variants were (in Section 2.2) i.e. through the techniques that have been developed for solving it. Furthermore, the term DVRP does not represent a single problem (in the way CVRP or VRPTW does), it encompasses multiple problems, and even identifying whether a problem belongs to this group is a complex task.

Before consideration is given to which problems can be classified as DVRPs, it seems sensible to consider the reason for their inception (and their continued research). In a similar vein to the development of the CVRP and VRPTW, one can argue from a real-life or mathematical perspective, though the reality is likely to be a combination of both of these factors.

Much of the early work on DVRPs took the form of case studies, and as such, one can see that DVRPs are apparent in real-life and there is some merit associated with the production of solution techniques. However, the scope for further variants within the traditional VRP model was quickly diminishing, and the research community may well have been eager to explore more complex environments within which to test their abilities.

At its most basic level, dynamicity introduces some form of uncertainty that must be accounted for when solutions to the problems are being created. However, only certain forms of uncertainty result in a problem being referred to as dynamic. The next subsection contains a formal definition of what constitutes a dynamic problem and discusses some problems and whether they adhere to this definition.

Once one is familiar with the types of problems (and importantly the idea regarding how to solve the problems) that one would classify as dynamic, the problem that this thesis is concerned with is introduced. This discussion relates to the problems

inception and solution methodology, rather than discussing a particular algorithm that has been used to solve the problem. These details will be discussed in Section 2.9.5 when the ACS-DVRP is introduced.

2.9.1 Problem Categorisation

One of the difficulties associated with dynamic routing problems is that when they were first developed, there was a certain amount of ambiguity regarding what actually constituted a dynamic problem. People often thought that the terms uncertainty and dynamic were synonymous (within the context of routing problems), and this lead to many works being incorrectly referred to as dynamic (when using the now accepted definitions).

Much of the early work into problems containing uncertainty took the form of very different real-life examples that people were faced with. In response to this somewhat varied body of work, Psaraftis (1995) produced a summary paper; collating the existing work and proposing a formal definition of the term DVRP.

Psaraftis (1995) argued that the presence of uncertainty does not mean that one is automatically faced with a dynamic problem, stating '*a vehicle routing problem is dynamic if information on the problem is made known to the decision maker or is updated concurrently with the determination of the set of routes... if all inputs are received before the determination of the routes... the problem is termed static*'.

This statement highlights the two different types of routing problem containing uncertainty (static and dynamic), and how they differ with regards to the information that is known. To further distinguish between the two types of routing problem containing uncertainty, Psaraftis (1995) presented definitions relating to the way in which the two types of problem are solved:

- Static Problem

  'the output of a certain formulation is a set of preplannned routes that are not reoptimized and are computed from inputs that do not evolve in real-time'

- Dynamic Problem

  'the output is not a set of routes, but rather a policy that prescribes how the routes should evolve as a function of those inputs that evolve in real time'

Using these definitions, one is able to consider a routing problem that contains uncertainty, and ascertain whether it is dynamic based on either the information known or how one would approach solving it.

Using a similar approach to Psaraftis (1995), it is useful to see how a problem that might appear dynamic (to one who is not familiar with the definitions provided by Psaraftis (1995)) can be shown to be static. Following this, a discussion of the most popular dynamic problem components (with details for further reading) will be presented.

2.9.2 Problems that One Might Incorrectly Assume to be Dynamic

Before consideration is given to the dynamic problems, it is important that one is able to distinguish between those problems that are dynamic and those that even if they appear to be, are not. This subsection presents two routing problems that contain uncertainty, but do not adhere to the strict definition of a dynamic problem given by Psaraftis (1995).

These definitions will assume knowledge of the TSP, VRP etc. and will only discuss the difference between the classic problem definition and the problem that was considered i.e. focusing on the uncertainty.

- The Time-Dependent VRP (TDVRP)

  The travel times between customers (i.e. the cost of particular arcs) differ according to the time of the day i.e. taking note of traffic intensity. However, the way in which they differ is known in advance, and as such, one can calculate an entire schedule a priori accounting for these variable conditions. One is directed to Malandraki and Daskin (1992) for further reading.

- The Probabilistic TSP (PTSP)

  The demand at each customer occurs with a known probability p (or does not occur with probability 1-p). The problem requires one to compute a schedule a priori; once this is completed one is made aware of which customers require servicing. Those that need not be visited are simply skipped, though the remainder of the route is not adjusted to compensate for this. One is directed to Jaillet (1985) for further reading.

With the idea that one cannot classify a problem as dynamic if one is able to produce the route a priori, consideration can be given to problems that one would classify as dynamic.

2.9.3 Problems that are Dynamic

Given the recent inception of this class of problem, there is a definite lack of structure to the work that has been conducted in this area. Many of the authors have proposed their own name for the dynamic problem that they were interested in solving, as the exact same problem had yet to be tackled. However, although in the strictest sense of the word they are new problems (as they all contain subtle differences) one can group some together as they contain similar dynamic components.

Although they share a basic premise of being routing problems with uncertainty apparent in a particular component, there are differences between the workings that have been classified in the same sub-group. To solve these problems, one would need a greater amount of detail than can be provided here, and it is suggested that one seeks the original papers should they require further details.

Dynamic Customers

These problems have a number of customers that are not known to the dispatcher when the routes are initially constructed. They become known as the vehicle(s) traverse the route(s) and must be incorporated into the dynamic schedule. Alterations can be made either periodically or in response to a change in the customers (this will be discussed in greater depth in Section 2.9.2).

Problems of this type have been given many different names, a selection of which (and the corresponding publications) are presented below:

- Real-Time Vehicle Routing and Dispatching in Gendreau et al. (1999)
- Real-Time Vehicle Dispatching in Ichoua et al. (2000)
- Dynamic TSP in Guntsch and Middendorf (2001)

Dynamic Demand

These problems have customer demand levels that are not known to the dispatcher when the routes are constructed. They only become known as the vehicle(s) arrive at the particular customer(s). Obviously, this can result in subsequent customer demands being unable to be fulfilled by the vehicle that was originally scheduled to service them. If this occurs, one is required to make a change to the proposed schedule in order to make the solution feasible.

Problems of this type have been given many different names a selection of which (and the corresponding publications) are presented below:

- The Dynamic TSP[17] in Psaraftis (1988)
- The Stochastic VRP in Dror et al. (1989) and in Yang et al. (2000)

Dynamic Time

These problems have either a service time or a travel time that is not known to the dispatcher until route execution. As one cannot rely on a schedule that has been produced a priori, one is required to update the schedule so as to maintain feasibility. Although this is currently one of the lesser-studied DVRPs, it is a problem that is likely to be prevalent in real-life situations and thus should not be ignored.

Problems of this type are yet to receive a significant amount of attention, though the travel time variant (which recalculates solutions based on current traffic conditions) is referred to as the Real-Time Dynamic VRP in Qili Zhu and Ong (2000).

[17] Although this problem has the same name as one of the dynamic customer problems, they are very different. This highlights the difficultly one is faced with when trying to introduce these problems.

Composite Dynamic Problems

It is also possible to consider a problem with more than one dynamic component, though the level of research into these is even less established. One example has been identified, whereby the schedule has to adapt to dynamic customers and dynamic service times. Bent and Van Hentenryck (2002) have given this problem the name Dynamic Vehicle Routing with Stochastic Requests.

It is sensible to assume once the level of research on the singularly dynamic problems has increased, problems such as this will receive a greater level of attention. It should be noted that although one may experience an increase in computational effort (when compared to a singularly dynamic problem) it would not be on the same scale as was experienced when comparing a static and singularly dynamic problem.

2.9.4 The Dynamic Vehicle Routing Problem (with Dynamic Customers)

Of the dynamic problems identified in the previous subsection, it is the dynamic vehicle routing problem (with dynamic customers) that is of particular interest to this thesis. It is therefore necessary to present some further details regarding this problem, so that one is aware of the work that has already been conducted and how one goes about producing solutions.

The benchmark datasets for the problem are described (including details regarding their creation), along with details regarding two potential solution strategies. After a preference has been stated, some further details necessary to solve the problem (and to allow a direct comparison with an existing piece of work) are presented. It should be noted that for the remainder of this thesis, the term DVRP is assumed to represent the DVRP (with dynamic customers).

Benchmark Datasets

As with any combinatorial optimisation problem, it is necessary to have datasets upon which ideas can be tested. For well-researched problems, the benchmark datasets will have had many analyses conducted upon them. Optimal or best-achieved solutions will be known, and new results can be compared with these. When Kilby et al. (1998) approached the DVRP, no such datasets existed, thus they proposed some based on popular VRP benchmarks:

- 7 datasets from Christofides and Beasley (1984)
- 2 datasets from Fisher and Jaikumar (1981)
- 12 datasets from Taillard (1994)

Using these as starting points, these datasets were transformed into DVRP datasets through the addition of three key details. The first relates to the dataset in general, the final two to each specific customer:

- The datasets required the concept of the working day (T), the time when all the customers must have been serviced and the vehicles returned to the depot. This was arbitrarily chosen to be eight times the maximum distance between two customers.

- Each customer required an available time, representing the time in which the customer first enters the system. These times were constructed using the uniform distribution, with its bounds based on T.

- Each customer also required a service time (how long the vehicle must remain at a customer in order to complete the service). These were chosen such that approximately the same time was spent travelling as servicing. This detail was required to prevent one from leaving the whole days work until late in the afternoon.

It is important to note that once the working day value was chosen, the other two details were chosen to complement it.

Two Possible Solution Techniques

The dynamic element of the DVRP comes from the arrival of new customers throughout the course of the working day. These new arrivals have to be incorporated into the existing schedule through the modification of the current routes (including the deployment of extra vehicles if required).

At present, two solution techniques have been used when solving the DVRP. Both allow the problem to be represented as a series of static CVRP-like problems, though with slight modifications as to current vehicle locations and residual capacities (the customers already serviced are also noted). The techniques differ in when they decide to create a new static CVRP-like problem. Note that once a new problem is created, a certain proportion of the previous solution becomes committed to (further details regarding this will be presented following this discussion).

- Reaction Method (Gendreau et al. (1999))

  This technique immediately creates a new problem each time a new customer arrives (and the next customer to be serviced by each vehicle becomes committed to). The benefits of this technique are its simplicity and the immediate incorporation of new customers. Problems could occur if a new customer arrives soon after another, with the optimiser not having the necessary time to find a good solution (to the previous static CVRP-like problem). This could result in a poor solution being used to decide which customers ought to be committed to.

- Timeslot Method (Kilby et al. (1998))

  This technique divides the working day into a predetermined number of timeslots (n_{ts}). A new problem is created when a timeslot has finished, regardless of the arrival of new customers (schedules are committed to for the duration of the next timeslot[18]). The benefits of this method include control over the time dedicated to each static CVRP-like problem (which ensures there is sufficient time to obtain a good solution). Problems could occur with the enforced delay in incorporating the recently arrived customers. Vehicles may be scheduled to service existing customers, regardless of whether a recently arrived customer is better placed.

As both of these techniques have their faults, and as there is no direct comparison between them, one is perfectly entitled to select either method to pursue. The most

[18] A further period (known as the commitment horizon) of the existing schedule also becomes committed to. This is discussed under the heading Further Details (within this subsection).

likely key factors to consider when making this decision are personal preference and a wish to have a work with which to make comparisons with.

It is for both of these reasons that the timeslot solution technique of Kilby et al. (1998) has been chosen for use in this thesis. This method allows one to treat the problem in a more static-like manner, which it is hoped, will help in establishing the DVRP as a more popular research problem.

If one uses the same benchmark problems and solution strategy as Montemanni et al. (2005), one will be able to directly compare the results produced. This will allow one to treat their results like the best-achieved results in a traditional combinatorial optimisation problem (i.e. one can attempt to better their results). Further benefits of the timeslot solution strategy will become clear as the work in the later chapters makes use of the technique.

Further Details

With the benchmark datasets and solution strategy described, almost all of the details necessary to produce solutions for the DVRP have been provided. There remain two key details (introduced by Kilby et al. (1998)) that one must be familiar with before producing solutions:

- Degree of Dynamicity (DOD)

  The DOD describes how dynamic the problem being solved is. Prior to solving the problem, a cut-off time is chosen (T_{co}). All customers with an available time after this, are treated as if they arrived yesterday (i.e. can be scheduled in the first timeslot, denoted ts_0).

- Commitment Horizon (CH)

  The CH relates to an extra period in the schedule that is fixed, beyond the end of the next timeslot. This is included to increase the realism by allowing extra time for communication between schedulers, drivers and customers before a particular customer is committed to.

With these definitions in place, Kilby et al. (1998) used a simple insert and improve method (based on the workings of Caseau and Larburthe (1999)) to produce solutions (note that pick-ups and deliveries were considered separately). The focus of their discussion was on the impact of various DOD and CH. Results were purely in graphic form and highlighted the increase in cost as either the DOD or CH increase.

Following the workings of Kilby et al. (1998), Montemanni et al. (2005) acknowledged that these benchmarks were indeed a suitable measure of performance, but that the DOD and CH should be fixed for their workings. They suggested a DOD of 50% (T_{co}=0.5), though based it on time rather than number of customers (due to the uniform distribution of customers this makes little difference) and a CH of 1% (T_{ac}=0.01). Note that only deliveries were considered, as is the case in this thesis.

As explained previously, it is beneficial to have results with which to compare the performance of the algorithms in this thesis, thus the parameter settings of Montemanni et al. (2005) (for T_{co} and T_{ac}) are embedded into the problem definition.

To formalise the timeslot solution strategy and the impact that these parameters have on the problem, pseudo-code detailing how the static CVRP-like problems are created and can be seen in Figure 2.5 (where T_{ts} is the duration of a timeslot).

```
Time = 0
Orders = customers known from day before (i.e. arrival time>T_co)
Do While        Orders ≠∅ or Time<T_co
                Create static CVRP-like problem with Orders customers
                Create solution for static CVRP-like problem
                Commit to customers with processing time ≤Time+T_ts+T_ac
                Orders = Orders–customers committed to
                Orders = Orders+customers appeared in last T_ts seconds
                Time = Time+T_ts
                Update starting positions, residual capacities and travel times
EndWhile
Commit all vehicles to the depot
```

Figure 2.5: Pseudo-code Representation of this DVRP

An example detailing the first few timeslots is also presented to help formalise the solution procedure:

- Customers that have arrival times $>T_{co}$ are assumed to have arrived yesterday, and form the set of customers n_0. A solution to this problem is then identified during ts_0.

- Customers scheduled to be serviced between T=0 and $T=T_{ts}+T_{ac}$ are committed to and the remainder of the timeslot (t_1) is spent trying to improve the route identified in t_0 (for those customers not yet committed to (n_1)). Any customers that arrive during t_0 are temporarily ignored until one reaches $T=T_{ts}$.

- When one reaches $T=T_{ts}$, one commits to customers that are scheduled to be serviced between $T_{ts}+T_{ac}$ and $(2xT_{ts})+T_{ac}$. The customers that arrived during t_1 are now considered along with any uncommitted customers in the new problem n_2.

- This process continues until one reaches $T=T_{co}$, after which no new customers arrive. Subsequent problems consist of just the uncommitted customers (which will reduce each time a timeslot is completed).

With the necessary details regarding producing solutions in place, consideration is now given to the existing algorithm, which has been used to create solutions to the static CVRP-like problems, the ACS-DVRP.

2.9.5 The ACS-DVRP

This is the final subsection of this literature review. It details the only existing algorithm that has been used to solve the exact problem (including the DOD and CH) that this thesis is concerned with (in Chapters Three through Five). Although this is a complex algorithm, its description is made somewhat simpler by the fact that the basis of the solution technique (the ACS algorithm) has already been presented in such detail.

This description details the algorithm by discussing the various components and if they differ from the traditional ACS algorithm. Although Montemanni et al. (2005) acknowledge that their algorithm is heavily based on the workings of Gambardella et al. (1999), as many of the details of this algorithm have already been provided, one can focus entirely on the ACS-DVRP. One algorithm specific detail regarding the problem transition is also presented.

This subsection concludes with some general comments about the ACS-DVRP and the literature review in general.

Solving a Static CVRP-like Problem with the ACS-DVRP

The algorithm used to create (and improve) the solutions does not differ between timeslots, thus one can describe it for a general static CVRP-like problem. It should be noted that one must remember that the vehicle locations and residual capacities will be updated prior to considering a new static CVRP-like problem.

As the algorithm is very similar to the classic ACS algorithm introduced in Section 2.4.4, one can explain the ACS-DVRP by considering the various components and if they differ from the traditional ACS algorithm:

- Construction

  No change to the construction technique i.e. it still uses the Pseudo-RPR. Note that a vehicle's tentative schedule is deemed complete when the depot is selected (it is included in the set F^k and may be selected by the Pseudo-RPR).

- Local Updating

  No change to the local update rule. This ACS algorithm uses $\Delta\tau_{ij} = \tau_0$ (the trail initialisation value), which has seemingly become the standard definition (in much the way Ant-Cycle has become favoured for the AS/EAS).

- Global Updating

  No change to the global update rule.

Once each ant has constructed a solution, a simple LS algorithm is used to try to improve the quality of the solution constructed. This algorithm is a very simple FI 1-Opt type neighbourhood that allows a single customer to be repositioned either within its existing route or another (i.e. intra and cross vehicle moves). Somewhat unusually, the exit criterion for the improvement phase is related to the exit criterion of the algorithm as a whole, and thus cannot be discussed independently.

In a rather unusual step, Montemanni et al. (2005) decided to base the exit criterion for the ACS-DVRP on time, with the algorithm run for 25 minutes (evenly distributed across the 25 timeslots). The LS component of the algorithm is run for 10 seconds each timeslot.

Parameters: n_{ts}=25, q_0=0.9, β=1, ρ=0.1, m=3 and $\tau_0 = 1/n \times Cost(PI)$, where *Cost(PI)* is the cost of a solution obtained by a greedy heuristic algorithm.

Static CVRP-like Problem Transition

In an interesting development, Montemanni et al. (2005) suggest that one ought to consider conserving some element of the information learnt from one static CVRP-like problem, for use in future timeslots. This has been proposed as there are obvious similarities between successive static CVRP-like problems.

The values in the pheromone matrix (τ_{ij}) are intended to reflect the quality of the solutions when customers i and j have been visited in sequence. By transferring some of this information (from timeslot to timeslot) one can retain the information learnt about those customers that are present in both static CVRP-like problems.

The pheromone transfer strategy is based on the workings of Guntsch and Middendorf (2001) and Guntsch and Middendorf (2002). A new parameter γ_r is introduced to regulate the pheromone transfer rate from timeslot to timeslot. For each pair of customers that appear in both static CVRP-like problems, the corresponding pheromone matrix entry is initialised to the values seen in Equation 2.20.

$$\tau_{ij} = (1-\gamma_r)\tau_{ij}^{old} + \gamma_r \tau_0 \qquad (2.20)$$

where τ_{ij}^{old} is the value of τ_{ij} in the previous static CVRP-like problem (a value of γ_r=0.3 was identified via a parameter review).

Entries in the pheromone matrix corresponding to customer pairs where one (or both) of the customers arrived during the previous timeslot are initialised to τ_0. The ability to transfer knowledge in this manner may well have been partially responsible for the selection of an ACO technique (few other metaheuristics would have lent themselves to this knowledge transfer).

Closing Comments

With the details of the ACS-DVRP presented, the literature review is effectively completed. One is required to be familiar with the details of the ACS-DVRP to understand much of the work in this thesis, though many further details will be presented in a greater depth throughout the early chapters of this thesis.

Note that a summary has not been included in this chapter, as it is recommended that one refer back to the appropriate section if required.

Chapter Three

Establishing a Base for Further Research

3.1 Introduction

The DVRP is a very recent problem, and as such, the current level of research can only be described as being in its infancy. Comparing this with a more established problem like the CVRP, one is unable to call upon a vast catalogue of research upon which further studies can be based. The only existing analysis of the DVRP (using the timeslot technique) comes in the form of the previously described ACS-DVRP algorithm introduced by Montemanni et al. (2005).

Chapter Two highlighted that this algorithm was designed as a DVRP adaptation of the already successful MACS-VRPTW metaheuristic. Its results appeared promising and it was a very interesting piece of research (partially inspiring this thesis), and as such, one would certainly not question the validity of the research. Building on the workings of Kilby et al. (1998), this implementation was both new and complex. It provided a method for solving problems which previously could only be discussed theoretically. However, with no other items of research upon which to base their decisions and to act as a guide, the application of such a complex and well-developed

algorithm (ACS) on a new problem could be considered as jumping in at the deep end.

Given that lack of research to act as a foundation, their identification of a direction in which research should take was presumably made on an interest level rather than based on the likelihood of producing competitive results. If one was to be interested in producing the best results possible, it may have been of benefit if details were known about which types of solution method were well suited to the problem.

In order to minimise the impact of this problem being in its infancy, it seems prudent to carry out some form of preliminary investigation to act as a foundation upon which more complex research can be based. The existence of the ACS-DVRP simplifies such an investigation greatly.

It is proposed that a series of simple metaheuristic implementations are made, in an attempt to recreate the natural evolution that the solution techniques have experienced on more traditional problems. Furthermore, these results will serve as a reference point; allowing additional comparisons to be made with the existing results (ACS-DVRP) and any other algorithms that are developed. It should be noted that the production of a concise base is never going to be exhaustive, though it should provide enough of an insight to safeguard against investigating unsuitable avenues of research. This Chapter will detail this investigation, and is structured as follows:

Firstly, the common implementation issues are tackled. This consists of a comprehensive discussion of the data sets, and clarification of the necessary computational details.

Secondly, a discussion of metaheuristic classification (including an amended technique) and the reasoning behind such classification are presented. The next two sections detail the implementation of various heuristics and metaheuristics respectively.

The chapter concludes with a discussion of which algorithms represent the most promising direction for the research.

3.1.1 DVRP Benchmark Datasets

As the DVRP was developed to simulate a real-life problem, one must have datasets that reflect this. The CVRP has many such datasets, and Section 2.9.1 detailed how these were adapted into suitable problems for the DVRP. These amended datasets are currently available on the University of St Andrews website at: http://www.dcs.st-and.ac.uk/~apes/apedata.html. These datasets range from 50 to 199 customers and the number of customers, along with the CVRP benchmark from which it was derived, can be inferred from the name of each instance (e.g. c100 implies that it is a dynamic adaptation of a 100-customer problem from the Christofides CVRP benchmarks).

It is common practice when solving a combinatorial optimisation problem to use public datasets, thus preventing the need for each piece of research to construct new ones. This is no different for the DVRP (Kilby et al. (1998) and Montemanni et al. (2005) were not afforded this luxury, explaining their need to create them) and two of the benefits are presented below:

- It allows a direct comparison of solution quality with the results achieved by Montemanni et al. (2005) (the ACS-DVRP algorithm). One can talk candidly and objectively about the performance of a particular algorithm in comparison with another.

- Kilby et al. (1998) were hopeful that their datasets would eventually become universal benchmarks for the DVRP. The obvious prudence demonstrated in the development of these datasets, along with the security that comes from amending existing datasets, allows one to have the necessary confidence in their ability to successfully represent the problem.

The development of good[1] benchmarks (as these appear to be) is important to the development of the problem. If the DVRP is ever to become a widely accepted problem and researched like other variants of the VRP (e.g. VRPTW and VRP with Backhauls) then one must have confidence in them.

[1] The notion of good in this case relates to ensuring they offer an example of the type of problem likely to be faced in real-life.

Due to the number of datasets in this benchmark set (21), it is not possible (due to time and computational limitations, see Section 3.1.2) to carry out all of the analysis on all the datasets. Montemanni et al. (2005) proposed the selection of one dynamic dataset from each of the original three CVRP benchmark sets to create a test set. This proposal is both sensible and justifiable, and thus the same datasets (c100, f71 and tai75a) are chosen.

The three test datasets can be seen geographically in Figure 3.1 (where the different colours relate to the timeslot in which that particular customer becomes known).

One should observe how the datasets differ in their customer dispersion. By optimising an algorithm on such varied datasets, one would hope to produce a suitably robust algorithm. However, as a result of this variation, it may be necessary to compromise on some of the parameter optimisations; this is symptomatic of the type of problem being solved, and is unavoidable.

3.1.2 Computational Details

The primary concern when solving a combinatorial problem is the cost of the solution that is produced. However, as explained previously, the need for metaheuristics stems not from their ability to identify the best solution (they often do not), but to identify a good solution in a reasonable time.

The concept of reasonable time is often both important and interesting in traditional problems such as the TSP, CVRP and VRPTW. The ability to produce a solution very quickly may be paramount to a potential user and as such will heavily influence which algorithm should be implemented. However, there is some notion of ambiguity as to whether it should be so prevalent in the DVRP.

Two different principals regarding time are now presented: the first considers the problem from a theorectical standpoint, while the second gives consideration to the thoughts one might have if they were faced with this problem in real-life.

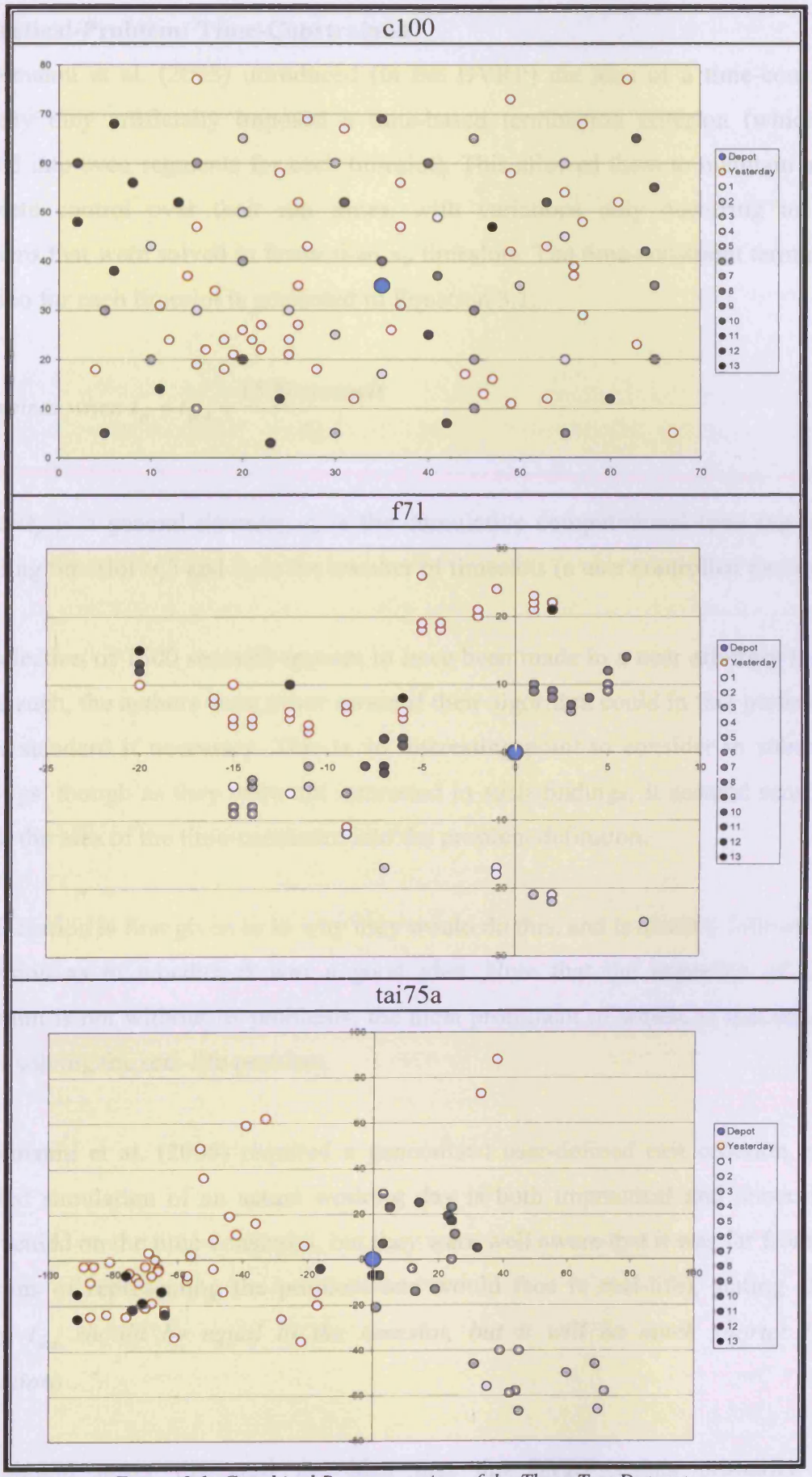

Figure 3.1: Graphical Representation of the Three Test Datasets

Theoretical-Problem: Time-Constrained

Montemanni et al. (2005) introduced (in the DVRP) the idea of a time-constraint, whereby they artificially imposed a time-based termination criterion (which was divided into even segments for each timeslot). This allowed them to maintain almost complete control over their run times, with variations only occurring in those problems that were solved in fewer than n_{ts} timeslots. The time-constraint termination criterion for each timeslot is presented in Equation 3.1.

$$ts_p \text{ expires when } t_p = t_{p-1} + \frac{1500 \text{ seconds}}{n_{ts}} \tag{3.1}$$

where ts_p is a general timeslot, t_p is the cumulative computational time (up to and including timeslot ts_p) and n_{ts} is the number of timeslots (a user controlled parameter).

The selection of 1500 seconds appears to have been made in a near arbitrary fashion, and as such, the authors were never aware if their algorithm could in fact perform to a higher standard if necessary. This is an interesting point to consider in subsequent workings, though as they were not interested in such findings, it seemed sensible to embed the idea of the time-constraint into the problem definition.

Consideration is first given as to why they would do this, and is directly followed by a discussion as to whether it was a good idea. Note that the imposing of a time constraint is not without its problems, the most prominent of which, is that one is no longer solving the real-life problem.

Montemanni et al. (2005) required a generalised user-defined exit criterion, as the repeated simulation of an actual working day is both impractical and unnecessary. They settled on the time-constraint, but they were well aware that it was far from ideal (in terms of representing the problem one would face in real-life), stating '*In the reality* t_{acs} *should be equal to the timeslot, but it will be much shorter in the simulations...*'.

One should never disregard the idea that the DVRP exists primarily as a representation of a real-life problem. However, one must remain aware that other

research may be conducted in the future, and as such, it would be beneficial to be able to directly compare results. This allows one to be much more objective when making conclusions between various algorithms.

This method benefits from its obvious consistency, with no one run ever taking longer than another. However, aside from the obvious lack of realism, there exists another significant argument as to why one should not impose such a restrictive time-constraint.

Although the time-constraint exit criterion is workable, one can envisage a situation where it would seem nonsensical to restrict the algorithm to this artificial limit of 1500 seconds[2]. If an algorithm is still improving the solution in that particular timeslot, why should one move to the next, given this is merely an artificial limit. Surely any real-life implementation would never consider such a counter productive idea.

From a computational point of view, there also exists an argument based on allocating too much time to a problem. If an algorithm does not require 1500 seconds to achieve the best solution it can, one should not need to wait until this limit has been reached. Although not as significant as not giving enough time (as it will not improve solution quality), one must be aware that using resources unnecessarily does not represent a sensible use of one's time.

The time-constrained technique is also susceptible to problems that arise due to different computing speeds (though attempts can be made to negate this by reducing the value of 1500 accordingly, discussed later in this section).

One should be aware that the imposing of any time constraint will significantly alter the problem being solved. Consideration is now given to how one would consider the problem if the idea of the time-constraint were to be disregarded.

[2] Computation time on an Intel Pentium 4 processor 1.5Ghz with 256MB of RAM

Real-Life: Non Time-Constrained

Time can be considered to not be a particularly prominent issue for the DVRP, if one is to consider it being implemented in its real-life environment. The problem involves simulating the deliveries made during the working day[3], and as such, if solutions are obtained in a shorter period of time than that, they are technically valid.

Kilby et al. (1998) supported this ideology, stating *'the main advantage in solving a problem dynamically is that there is a lot of time to devote to improvement'* and further backed up this claim, proclaiming *'... we have all day so we might as well use it'*.

However, when presented with a problem that essentially could be solved over the course of an entire day, one is faced with the decision of just how to terminate a solution algorithm. Obviously the repeated simulation of a working day is not feasible (run-times will be excessive), thus if one were to wish to tackle it in as close to a real-life implementation as one could, a new exit criterion would be needed.

This is particularly problematic given that any potential criterion will have to be applicable to every algorithm that is going to be implemented. Such a requirement leaves one faced with an extremely limited choice. This may have been in Montemanni et al. (2005) thoughts, when they bypassed this problem through the introduction of the time based exit criterion. One could increase the time available to try to give the algorithms a better chance, but again there exists the problem of such a static exit criterion when comparing techniques which have vastly different run times in order to produce good solutions.

Any new generic method cannot be based on some number of iterations/generations etc., as no such constant exists in every algorithm. A more dynamic termination philosophy is proposed.

If one is to consider the metaheuristics movement in the search-space, one will eventually reach a point at which the algorithms will have performed to the best of

[3] One would expect this to be a substantial period of time, often 7 hrs or more.

their ability. Any subsequent computational effort will essentially be wasted (beyond reasonable doubt), as no benefit will come of it. Obviously one will need to pick an individual exit criterion based on iterations, cycles etc. for every algorithm, to support this philosophy.

As a result of this, the duration of each algorithm will obviously vary significantly. By giving the algorithm the best opportunity (time wise) to achieve all it can, one is ensuring that decisions will be made purely on performance.

This philosophy is not without its problems, one will need to retain some awareness of time to prevent run times from becoming ridiculously large (with respect to the notion of the working day). Furthermore, as one needs to be sure that they are allowing the algorithm necessary time to perform to the best of its ability, it is possible that the algorithm may be run for longer than is strictly necessary. One will naturally err on the side of caution, and as such, certain parameters (controlling the exit criterion) may be chosen to be larger than is strictly necessary.

With the introduction of this new philosophy, there are some interesting points that arise and are duly noted:

- Results will not be directly comparable to those achieved by the ACS-DVRP. This would be unfair, as one does not know if this algorithm could have performed to a higher standard if it were to be run for a longer period of time.

- The run-time will no longer represent the value T, as proposed by Montemanni et al. (2005). Under this new criterion each timeslot will not be given an equal proportion of the computational time. This is due to the varying levels of complexity, caused by the number of customer arrivals and commitments. If one wanted to compute a comparable value for T, it would be necessary to identify the maximum time spent on any one timeslot and multiply this by n_{ts}.

Balancing the Two Philosophies

Given significant arguments exist for and against both philosophies, a decision has been made to try to give a good account of both variants of the problem. Substantial consideration was given to this, as it will have a significant impact on the direction that this thesis will take.

Not specialising in either variant presents many problems of its own, not least increasing the complexity and size of the workings required. However, given the lack of any logical reason for discounting either variant of the problem, no other decision seems justifiable.

A basic pattern will be followed for all chapters (with the exception of Chapter Seven which is specifically interested in solutions produced in short periods of time). The algorithms will not be considered in their time-constrained variant until towards the end of each chapter. The introduction of new ideas and optimisation of parameters will be made on the non time-constrained variant allowing one to make decisions based on results that are the best the algorithm can achieve.

Any decisions made will be privy to the trade-off between solution quality increase and time increase, thus all decisions can be made objectively. Any ideas that significantly increase run-times will not be considered due to their inappropriateness in the time-constrained variant. It should be noted that an additional benefit of this dual variant ideology comes from the increased likelihood of producing a fairly robust algorithm.

Upon production of a complete set of non time-constrained results one will consider the run-times and decide whether any further amendments are needed to adjust the optimised algorithm to the time-constrained variant of the problem. If required, the necessary alterations will be made, and will be followed by a complete set of time-constrained results.

With this dual ideology being followed, confusion between the variants is an obvious concern. To avoid such pitfalls, all time-constrained results will be presented in blue[4]. Any comparisons between these blue results can be made purely on solution quality, whereas time may be a factor in decisions and conclusions made in the non time-constrained variant.

With a decision made to continue with both problem variants (though with the majority of decisions being based on the non time-constrained variant), one can now consider the notion of trials.

Trials

Due to the random nature of the algorithms that are going to be applied to this problem, it is necessary to run each algorithm for more than just a single run. This is common with many combinatorial problems, but is likely to be even more prevalent due to the variability posed by a dynamic problem.

Montemanni et al. (2005) suggested that five trials be conducted for each analysis, and reported the minimum, average and maximum cost achieved (denoted Min, Ave and Max respectively). This reporting system seems sensible and is thus continued. The number of trials is increased to ten, to improve the confidence one has in the conclusions one makes when judging the results.

Computational Performance

All of the computations in this thesis will be conducted on a personal computer with an Intel Pentium 4 processor 3.06 GHz with 1GB of RAM and all of the algorithms have been programmed in FORTRAN. All times reported will be the times taken on this machine.

As identified earlier, a problem exists for comparing results in the time-constrained variant of the problem, as this machine operates at a faster speed than that used by which Montemanni et al. (2005). Dongarra (2002) suggests that their machine

[4] Results not shown in blue may have still been produced in less than 750 seconds (see Computational Performance for details).

operates at half the speed of the one that is to be used in this thesis, thus the artificial limit of 1500 seconds is reduced to 750 seconds.

3.2 Metaheuristics, Details and Discussion

The primary focus of this thesis is the production of high-quality (if not optimal) solutions for the DVRP and other variants of it. These solutions are to be produced using a variety of carefully selected metaheuristics. A review of the more popular metaheuristics was provided in Chapter Two, coupled with a discussion of why non-optimal techniques are used given that exact methods (such as branch-and-bound) exist.

This section presents a discussion on why one would be interested in the classification of metaheuristics as well as a modification to an existing metaheuristic classification system.

3.2.1 Metaheuristic Classification

Since their inception, considerable research has been conducted in the field of metaheuristics. Multiple metaheuristics have already been developed, and there exists the distinct possibility that others will be suggested in time. Given the continuously expanding nature of this field, various authors have felt that it would be beneficial to classify the existing metaheuristics into sub-groups.

Within this metaheuristic classification section, there are two key ideas to consider:

The first of these concerns itself with the central issue, why one would want to classify metaheuristics. The reasoning behind ones need (or want) to classify metaheuristics could vary considerably, thus a selection of ideas are presented.

Secondly, given the established benefits of classification, it is necessary to consider how one wants to classify. There does not exist a commonly accepted classification scheme, thus it has always been at the author's discretion to propose one. The existence of multiple classification schemes not only presents the possibility of confusion (as to which to use), it can result in overly bespoke schemes. One can

design a scheme to fit their own work well, but it may not be applicable to others work, and as such may not serve the wider research community.

An adaptation to an existing scheme is presented, in the hope that it clarifies the process, allowing for the better classification of metaheuristics.

Why Classify

There does not exist one sole reason behind ones desire to classify the existing metaheuristics. It is likely to vary from person to person, and as such, only possible reasons can be presented. Supplementing this list is a discussion of which of these are applicable to the successful establishing of a base, on which this thesis can build.

- Ease of reference i.e. one can state they are using a specific type of algorithm.
- Why two techniques might complement each other in a hybrid algorithm.
- Why particular algorithms might be best applied to certain problems.
- Understanding common properties shared by certain algorithms.
- Understanding the different ways in which algorithms can find good solutions.

With regards to this thesis, it is the final three reasons that are pivotal. As a relatively new problem, little is known about what techniques are better suited to the dynamic environment. The classification will hopefully safeguard against the selection of unsuitable or overtly similar algorithms, whilst allowing a fair comparison of those that contain common features.

The Classification Scheme of Osman and Hindi (2004)

As with the definition of the term metaheuristic, there is no commonly accepted scheme for metaheuristic classification. One is faced with a choice of designing one for oneself (as many authors have chosen to) or using an existing technique (which is likely to have been designed for a different purpose).

The discussion in this thesis is to focus on the adaptation of an existing technique, in the hope that it may clarify an existing ambiguous technique, such that, it may be beneficial regardless of one's reason for wanting to classify.

Osman and Hindi (2004) proposed the scheme that is to act as a basis for the classification scheme in this thesis. They proposed that the family of metaheuristics could be divided into a trio of sub-groups each with a different inherent theory for finding solutions with good/optimal costs:

- Construction

  These algorithms generate solutions by the appending of components to an (initially empty) partial solution, until that solution is complete. It is hoped that using a careful selection procedure, a good solution will be constructed.

- Local Search

  Starting with an initial solution (usually found via some heuristic) these algorithms iteratively replace the current solution by another solution in a user-defined neighbourhood.

- Population

  These use a number of solutions and combine them with the hope of utilising the positive components of various solutions. The result of this is the creation of new solutions with differing and hopefully better costs.

Having such a small number of subgroups could present a problem depending on the reason behind one's wish to classify. They may not allow for enough distinction between techniques to reap the benefits of classifying in the first place. Furthermore, with the technique in its current guise, a serious problem presents itself when one actually tries to allocate a metaheuristic. Given such a limited number of subgroups, one is faced with deciding which best represents an algorithm that could be argued for inclusion in two (or even three) subgroups.

This user interpretation provides a lack of consistency; relying on a series of compromises is unlikely to ever represent an ideal situation. However, for all its faults, this seems a more useful system (for this particular classification) than other proposed systems, such as Bitattari et al. (2001) and Blum and Roli (2003). These

relied on repeated separation algorithms into two distinct classes based on certain attributes.

To alleviate the problem associated with the current technique, a Venn diagram is proposed as a better way of representing the three subgroups. This modification to the existing classification system, allows one to utilise the overlaps between the subgroups (thus classifying a metaheuristic as an amalgam of two or more subgroups).

A list of some of the most popular metaheuristics is given below, along with an example of how one would allocate a specific algorithm into this modified system:

- Ant Colony Optimisation (ACO)
- Genetic Algorithm (GA)
- Greedy Randomised Adaptive Search Procedure (GRASP)
- Memetic Algorithm (MA)
- Simulated Annealing (SA)
- Tabu Search (TS)

The classification of the ACO[5] metaheuristic is now considered. Osman and Hindi (2004) deemed ACO to belong to the 'construction' subgroup. This is a perfectly sensible proposal as each ant tries to construct a good solution. However, given that there are multiple ants, interacting via the trail matrix; one could easily argue that it belongs in the population subgroup. Under Osman and Hindi's scheme, one was faced with a choice, as to which was the more prevalent feature. With the introduction of the Venn diagram, it can be placed in the intercept between these two subgroups.

The remainder of the metaheuristics that were listed have been classified according to this Osman and Hindi-like system (see Figure 3.2). The introduction of intercepts has greatly simplified the classification procedure through a reduction of subjectivity and an increase in the scheme's flexibility.

[5] Assumes that a descent phase has not been included in ACO

Two of these metaheuristics, ACO and TS (though TS was applied in a reactive system on different datasets) have already been applied to this problem. As a result of this, basic forms of these algorithms will be investigated in this preliminary analysis. Supplementing these implementations will be two heuristics and a further two metaheuristics. Details regarding the selection of these algorithms can be found in Sections 3.3 and 3.4 respectively.

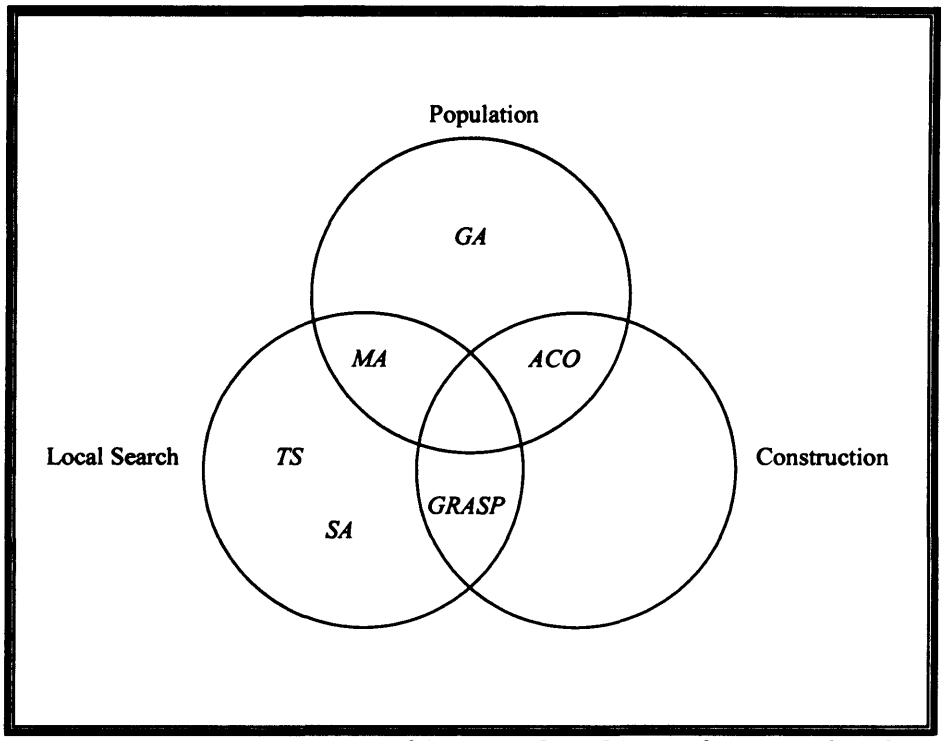

Figure 3.2: Venn Diagram Representation of Osman and Hindi's Metaheuristic Classification System

3.3 Implementation of the Chosen Heuristics

With the problem and the history of possible solution techniques suitably introduced in Chapter Two, focus now switches to the production of results for the DVRP. This section implements two simple heuristics, to act as a precursor to the more complex metaheuristics that are to be considered later. Although not strictly necessary, as solution quality is likely to be poor in relation to the ACS-DVRP, there are two beneficial reasons for starting the analysis at this point:

- It gives some basic results, which represents a very simple starting point for the analysis. Results like these would have been produced for other combinatorial optimisation problems when they were first developed.

- As the definitions of the word metaheuristic explained, the techniques rely on the intelligent use of heuristics. By implementing the heuristics, one is able to gain a greater understanding of the mechanics of these algorithms, as well as simplifying the descriptive process of the latter techniques.

There exist a considerable number of heuristics that have been applied to the CVRP, and therefore, through the use of the timeslot technique could be easily adapted to the DVRP. However, with metaheuristics forming the primary focus of this thesis, it makes sense to consider those that are contained within some of the metaheuristics that are going to be considered. The heuristics chosen are two of the simplest, yet widely used (for the CVRP): The Nearest Neighbour algorithm and the descent algorithm.

3.3.1 Nearest Neighbour (NN)

The application of a NN algorithm is a sensible starting point for any routing based problem. It gives a quick and simple (deterministic) solution, with which other techniques can be compared. Its simplicity is in relation to the other methods that are to be implemented; applying it to the DVRP is still a more complex procedure than the TSP implementation discussed in Chapter Two.

The increased complexity comes from the dynamic factor, as well as the need to consider multiple vehicles. For ease and continuity they are considered sequentially, in a similar fashion to the construction process in the ACS-DVRP. The decision as to which customer should be appended to the incomplete solution, is made from the set F (where the set F contains all the feasible customers for the current vehicle[6]). A single customer is selected from this set (denoted j), whereby the decision is based purely on the minimisation of distance, as seen in Equation 3.2.

$$j = \arg\min_{l \in F} \{c_{il}\} \tag{3.2}$$

where i is the current vehicle location.

[6] Depot is not included in the set F.

This process is continued until there are no feasible expansions remaining i.e. $F=\varnothing$. Upon reaching this situation, one of two scenarios will have been reached:

- The current vehicle has been filled with all the customers that it can, but customers remain unallocated. The current vehicle is then scheduled to return to the depot (only temporarily[7]) and another vehicle must be deployed. This new vehicle then will have a new set F, associated with it, and selections can again be made according to the NN philosophy.

- All customers have been allocated to a vehicle. This represents the termination criterion of the NN algorithm i.e. the solution for that timeslot is complete.

The timeslot technique greatly simplifies the descriptive process, as each timeslot is solved in a similar fashion. One important difference in the latter timeslots (i.e. not ts_0), is noticeable when (and if) an extra vehicle is required. As in ts_0, the new vehicle will begin at the depot, but the departure time will be equal to the end of the current timeslot + T_{ac} (as that driver has already committed to not working that timeslot and the advance commitment time). Please refer to Section 2.9.4 for details of the timeslot solution strategy and commitment policy.

Pseudo code for the NN algorithm can be seen in Figure 3.3.

With the mechanics of the algorithm suitably described, one can now implement the described NN algorithm on the DVRP. It should be noted that due to the non-stochastic nature of the NN algorithm, it is not necessary to conduct the aforementioned trials (and display Min, Ave and Max results) as the same solution will be produced every time.

These results can be seen in Table 3.1.

[7] As this part of the schedule will be discarded in the next timeslot.

Input: Problem data

Output: Solution

While customers have not been allocated

If $F \neq \varnothing$ then

Select $j \in F$ such that c_{ij} is minimised

$i=j$

Update F

Else return current vehicle to depot and consider next/new vehicle

End If

End While

Figure 3.3: NN Pseudo Code

Method		c100	f71	tai75a
NN		1394.84	397.72	2786.79
ACS-DVRP	Min	973.26	311.18	1843.08
	Ave	1066.16	348.69	1945.20
	Max	1100.61	420.14	2043.82

Table 3.1: Results Achieved by the NN Algorithm and the ACS-DVRP

These results are only the second set of results available for these specific datasets and as such any comparison is fairly limited. As one would expect, the solution quality is considerably worse than those produced via the ACS-DVRP (time-constrained). If one compares these with the averages achieved by the ACS-DVRP, one can see an increase in distance of between 14-43%.

Comparing the distances with the maximums achieved by the ACS-DVRP (i.e. worst case scenario), they range from a 6% reduction to a 36% increase. This demonstrates that there is definite scope for improvement (with regards to the ACS-DVRP), as even a simple algorithm betters one of the metrics reported.

One would never expect a simple NN heuristic to produce highly competitive results (when compared with the complex ACS-DVRP), but the early indications are perhaps more promising than one would have envisaged. These solutions should be retained as

a worst-case scenario; any technique that produces solutions of a lower quality is simply not suitable.

Probabilistic NN Adaptation (PNN)

To conclude this discussion of the NN algorithm, a stochastic variant of the technique is presented. Although one would not consider this as a heuristic technique in its own right (it is very unlikely to produce better results) it does form an important component of some of the more complex metaheuristics that will be considered. Described here for the sake of simplicity, due to its obvious link to the NN algorithm, it chooses the next customer to append to the current solution probabilistically (based on distance) from the set of all feasible expansions (F).

This choice is made using a simplified RPR-like technique (Equation 3.3), where distance is the sole component considered.

$$p_{ij} = \begin{cases} \dfrac{(1/c_{il})}{\sum_{l \in F}(1/c_{il})} & \text{if } j \in F \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

One reason for such an adaptation comes from the idea that a greedy decision at one point in the construction may well lead to a worse decision having to be made towards the end of the construction process. This is a well-known problem with the NN algorithm, and as such is often responsible for the algorithms lack of competitiveness.

Solutions produced via the PNN technique are likely to have distances higher than the NN algorithm due to randomly selecting a poor customer to append, but they can act as a diversification technique in techniques that involve a population of solutions. No results are presented for this algorithm (they were of poor quality), but it will be included in some of the metaheuristics implemented later in this chapter.

3.3.2 Descent

The basic premise of the descent algorithm was described in some detail in Chapter Two; thus only a basic overview is presented here. Note that the primary focus of this

section is on the implementation of the technique; rather than further discussion of its underlying principals.

The traditional descent algorithm involves the minimisation (or maximisation) of the cost function, $f(x)$, subject to $x \in X$ (where X is a set containing all feasible solutions). The descent algorithm begins with a solution (referred from here on as the initial solution), which represents one particular point in the search-space. The search then moves, in an iterative fashion, to other points in the search-space via neighbourhood moves.

The potential moves consist of all solutions that belong to the user-defined neighbourhood. Each $x \in X$ has a neighbourhood, $N(x)$, and from that, some $x' \in N(x)$ is chosen as the next place in the search space to move to, subject to $f(x') < f(x)$. The position x' then replaces x, and the process is continued until some chosen termination criterion is met.

When operating a descent algorithm, one must decide which descent strategy one wants to adopt. As explained in Chapter Two, there exist two quite different types of descent strategy. Given the need to make a choice as to which one is going to be pursued in this implementation, a brief summary of them both is provided below:

- First Improvement (FI)
  One iteratively tests random moves in the neighbourhood, and implements a move each time x' is found, such that $f(x') < f(x)$. One cannot be sure of obtaining a local optimum with this technique, but it may offer speed advantages dependent on the size of the neighbourhood.

- Best Improvement (BI)
  One evaluates the entire neighbourhood i.e. all $x' \in N(x)$, and the move that maximises the reduction in the cost function is made. If the best neighbourhood move available does not reduce the cost function i.e. $f(x') > f(x)$, then one is assured that a local optimum has successfully been located.

There seems to be little conclusive evidence with regards to whether one type of descent strategy consistently outperforms the other. One can envisage a situation where either one would be preferable, dependent on the neighbourhood definition or some unavoidable time constraint (but that is not the case here). Neither strategy is likely to take significantly longer than the other, given the desire to give the algorithm the best chance possible.

With any choice being purely subjective, it seems sensible to follow the precedent set by the ACS-DVRP which utilised a FI descent strategy (in its improvement phase). The exit criterion for their implementation was based on the time constraint (or more specifically a percentage of it, dedicated to descent), thus some new criterion for this FI implementation will be required.

As one is attempting to give the algorithm the best chance possible, one must be confident that the algorithm is being run until a local optimum is identified. As discussed earlier, this is a relatively simple idea in relation to the BI descent strategy, but needs a more considered approach in the FI descent strategy adopted here.

One is aware of the number of iterations of the descent algorithm that have been completed i.e. each tested $x' \in N(x)$, and thus is able to monitor the performance of the algorithm. This provides two potential ways of monitoring performance, the number of iterations that have been completed, or, perhaps more interestingly, the number of iterations that have passed since an improvement was identified.

This second method allows one to base the termination of the algorithm on its performance level, which adheres to the philosophy of giving the algorithms the best chance to achieve all that they can. The decision has been made to select this exit criterion and therefore one is required to introduce a parameter to control it. The parameter z is selected, and is set arbitrarily high (z=3000) to ensure that (beyond a reasonable doubt) a local optimum is achieved.

Pseudo code detailing this procedure can be seen in Figure 3.4.

```
Input: Problem data and initial solution
Output: Improved solution

While   Iterations since last acceptance < z
        Test random x' ∈ N(x)
        If f(x') < f(x) then
                Designate x' to be the new x
        End If
End While
```

Figure 3.4: Descent Pseudo Code

With the algorithm's mechanics suitably introduced and a descent strategy chosen, one must now select an appropriate neighbourhood in which the technique can operate. Given that this is only a basic implementation, the 1-Opt neighbourhood (as chosen by Montemanni et al. (2005) in the ACS-DVRP improvement phase) has been chosen.

This will be used as the only neighbourhood in this chapter (including the metaheuristics that rely on a local search component), thus allowing a fair comparison with the other techniques. To make comparisons between techniques utilising different neighbourhoods would unnecessarily distort results, which consequently, would further complicate the selection process.

In order to implement a neighbourhood in a VRP (such as the DVRP), one must not only account for the possibilities of customers moving within a vehicle (intra, like the TSP), but also moving between two vehicles (cross). Many popular neighbourhoods were originally conceived for the TSP, and hence have required adapting to make them applicable. The 1-Opt neighbourhood originated as a TSP neighbourhood but has long since been applied to VRP.

New notation is now introduced, to allow one to identify which moves are (or are not) included in a specific neighbourhood. This is not strictly necessary for a neighbourhood as rudimentary as 1-Opt but it will allow for the use of consistent notation throughout this thesis.

Consider the general form of a move to be the swapping of a set of customers (A, can be 0) from one vehicle with another set of customers (B). The second set can either come from within the same vehicle (i) or from another (x). Thus one is able to denote the two types of moves in the form i:A,B and x:A,B, where A and B are the number of customers being relocated.

Using this new notation, one is able to refer to the two composite moves that define the 1-Opt neighbourhood as i:1,0 and x:1,0. An example of the way in which these moves can alter the current solution can be seen in Figure 3.5.

As the FI descent strategy has been selected, potential moves need to be tested in an arbitrary fashion. Ideally one would randomly select each potential move from the entire neighbourhood, however that would unnecessarily complicate proceedings given that the two neighbourhood components (i:1,0 and x:1,0) require distinct programmes to be written. As a compromise, prior to a move being selected and tested, a random choice is made between the two neighbourhood components.

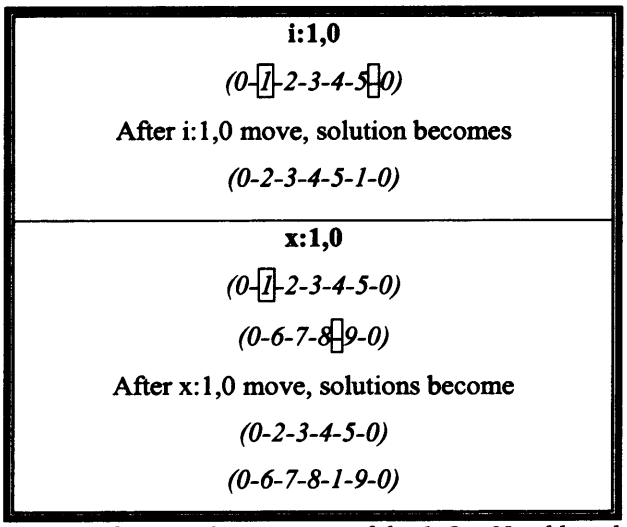

Figure 3.5: The Two Components of the 1-Opt Neighbourhood

With the neighbourhood introduced, one merely needs to identify an initial solution and the descent algorithm can be implemented. It makes sense to utilise the initial solutions constructed via the NN algorithm. The results for the descent algorithm can be seen in Table 3.2.

Method		c100	f71	tai75a
Descent	Min	1124.54	317.88	2060.45
	Ave	1215.56	370.76	2248.18
	Max	1338.24	424.97	2422.52

Table 3.2: Results Achieved by the Descent Algorithm (NN Initial Solution)

These results (and all those that follow) contain some form of random element, and as such one is required to conduct the trial of ten runs that was discussed in Section 3.1.2. As expected, the descent algorithm has improved results, though this is not as clear-cut, as one would achieve were they investigating a static problem.

Usually, a descent algorithm cannot result in solution quality being reduced from that achieved via the heuristic that created the initial solution. Moves that increase the cost function are not allowed and as such, the worst that could occur would be that no improving moves were located (and the descended solution would equal the initial solution).

However, with this problem, a better solution in one timeslot does not always result in a better overall solution (when all of the timeslots have been considered). This can be seen when looking at the Max results from the trial of ten. The worst result produced for the f71 dataset is worse than was achieved by the NN heuristic. Unfortunately, this is somewhat symptomatic of the problem being solved (or specifically the timeslot solution technique).

When comparing the Min and Ave results of the descent algorithm with those produced via the NN heuristic, one can see that result quality has improved by between 19-26% and 7-19% respectively. Although a worse solution (for f71) was produced, one would have no hesitation in stating that the descent algorithm has improved upon the initial solution. This is important, as it means that the problem is behaving in a similar (if not exactly the same) manner as the static routing problems that preceded it.

3.3.3 Summary of Heuristics

With the implementation of the two basic heuristics complete, some general statements about the quality of result produced can be presented. These are intended to demonstrate the higher quality of the results produced by the ACS-DVRP of Montemanni et al. (2005).

The results achieved by the heuristic techniques are of a noticeably lower standard, though the uncompetitive nature of these solutions is not surprising. As stated earlier; these heuristics are merely the building blocks of the more complex metaheuristics that are to be introduced shortly. Given the high level of discrepancy in the solution quality between the results produced thus far and those achieved by the ACS-DVRP, it does not seem unreasonable to speculate that the results produced by Montemanni et al. (2005) are likely to be of a high standard (for any technique).

With the heuristics seemingly uncompetitive, consideration is now given to the more complex metaheuristics. One would hope that these offer some improvement to the heuristic results, though it would not be surprising if they still do not compete with the ACS-DVRP. The ACS-DVRP is a complex implementation of the ACO ideology, whilst the metaheuristics considered in this chapter are going to be of a less developed nature.

3.4 Implementation of the Chosen Metaheuristics

After a discussion concerned with the meaning and origins of the term metaheuristic, Section 3.2 identified and expanded upon a well-known metaheuristic classification system. The aim of this classification was to ensure that the metaheuristics chosen for this initial implementation are not all of a similar type.

However, with a variety of metaheuristics going to be considered, it seems sensible to discuss why it is beneficial to implement a variety of algorithms. This discussion focuses upon the notion that there exists no metaheuristic that is better than another. This idea was formalised by Wolpert and Macready (1997), through their 'no free lunch' theorems.

'No Free Lunch'

Glover (1986) coined the term metaheuristics, and since that paper, there has been a considerable amount of research conducted in this field. Given this significant volume of work, it does not seem unreasonable to assume that one could collate the various workings and identify metaheuristics that achieve consistently better results than others. However, when people came to look at the performance levels, no such conclusion was forthcoming. The discrepancy in performance levels was significant and led to the ideology that better metaheuristics exist to be refuted by Wolpert and Macready (1997). They stated '*... that for any algorithm, any elevated performance over one class of problems is offset by performance over another class*'.

This was an incredibly important conclusion, as one would not be able to disregard metaheuristics on the basis of their poor performance on other problems. With the knowledge that no algorithm is of an inherently higher standard than any other; basing one's work on a single metaheuristic because it has been successful in the past could seem somewhat foolhardy.

As the DVRP is a new problem, Montemanni et al. (2005) (ACO) and Gendreau et al. (1999) (TS) tried to safeguard against this by selecting algorithms that have had proven success on problems of a similar nature, namely the CVRP and the VRPTW. Although this will not guarantee good performance on the DVRP it seems a prudent step to take. Whether the high level of performance follows through to the DVRP, is not currently known, however, it is hoped that this analysis may well shed some light on the issue.

The Selection of Metaheuristics

The classification of the various metaheuristics in Section 3.2 offers extra information that one can use when trying to select appropriate metaheuristics to implement. Firstly, given that other authors have chosen ACO and TS, it is important that their suitability is assessed. One should know if good performance on other VRPs is a suitable guide for selecting DVRP metaheuristics, or whether they were misguided in their selection.

Supplementing these are two other metaheuristics that have been chosen for various reasons (explained below):

- Greedy Randomised Adaptive Search Procedure (GRASP)
  It was felt beneficial to analyse an algorithm with which one could draw a number of parallels with the ACO technique. The lack of any other metaheuristics being classified in the construction/population sub-group meant that the selection had to be made in a subjective manner. GRASP was selected as it offered some of the desired characteristics; most prominently the production of multiple solutions. Moreover, if one is to include an improvement phase into an ACO algorithm then the parallels with GRASP become even more prominent.

- Simulated Annealing (SA)
  It was felt necessary to test a second LS technique. This will enable one to see if TS is the most suitable for implementation on the DVRP. The neighbourhood used for all LS in this precursory analysis is the simplistic 1-Opt neighbourhood (discussed in 3.3.2). Given that the strength of these algorithms is linked to the quality of the neighbourhood, this should enable one to see if the mechanics of one technique outweighs that of the other.

The remainder of this section details the implementation of the four different metaheuristics; starting with ACO (though a far simpler variant than that proposed by Montemanni et al. (2005)), then GRASP, TS (again a far simpler implementation than the existing one by Gendreau et al. (1999)) and finally SA.

Each of the implementations will adhere to the same similar structure. They will start with a brief overview of the algorithm, stripped-down to its most basic form (more complex implementations were detailed in Chapter Two). A discussion of any practical issues pertaining to this implementation is then made, along with a discussion of the various parameters. Pseudo-code detailing this specific implementation is then presented, before a parameter optimisation is carried out.
The performance of these algorithms will be analysed in Section 3.5.

3.4.1 Ant Colony Optimisation (ACO): Ant System (AS)

Chapter Two identified the many forms that the ACO metaheuristic can take, ranging from the seminal AS to the more recent and complex MMAS and ACS algorithms. As stated previously, simplistic, stripped-down versions of the metaheuristics are being assessed in this section, thus it seems sensible to choose the AS/EAS guise here. Elitist ants have been considered for inclusion as they appeared very early in the algorithms evolution, see Dorigo et al. (1996).

This analysis follows from the description of AS/EAS given in Section 2.4.1; construction is made via the RPR (seen again in Equation 3.4) and the Ant-Cycle definition has been selected for the learning phase ahead of Ant-Density and Ant-Quantity. All ACO related parameter definitions remain the same as previously given.

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in F^k}(\tau_{il})^{\alpha}(\eta_{il})^{\beta}} & if\, j \in F^k \\ 0 & otherwise \end{cases} \tag{3.4}$$

The algorithm constructs m (the number of ants) solutions each cycle and trail values remain constant until a cycle is completed. Construction of solutions will be made in the same sequential manner as in the ACS-DVRP, with the set F^k containing all feasible customers that can be appended to the current vehicle's schedule. If $F^k \neq \varnothing$ then a new vehicle is deployed, thus requiring the set F^k to be redefined accordingly.

Upon completion of a cycle the trail values are adjusted according to the relevant update rule (seen again in Equations 3.5, 3.6 and 3.7).

$$\tau_{ij} = \rho\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs} \tag{3.5}$$

where

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if arc } (i,j) \text{ belongs to } T^k \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

and

$$\Delta\tau_{ij}^{bs} = \begin{cases} Q/L_{bs} & \text{if arc } (i,j) \text{ belongs to } T_{bs} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

It should again be noted that the AS/EAS algorithm differs in philosophy to the ACS algorithm due to the use of a positive feedback loop i.e. trail values on arcs belonging to better solutions are reinforced (as opposed to being diminished). The elitist ants will further intensify the search on the promising regions of the search space.

The algorithm is run until t (a user-defined parameter that acts as the exit criterion) cycles have been completed. The best solution constructed (out of the entire $m \times t$) is returned as the final solution. Figure 3.6 shows pseudo code that details the mechanics of the applied AS/EAS.

```
Input: Problem Data
Output: Solution

While number of cycles completed is < t
    While number of solutions that cycle is < m
        While all customers have not been allocated
            While F ≠ ∅
                Use the RPR to select a customer to append to current vehicle
            End While
            Deploy new vehicle
        End While
    End While
    Update τ according to Ant-Cycle learning phase (can include elitist ants)
End While
```

Figure 3.6: AS Pseudo Code

With the algorithm suitably introduced and any necessary details clarified, one only requires a set of default parameters to be able to implement the AS/EAS on the DVRP.

As this is such a parameter heavy algorithm, it seems prudent to turn to prominent ACO literature to try to identify a suitable set of default parameters. These can then be

adjusted by way of a parameter review to tailor the algorithm specifically to the DVRP. In the case of ACO (or more specifically AS/EAS) one would naturally turn to the many workings of Dorigo. These default values, were taken from Dorigo et al. (1996) and were supplemented with one additional parameter (t) identified by some preliminary runs[8]: e=0 (no elitism), Q=1, m=n, α=1, β=5 and t=3000.

No default value for ρ has been selected, as this is to be the first parameter that is formally analysed in the parameter review. There is often no specific reason for the order in which parameters are assessed, with one often working in an arbitrary manner. The value of ρ selected by Dorigo et al. (1996) (when working with the Ant-Cycle learning phase) is somewhat high when compared with subsequent analyses and as such, seems a sensible place to start the parameter review. Note that in the preliminary runs a value of ρ=0.5 (as per the workings of Dorigo et al. (1996)) was used.

The preliminary runs were required for two reasons: the first of which being the obvious disparity between the TSP implementation of Dorigo et al. (1996) and the DVRP being tackled here. The time taken to identify good solutions in both may be considerably different. Secondly, one must adhere to the previously discussed philosophy of giving the algorithms the best chance possible to showcase their strengths. The value of t must be set at such a level that the algorithm is given the maximum chance of success.

To support the selection of t=3000, Figure 3.7 presents the current best solution against the cycle number, from a single run on the c100 dataset in ts_0.

As one can see, the selection of t=3000 errs on the side of caution considerably. The best solution was identified after less than 500 cycles, and as such, is technically being run for much longer than was strictly necessary. However, with other timeslots maybe requiring further cycles, this seems to represent a value with which one can have confidence that the algorithm will have enough time to showcase its potential.

[8] These runs were used solely to understand parameter interaction, thus results are not presented.

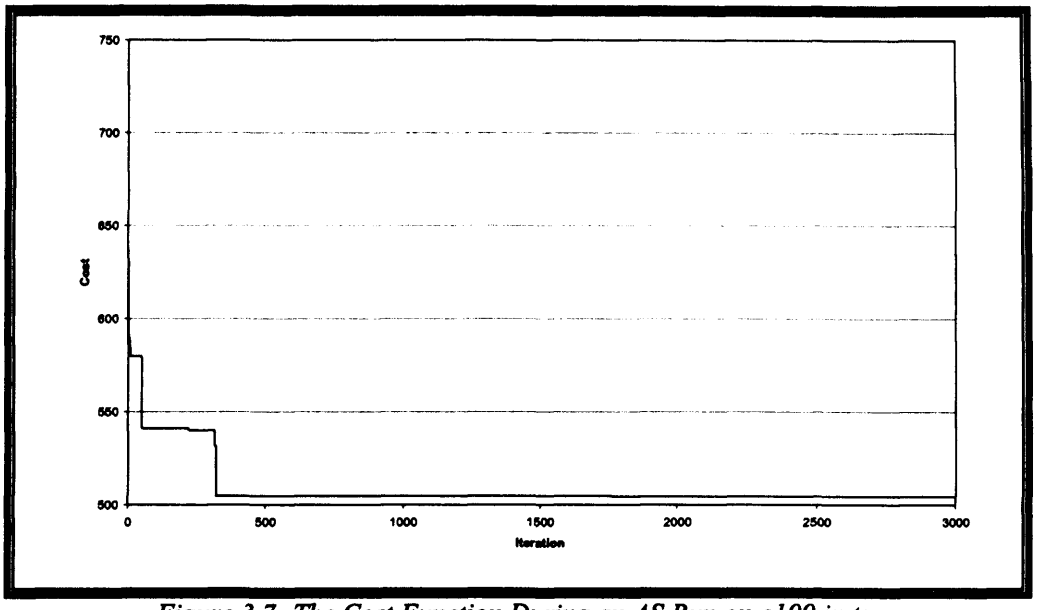

Figure 3.7: The Cost Function During an AS Run on c100 in ts_0

With a default set of parameters in place, attention switches to the optimisation of these (an integral part of any metaheuristic implementation).

AS/EAS Parameter Optimisation

Often cited as one of the main problems with the ACO metaheuristic, the parameter optimisation can be a time consuming affair. The sheer number of parameters makes this a cumbersome algorithm to get working with any sense of optimality. This review concerns itself with what one could consider to be the most influential parameters, namely: ρ, α, β and e. The remaining parameters (specifically m and Q), although important, are not considered in this review.

Within traditional routing problems, the number of ants (m) has long since been shown to work well when it is set to equal the number of customers (n). Though this was not used in the ACS-DVRP, that decision related to the fact that it was the less traditional ACS implementation (with an unusual interplay of its complex update rules), rather than it being an unsuitable selection for the DVRP.

The AS/EAS CVRP implementation of Bullnheimer et al. (1999b) with which this implementation shares many common features, continued to use $m=n$. Given the dynamic nature of the problem being considered, a decision is made to set $m=n_p$ (thus a different number of ants will be used for each CVRP-like sub-problem).

The parameter Q, which Dorigo et al. (1996) described as the 'quantity of trail', has been shown to be a fairly robust parameter, and as such, seems an unnecessary parameter to consider. Most AS/EAS implementations have abandoned the optimisation of the Q parameter, often replacing it in the Ant-Cycle learning phase definition with the value Q=1. This seems a sensible policy to continue with, and as such, one does need to optimise in this review.

These parameters were included in the preliminary runs and did not seem to adversely affect performance. With values for these parameters known, one can begin the optimisation process. As mentioned previously, this will begin with considering the most appropriate value for the parameter ρ.

Results can be seen in Table 3.3.

ρ		c100	f71	tai75a
0.5	Min	**1080.26**	**333.59**	2118.33
	Ave	**1141.88**	354.57	**2265.54**
	Max	1214.26	385.01	2566.94
0.1	Min	1102.09	336.49	**2070.90**
	Ave	1163.24	**351.23**	2313.45
	Max	1254.36	**366.20**	2604.03
0.01	Min	1095.68	345.37	2177.07
	Ave	1144.86	372.43	2306.25
	Max	**1181.75**	426.87	**2457.29**

Table 3.3: Analysing Various Values of ρ in AS

It is immediately clear that there is a considerable amount of variability in the results; therefore any choice is going to be a compromise across the three datasets. However, with the intention of making the algorithm as robust as possible, ρ=0.5 is selected. This choice is made in the knowledge that it supports the findings of Dorigo et al. (1996), and thus enhances the confidence one has in its selection.

With ρ now selected, the balance between the trail and visibility factors (within the RPR construction) can be considered. Controlled by α and β, they dictate the bias associated with each of the two components. Given that there is a substantial volume

of literature that suggests that $\alpha \leq \beta$ provides the highest quality solutions, this seems a sensible policy to adhere to.

With the decision to place a greater emphasis on the visibility than the trail, it seems sensible to fix $\alpha=1$, and only consider varying the value of β. The results of this analysis can be seen in Table 3.4.

β		c100	f71	tai75a
1	Min	1279.78	327.88	2230.80
	Ave	1360.34	351.21	2420.60
	Max	1506.01	373.08	2514.79
2	Min	1128.13	321.13	2153.26
	Ave	1194.79	**351.28**	2279.01
	Max	1262.62	376.37	**2341.55**
3	Min	**1048.35**	339.62	2132.37
	Ave	**1127.20**	355.16	**2245.49**
	Max	**1191.31**	**370.91**	2351.58
4	Min	1063.76	**317.73**	2145.88
	Ave	1135.64	365.55	2255.91
	Max	1256.68	400.15	2516.20
5	Min	1080.26	333.59	**2118.33**
	Ave	1141.88	354.57	2265.54
	Max	1214.26	385.01	2566.94

Table 3.4: Analysing Various Values of β in AS

These results show that the default setting of $\beta=5$, was placing too much of an emphasis on the visibility, resulting in the system acting in too greedy a fashion. The search was not acting in a diverse enough manner to locate solutions of the highest possible quality. It appears that by reducing this bias parameter to $\beta=3$, there exists a better trade off between the two composite factors.

To complete this investigation into the AS/EAS algorithm, it is necessary to consider the impact that elitist ants can have. The inclusion of elitist ants in the early workings of Dorigo et al. (1996) had a significant affect on the algorithm's behaviour. They achieved a $\approx$0.5% decrease in distance, with run times reducing to less than a ninth of those without the elitist ants.

The benefits associated with the run time reduction are somewhat negated in this investigation (though they may be of greater significance at a later date), but given their ability to improve solution quality as well, they are worthy of investigation.

The parameter *e* controls the number of elitist ants, and will undoubtedly have a significant impact on the quality of the solution achieved. Too few (or none) and the search may not be able to intensify on the promising areas of the search-space. Too many and they will cause the AS/EAS to stagnate (because of the autocatalytic nature of the algorithm). Stagnation can be a real problem with the AS/EAS as the algorithm does not contain a mechanism for alleviating this problem. More complex ACO implementations often include a variety of diversification measures that can react should stagnation occur.

All of the AS/EAS results that have been produced up to this point, were done so without the use of elitist ants i.e. one has set the parameter e=0. This investigation considers the impact of increasing this value, in a similar range to those discussed by Dorigo et al. (1996).

The results of this analysis can be seen in Table 3.5.

e		**c100**	**f71**	**tai75a**
0	Min	1048.35	339.62	2132.37
	Ave	1127.20	355.16	**2245.49**
	Max	1191.31	370.91	**2351.58**
4	Min	1087.07	**328.04**	2226.99
	Ave	1118.62	346.12	2315.96
	Max	1175.24	363.59	2385.36
8	Min	1060.01	330.24	2149.86
	Ave	1092.76	**345.97**	2299.42
	Max	1143.04	**354.73**	2474.91
12	Min	**1022.04**	346.56	**2131.80**
	Ave	**1082.77**	354.64	2287.64
	Max	**1127.88**	361.85	2405.74

Table 3.5: Analysing Various Values of e in AS

As an initial observation, it is clear that there is a large amount of variability apparent in the results (though there does not appear to be any obvious trend). Given that one requires the algorithm to be as robust as possible, one must give careful consideration to which parameter setting represents the best compromise across the three datasets.

The level of improvement evident in the c100 dataset is significant, with the greater the number of elitist ants, the higher the performance level. Unfortunately, as this trend does not appear to be evident when considering f71 and tai75a, it seems sensible to exclude the extreme values that were considered. With a preference for the less extreme values of e (i.e. 4 and 8), one needs to consider which of these performs best. It appears that e=8 performs to the highest standard (which also supports the findings of Dorigo et al. (1996)), and as such, is selected.

To complete the elitist ant analysis, it is important to check that the AS/EAS algorithm is not stagnating. To ensure that this is not occurring, a plot of the average distance produced by the m ants that cycle, is presented. Were stagnation to be occurring, one would expect this average distance to become constant (a straight line).

This plot can be seen in Figure 3.8 for ts_0 in the c100 dataset, whilst the best solution is also presented to show the algorithms success in producing good solutions.

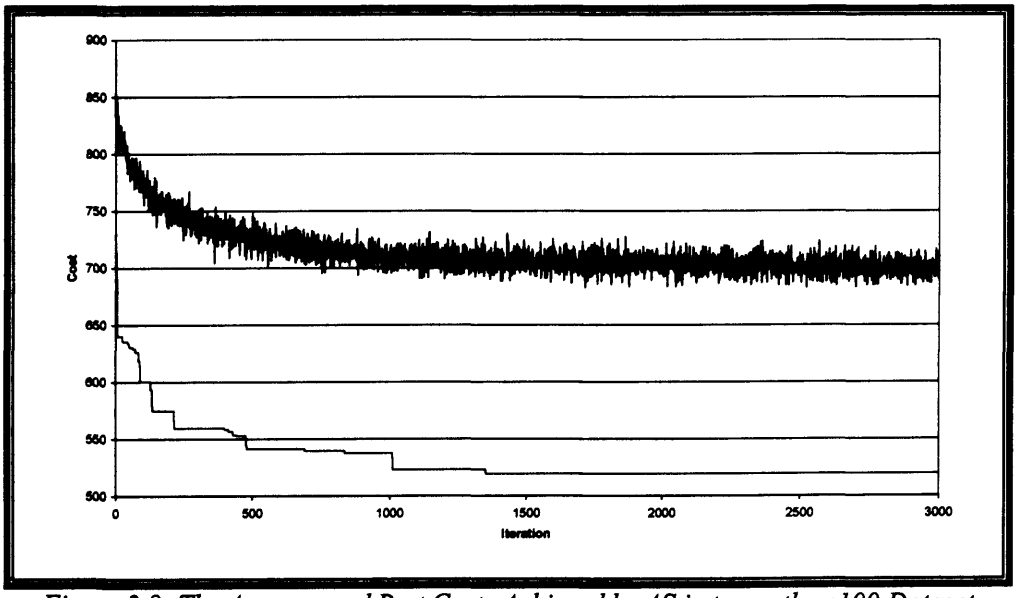

Figure 3.8: The Average and Best Costs Achieved by AS in ts_0 on the c100 Dataset

This plot demonstrates that there is still the desired variation in the results produced, and as such, for this time period and dataset, stagnation is not occurring. One can never be sure that this is representative of all timeslots, but this is symptomatic of the problem being solved.

This figure also identifies that improvements to the solution were still occurring up until just before halfway through the run. This enables one to have confidence in the parameter balance, as were improvements only occurring at the very beginning, it could indicate that the algorithm had placed too much emphasis on intensification. Furthermore, one can see that the selection of t=3000 adheres to the philosophy regarding giving the algorithm the necessary time to achieve all it can. This concludes the parameter review of the AS/EAS and the final parameters settings are now presented.

AS/EAS Final Parameters

The final parameters for this AS/EAS implementation are: ρ=0.5, β=3 and e=8 (whilst α=1, t=3000 and Q=1).

3.4.2 Greedy Randomised Adaptive Search Procedure (GRASP)

The basic theory behind the GRASP metaheuristic was discussed in some detail in Chapter Two. Consideration was given not only to its development as a two-phase metaheuristic, but also as to why it appears to be successful.

The key strength of the GRASP algorithm comes from the repetition of a technique that one might consider worthy of implementation just once. This repetition allows one to consider (according to Resende (1998)) GRASP as a sampling technique, whereby a population of good solutions are produced, and the best selected. This is obviously a sound principal, and one can understand how it represented a good starting point for this new algorithm.

Consideration was given to how this successful idea could be further improved, most noticeably through the control of the algorithms intensification/diversification balance. The most basic of the methods to control this is the restricted candidate list

(RCL), though more complex techniques based on memory functions and pools of successful solutions have also been considered.

The GRASP algorithm is a composite algorithm, whereby each cycle consists of a single construction and a subsequent improvement phase. Its classification under the old Osman and Hindi (2004) technique was greatly subjective, though in the adapted system (Figure 3.1) the procedure was greatly simplified. One could have considered it as a population technique as well, but as there is not always a transfer of knowledge from one solution to another it was felt unnecessary.

The first phase (solution construction) is made according to a slightly amended PNN technique, similar to that which was discussed in Section 3.1.1 (and shown in Equation 3.3). Before considering this amendment, consideration is given as to why one would use the PNN as opposed to the better performing NN.

In a single iteration one would prefer to remove the random element (allowing all choices to be made in a greedy manner), as it is likely to produce a solution that is of a higher quality than a single cycle of the PNN. However, when one is repeating the process there is a need for some form of diversification, hence the introduction of the random element.

As for the possible amendment, it allows for an adjustment to be made to the customers that have the potential to be selected as the next customer to be appended. As in the AS/EAS implementation, the set F contains all feasible expansions and thus any selection of the next customer to visit must belong to this set. However, one need not have a selection being made directly from this set, they are free to impose a greater level of restriction on this if they so wish.

Instead of allowing choices to be made from the entire set F, through the use of a RCL, one can ensure that the selection comes from a smaller subsection of this. The RCL can either limit the selection to some fixed number (cardinality) e.g. the seven best customers (as seen in Pitsoulis and Resende (2001)) , or they can limit selection to a certain proportion e.g. only select from the best 25% of customers.

Given the variable nature of the problem sizes being solved (in the various timeslots); the second, more changeable RCL definition seems better suited. Equation 3.8 demonstrates how the parameter (*RCL*) will control the proportion of customers that are available for consideration.

$$NewSize(F) = \frac{OldSize(F)}{RCL} \tag{3.8}$$

where *OldSize* is the number of elements in the set F using the previous definition, and *NewSize* is the reduced number of elements in the set F when an RCL is used.

This newly introduced parameter has been assigned the moniker *RCL*. Traditionally one would use the parameter n, but this was not considered due to its presence in the previously considered AS/EAS algorithm. The higher the value of the user-defined variable *RCL*, the greater the level of restriction placed on the build process.

The algorithm is run until t (a user-defined parameter that acts in the same way as it does in the AS/EAS algorithm) cycles are completed. However, one must now be aware that the number of cycles now equals the number of solutions that are produced. GRASP does not use a parameter m (or alternatively one could say m=1) to denote the number of solutions each cycle, as there is no benefit from producing multiple solutions per cycle due to their independence.

Pseudo code detailing the workings of this GRASP metaheuristic for the DVRP can be seen in Figure 3.9.

With the algorithm suitably introduced and the necessary details clarified, one only requires a set of default parameters to implement the algorithm.

```
Input: Problem Data and Initial Solution
Output: Vector of Nodes and Total Cost

While number of cycles completed is < t
    While all customers have not been allocated
        While F ≠ ∅
            Probabilistically select a customer to append to the current vehicle from
            the best (F/RCL), where the term best relates to minimising distance.
        End While
        Deploy new vehicle
    End While
    Descend using FI descent strategy (Pseudo-Code presented in Figure 3.4)
End While
```

Figure 3.9: GRASP Pseudo Code

One could use the literature to identify suitable parameters, but there appears to be a lack of CVRP implementations to use as a base. Although the VRPTW is a similar problem, the descent phase is a much more complicated affair (as each move needs to satisfy the time window constraints) and thus is unlikely to provide a sensible parameter setting. However, given the similarities between GRASP and the standard descent algorithm, one can use this as a method of identifying one of the necessary parameters.

The results that are produced via the PNN construction are likely to be of a lower quality than those produced via the NN algorithm, with which the descent algorithm was previously run. However, given that the parameter descent exit parameter (z) is defined as the number of iterations without improvement, it should be suitable even if the initial solutions are worse.

The descent implementation does little to suggest what parameter settings should be used for the number of cycles completed (t) and the RCL (RCL). One requires a default parameter for only one of these, while the other can be assessed in the first stage of the parameter review. Given the importance of allowing the algorithm to perform to the best of its ability, it seems sensible to begin by optimising t.

With this decision made, one needs a suitable value for *RCL* with which the analysis can begin. Starting with the no restriction on the build process (*RCL*=1), should result in the greatest level of variability in the results (pre-descent) and should therefore operate as a worse case with regards to identifying a suitable value for *t*.

This completes the selection of the default parameters; the algorithm can now be optimised.

GRASP Parameter Optimisation

In its basic guise (which this most certainly is), one does not need to concern oneself with the discussion of memory functions or the creation of pools of successful solutions. The strength of this GRASP algorithm comes primarily from its repetition, and as such, it is not a particularly parameter heavy algorithm (in contrast to the AS/EAS algorithm). There are only the three parameters for which default values were considered.

Given one of these parameters has already been successfully identified in the basic descent implementation (*z*) only *t* and *RCL* need be considered. For the reasons previously identified, the parameter *t* is to be considered first.

As the GRASP metaheuristic is a local search/construction amalgam, it might seem sensible to initially match the value of *t*, to that of AS/EAS algorithm (another algorithm utilising a construction element). However, due to the lack of a learning process i.e. the first solution is as likely to be the best produced as the last, it is clear that substantially less runs are required. The descent element of the GRASP metaheuristic further negates the need for very high values of *t*, as even initial poor solutions can be transformed into competitive ones. A variety of potential values for *t* have been analysed, and the results can be seen in Table 3.6.

It is evident that there is a trend towards the higher values of *t*, though one would expect this behaviour. However, if one were to continue to increase the value of *t*, there would come a point at which no further improvements would be identified. One would expect *t*=1000 to perform at a higher standard than *t*=500, though the results do

not support this. With this is mind, it appears that this limit may well have been reached somewhere between these values.

t		c100	f71	tai75a
100	Min	1109.04	314.87	1836.29
	Ave	1168.89	332.50	1967.54
	Max	1214.60	355.98	2085.37
200	Min	1074.02	314.63	1855.68
	Ave	1129.66	322.41	1965.87
	Max	1194.54	338.95	2115.43
500	Min	1061.02	**300.28**	**1759.79**
	Ave	1097.94	**320.09**	**1928.45**
	Max	**1136.07**	**327.15**	**2044.39**
1000	Min	**1016.77**	308.30	1817.84
	Ave	**1080.48**	320.72	1963.22
	Max	1136.89	330.61	2059.33

Table 3.6: Analysing Various Values of t in GRASP

With the knowledge that t=500 cannot theoretically perform better than t=1000, the selection of t=1000 seems sensible. This should it give the algorithm the necessary number of runs to achieve the best results possible. Note that the selection of t=1000 supports the earlier claim that this GRASP implementation will require less cycles to achieve its best solutions than the AS/EAS algorithm did.

With a suitable exit criterion identified, the impact that *RCL* will have on the construction phase of the GRASP algorithm is considered. Candidate lists, as described previously, limit the possible arcs that can be appended to an incomplete solution. The higher the value of *RCL*, the more restrictive the build process is. A good selection will rule out the possibility of selecting arcs that are unlikely to be included in a good solution. However, should too high a value of *RCL* be selected, the search will intensify too much and results will not contain the necessary diversity.

To gain an understanding of the impact that the various values of *RCL* will have upon the build process, the descent phase has been removed and the first time period ts_0 has been investigated. Using the previously determined t=1000, 100 results are sorted into

class intervals of 50, 20 and 100 (for datasets c100, f71 and tai75a respectively) and the frequency with which the results are achieved, is considered (see Figure 3.10)[9].

Datasets c100 and f71 show an obvious improvement when a candidate list is introduced; these plots show a trend towards the higher, more restrictive values of *RCL*. Unfortunately, the final dataset (tai75a) does not seem to behave in the same manner. It also appears that the selection of *RCL*=20 is too restrictive, with result quality beginning to deteriorate in datasets f71 and tai75a.

The next highest trio of values (5, 10 and 15) have been chosen for a full trial, with the results of *RCL*=1 (as previous analysed) included for comparative purposes (see Table 3.7). Note that the descent component has been reintroduced at this point.

The introduction of an *RCL* (of any size) has reduced the average cost for datasets c100 and f71. This validates the trends first identified in Figure 3.10. The performance on the tai75a dataset is not so encouraging, no obvious trends were identified in the preliminary graphs, and results appear to show a reduction in solution quality at the higher, more restrictive values of *RCL*.

The need for a compromise is obvious; the least restrictive of the RCL values i.e. *RCL*=5 will improve the performance on c100 and f71 without being so restrictive that tai75a will suffer. This concludes the parameter review of GRASP and the final parameter settings are now presented.

GRASP Final Parameters

The final parameters for this GRASP implementation are: t=1000 and *RCL*=5 (whilst z=3000).

[9] Individual histograms would have been statistically correct (as these graphs should not have been continuous), but one would not have been able to draw the necessary conclusions.

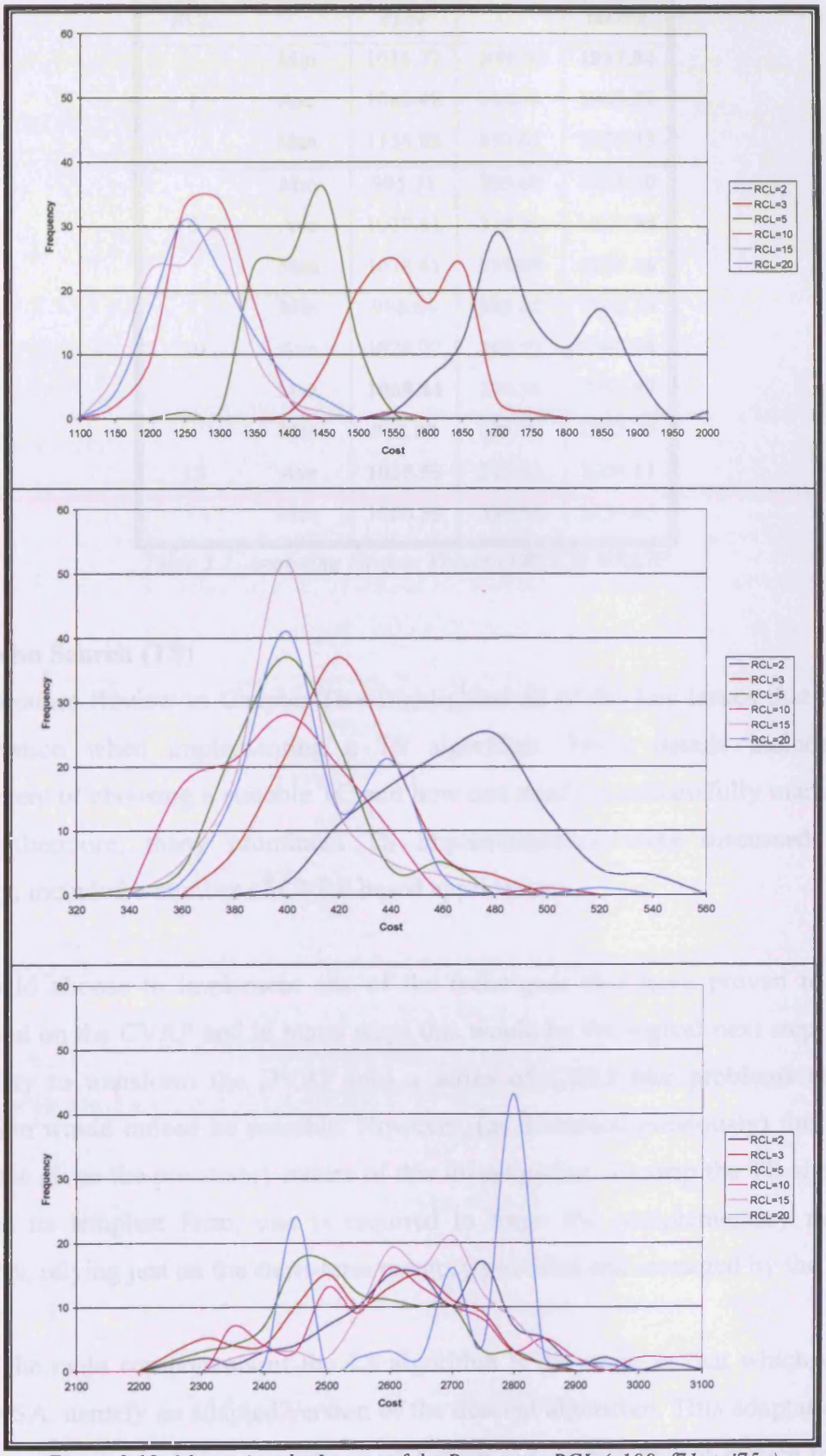

Figure 3.10: Measuring the Impact of the Parameter RCL.(c100, f71, tai75a)

RCL		c100	f71	tai75a
1	Min	1016.77	308.30	**1817.84**
	Ave	1080.48	320.72	1963.22
	Max	1136.89	330.61	2059.33
5	Min	995.51	305.08	1893.30
	Ave	1039.51	314.10	**1961.94**
	Max	1078.61	**319.90**	**2051.36**
10	Min	968.64	**303.42**	1882.54
	Ave	1029.27	**313.72**	2007.54
	Max	**1068.44**	334.56	2092.87
15	Min	**963.08**	306.36	1893.74
	Ave	**1025.50**	315.91	2000.11
	Max	1086.39	330.66	2130.65

Table 3.7: Analysing Various Values of RCL in GRASP

3.4.3 Tabu Search (TS)

The Literature Review in Chapter Two highlighted all of the key issues that require consideration when implementing a TS algorithm. These details included the requirement of choosing a suitable TC and how one needs to successfully manage the TL. Furthermore, many prominent TS implementations were discussed which crucially, included a number of CVRP based applications.

One could choose to implement one of the techniques that have proven to be so successful on the CVRP and in many ways this would be the logical next step. Given the ability to transform the DVRP into a series of CVRP-like problems such an adaptation would indeed be possible. However, (as discussed previously) this would be unwise given the precursory nature of this investigation. To strip the TS algorithm down to its simplest form, one is required to forgo the complementary memory structures, relying just on the short-term memory provided and managed by the TL.

One of the main components of the TS algorithm is the same as that which will be used by SA, namely an adapted version of the descent algorithm. This adaptation (for both algorithms) comes via acceptance of certain moves that do not reduce the cost function. Not only do they obviously differ in how they do this, (described in Chapter Two), but they make use of different descent strategies. The TS algorithm makes use

of the BI descent strategy (developed by Glover (1989) and Hansen and Jaumard (1990)), in contrast to the FI descent strategy employed by SA and the earlier descent algorithm. The differences between these descent strategies have been presented in Section 3.3.2, and as such, are not recounted here.

Before one can proceed with the identification of a default set of parameters (and the subsequent optimisation of these), one must identify an appropriate TC with which the TS algorithm can successfully distinguish between solutions.

Tabu Characteristic

As stated in Chapter Two, it is common practice to remember some characteristic that can be used to identify a previous solution, rather than storing entire solutions. However, unless this characteristic uniquely identifies a specific solution (difficult to achieve without storing a large number of attributes), unvisited regions of the search-space will also be declared tabu. Though this in itself is not a significant problem (as one can never explore the entire search-space), the selection of what one stores is crucial to the success of a TS algorithm.

A very basic TC is now proposed.

Prior to a move being made, it is necessary to check whether this move will result in the search returning to the previously visited region of the search-space. With this in mind, it seems sensible to store information relating to the move that has just happened i.e. the following information:

- The vehicle from which a customer was moved.
- The customer number.
- The place within the vehicle from which they were moved.

Storing this information results in an entry in the TL taking the form ($P_{vehicle}$, $P_{customer}$, $P_{position}$), where the P relates to the previous solution. The first two details recorded ($P_{vehicle}$ and $P_{customer}$) strongly resembles the workings of Archetti et al. (2006) where

'if a customer, say i, is removed (inserted) from a route, say r, then it is tabu to insert (remove) i in r for a certain number of iterations'.

The decision to supplement this technique with the $P_{position}$ was not taken lightly, as it will complicate the procedure (discussed below). However, were one to stick with the existing technique of the TC preventing customers returning to vehicles that they had previously belonged, one would greatly restrict the search. The added complications of including the customer position were felt necessary to give the TS algorithm the potential to achieve the best solutions it could.

The quote from Archetti et al. (2006) suggests that information can be stored about the route to which a customer is inserted, as well as removed. This will help identify the previous solution as well as the current one, acting as a sort of safety net for the technique. This system of information storage has been assigned the moniker dual storage, and will be utilised in this implementation.

Therefore, the TL will also contain the three pieces of information presented previously ($P_{vehicle}$, $P_{customer}$, $P_{position}$) as well as the similar details ($C_{vehicle}$, $C_{customer}$, $C_{position}$), of the current solution, note that $C_{customer}=P_{customer}$.

An example of this technique preventing a previously visited solution from being revisited (cycling) in both the i:1,0 and x:1,0 components of the 1-Opt neighbourhood can be seen in Figure 3.11.

Vehicle 1

Position

Iteration	1	2	3	Store 1	Store 2
1	a	b	c		
2	c	a	b	(1,c,3)	(1,c,1)
3	a	b	c		

Iteration	Vehicle 1			Vehicle 2			Store 1	Store 2
	Position			Position				
	1	2	3	1	2	3		
1	a	b	c	d	e			
2	a	c		d	b	e	(1,b,2)	(2,b,2)
3	a	b	c	d	e			

Figure 3.11: Examples of the TL Preventing Cycling, i:1,0 and x:1,0.

The way in which the TL (with this specific TC) prevents cycling is very similar for the i:1,0 and x:1,0 moves. The examples for both neighbourhood components, begins

with a current solution, denoted iteration 1. One need not be concerned with how this solution was identified (it could be the result of a construction heuristic, or midway through the search), what is important is the move that takes them to iteration 2.

In both cases new solutions are obtained and information (according to the TC) about the previous solutions and current solution are entered into the TL. This information is then stored and when one attempts iteration 3, the moves that will take the solution back to the solution shown in iteration 1, cannot be selected (as indicated in red) as they are deemed tabu.

However, when investigating these neighbourhoods, it becomes clear that there is another form of cycling that requires consideration. This kind of cycling is subtler, and unfortunately is not prevented via the existing TC. This was only observed in the i:1,0 component of the 1-Opt neighbourhood, with the x:1,0 component seemingly unaffected.

It occurs in an indirect manner i.e. a move is not undone directly, but a previous solution is revisited due to a series of moves that indirectly recreate it (see Figure 3.12).

		Vehicle 1 Position				
		1	2	3	Store 1	Store 2
Iteration	1	a	b	c		
	2	c	a	b	(1,c,3)	(1,c,1)
	3	a	c	b	(1,a,2)	(1,a,1)
	4	a	b	c	(1,b,3)	(1,b,2)

Figure 3.12: Example of Indirect Cycling Occurring (i:1,0)

This example shows how a solution that has previously been visited can be revisited via a series of i:1,0 moves. The solutions in iterations 1 and iteration 4 are identical, and from looking at the information in the Store 1 and Store 2 columns, one can see that no moves that were declared tabu have been made. Unfortunately this

demonstrates that this TC is susceptible to cycling, the very thing short-term memory is attempting to counteract.

This is a difficulty associated with the inclusion of an intra vehicle component in the neighbourhood, and there is little that one can do to prevent this behaviour from occurring (with this TC). One could increase the length of the TL (controlled by the parameter *LS*) and allow repetition to occur; one would then store these details making it less likely that it could happen again. However, an increase in *LS* is likely to present other problems. If the TL is too long, then regions of the search-space are considered tabu for too long and good solutions may well be missed.

Although not ideal, a split is proposed between the two components of the neighbourhood. It is proposed that only the x:1,0 moves being made are party to the tabu procedure. The split allows *xt* (a user defined parameter) moves to be made of a x:1,0 type, before descending to a i:1,0 local optimum (BI strategy). Any moves made in this secondary phase are not subject to the constraints imposed by the TL and furthermore, do not count towards the number of iterations required before an item is removed from that list.

This algorithm is executed until a termination criterion is met, in this case defined by the total number of x:1,0 moves, controlled by the user-defined parameter *l*. Upon completion, the best solution achieved is returned as the final solution. Pseudo code detailing this algorithm is now presented (Figure 3.13).

One should note that the split in the neighbourhood is far from ideal and it is likely that the solution quality may suffer as a result of the enforced rigid interplay of the two neighbourhood components. Were this metaheuristic to be chosen for further investigation a TC that allows the different neighbourhood components to work together will need to be developed.

With all of the necessary explanations in place, one only requires a set of default parameters before the TS algorithm can be implemented. Referring back to the

philosophy that time issues are not of paramount importance, the number of iterations (denoted by *l*) is initially set arbitrarily high at 15000[10].

```
Input: Problem Data and Initial Solution
Output: Improved Solution

While  Termination criteria not met
       While  number of iterations completed is less than xt
                     Compute f(x') ∀ x' ∈ N(x)  (where N(x) is restricted to x:1,0)
                     Select x' such that f(x') is minimised (Tabu moves not included)
                     Designate x' to be the new x
                     Update tabu list (Previous and Current solutions)
       End While
       Descend to i:1,0 local optimum (BI strategy)
End While
```

Figure 3.13: TS Pseudo Code

With l=15000 and xt=1 (the simplest case) it is possible to begin the optimisation process.

Tabu Search Parameter Optimisation

This particular TS algorithm relies on a trio of parameters, all of which require optimisation. The obvious parameter to consider first is *LS*, the size of the tabu list. This is critical to the algorithms success; too low and the algorithm will cycle, too high and promising areas of the search-space will not be considered. Once a suitable value for *LS* has been identified, complementary values of l and xt can be established via further parameter assessments.

This parameter (*LS*) represents the number of moves that are made, until that move can be repeated. Complex implementations make use of a dynamic *LS* (a parameter that can change), adapting to meet the requirements of the search. Although not a particularly complex idea to investigate, given the simplistic nature of this implementation the decision has been made to assume a fixed value for *LS*.

[10] l moves will actually be made in TS, as opposed to simulated annealing where l represents the number of potential moves.

One would expect this to not have a dramatic effect on solution quality; many highly successful TS implementations use a fixed value. Glover (1989) noted, from an empirical study, that the list size *'has a highly stable range of values that both prevent cycling and lead to remarkably good solutions'*.

The proposed values for *LS* are 20, 30, 40 and 50, with the relevant results presented in Table 3.8. It should be noted that due to the dual storage technique the number of solutions that are actually prevented from repetition are *LS*/2.

LS		c100	f71	tai75a
20	Min	1176.43	332.76	2287.49
	Ave	1380.17	429.62	2655.43
	Max	1633.75	526.90	3073.95
30	Min	1117.43	329.95	2153.79
	Ave	1174.63	**352.72**	2455.09
	Max	1273.76	397.45	2807.56
40	Min	**1033.73**	**318.78**	**1978.07**
	Ave	1104.10	355.59	**2226.14**
	Max	1193.60	**395.74**	**2441.89**
50	Min	1040.28	331.36	2163.39
	Ave	**1103.02**	352.98	2298.50
	Max	**1157.60**	409.92	2539.47

Table 3.8: Analysing Various Values of LS in TS

These results suggest that a selection of *LS*=40 produces significantly better results than the other values that have been assessed. It is interesting to note that *LS*=50, although highly competitive on two of the datasets (within this assessment), appeared to be too restrictive to deal with the tai75a dataset.

With a suitable value of *LS* successfully identified, the parameter *l* (currently set at 15000) needs to be investigated. Although this was arbitrarily chosen to be a high value, the run times were sufficiently low that it seems sensible to assess higher values in the hope of identifying some level of improvement. To see if the algorithm could benefit from a larger number of iterations, various other values for *l* (all multiples of 15000) have been tested, with the results presented in Table 3.9.

There is a high level of variability apparent in these results, with no obviously best performing value. One could argue that there is a slight improvement in solution quality as *l* increases in datasets c100 and f71, though tai75a suggests otherwise. The logic behind the technique supports the notion that the higher the value of *l*, the better the chance of identifying good results, until some saturation point is reached (performance will not decrease after this point has been reached). With this in mind, *l*=45000 is chosen, it should provide the algorithm all the time necessary to locate the best solution it can.

l		**c100**	**f71**	**tai75a**
15000	Min	1033.73	318.78	1978.07
	Ave	1104.10	355.59	**2226.14**
	Max	**1193.60**	395.74	**2441.89**
30000	Min	1048.40	319.50	2107.47
	Ave	1111.62	**341.75**	2442.96
	Max	1205.72	**371.15**	2763.18
45000	Min	**1001.81**	**313.72**	**1975.49**
	Ave	**1095.72**	344.04	2372.34
	Max	1322.7	384.43	2863.76

Table 3.9: Analysing Various Values of l in TS

To conclude the parameter review it is necessary to investigate the impact of the parameter *xt*. This is an unusual parameter to assess, as it would not be needed were a TC selected that was not susceptible to the indirect cycling as seen in Figure 3.11. The current parameter setting investigates each vehicle individually after each and every x:1,0 move, (returning the solution to a i:1,0 local optimum). This is not ideal, as it reduces the potential movement through the search-space, but it may still represent the best that can be done with this TC.

Various values of *xt* are now investigated (see Table 3.10), where the higher the setting, the more diversified the search i.e. *xt*=3 means 3 x:1,0 moves are made before returning to the i:1,0 neighbourhood, and the associated local optimum.

These results appear fairly conclusive, with *xt*=1 appearing to be the best performing parameter selection i.e. alternating between x:1,0 TS moves and i:1,0 descent moves.

With the identification of an optimal value for *xt*, the parameter review of this TS implementation is completed.

xt		c100	f71	tai75a
1	Min	**1001.81**	**313.72**	**1975.49**
	Ave	**1095.72**	**344.04**	2372.34
	Max	1322.70	**384.43**	2863.76
3	Min	1046.98	331.11	1999.61
	Ave	1132.00	372.62	**2329.31**
	Max	1241.66	434.85	**2811.09**
5	Min	1054.26	329.52	2143.35
	Ave	1120.74	361.47	2420.26
	Max	**1221.39**	406.81	2843.09

Table 3.10: Analysing Various Values of xt in TS

To complete the analysis, it is interesting to look for signs of cyclic behaviour within the local search component of the algorithm. If the TC is working correctly then for the length of TL there should not be any repetition of solutions, as they will be declared tabu. Furthermore, if a suitably large value of *LS* has been selected, the search should have deviated enough, such that by the time certain moves are removed from the TL solutions would not be revisited anyway.

Various timeslots (across all the datasets) were graphed, with almost all showing no signs of cycling. Interestingly, there did appear to be signs of cycling in two of the earlier periods of the c100 dataset. An example of this can be seen in Figure 3.14.

The cycling appears to be happening after a fairly substantial number of moves (over 9000), but as a portion of the 45000 one could potentially consider this to be early on. This results in 36000 essentially wasted iterations, whereby the search never leaves the cycles from which it has become trapped.

When repetition occurs, one is immediately drawn to the period size, defined as the number of iterations over which the repetition occurs. This graphs period is 20, thus after 20 moves (of the x:1,0 neighbourhood), the next move returns the search to the solution that was found 20 iterations ago. This is as a result of the selection of *TL*=40,

which due to the dual storage technique actually means that moves are stored for only 20 iterations.

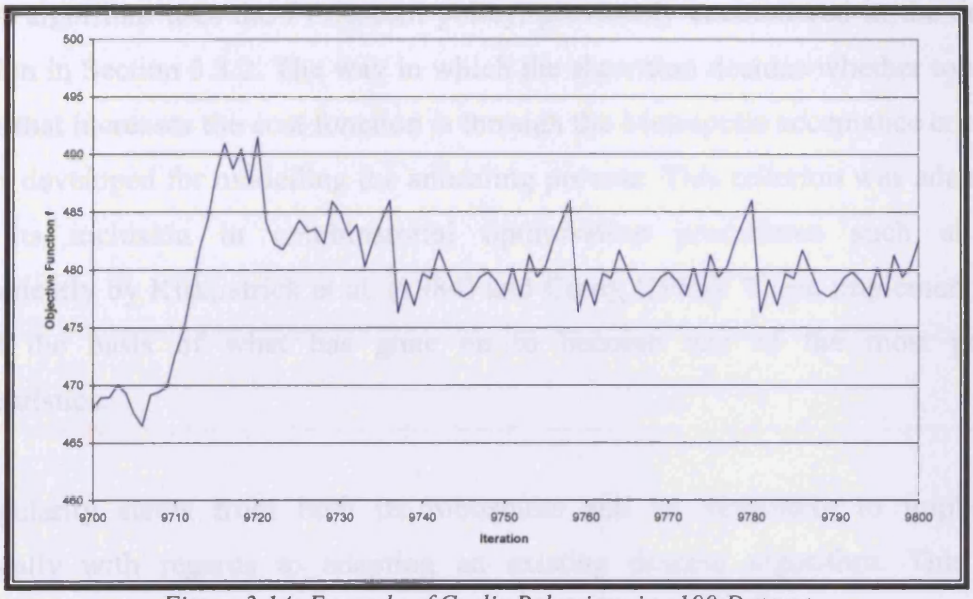

Figure 3.14: Example of Cyclic Behaviour in c100 Dataset

The selection of TL=40, is clearly not large enough for this timeslot in this dataset, but one should not be overly concerned as solutions of a relatively high quality were identified prior to the cyclic behaviour.

The current TC and specifically the *LS* has prevented cycling from iteration to iteration but may not have completely eradicated indirect cycling. The low number of instances of cycling suggests that this is not a particularly significant problem; moreover, one could put it down to the result of an unfortunate set of circumstances. However, with that said, were research to be continued in this area, it is a certainly an issue that would require addressing. This concludes the parameter review of the TS algorithm, and the final parameters settings are now presented.

Tabu Search Final Parameters

The final parameters for the TS algorithm are: LS=40, l=45000 and xt=1.

3.4.4 Simulated Annealing (SA)

As the Literature Review showed, the primary distinction between the SA metaheuristic and the basic descent algorithm is the ability to accept moves that are

counter productive in the short term. Although in theory this is a very similar idea to the TS algorithm, the way in which this is done is very different.

The SA algorithm uses the FI descent policy, previously encountered in the descent algorithm in Section 3.3.2. The way in which the algorithm decides whether to accept a move that increases the cost function is through the Metropolis acceptance criterion; initially developed for modelling the annealing process. This criterion was adapted to allow its inclusion in combinatorial optimisation procedures such as this, independently by Kirkpatrick et al. (1984) and Cerny (1985). These implementations formed the basis of what has gone on to become one of the most popular metaheuristics.

Its popularity stems from both its robustness and its simplicity to implement, specifically with regards to adapting an existing descent algorithm. This basic implementation does just this, simply taking the previously described FI descent strategy and introducing the Metropolis criterion. The following details are summarised from the workings of Gendreau et al. (2002).

An iteration of simulated annealing involves a solution x' being drawn from $N(x_l)$. If $f(x') < f(x_l)$ then x_{l+1} is set equal to x'; otherwise a decision is made according to Equations 3.9 and 3.10.

$$x_{l+1} = \begin{cases} x' & \text{with probability } p_l \\ x_l & \text{with probability } 1 - p_l \end{cases} \tag{3.9}$$

where p_l is calculated in accordance with equation 3.10, within which t=temperature.

$$p_l = \exp\left(\frac{-(f(x') - f(x_l))}{t}\right) \tag{3.10}$$

One common issue with SA is the lack of user control over the algorithm itself. Obviously the selection of an appropriate neighbourhood is critical to the algorithms success, but beyond that choices are somewhat limited. The one area with which one

can alter the basic SA model is through the parameter t, which simulates the temperature. This parameter requires initialisation and termination values, t_i and t_t respectively, and some method of changing from one to the other, known as the cooling schedule.

Chapter Two, identified many different techniques that have been proposed for controlling the cooling schedule: including geometric, Lundy and Mees (1986) and Aarts and Korst (1989). Given that this is supposed to be a stripped down version of the algorithm, it seems appropriate to make use of the basic geometric reduction function. This requires a parameter α to be selected (α<1), and is shown in Equation 3.11.

$$t=\alpha(t) \tag{3.11}$$

Pseudo code for this SA implementation can be seen in Figure 3.15.

Input: Problem data, initial solution, α, t_i and t_t
Output: Improved solution

While temperature $(t)<t_t$
  While number of iterations<l
    Randomly select $x' \in N(x)$
    If $f(x') < f(x_l)$ then
      $x_{l+1} = x'$
    Else
      If rand $< \exp\left(\frac{-(f(x')-f(x_l))}{t}\right)$ then
        $x_{l+1} = x'$
      End If
    End If
  End While
  $t=\alpha(t)$
End While

Figure 3.15: SA Pseudo Code

This concludes the descriptive section for the SA algorithm, and attention can now be given to the selection of a default set of parameters. As explained in the algorithms description and pseudo code, there are four user-defined parameters within this SA implementation, namely l, α, t_i and t_t.

Due to the philosophy of giving each algorithm as much time as it needs to perform to the best of its ability it seems sensible to initialise the temperature high and terminate low. As long as an appropriate cooling schedule (i.e. the way the temperature is reduced) is selected, then this should allow the algorithm the necessary time to demonstrate its ability.

Preliminary runs were made and they supported this assumption, and t_i and t_t were fixed at 100 and 0.1 respectively. Interestingly, by fixing the values of t_i and t_t it allows one to control the remaining parameters (l and α) in such a way that all of the runs are analysed across a similar number of iterations.

Simulated Annealing Parameter Optimisation

As SA is operating using the FI type descent moves i.e. moves are assessed individually rather than investigating the complete neighbourhood (BI), the total number of iterations may need to be considerably higher than it was in the TS algorithm. The parameter l represents a different measure of control within the two descent policies; here it represents the number of attempted moves (whether accepted or not) at each temperature, as opposed to the number of actual moves.

As the starting temperature and exit temperature are fixed at 100 and 0.1 respectively, one can calculate, for a specific cooling rate, the number of temperature reductions (t_r) used to meet the termination level t_t. This allows the total number of iterations ($l \times t_r$) to be fixed for varying values of l and α.

Literature suggests that high values of α, corresponding to a slow cooling rate, will perform best. Results of this analysis can be seen in Table 3.11.

l	t_r	$l \times tr$	α		c100	f71	tai75a
				Min	1034.99	304.14	1923.85
2500	688	1720000	0.99	Ave	1100.20	**315.53**	2312.25
				Max	**1149.93**	**334.09**	**2457.30**
				Min	1035.46	307.76	2038.42
6300	273	1719900	0.975	Ave	1108.51	343.10	2292.05
				Max	1219.29	421.36	2495.31
				Min	1039.41	306.28	**1849.77**
12740	135	1719900	0.95	Ave	1095.19	336.13	2318.43
				Max	1152.04	356.79	2720.08
				Min	**1016.29**	**294.23**	2011.09
26060	66	1719960	0.9	Ave	**1094.15**	321.84	**2194.79**
				Max	1179.67	339.53	2721.79

Table 3.11: Analysing the Relationship between l and α in SA

There does not appear to be an obvious trend in these results, and as such any selection is going to be more subjective than one would like. However, it does appear that the lowest value tested (α=0.9) outperforms the others. Though it is possible that with a greater number of iterations (l) the slower cooling rates would work better, it seems sensible to continue with the best-performing at this stage.

One must always select the order in which the parameters are optimised and as such, this is a common problem with many metaheuristics. Interestingly, the highest value tested (α =0.99) would appear to be the second best parameter selection which may suggest a lack of sensitivity associated with the parameter α (within this specific implementation).

With an appropriate value of α selected, one needs to test whether the algorithm is being given sufficient time at each temperature level, such that the philosophy of giving the algorithm the best chance possible is being adhered to. This is controlled by the parameter l, and the results of this investigation can be seen in Table 3.12.

The first conclusion that one can draw from these results is that there appears to be little benefit (in terms of solution quality) associated with an increase in the number of iterations at each temperature. This suggests that the initial setting of l=26060 is sufficient, and thus no alteration is made to this parameter. It is possible that this is

due to the fact that the cooling rate was optimised with respect to this number of iterations, but one is always going to be faced with problems of this nature. A similar problem (though reversed) would be encountered, if the number of iterations at each temperature was considered before the cooling rate.

l	t_r	$l \times tr$	α		**c100**	**f71**	**tai75a**
				Min	**1016.29**	**294.23**	2011.09
26060	66	1719960	0.9	Ave	1094.15	321.84	**2194.79**
				Max	**1179.67**	339.53	2721.79
				Min	1033.77	305.04	**1988.21**
52120	66	3439920	0.9	Ave	1112.34	322.40	2252.84
				Max	1276.99	347.14	2509.73
				Min	1041.06	316.10	2029.94
104240	66	6879840	0.9	Ave	**1092.32**	**320.80**	2227.34
				Max	1212.83	**331.55**	**2487.11**

Table 3.12: Analysing Various Values of l in SA

As with the TS algorithm, it is important to check that the search is behaving in the manner that one expects. Although cycling is not an issue for SA (due to the FI descent policy) a plot of the search can be very informative with regards to quenching.

The SA algorithm is said to have quenched if the temperature has been reduced too quickly, causing the search to become stuck in a particular region of the search-space. A plot of the cost function allows one to see if this has happened. If it has, the cost will stagnate early in the search procedure.

Figure 3.16 shows the behaviour of this SA algorithm for timeslot ts_0 in the c100 dataset.

The initial area of interest in this graph are the fact that there appear to be a considerable number of increasing moves in the earlier iterations, with the size of these moves reducing as the number of iteration increases. This is the behaviour one would expect when a SA algorithm is performing as desired. The large amount of variability at the beginning is due to the high initialisation temperature.

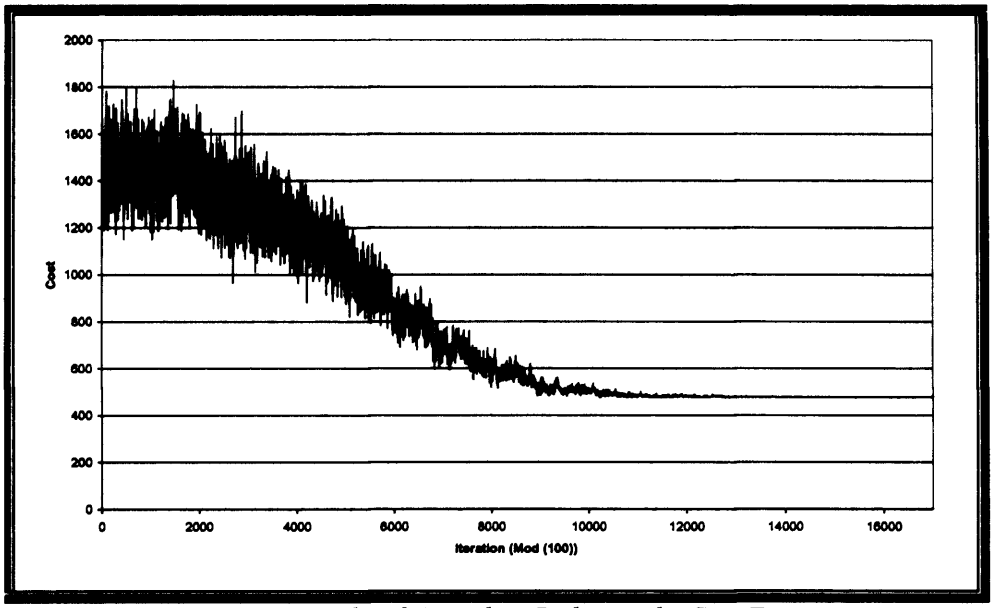

Figure 3.16: Simulated Annealing Reducing the Cost Function

Given that increases are still being made after a significant number of iterations have been completed, one can assume that quenching is not a problem with this example. This is further supported by the fact that the best solution achieved does not occur early on in the search, something one might expect if the search had become trapped. This concludes the parameter review of the SA algorithm and the final parameter settings are now presented.

Simulated Annealing Final Parameters

The final parameters for the SA algorithm are: t_i=100, t_f=0.1, α =0.9 and l=26060.

3.5 Directions for Further Research

It is possible to view this as an entirely subjective issue, and as such one cannot make a wrong decision. The results that have been produced throughout this chapter were (as according to the introduction of this chapter) made with a purpose of basing the decisions on the performance of the specific techniques.

However, through the workings of this chapter further (and equally important) details that will influence ones decision have come to light. An understanding of what it is like to implement these algorithms on such an evolved and complex problem as the DVRP has been achieved.

This section is divided into three subsections, the contents of each is now presented.

Firstly, and perhaps most obviously, one is required to discuss the relative competitiveness of the algorithms in their basic guise. One should note that for only one of the basic implementations (see Section 3.5.1) was the performance comparable (in terms of solution quality) with the results of the ACS-DVRP.

Secondly, and perhaps more interestingly, the basic implementations have allowed one a glimpse into the potential benefits and pitfalls associated with one's attempts to solve this problem using various metaheuristics. It is likely that were a researcher to produce a more complex metaheuristic implementation, they will be faced by numerous further problems. One must also consider their particular interest in certain areas, as investigating any of these metaheuristics is likely to be an sizable undertaking.

Finally, the ideas in these last two subsections are collated and decisions are made as to what direction the research should take.

3.5.1 Metaheuristic Performance Comparison

The four metaheuristics (AS/EAS, GRASP, TS and SA) have all been optimised with regards to performance (not time) and as such, results should be of the highest quality achievable with these basic implementations. The latter three of these metaheuristics all rely on a similar local search element; the somewhat primitive 1-Opt algorithm, as used by the ACS-DVRP. Given the existence of such a common element, comparisons between these three algorithms should be particularly interesting.

The results achieved by each of the metaheuristics (with the parameter settings that were presented at the end of each parameter review) are presented in Table 3.13.

These results clearly show that the GRASP metaheuristic is the best performing of the four that have been assessed; eight of the nine metrics achieved their lowest cost with this metaheuristic. In second place one could argue the merits of either AS/EAS or SA, based on issues of better individual results or minimising the variability. In last

place is the TS algorithm, it is clearly the worst performing, with poor best results and high levels of variability.

Met'tic		c100	f71	tai75a
AS/EAS	Min	1060.01	330.24	2149.86
	Ave	1092.76	345.97	2299.42
	Max	1143.04	354.73	2474.91
GRASP	Min	**995.51**	305.08	**1893.30**
	Ave	**1039.51**	**314.10**	**1961.94**
	Max	**1078.61**	**319.90**	**2051.36**
TS	Min	1001.81	313.72	1975.49
	Ave	1095.72	344.04	2372.34
	Max	1322.70	384.43	2863.76
SA	Min	1016.29	**294.23**	2011.09
	Ave	1094.15	321.84	2194.79
	Max	1179.67	339.53	2721.79

Table 3.13: Comparison of Various Metaheuristics Implemented on the DVRP

The GRASP metaheuristic is performing to such a standard that one ought to compare it to the level of performance of the ACS-DVRP[11]. Both techniques make use of the same local search neighbourhood, with what one could assume to be better initial solutions being constructed via the ACO implementation.

The results for the ACS-DVRP (for the test datasets) are presented in Table 3.14.

Met'tic		c100	f71	tai75a
ACS-DVRP	Min	973.26	311.18	1843.08
	Ave	1066.16	348.69	1945.20
	Max	1100.61	420.14	2043.82

Table 3.14: ACS-DVRP Results

[11] Comparison between the non time-constrained (GRASP) implementation and time-constrained (ACS-DVRP) implementation.

One must consider two issues in this comparison, firstly the fact that the GRASP implementation was not made on the time-constrained variant of the problem (as the ACS-DVRP was), and secondly which has performed to a higher standard.

One must be aware that the time which the algorithm takes often has a significant impact on the level of performance of a particular metaheuristic. In the basic implementation in this chapter each of the algorithms was probably given more time than was strictly necessary, but this was done in an effort to ensure solution quality was of the highest standard. The run times for the fully optimised implementations of each of the four assessed metaheuristics can be seen in Table 3.15.

Met'tic	c100	f71	tai75a
AS/EAS	324.34	138.58	123.96
GRASP	96.66	76.67	61.04
TS	122.04	56.59	56.34
SA	37.08	44.04	28.83

Table 3.15: Average Time Comparison of Various Metaheuristics (in Seconds)

What is immediately apparent is that every (average) time is considerably less than the 750 seconds that are allowed in the time-constrained variant of the problem. This is particularly interesting in the case of the GRASP metaheuristic where one would not hesitate in proceeding with a solution quality comparison with the time-constrained results of the ACS-DVRP (as even if one were to calculate T exactly it would be beneath 750 seconds).

Comparing the result quality of the GRASP (non time-constrained, but with run times well beneath 750 seconds) and the ACS-DVRP (time-constrained) offers up some interesting observations. GRASP appears to be producing average solutions of a higher quality (specifically on the c100 and f71 datasets), though it seems to be somewhat lacking in the ability to identify the really good solutions that the ACS-DVRP manages too.

This is particularly interesting as there is a common conception that ACO techniques lack the required aggressiveness to identify the very good solutions (instead

producing a population of good solutions). As a result of this, hybridising is often felt necessary, and it may well be that the inclusion of the local search phase in the ACS-DVRP is responsible for its ability to identify the very high quality solutions.

With regards to the other metaheuristics (only being compared with each other, not the ACS-DVRP), the performance of the AS/EAS is quite interesting. It appears to be performing to a reasonably good standard (relatively), without the inclusion of a local search phase. The GRASP implementation suggests that the repeated use of a local search phase can offer high quality results, and with ACO likely to identify higher quality pre-descent solutions (one would hope), the potential in the ACO algorithm is interesting.

It is the slowest of the basic implementations, but even this algorithm operates at considerably less than the 750 seconds (though not evenly distributed across the timeslots), that one would need if they were to choose to implement it on the time-constrained variant of the problem.

Usually, one would assume that the fairest comparison that could be made with these results would be between the TS and SA algorithms. They are both classified as local search techniques, they make use of the same neighbourhood and they use the same initial solution.

However, the results in this chapter are not as useful as one might hope, as the TC in the TS algorithm required the components of the 1-Opt neighbourhood to be split. The results suggest that either the TS algorithm is not up to the job of solving the DVRP (unlikely given it is considered better than SA at solving the CVRP and VRPTW) or that the split has reduced the quality of the solutions achievable. The quality of the results for the TS algorithm (in particular the unusually high levels of variation) suggests that this algorithm requires significant alterations were one to make it competitive.

The SA algorithm has performed to a standard that suggests the technique has merit as a DVRP solution technique, and importantly, does not suffer from neighbourhood complexities in the way that the TS algorithm did. It was also the fastest of the

algorithms to implement, suggesting that one would have a considerable amount of time available to identify further improvements (if adjustments to the technique are identified).

Obviously, one should not use time as an overriding factor when making the decisions on these results, as they were never optimised to be time efficient, but one should be aware that time may well become a significant factor in more complex implementations of these algorithms.

Consideration is now given to the other factors that will influence the decision process.

3.5.2 The Information Learnt and the Potential for Improvement

This section has the potential to be as large as one chooses to make it. Understanding the subtleties of the techniques implemented is an immense task and one that cannot truly ever be completed. Instead, this section acts as a guide to the initial decision process that one would undertake if they were to choose to continue research in this area.

Some of the factors that may seemingly belong in this section have already been detailed when discussing the metaheuristics solution quality, and as such one will be referenced to Section 3.5.1 when required.

Each of the metaheuristics is now addressed in turn, with a brief details on the information learnt, and the potential for improvement. The improvements can either be adjustments to the algorithm in its current guise, or more drastic alterations involving changes to the algorithms' mechanics.

It should be noted that as this subsection only covers the some initial thoughts: for those methods that are continued (in Chapters Four and Five,) many other details will be discussed. The more complex implementations will not stick to just making use of the information discovered here, they will look to build on it; as research progresses, one is likely to discover further intricacies and problems.

AS/EAS

Given the existence of the ACS-DVRP, the main thing that the AS/EAS implementation showed was that fact that Montemanni et al. (2005) were correct in assuming that a technique that has been shown to be successful on the CVRP and VRPTW can be adapted into a high quality algorithm for the DVRP.

The inclusion of a local search phase (as discussed in Section 3.5.1) is an obvious area for improving any ACO algorithm; it will always improve the performance of the algorithm (outside any abnormalities associated with the timeslot solution strategy), unless one is forced to reduce the number of cycles i.e. if one were solving the time-constrained variant of the problem.

If one was to proceed with the ACO metaheuristic, one has to decide whether to continue with the AS/EAS algorithm or whether one of its extensions would represent a better algorithm to work upon. The higher level of performance of the ACS-DVRP suggests that there may well be mileage in abandoning AS/EAS and favouring one of its more complex siblings.

Once an algorithm has been selected/implemented there is considerable scope for work outside the inclusion of a local search phase. There exist a multitude of intensification and diversification techniques, allowing the user a greater level of control, depending on the successfulness of the search.

GRASP

The first impression has to be that the GRASP implementation is incredibly significant, simply because it has shown that one can better the results achieved by the ACS-DVRP. This offers up considerable hope that further improvements can be identified in the subsequent chapters.

A GRASP algorithms main strength is from its repetition, and given the low run times, one could choose to repeat the algorithm many more times if they wished. However, Table 3.8 showed that this might not be the best area in which to try to improve the algorithms performance. The current neighbourhood is somewhat limited, and given the existence of multiple other neighbourhoods (which have been

shown to work better on the TSP, CVRP and VRPTW) one would want to explore this avenue.

If one were to make greater changes to the actual algorithms mechanics, they would have to look into the ideas of complemetary memory structures. These will enable the algorithm to remember promising components of solutions, or regions of the search space, and make use of this when constructing the initial solutions.

Tabu Search

The TS algorithm's performance can only be described as somewhat of a disappointment given its pedigree when it comes to routing problems. This in itself results in more questions being posed than it offers answers. The need to split the neighbourhood into two components has probably prevented the TS algorithm from producing competitive results and that somewhat distorts the comparison that has previously been presented.

If one were to continue research in this area one would have a multitude of issues that would need to be addressed e.g. the poor interaction of the neighbourhood components and the cyclic behaviour. However, many of the problems may be down to the poor selection of TC, a new TC may hold the key to rejuvenating the TS algorithm and making it competitive.

With regards to alteration to the mechanics of the algorithm, there are many TS extensions (see Chapter Two) such as aspiration criterion and medium/long-term memory structures that alter the intensification/diversification balance. In a similar way to GRASP, any technique that utilises a local search component will always benefit from the inclusion of a better (and usually more complex) neighbourhood.

Simulated Annealing

The SA implementation has outperformed the TS implementation quite significantly, and that suggests that the technique is not without merit. It was not able to match the result quality of the GRASP implementation; this may suggest that a technique that makes use of a population of solutions may be better suited to the dynamic environment in which these methods are being assessed.

The possibility of taking the SA further is somewhat stifled by the fact that there exist very few directions in which the research could go. The most sensible alterations would be to the cooling schedule (other schedules explained in Chapter Two) and the inclusion of some diversification measures to help the search escape if it becomes trapped.

As with many local search techniques, the biggest improvement one can make to the algorithm will come in the form of a new neighbourhood (in much the way it will improve GRASP and TS). The benefit of SA ahead of TS would be that the SA algorithm is able to accept a new neighbourhood without the need to change the mechanics of the algorithm. Although this may seem a somewhat small issue, one can see from the TS performance what can happen if the algorithm cannot make use of the neighbourhood properly.

3.5.3 Metaheuristics to Take Forward (Chapter Conclusions)

The decision is not a simple one to make. One can argue the merits of each of the techniques, and in reality one would hope that eventually each of these techniques would be fully explored with regards to its application to this problem. Unfortunately this would be outside the scope of a single thesis and as such, two of the four methods are rejected and their evolution will go no further.

The first metaheuristic chosen is the ACO. When implemented in a hybrid technique with some form of local search, it shares a strikingly similar two-phase approach to the GRASP metaheuristic. The GRASP results are currently better, but one has to assume that better constructed (initial) results cannot harm the performance of a local search neighbourhood.

The scope for work, along with the fact of some genuinely competitive results (in the form of the ACS-DVRP) makes this an exciting area in which to conduct further research. The possibility of bettering the results using the same ideology represents the kind of challenge one would normally find when researching a well-developed problem.

The second metaheuristic, chosen for further exploration is TS. Its poor performance in this implementation aside, it is generally considered one of the best performing metaheuristics for routing problems. It offers up many interesting issues, such as, whether it is possible to use the technique to produce good solutions and can the problems regarding the TC and the neighbourhood components be overcome. Although this goes slightly against the initial reasons for doing the preliminary results, it presents an interesting avenue for research.

Chapter Four

ACO: Enhancing its Performance

4.1 Introduction

In Chapter Three the ACO metaheuristic was identified as one that had great promise (and was worthy of further research) with regards, to its application to the DVRP. The purpose of this chapter is to realise this potential, improving the results achieved in the previous chapter, and challenging those achieved by Montemanni et al. (2005).

In Chapter Two, the ACO family was shown to comprise of a series of algorithms that evolved from the AS/EAS proposed by Dorigo (1992). Although they were clearly defined (with regards to their TSP application), only summary details were provided about the performance levels of the various implementations. This chapter begins with a detailed comparison of the algorithm's performance levels (though this is preceded by a brief recap of the different algorithms).

Ideally one would want to conduct this type of performance comparison on a variety of routing problems, but such wide-ranging research has yet to be conducted. As a result of this, one will have to base the majority of this performance comparison on the TSP (it is the only routing problem for which all of the ACO family members

have been implemented). Details regarding some CVRP implementations will also be presented, but as stated, these are not available for all of the algorithms considered.

This performance comparison is presented with the aim of helping one make an informed choice as to which of these algorithms is most likely to perform to the highest standard on the DVRP (this relies on the same assumption as Montemanni et al. (2005) i.e. good performance on these problems is likely to be mirrored on the DVRP).

Ideally, one would have read this comparison prior to knowing its conclusions, however, it is necessary (with regards to detailing the structure of this chapter) to outline two of its key findings:

- The ACS and MMAS algorithms are the best performing of the members of the ACO family.

- ACS is likely to be the best performing of the ACO family in scenarios when time is at a premium.

From these (and the other less significant) conclusions, it seems that the ACS is the most suitable of the ACO algorithms to implement on the DVRP (if one is trying to produce the best results possible). Obviously this presents a possible issue with regard to the fact that there already exists an ACS implementation (the ACS-DVRP), which has been discussed in detail (previous chapters).

After much consideration about the potential scope of the algorithm, it was felt that one could still conduct a sizable amount of different research within the same algorithm framework. Furthermore, it did not seem sensible to discard the findings of the performance comparison simply because a similar piece of research had already been conducted. The decision is made to use the ACS-DVRP as a basis for this chapter, and consider ways in which its performance can be improved.

Secondly, some further details of the ACS-DVRP are presented. Included in this section is a discussion of the parameter settings selected by Montemanni et al. (2005) (and how they were identified), along with some details regarding some potential shortcomings of the algorithm.

Thirdly, a series of remedies for the previously identified potential shortcomings are presented. Included in this section will be a discussion of the theory behind them, along with the necessary details regarding their implementation.

Sections Four through Six are concerned with the production (and hopeful improvement) of results. Section Four deconstructs the ACS-DVRP into its component parts, each of which is investigated and optimised individually (in the hope of improving the overall performance). Section Five details the inclusion of the previously proposed remedies to address the algorithm's potential shortcomings, while Section Six presents an analysis concerning the hybridisation of the algorithm with some form of local search.

Finally, the competitiveness of this new algorithm will be scrutinised on both the standard and time-constrained variants of the problem[1].

4.1.1 A Brief Recap of the ACO Algorithms

As seen in Chapter Two, the ACO family contains four key algorithms: AS/EAS, AS_{rank}, MMAS and ACS. Although using similar fundamentals (the other three evolved from AS/EAS), they do differ considerably with regards to their pheromone update rules.

The way in which the pheromone trails are altered during the course of a run, will have a serious impact on the success (or failure) of a particular algorithm. As the four algorithms all use different rules, they will obviously vary in their performance level. Consideration needs to be given to their performance (see Section 4.1.2), however, prior to this, it seems sensible to recap the algorithms.

[1] All previous analysis will be conducted on the non time-constrained variant of the problem, though computationally expensive alterations will not be considered.

ACO Algorithm	TSP Application	CVRP Application	Construction	Pheromone Update Details
AS/EAS	Dorigo (1992)	Bullnheimer et al. (1999b)	RPR	$\tau_{ij} = \rho\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs}$ where $\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if arc } (i,j) \text{ belongs to } T^k \\ 0 & \text{otherwise} \end{cases}$ and $\Delta\tau_{ij}^{bs} = \begin{cases} Q/L_{bs} & \textit{if arc } (i,j) \textit{ belongs to } T^{bs} \\ 0 & \textit{otherwise} \end{cases}$
AS_{rank}	Bullnheimer et al. (1999a)	Bullnheimer et al. (1999c)	RPR	$\tau_{ij} = \rho\tau_{ij} + \sum_{\mu=1}^{\sigma-1}(\sigma-\mu)\Delta\tau_{ij}^{\mu} + \sigma\Delta\tau_{ij}^{bs}$ where $\Delta\tau_{ij}^{\mu} = \begin{cases} 1/L^{\mu} & \text{if arc } (i,j) \text{ belongs to } T^{\mu} \\ 0 & \text{otherwise} \end{cases}$ and $\Delta\tau_{ij}^{bs}$ as AS/EAS but with $Q=1$.

ACO Algorithm	TSP Application	CVRP Application	Construction	Pheromone Update Details
MMAS	Stützle and Hoos (1997)	N/A	RPR	$\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}^{bs}$ $\Delta\tau_{ij}^{bs}$ as AS/EAS but with $Q=1$ Trail is restricted to range $[\tau_{min}, \tau_{max}]$ where $\tau_{min} = \tau_{max}/a$ and $\tau_{max} = 1/\rho L_{bs}$
ACS	Dorigo and Gambardella (1997)	N/A	Pseudo-RPR	2 Update Rules Global: $\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij}^{bs}$ $\Delta\tau_{ij}^{bs}$ as AS/EAS but with $Q=1$ Local: $\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_0$ where $\Delta\tau_0$ is a user defined parameter (often some multiple of $1/L_{bs}$)

Figure 4.1: Summary of Key Differences among the Members of the ACO Family.

As explained previously, the primary difference between these algorithms comes in the form of the pheromone update rules, although ACS also modifies the construction phase. Figure 4.1 provides concise details with regards to the key differences, as well as noting their early implementations on both the TSP and the CVRP (if applicable).

As stated previously, further details regarding the algorithm's implementation details and development can be seen in Chapter Two. With the algorithms suitably reintroduced one can now critique their performance.

4.1.2 ACO Performance Review

As the four different members of the ACO metaheuristic family all contain certain strengths and weaknesses, it is important to identify which is likely to represent the best avenue for pursuing high quality results for the DVRP.

The algorithms were developed for/on the TSP, and as such, comparing the results achieved by the algorithms on the TSP seems a sensible starting point. The algorithms are also in their most developed state on the TSP, as many of the details in the adaptations were honed over a series of TSP based publications. Furthermore, given the obvious parallels to the DVRP (DVRP is related to CVRP which is related to TSP) it would seem somewhat foolhardy not to utilise this information.

However, one should not concern oneself solely with the TSP, as it is clearly a different problem from the DVRP. Given that the CVRP represents a link between the two problems (DVRP and TSP) the performance of ACO algorithms on the CVRP is clearly of interest too. The performances of the ACO algorithms on these two problems are now considered.

ACO for the TSP: Performance Comparison

Fortunately, this is a topic of great interest to many other researchers, and as such, investigations into this very subject already exist. Two such comparisons are considered here: Dorigo and Stützle (2004) and Stützle and Hoos (1997).

Dorigo and Stützle (2004) produced a word-based analysis[2] of the various ACO algorithms performance on the TSP. The conclusions of these variants' performance is too lengthy a write-up to include in its entirety, thus only two of the central findings are presented here:

- '*...all extensions of AS achieve much better final solutions than AS*'

- '*Comparisons among the several AS extensions indicate that the best performing variants are MMAS and ACS, closely followed by* AS_{rank}.'

The first of these points clearly shows the merits of the extensions, whilst the second tells of the strength of the MMAS and ACS algorithms. Interestingly, the two authors producing this summary were involved in the development of the two methods highlighted for their good performance.

The workings of Stützle and Hoos (1997) included a table of results showcasing the relative performances of the ACO variants (shown here in Table 4.1). These results were for seven well-known TSP instances, the size of which can be inferred from the dataset's name. The optimal results are also included for comparative purposes.

Instance	AS	EAS	AS_{rank}	ACS	MMAS	Opt
eil51	437.3	428.3	434.5	428.1	427.6	**426**
kroA100	22471.4	21522.8	21746	21420	21320.3	**21282**
d198	16702.1	16205	16199.1	16054	15972.5	**15780**
ry48p	15296.4	14685.2	14511.4	14565.4	14553.2	**14422**
ft70	39596.3	39261.8	39410.1	39099	39040.2	**38673**
kro124p	38733.1	37510.2	36973.5	36857	36773.5	**36230**
ftv170	3154.5	2952.4	2854.2	2826.5	2828.8	**2755**

Table 4.1: Comparing the Performance of the ACO Algorithms on the TSP

The most immediate observation from these results is that they seem to support the first finding of Dorigo and Stützle (2004), i.e. that all of the extensions offer some

[2] Utilising results they produced themselves as well as those from a selection of papers.

form of improvement to those of the original AS algorithm. The second finding (expressing a preference for ACS and MMAS) is also confirmed. These two analyses can be seen to support each other's findings; they offer a similar message with regards to which algorithms produce better solutions for the TSP. Were one to base decisions on the algorithm's performance on the TSP alone, one would be likely to favour the MMAS (as it slightly outperforms ACS).

ACO for the CVRP: Performance Comparison

The CVRP has received far less attention (with regards to ACO implementations) than the TSP, and as such, one does not have the same volume of research with which to make a performance comparison. There does not appear to be a published comparison for the CVRP (in the way in which there was for the TSP), presumably because of this lack of implementations.

As seen in Figure 4.1, there do not appear to have been MMAS and ACS implementations on the CVRP, and as such, one can only consider which of AS/EAS and AS_{rank} is the better performing.

The two workings that one needs to compare are Bullnheimer et al. (1999b) and Bullnheimer et al. (1999c) (which consider the AS/EAS and AS_{rank} variants respectively). It should be noted that these papers differ with regards to other components than just their ACO technique, but unfortunately without producing new algorithms, a direct comparison is not possible.

When one considers the performance of these algorithms, one can see that the results follow the same trend as for the TSP (i.e. AS_{rank} outperforming AS/EAS). This is important as it shows that the performance on one routing problem is mirrored on another. This gives hope that the TSP and CVRP may be good indicators of performance levels for the DVRP (a fact which Montemanni et al. (2005) considered when producing the ACS-DVRP).

Using this assumption, one would expect the more complex MMAS and ACS algorithms to better both of these algorithms (on the CVRP), but such research has yet to be conducted.

Selecting a Variant for Further Research

As there is clearly a lack of CVRP implementations, the primary factor on which one must base this decision is the performance of the algorithms on the TSP. As explained previously, one would be likely to favour the MMAS algorithm, but making such a selection would be inappropriate before consideration is given to one further factor.

As with many metaheuristics (particularly for dynamic problems) the time which an algorithm takes to produce its best solutions is an important factor. It is widely known that ACO algorithms by their very nature require a sizable amount of time to learn their way to good solutions. However, within the ACO family, the time required will differ between algorithms.

If one were to only be concerned with the non time-constrained variant of the DVRP, this issue would be less significant as the algorithm could be run for a longer period of time. However, with the time-constrained variant constituting a significant part of these workings, one must ensure that the selected algorithm will have sufficient time to showcase its potential.

This issue will become more prominent as the problem size increases[3] as the number of cycles that can be completed in the set time will diminish. It is therefore prudent to select one of the faster performing algorithms.

Dorigo and Stützle (2004) make reference to the behaviour of the two best performing algorithms MMAS and ACS, stating '*...it can be observed that ACS is the most aggressive of the ACO algorithms and returns the best solution quality for very short computation times. Differently, MMAS initially produces rather poor solutions and in the initial phases it is outperformed even by AS'.*

With such poor performance at the beginning of the algorithm, it is not difficult to imagine a situation when the time-constrained variant of the problem would not allow MMAS the necessary time to identify the good solutions it is likely to be capable of. With the aggressiveness of the ACS algorithm noted, current research seems to

[3] The maximum problem size in the full dataset is 150 customers, though one could easily produce larger problem instances.

suggest that the ACS algorithm would be the best route to pursue for this DVRP analysis.

This selection obviously supports the decision of Montemanni et al. (2005) to develop an ACS based algorithm for the DVRP. Although this is a significant piece of work, one should remain undeterred as there remains plenty of scope for research.

A distinct advantage of this selection is that one can begin the research at the point that the ACS-DVRP concluded (rather that developing the algorithm from its TSP or CVRP roots). The ACS-DVRP will act as a starting point for this analysis, with an aim of finding suitable improvements to it.

4.2 The ACS-DVRP

One problem of using such a well-developed starting point stems from the fact that a considerable amount of effort has already been put into making the algorithm as good as possible. It is not unreasonable to consider improving this algorithm will be a greater challenge than one would face if they were developing a new algorithm based on AS_{rank} or MMAS.

Given the complexity of the ACS-DVRP (the full details of which can be found in Chapter 2) one should give careful consideration to any changes being made; the algorithm should be treated with respect. Were one to consider altering components of it without the proper understanding, one may not fully understand the impact that a decision may have.

This section begins with a recap of the parameter settings chosen by Montemanni et al. (2005); these are to form the default parameter set for this chapter. Given that these parameters were selected via a parameter review, one should have sufficient confidence to avoid the need for any precursory runs.

An investigation into some of the algorithm's potential shortcomings (i.e. those areas where it is believed that the algorithm could perform better) is then presented. These are discussed in the hope that in the subsequent sections, suggestions of how to remedy these problems can be presented.

4.2.1 The ACS-DVRP Parameter Settings

The advantages of utilising an existing piece of work as a basis for further research extend to the usually contentious issue of selecting an appropriate set of default parameters. This is a particularly prominent problem in ACO algorithms; given the high number of parameters that one is required to optimise in order for the algorithm to work as well as possible.

In Chapter Three, the AS/EAS implementation default parameters were identified from prominent TSP and CVRP workings. Although this was successful, with regards to identifying sensible starting values, the fact that they were identified on very different problems may have resulted in them not being as good a set as they could have possibly been. However, in this chapter, one is able to use the optimised parameters of the ACS-DVRP as a default set in this analysis. One should be aware that the ACS algorithm requires even more parameters than the AS/EAS algorithm, and as such underlines just how beneficial this default set is likely to be. These optimised parameter settings were presented in Chapter Two, but given their importance to this chapter, it seems sensible to recap not only what they are, but where they came from.

The ACS-DVRP parameter settings were established from two sources:

- MACS-VRP (Gambardella et al. (1999))
  This paper was the first ACS implementation on any type of VRP (the VRPTW), and formed the basis for the ACS-DVRP. Many of the same parameter settings were used, in the hope that a parameter optimised for the CVRP would also prove a sensible choice for the DVRP.

- Parameter Review
  These are an important process for all ACO based algorithms, and in this paper (as explained previously) took the form of a trial of five runs on the three test datasets. Decisions were made one parameter at a time and seemingly in no specific order.

The ACS-DVRP parameters, which will be referred to from here on as the default parameter set, are as follows: n_{ts}=25, q=0.9, β=1, γ=0.3, ρ=0.1, m=3 and $\tau_0 = 1/n \times Cost(PI)$, where *Cost(PI)* is the cost of a solution obtained by a greedy insertion heuristic for t_0. The exit criterion (for the non time-constrained variant) is controlled by the parameter t, which controls the number of cycles. A value of t=15000 was selected as it should give the algorithm sufficient time to produce the best solutions it can.

As this problem is supposed to represent a real-life situation, whereby new orders arrive throughout the working day, it seems odd to have a parameter (τ_0) that requires one to know the total number of customers in advance. The size of each benchmark problem is known (it is in the dataset's name), but one has to consider whether this information should be used when solving the problem.

It is apparent that to use the information would offer distinct advantages (though not specifically in the selection of this parameter), and is simply information that one would not have, were they faced with the problem for real. It seems that to use this information would not lend itself well to the idea of uncertainty associated with dynamic problems.

However, it is clear that an alternative to this is readily available, and it will not greatly impact upon the algorithm's performance (as it results in a very similar value being selected). Given that one does know for what proportion of the day customers are not known in advance (known as the DOD), one can approximate the total number of customers. This technique is only possible as the datasets have random arrivals (thus the DOD which relates to time) is likely to be very similar to the proportion of customers that are not known.

The DOD of these datasets is 50%, thus it can be assumed that ≈50% of the customers are known in advance (i.e. n_0). By making use of this figure, one can eliminate the need to know the exact number of customers in advance, thus fitting in with the ethos of uncertainty.

Adopting this idea seems prudent and the new definition for τ_0 can be seen in Equation 4.1.

$$\tau_0 = 1/2n_0 \times Cost(PI) \tag{4.1}$$

With regards to all of the default parameters, investigating and (if performance is improved) altering them, is one of the most obvious (and simple) changes that one can make to the ACS-DVRP. Although they do not represent the most substantial changes one can make, their importance should not be underestimated. An ACO algorithm's performance can deteriorate significantly if the parameters selected are poorly chosen.

4.2.2 Potential Shortcomings of the ACS-DVRP

In its present form, there appear to be two key features of the ACS-DVRP that appear to offer considerable scope for improvement. The first relates to the somewhat temperamental and computationally expensive technique for deciding when to deploy further vehicles. The second tackles a problem more inherent to the solution technique, notably the sequential nature of the RPR construction phase.

One must remember that the identification of these problems relies on the existence of the fully operational ACS-DVRP algorithm, and as such, one can understand why these were not considered during the algorithms development.

Further Vehicle Deployment

The ACS-DVRP criterion for tentative schedule completion is currently the selection of the depot. This can occur in two ways, either through a RPR selection or when the set $F^k = 1$ (i.e. it remains the only possible expansion to the current partial solution).

There are two distinct problems with the method in its current guise, the first affecting the cost of the solutions constructed and the second relating to the substantial computational time required:

- Early Returns

  The ACS-DVRP defines the visibility of a customer as the reciprocal of the distance between the current customer and the members of the set F^k. As the depot is included in this set, one can envisage a situation where a vehicle returns to the depot soon after departing. The first customer to be serviced is chosen through the RPR and (due to the visibility definition) is likely to be close to the depot. A decision must then be made as to where to go next. Due to the depot's close proximity to the current vehicle location, there is a relatively high chance it will be selected; resulting in a very short vehicle route. This will result in a greater number of vehicles being deployed than is strictly necessary; although this need not be minimised, it can generally be assumed that good solutions are likely to have high vehicle capacity utilisation.

- Computationally Expensive

  As the set F^k contains all the possible feasible expansions, the feasibility (with regards to both capacity and time) must be calculated for all non-allocated customers prior to every RPR selection. Although singularly this is not a time-consuming procedure, when one considers the number of times it must be repeated, this calculation is clearly a very computationally expensive process.

In response to these observations, a tentative schedule criterion that is not susceptible to such temperamental behaviour regarding the production of short vehicle tours, whilst also being computationally less expensive is presented in Section 4.3.1. Note that discussions relating to computational expense will only affect the time-constrained variant of the problem (the greater the computational effort, the fewer cycles that will be completed).

Sequential Construction

The ACS-DVRP relies on the sequential building of the vehicle routes i.e. one vehicle's tentative schedule is completed before consideration is given to the next. This methodology was seen in the AS/EAS implementation of Bullnheimer et al.

(1999b) when solving the CVRP, and has been a component of all subsequent ACO implementations on the various VRPs.

One drawback with the technique can be seen, if one assumes that the best solutions are produced when one is able to give consideration to the maximum number of customers in the construction phase. As a new vehicle is deployed, many of the potential arcs (customer pairings) are not considered, as one of the customers has already been allocated to a previous vehicle.

This can lead to an interesting and potentially unfavourable situation regarding the previous customer's allocation. Given that a new vehicle has been deployed, one ought to consider if this new vehicle is better placed to service the previously allocated customers.

The lack of knowledge about the other vehicles is detrimental to the existing technique. It acts in a greedy manner, concerning itself only with the current vehicle, whilst displaying little foresight for the problem as a whole.

The existing technique will result in (relatively) short computation times as one is theoretically only considering subsections of the problem. However, it may well be too restrictive to enable one to produce the best solutions possible.

A non-sequential construction technique is presented in Section 4.3.2. This will prevent this unfavourable situation occurring, whilst hopefully improving solution quality. This technique will not be without its problems, given that one will be considering many more customer arcs in the RPR, it is likely to result in a increase to the run-time.

4.3 Improving the ACS-DVRP

The focus of this section is to address the potential shortcomings that were identified previously. Although only two potential shortcomings were identified, this section is divided into three subsections.

Firstly, a potential solution to the further vehicle deployment issue is presented. This is proposed as a remedy for a specific problem, though the way in which it impacts upon the behaviour of the other aspects of the algorithm are unknown. One will have to discuss whether it is having a positive impact on the performance of the algorithm or if its inclusion will result in further problems.

Secondly, consideration is given to the development of a non-sequential construction technique. This represents a significant change from traditional ACO constructions in both CVRP and VRPTW implementations; no existing work considers a construction method of this type. As this is a new and untried technique, one cannot be sure of the impact this will have on the performance levels.

Thirdly, and somewhat unfortunately, one can envisage a problem being caused by the non-sequential construction technique; namely an increase in the number of vehicles deployed. This somewhat negative consequence must be compensated for via some other technique, were one to deem the inclusion of the non-sequential construction technique appropriate. The reasoning for this increase will be discussed, and two potential solutions to this new problem are discussed.

These methods are not implemented in this section; this work concerns the theoretical reasons for the adjustments, and details as to how one would make them to the algorithm. Implementation of these techniques will be in Sections 4.4.1, 4.6.1 and 4.6.2 respectively.

4.3.1 New Criterion for Tentative Schedule Completion

The method proposed here seeks to prevent the potential for the production of short vehicle routes. As explained previously, these occur due to the current definition of the set F^k, and the fact that the visibility encourages the selection of customers close to the current vehicle location. With one not wanting to begin by altering the visibility matrix (although consideration will be given to whether this is a good idea, in Section 4.5.2) from the reciprocal of the distance, it seems that any change must come from the definition of the set F^k.

To prevent the selection of the depot early on in the construction of a vehicle route, it seems prudent to remove the depot from the set F^k. However, as the depot must be scheduled eventually (to complete the vehicle's tentative schedule) one must consider the alternative ways in which it could be selected.

The simplest method, would be to use the current technique with this newly defined set F^k and return to the depot when $F^k = 0$. Unfortunately, this is not without its problems; it will lead to somewhat irrational choices of customer due to their demand rather than their proximity to the current position of the vehicle. It seems clear that a more subtle technique is required.

One could reintroduce the depot into the set F^k; once a route has suitably ventured from the depot, the depot's selection would only be made if it seemed sensible for that vehicle to complete its tentative schedule. However, as a discussion was presented about the computational expense of the repeated calculation of the components of the set F^k, it seems sensible to consider a technique that would alleviate this problem too.

It is proposed that the set F^k should be expanded so that it includes all non-scheduled customers (both feasible and infeasible), with the exception of the depot. As all customers are included in this set, one does not need to test the feasibility of every customer prior to each RPR selection (thus reducing the computational effort significantly).

As a result of this, it will be necessary to assess the feasibility of a move after the selection has been made (thus one needs to check only one customer's feasibility for each RPR selection). This feasibility check will lead to one of two scenarios:

- The Customer is Feasible
  The customer chosen via the RPR does not cause any feasibility problems; it would have belonged to the previous definition of F^k. The customer is then appended to the current solution (in much the same as it would have been previously) and the process is repeated i.e. another RPR selection is made.

- The Customer is Infeasible

  This means that the customer chosen would not have belonged to the previous definition of F^k. The attempted allocation of the selected customer causes either the capacity or time constraints of that vehicle to be violated. In a variation to the previous method, this is proposed to be the trigger for declaring a vehicles tentative schedule to be complete. That vehicle is referred to as a temporarily taboo vehicle (TTV), and consideration is then given to the next vehicle (for which a new RPR selection is made).

The TTV method varies considerably from the technique proposed by the ACS-DVRP and as such, one cannot be sure of exactly how it will impact upon the performance of the algorithm. However, this alteration was not made purely in the desire to see an improvement in solution quality, it had other objectives. It should prevent the production of short vehicle routes (which may improve the cost), whilst it will be computationally less expensive as it does not rely on the repeated computation of the set F^k for each RPR selection.

Results supporting these claims will be shown in Section 4.4.1.

4.3.2 Non-Sequential Construction

With the impact that a sequential construction technique has on the way the algorithm performs already detailed, one can focus on the discussion of an alternative technique. If the construction is going to be non-sequential, one must propose a system that builds multiple vehicle routes simultaneously.

This will greatly increase the complexity of the build process, and with it, the computational time required to produce a solution. One must balance the benefits that the technique will provide, against this increase in time to determine whether this is a sensible adaptation to the algorithm.

Before the discussion of how one can adapt the existing method in such a way that the construction is non-sequential, an analogy is presented in the hope of clarifying how selections are made via the RPR.

The Roulette Wheel

In the book 'Genetic Algorithms in Search, Optimization an Machine Learning', Goldberg (1989) first proposed what has become a commonly accepted analogy regarding the RPR. He allowed one to visualise how the rule worked in a way that made the RPR accessible to non-mathematicians; he compared it to the game of roulette.

Roulette, or more importantly the wheel itself, is understood by most. It contains a number of pockets (37 or 38 depending on country), each with equal probability of being selected. He proposed that each RPR selection could be considered as a spin of a roulette wheel, albeit one with a varying number of pockets of varying sizes.

The number of pockets can be equated to the number of elements in the set F^k, and the size dependent on the probabilities calculated from the RPR. It should be noted that a new wheel is constructed after each spin (RPR selection) as the probabilities and elements in the set F^k will have altered. One can use this analogy to equate the problem of sequential route construction to the idea that there are not enough pockets included on the wheel.

The Extended Roulette Wheel

It is proposed that the RPR (or the roulette wheel) is extended in such a way that the customer arcs (or pockets) available, allow the algorithm to append a customer to one of multiple vehicle routes. This will allow for routes to be constructed simultaneously, bypassing the greedy nature of the existing technique. It is hoped that by removing the need to concentrate on individual routes, one will retain a more global view of the problem, and consequently improve the quality of the solutions built.

Whilst this technique will retain many of the existing features of the ACS-DVRP, it makes an alteration to the definition of an element in the set F^k. Consider the situation where a vehicle is currently at customer 1 and it has a choice of three customers that it can visit next (regardless of which criterion for tentative schedule completion is being utilised). The current defined elements of F^k would be {2,3,4}, but it is proposed that this is redefined as a set of customer arcs from the current vehicle location e.g. {(1,2),(1,3),(1,4)}.

This change in definition is key to understanding how one is able to extend the roulette wheel to construct multiple vehicle routes. If one can allow further vehicles to be deployed before the existing one has met the criterion for tentative schedule completion, one can include customer arcs from multiple vehicles in the RPR selection process.

It is proposed that a new vehicle be considered for deployment in every RPR selection, thus negating the need for tentative schedule completion to be met for new vehicle deployment. The current locations of all vehicles that have not completed their tentative schedules are noted (including the depot for a new vehicle), and the customer arcs that could be appended to these are included in the RPR selection process i.e. for previous example the set F^k would contain the elements $\{(1,2),(1,3),(1,4),(0,2),(0,3),(0,4)\}$.

Once a RPR selection has been made (i.e. one arc appended to one vehicle's tentative schedule), the process can then be repeated until all customers have been allocated.

This technique will significantly increase the number of arcs (pockets on the roulette wheel) that are being considered, which unfortunately is likely to result in an increase to the computation time[4]. Consideration will be given as to whether this increase is acceptable, given that the time-constrained variant will be analysed at the end of this chapter.

4.3.3 Encouraging the Use of Fewer Vehicles

Although the extended roulette wheel will enable the problem to be considered in its entirety, one should be aware that it is likely to have a negative impact associated with it too. Given the requirement for a new vehicle to be considered in each RPR selection, one might expect the number of vehicles being deployed to increase.

Although minimising vehicles is not a necessity with regards to the objective function, it is likely that solutions with fewer vehicles often result in lower distances.

[4] It should be noted that one would hope that the savings achieved via the proposed TTV criterion (page 176) would offset some of this increase.

Consideration is now given to how one could prevent the unnecessary deployment of further vehicles. It seems clear that any method for achieving this must be flexible and just encourage the use of fewer vehicles, as it would be nonsensical to force the algorithm to minimise the number of vehicles if better solutions exist with more vehicles.

Two techniques are proposed that encourage the use of the existing vehicles ahead of further vehicle deployment. The theory behind both of the techniques is to alter components of the RPR, though how they exert their influence differs between the two methods.

- Capacity Utilisation (CU)

  This technique implements one of the problem specific improvements suggested by Bullnheimer et al. (1999b), when solving the CVRP with an AS/EAS. Noticing the need for a *'high degree of capacity utilization'* they altered the components in the RPR to encourage the selection of customers that better filled the vehicles.

  They introduced the parameter D_i to represent the vehicle capacity utilisation, including the demand d_i of the vehicle's current location (customer v_i). For each potential next customer v_j, $\mu_{ij} = (D_i + d_j)/capacity$, is calculated and included in an altered RPR (see Equation 4.2).

$$p_{ij}^k = \begin{cases} \dfrac{(\tau_{ij})^\alpha (\eta_{ij})^\beta (\mu_{ij})^\lambda}{\sum_{l \in F^k} (\tau_{il})^\alpha (\eta_{il})^\beta (\mu_{il})^\lambda} & if j \in F^k \\ 0 & otherwise \end{cases} \tag{4.2}$$

  High[5] values of μ_{ij} will indicate a greater level of capacity utilisation, and are therefore encouraged. The bias parameter λ is introduced to allow a level of

[5] $\mu_{ij} \leq 1$ is required for a feasible solution.

control over the influence of this new component, in much the same way as α and β do for the trail and visibility.

- Decreased Depot Visibility

  This simple technique benefits from the lack of extra computational effort required for its implementation (which is in direct contrast to the CU technique). The visibility of all customer arcs in the set F^k that begin at the depot i.e. those taking the form $\eta_{0,j}$ are reduced.

  The user-defined parameter $\theta\,(0 \leq \theta \leq 1)$ is introduced and is multiplied by the visibility to create an artificially lowered visibility. This will discourage the selection of these arcs, which consequently discourages the deployment of further vehicles.

Unfortunately, both of these methods require the use of an extra parameter, something that would be beneficial to avoid given the already sizable number. However, if one wants to exert a greater level of control on the system, such an increase is somewhat unavoidable. If either (or both) of these methods were to be adopted, the relevant parameter (λ and/or θ) will require optimisation (this optimisation will be conducted as and when required).

4.4 The First Implementation and the First Proposed Remedy

All the information required to reproduce the ACS-DVRP has been presented; were one to implement it, one would expect to produce results of similar quality to those achieved by Montemanni et al. (2005). Although reproducing these results would represent a sensible starting point for this analysis, it is possible that a different set of results will allow one to draw conclusions with more clarity.

It is proposed that the ACS-DVRP (a hybrid algorithm, consisting of a construction phase and a local search phase) has its local search phase removed for the majority of the analysis. The benefits of such an adaptation are twofold:

- Ease of Optimisation

  It is easier to compare two or more sets of results if a local search phase is not compensating for an inherently bad technique. If the local search phase is retained, improvements made to the solution quality may not reflect an actual improvement to the quality of results constructed.

- Time

  ACO based algorithms are time consuming by their very nature, and this time increases further if a local search phase is included. Given that ten runs are required for each of the three datasets, it seems sensible to only implement the improvement phase on an optimised algorithm.

Once the parameter optimisation has been completed, and one has given the necessary consideration as to whether any of the proposed improvements deserve inclusion, the local search phase will be reintroduced. Given the existence of various neighbourhoods, it also seems prudent to assess whether further improvements are achievable via a change of neighbourhood.

The reintroduction of the local search phase will have a larger impact on the time-constrained variant of the problem than the non time-constrained variant (where the number of cycles will remain constant).

4.4.1 Criterion for Tentative Schedule Completion: Current vs. TTV

Given that one is interested in optimising the performance of the algorithm, it seems sensible to begin by considering whether the proposed remedy regarding further vehicle deployment has the impact that one would expect. The benefit of considering this first (i.e. prior to the parameter optimisation), is that one will be privy to extra information about the algorithms structure when they come to consider suitable parameter settings.

The two methods (denoted Current and TTV) are to be investigated using the default parameters identified in Section 4.2.1. However, before producing the results in the

standard fashion, one should consider the purpose of introducing the TTV and whether some better form of assessment exists.

One is not attempting to improve solution quality directly through the introduction of the TTV; it was proposed as a remedy for two potential shortcomings. It was felt that too many vehicles might be deployed when using the current criterion, whilst it also relied on the computationally expensive repeated evaluation of the set F^k. As the number of vehicles deployed and time taken to produce the solutions are both measurable entities, it would seem sensible to measure the impact (of the TTV) directly.

One could conduct the trial of ten runs and report these new metrics, but given the change in information reported, one should consider if this type of trial is the most suitable.

It has been shown how the timeslot technique creates a series of CVRP-like problems, which are solved subsequently in order to produce a final solution. Whilst any final solutions will always represent a solution to the whole problem (and are thus comparable), the CVRP-like problems solved along the way will vary from run to run. As different customers are committed too, the remaining customers (which constitute the next sub-problem), will vary. As they contain different customers (and possibly a different number of customers), these sub-problems will be of varying complexity; rendering any comparisons (with regards to the newly proposed metrics) somewhat unreliable.

Given the volatile nature of the metrics being considered, it seems that the most appropriate way of producing a fair assessment is if they are solving exactly the same problem. Exerting this level of control over the sub-problems contained in the DVRP is not possible (as one could not ensure that the same customers were committed too), it is clear that a new test-bed is required.

ts_0= CVRP

Any dynamic problem, will, by its very nature, be difficult to control in such a way that this type of assessment would be completely fair. The timeslot method iteratively

solves adapted CVRP-like problems, and given that the CVRP does not suffer from these issues, one has a similar environment within which to conduct this assessment.

The similarities between these problems (DVRP sub-problems and the CVRP) were fundamental to the workings of Montemanni et al. (2005), and given the use of their algorithm as a starting point, it seems reasonable to continue this logic.

One could make use of the traditional CVRP benchmarks to conduct this analysis, but on closer inspection a complete CVRP is already apparent in the existing solution technique. Whilst a general timeslot will have varied vehicle start locations, residual capacities and times (and as such, is not a true CVRP), these variations are not apparent in the first timeslot (ts_0).

The complete CVRP solved in timeslot ts_0 is likely to be one of the most complicated sub-problems in terms of customer numbers (as 50% of the customers have arrived and none have been committed to). It is hoped that a technique that is successful in solving ts_0 (in terms of vehicle deployment and time) will mirror this success in the general timeslots of lesser or equal complexity.

Given the time required to produce a solution for a single timeslot is considerably less than required to solve the complete problem, a greater number of runs are to be conducted. This should increase the confidence which one has in any subsequent results, which is especially beneficial as any decision to alter the tentative schedule completion criterion could have a significant impact.

The results for a trial of 50 runs are presented in Table 4.2.

Metric	Criterion	**c100**	**f71**	**tai75a**
Ave. No. Vehicles	Existing	5.73	2.33	7.43
	TTV	4.02	2.00	6.09
Ave. Time (secs)	Existing	12.26	4.51	9.41
	TTV	9.87	3.67	6.52

Table 4.2: Comparing the Two Tentative Schedule Completion Criterion, Where Existing Refers to the Criterion Chosen by Montemanni et al. (2005)

In this analysis both criterion produced their best solutions with the same number of vehicles (4, 2 and 6 respectively). However, although both techniques provide solutions with these numbers of vehicles, the TTV achieved these solutions with a far greater level of consistency (with averages very close to these minimums). This consistency is beneficial to the ant's learning process, and should result in better solutions being constructed.

The decrease in the number of vehicles deployed, is accompanied by a large decrease (22% averaged across the three datasets) in the time taken to produce the solutions. This reduction in time taken will allow more cycles to be completed when considering the time-constrained variant of the problem, whilst allowing results to be produced faster for the non time-constrained variant.

The results clearly show that the TTV outperforms the Current criterion; superior results are achieved across all three datasets for both of the metrics considered. The TTV is therefore selected as the criterion for tentative schedule completion, and will be included in all subsequent workings.

4.5 Parameter Optimisation

As with any ACO based technique, an in-depth parameter review is required for the algorithm to perform to its potential. Unfortunately, this requires a considerable amount of computation, analysis and interpretation of results. This parameter review is divided into two sections, each focusing on a different component of the ACS algorithm.

The first of these sections concerns itself with whether the Pseudo-RPR is more effective than the traditional RPR and what values should be selected for the parameters that impact calculations in each of these formulae. Alterations to these parameters will not impact the trail or visibility matrices in any way; just how these pieces of information are combined in order to calculate the various probabilities.

The second section focuses on those parameters that impact the actual values contained within the trail and visibility matrices, altering the learning process and a priori information with the aim of obtaining results of a higher quality.

4.5.1 Combining Information for Solution Construction

The Pseudo-RPR (see Page 50) evolved from the more commonly utilised RPR, the build process belonging to the other members of the ACO family (AS/EAS, AS_{Rank} and MMAS). This evolution was designed to intensify the search, a move presumably made as a sort of compensation for the newly introduced diversification measures included in the update rules.

However, the traditional RPR already contains a method for increasing the intensification of the search, the bias parameters α and β. These allow the user to not only control the overall greediness of the search, but also the relative importance of the trail and visibility i.e. encouraging bias towards one component over the other. Montemanni et al. (2005) set $\alpha=\beta=1$, thus minimising the level of bias, then encouraged a far more rigid form of greediness through the selection of $q_0=0.9$.

The trio of parameters assessed in this sub-section seek to identify whether the Pseudo-RPR outperforms the traditional RPR, when working within the ACS framework. The reduction and removal of the parameter q_0, is considered first (see Table 4.3).

q_0		c100	f71	tai75a
0	Min	1144.24	323.46	1936.54
	Ave	1266.26	336.96	2094.29
	Max	1372.53	357.21	2307.48
0.3	Min	1068.12	319.82	1988.00
	Ave	1120.97	336.86	**2066.39**
	Max	1203.17	375.62	**2152.50**
0.6	Min	1034.08	326.38	1972.97
	Ave	1080.02	335.59	2082.36
	Max	1141.67	347.43	2351.27
0.9	Min	**1008.04**	**307.39**	**1933.70**
	Ave	**1078.89**	**330.94**	2074.47
	Max	**1136.83**	**339.28**	2201.42

Table 4.3: Analysing Various Values of q_0

This table clearly shows that the selection of q_0=0.9 is sensible; it easily outperforms the various other values. However, it is not simply the removal of q_0 that is being assessed, but the Pseudo-RPR as a concept in itself. To determine whether the RPR can perform to a similar or better standard, it is necessary to evaluate sensible values for α and β and compare this against the result achieved via the Pseudo-RPR.

Ideally one would optimise α and β in unison, but due to the significant computational effort required this is abandoned in favour of optimising them individually. Given the logic identified in Section 3.4.1, where it was assumed that better solutions were achieved when $\alpha \leq \beta$, it seems prudent to analyse β first (see Table 4.4). Should a value for β>1 be selected, then the appropriate values of α can be analysed to remain consistent with the previously identified logic.

β		c100	f71	tai75a
1	Min	1144.24	323.46	1936.54
	Ave	1266.26	336.96	2094.29
	Max	1372.53	357.21	2307.48
2	Min	993.88	**297.25**	1837.80
	Ave	1052.02	**322.96**	**1961.08**
	Max	1119.41	351.10	**2116.66**
3	Min	**987.83**	305.27	1837.93
	Ave	1054.25	324.47	1984.03
	Max	1113.87	**336.39**	2159.86
4	Min	1023.97	320.98	**1793.18**
	Ave	**1041.84**	338.07	1999.48
	Max	**1086.95**	356.80	2256.67

Table 4.4: Analysing Various Values of β (given that α=1 and q_0=0)

The selection of an optimal value of β is not simple, with little consistency across the three datasets. Prior to a decision being made, it is interesting to note the performance of β=1. It is producing results of a considerably lower quality than the other values assessed, highlighting the validity of this investigation.

With β=1 not being considered, there is little to distinguish between the three remaining options. Their performances are all comparable, but a choice of β=2 is made, due to the result quality being arguably better and its computational simplicity[6].

With a sensible value for β identified, it is necessary to consider the possible values for α, thus completing this investigation into the RPR. If one assumes that better solutions are achieved when $\alpha \leq \beta$, one need only consider α=1 and α=2 (see Table 4.5) to complete the analysis.

α		c100	f71	tai75a
1	Min	**993.88**	**297.25**	**1837.80**
	Ave	**1052.02**	**322.96**	**1961.08**
	Max	1119.41	351.10	**2116.66**
2	Min	1047.59	307.21	1878.94
	Ave	1083.64	337.40	2016.26
	Max	**1116.68**	**351.02**	2182.79

Table 4.5: Analysing Various Values of α (given that q_0=0 and β=2)

There is a clear decrease in solution quality when α=2, thus the previously utilised value of α=1 remains a sensible choice for this parameter. With the RPR optimisation complete, one can compare it with the Pseudo-RPR (q=0.9, α=β=1, as used by Montemanni et al. (2005)). These results are presented in Table 4.6.

Construction		c100	f71	tai75a
Pseudo-RPR	Min	1008.04	307.39	1933.70
	Ave	1078.89	330.94	2074.47
	Max	1136.83	**339.28**	2201.42
RPR	Min	**993.88**	**297.25**	**1837.80**
	Ave	**1052.02**	**322.96**	**1961.08**
	Max	**1119.41**	351.10	**2116.66**

Table 4.6: Comparing the Pseudo-RPR with the Optimised RPR

[6] Raising a number to a power is computationally expensive in FORTRAN

It seems clear that the evolution from the RPR to the Pseudo-RPR is counter-productive in this instance, with result quality consistently better with the more traditional technique.

Although these results clearly show a preference for the RPR, one should not discredit the Pseudo-RPR, as it has been shown to be a component of some highly effective algorithms (e.g. the MACS-VRPTW). Continuing with an ACS-type framework though abandoning the Pseudo-RPR is an unusual step, as it is significantly altering the mechanics of the algorithm. However, one is concerned with the production of the best algorithm possible, and as such, one cannot favour a technique which appears to produce consistently lower quality solutions.

In all subsequent analysis, the Pseudo-RPR has been dropped in favour of the more traditional RPR with the optimised parameters α=1 and β=2. It should be noted that this change may have reduced the level of intensification in the algorithm, and consideration may need to be given to this, if result quality is not improved later in this analysis.

The two matrices that the RPR takes information from (and combines to compute the respective probabilities) are now considered for optimisation.

4.5.2 RPR Component Parameters

The RPR computes a probability for each arc that could[7] be appended to the current solution based on a composition of two distinct components (the trail and the visibility). As these are independent, it seems sensible to consider optimising each of these components separately.

The Trail Matrix

It seems sensible to begin with the more interesting and far more complex trail component. There are five parameters that are going to be considered in this review: ρ, γ_r, m, t and τ_0. The definitions of each of these parameters has been well explained

[7] Using the TTV criterion for tentative vehicle schedule completion, the term 'could' relates to all currently unallocated customers.

in the previous description of the ACS-DVRP, though they will be briefly recapped prior to each step of the optimisation process.

The evaporation rate ρ is crucial to the ACS algorithm with it impacting both the local update and global update rules. The use of the same parameter for these clearly distinct purposes is possibly questionable (in the original algorithm development they were separate parameters), but with Gambardella et al. (1999) producing competitive results for the VRPTW this idea is not explored.

The higher the value of ρ, the faster the local updating rule will drag the trail values back to their starting value of τ_0 (diversifying the search). Conversely, as ρ increases, the global updating will intensify the search further (note that one must be careful not to select too high a value, as this will cause the search to stagnate). A variety of values for ρ are assessed in Table 4.7.

ρ		c100	f71	tai75a
0.05	Min	1009.32	317.93	**1799.22**
	Ave	1060.14	331.80	**1931.58**
	Max	**1089.00**	**346.98**	**2017.74**
0.1	Min	993.88	**297.25**	1837.80
	Ave	**1052.02**	**322.96**	1961.08
	Max	1119.41	351.10	2116.66
0.3	Min	**992.12**	319.24	1824.16
	Ave	1053.94	333.49	2019.16
	Max	1120.13	357.36	2361.60
0.5	Min	1031.57	305.76	1896.50
	Ave	1063.31	329.65	2020.29
	Max	1110.63	350.67	2162.05

Table 4.7: Analysing Various Values of ρ

It seems clear that the lower values of ρ are producing results of a higher quality. Given that Montemanni et al. (2005) proposed a value of ρ=0.1, and these results show little to suggest this is not an appropriate selection, no alteration is made to this parameter. All subsequent analysis will continue to use ρ=0.1.

In a similar vein to the impact that ρ has in the local updating rule, the parameter γ_r dictates the level of drag back in the transition-updating phase. This is an important feature of the ACS-DVRP, as it is one of the primary reasons why the ACO metaheuristic seemed such a sensible choice for implementation on this problem. This update rule was included to retain some of the information that has been learnt in previous timeslots for customer arcs still available in subsequent timeslots.

If too much information about previous timeslots is retained then the search is not likely to be diverse enough to consider the customers that have just arrived. Conversely, it would be foolish to discard all of the information learnt in the previous timeslot, only to relearn the same information. A variety of values for γ_r have been assessed and the results are presented in Table 4.8.

γ_r		c100	f71	tai75a
0.1	Min	1017.05	309.30	1873.85
	Ave	1057.21	327.04	1999.17
	Max	**1112.13**	355.17	2126.32
0.3	Min	**993.88**	**297.25**	1837.80
	Ave	**1052.02**	**322.96**	**1961.08**
	Max	1119.41	351.10	**2116.66**
0.5	Min	1021.97	317.74	**1792.73**
	Ave	1073.27	330.69	1963.05
	Max	1117.34	**338.57**	2280.79
1	Min	1033.41	321.09	1872.86
	Ave	1087.15	331.29	2006.20
	Max	1150.87	345.30	2152.41

Table 4.8: Assessing Various Values of γ_r

The results of this analysis appear to support the findings of Montemanni et al. (2005) with a decline in solution quality apparent either side of their proposed value of γ_r=0.3. It is interesting to note that the worst performing of the four values assessed is when γ_r=1, which equates to no transition of information. This poor solution quality supports the inclusion of the third update rule, though no alteration to its present value is proposed.

With suitable values for ρ and γ_r identified, the parameter m (that dictates the frequency of the global update rule's implementation) is assessed. It does not represent a fair assessment if various values of m are simply analysed, due to m partially dictating the total number of solutions that are constructed ($m \times t$). Therefore, the parameter t must be reassessed in conjunction with the parameter m, thus allowing the number of solutions constructed to remain a constant (see Table 4.9).

m	*t*		**c100**	**f71**	**tai75a**
		Min	1007.71	318.15	1860.28
2	22500	Ave	1066.08	332.64	1964.52
		Max	1121.80	361.19	2090.39
		Min	**993.88**	**297.25**	1837.80
3	15000	Ave	**1052.02**	**322.96**	**1961.08**
		Max	1119.41	351.10	2116.66
		Min	1018.99	312.05	**1805.68**
5	9000	Ave	1061.26	336.05	1966.93
		Max	**1110.66**	351.26	**2055.15**
		Min	1043.99	315.96	1830.91
9	5000	Ave	1089.76	331.82	1986.84
		Max	1135.05	**349.82**	2182.29

Table 4.9: Assessing the Relationship between m and t

This analysis is of a similar form to the one carried out by Montemanni et al. (2005), whereby they assess various values of m, though their exit criterion determined by T_{ts}, automatically compensates. It reaches the same conclusion, with m=3 outperforming the other values assessed. This selection is supported across all three datasets, thus the previously utilised values of m=3, t=15000 remain.

With four of the five parameters being assessed and no changes being found, it suggests that the algorithm developed by Montemanni et al. (2005) is not a simple one to improve. Consideration is now given to the final parameter considered in this analysis, namely τ_0.

Somewhat surprisingly Montemanni et al. (2005) did not consider any other values for this parameter, presumably relying on the workings of Gambardella et al. (1999)

i.e. assuming the most appropriate value had already been selected. Although the solution quality of the ACS-DVRP suggests that this choice was not a bad selection, for completeness it seems sensible to see if better values exist.

Gambardella et al. (1999) proposed $\tau_0 = 1/(n \times Cost(PI))$ and Montemanni et al. (2005) continued with this definition. This was amended to $\tau_0 = 1/(2n_0 \times Cost(PI))$ (see Section 4.2.1) so one could claim to be solving a truly dynamic problem (i.e. one did not require to know the exact size of the problem being solved). To assess whether this is a good value for this parameter, one needs to consider a variety of other possible values. It is proposed that $\tau_0 = 1/(\upsilon \times Cost(PI))$, where $\upsilon > 0$ (see Table 4.10).

υ	Dataset	c100	f71	tai75a
	Min	1315.49	342.37	2036.56
1	Ave	1419.01	356.91	2160.62
	Max	1555.35	381.49	2375.44
	Min	1058.68	310.57	1924.32
n_0	Ave	1107.14	328.22	2015.17
	Max	1178.80	**345.44**	2128.99
	Min	**993.88**	**297.25**	1837.80
$2n_0 \approx n$	Ave	1052.02	**322.96**	1961.08
	Max	1119.41	351.10	**2116.66**
	Min	1007.99	309.88	1804.19
$4n_0 \approx 2n$	Ave	**1046.73**	329.72	**1959.67**
	Max	**1096.52**	345.79	2159.06
	Min	1002.93	315.92	**1773.47**
$6n_0 \approx 3n$	Ave	1061.03	333.02	2016.69
	Max	1101.23	350.89	2245.18

Table 4.10: Analysing Various Values of υ

Note that as υ increases, the initialisation value for the trail will decrease. If one examines the impact that this will have on the construction process, one can see that the search will be intensified.

The majority of the good solutions seem to be covered by two of the proposed values, namely $\upsilon=2n_0$ and $\upsilon=4n_0$. Making a choice between these is not easy, as their performances are quite similar (signifying a fairly robust parameter). However, with two of the three lowest averages, one could argue that $\upsilon=4n_0$ slightly outperforms $\upsilon=2n_0$, thus for the subsequent analysis $\upsilon=4n_0$ is utilised.

The analysis of the parameters that impact on the values contained in the trail matrix is now concluded. The choices made by Montemanni et al. (2005) via their parameter review were supported in this analysis, as was the majority of the information that they took from Gambardella et al. (1999). An alteration to the definition of τ_0 has been suggested. This change allowed it to be optimised; the result of this was a change in the proposed value of υ, though the improvement found was only marginal.

The Visibility Matrix

This is a far less complex entity to consider than the trail matrix, as the values remain fixed throughout the duration of the algorithm. The only control one has is in what formula is used to calculate the values in the first place. The current definition $\eta_{ij} = 1/c_{ij}$ (based on the NN algorithm) is certainly the traditional one used for routing problems, though Doerner et al. (2002) have found interesting results on the CVRP when using a different visibility definition.

They were using an AS_{rank} algorithm to solve the CVRP and considered incorporating a problem specific heuristic into the construction process. They argued that the current NN-type approach was too heavily influenced by the TSP, and did not acknowledge the clear distinction between the two problems.

They redefined the visibility as the problem specific C&WS function (Clarke and Wright (1962), see Section 3.3.2), in the hope that this tailored approach would yield further benefits. Their redefined visibility, η_{ij}, can be seen in Equation 4.3.

$$\eta_{ij} = s_{ij} = c_{i0} + c_{0j} - c_{ij} \tag{4.3}$$

This technique was given the moniker Savings-Ants, and the results that they achieved, improved upon the then best-known ACO implementation (AS/EAS approach by Bullnheimer et al. (1999c)). Note that it is not known if this improvement was due to the redefined visibility, as they also included other amendments and were comparing against a less complex ACO algorithm.

However, with encouraging results, and what seems an interesting approach, the Savings-Ants are considered for incorporation into this ACS. The visibility function is altered from its current NN-type to that given in Equation 4.3 and the results are presented in Table 4.11.

Visibility		c100	f71	tai75a
NN	Min	**1007.99**	**309.88**	**1804.19**
	Ave	**1046.73**	**329.72**	**1959.67**
	Max	**1096.52**	**345.79**	**2159.06**
Savings	Min	1090.59	316.07	2002.93
	Ave	1106.16	334.16	2136.05
	Max	1129.18	357.85	2336.89

Table 4.11: Comparing NN and Savings definitions for η_{ij}

It is clear that in this instance the use of the redefined visibility has a detrimental effect on the result quality. The size in the performance drop is somewhat unexpected; it has significantly reduced the quality of the solutions (across all three datasets). This is in stark contrast to the increase in performance that Doerner et al. (2002) experienced, and suggests that Savings-Ants are either better suited to the AS_{rank} framework or the simpler CVRP (or both).

There do not appear to be any other alternative visibility definitions that have been considered for the CVRP (and would be appropriate for the DVRP), thus the decision is made to continue with the NN based definition. This concludes the parameter review for the visibility matrix.

4.6 The Proposed Improvements

With all of the existing parameters analysed, one now needs to consider implementing the proposed improvements of Section 4.3 (TTV excluded as this has already been implemented). The first improvement concerns extending the roulette wheel (so as to prevent sequential route construction), while it is expected that this will require the introduction of the VRT in order to be competitive.

After the analysis regarding the inclusion of the two new elements has been completed, the improvement phase will be reintroduced, as all of the optimisation and proposed improvements to the algorithm will be complete.

4.6.1 The ERWACS-DVRP

With the technique described in Section 4.3.2 the focus of this section is the actual implementation of the extended roulette wheel. The basic premise is simple; the optimised ACS-DVRP is adjusted (and renamed the ERWACS-DVRP) to incorporate this new feature, and the results are compared.

As the technique removes the sequential construction (which has never previously been considered), and allows for the frequent deployment of further vehicles, it is difficult to be sure as to what impact this is likely to have on the results. As stated previously, one may well be required to introduce some further kind of restriction on the vehicles (VRT), though these will be assessed subsequently.

The results of the ERWACS-DVRP are presented in Table 4.12.

Method	Dataset	**c100**	**f71**	**tai75a**
ACS	Min	**1007.99**	**309.88**	1804.19
	Ave	**1046.73**	329.72	1959.67
	Max	**1096.52**	345.79	2159.06
ERW ACS	Min	1158.83	314.12	**1769.23**
	Ave	1214.26	**325.36**	**1867.99**
	Max	1291.24	**336.37**	**1945.37**

Table 4.12: Comparing the ACS-DVRP with the ERWACS-DVRP

These results suggest that the technique can be very effective, though seemingly only in certain circumstances. Results for tai75a have been improved considerably (4.68% decrease in average cost), while less significant improvements can also be seen in f71 (1.68% decrease in average cost). However, the most startling impact is the sizable increase in the distances associated with the c100 dataset (16.01% increase in average cost).

This increase clearly offsets the positive results achieved across the other two datasets, and as such, in its current state one would not be able to select the ERWACS-DVRP as a method for solving these three benchmarks (one is trying to produce a single robust algorithm that can cope with a variety of different problem scenarios). If the extended roulette wheel is to prove its worth, the poor performance in c100 needs to be eliminated.

It should also be noted that the ERWACS-DVRP requires more time to produce the solutions than the ACS-DVRP (this increase will be discussed in Section 4.7.2).

Prior to implementation, it was suspected that the continued potential for further vehicle deployment might result in an increase in the number of vehicles being used. It is necessary to investigate if this, or some other occurrence is responsible for the poor performance on the c100 dataset. The numbers of vehicles deployed in the final solutions (for the c100 dataset) were compared for the two methods.

The optimised ACS-DVRP deployed an average (still from a trial of 10) of 8.7 vehicles, which when compared with the average of 11.5 for the ERWACS-DVRP supports the aforementioned belief that this could be a problem. Referring back to the geographical representation of c100 (see Section 3.1.1), one can see that there is a more random dispersion of customers (in each timeslot) than for the other datasets. If one considers the way in which the extended roulette wheel operates, one would be more likely to favour a new vehicle if there are no customers very close to one of the existing vehicles (which is more likely to occur in a random dataset than a clustered one).

It seems reasonable to assume that c100 is more susceptible to the further deployment of vehicles than those datasets that appear more clustered (i.e. f71 and tai75a). Two methods for counteracting this problem were discussed earlier and are now considered.

4.6.2 Vehicle Reduction Techniques

The two distinct techniques were discussed in detail in Section 4.3.3 and are now ready to be implemented with the ERWACS-DVRP (thus allowing one to assess their impact). Note that given a new parameter is required for each of these methods, and there is little in the way of precedent (especially for the DDV), a precursory analysis represented the best method for identifying suitable values. However, before the results of this analysis are presented, it seems sensible to consider why it is so important that a suitable parameter value is identified for each of these techniques.

For either of these techniques, if too extreme a value is chosen, then the algorithm will either force the number of vehicles down (making decisions on capacity rather than distance), or not be restrictive enough to have the required impact (i.e. too many vehicles are still deployed).

The precursory analysis suggested that a value of λ=1 would operate best for the CU factor (which is the same value as chosen by Bullnheimer et al. (1999b) in their CVRP implementation) and that a value of θ= 0.5 would operate best for the DDV. As all of the necessary details regarding the inclusion of these two VRT techniques in the ERWACS-DVRP were presented in Section 4.3.3, their impact is now assessed.

Prior to considering the actual distances produced, it is worth noting that both techniques did succeed in their objective of reducing the number of vehicles being deployed. Before their inclusion (on the problematic dataset (c100)), the ERWACS-DVRP was deploying an average of 11.5 vehicles compared with the 8.7 for the optimised ACS-DVRP. The inclusion of the VRTs significantly altered this figure, dropping to 8.4 when the CU factor was included and to 9 when the DDV was included.

Obviously these reductions are desirable, but these figures alone do not allow one to judge if the methods have been a genuine success. It is necessary to consider what impact they have had on the distance. Any reduction in the number of vehicles is immaterial if it is achieved at an increase in the distance travelled. Results are presented in Table 4.13.

VRT	Dataset	c100	f71	tai75a
None	Min	1158.83	314.12	**1769.23**
	Ave	1214.26	325.36	**1867.99**
	Max	1291.24	336.37	**1945.37**
CU (λ=1)	Min	1020.85	**309.79**	1878.46
	Ave	**1063.15**	**318.14**	1923.08
	Max	**1114.99**	334.57	1971.39
DDV (θ=0.5)	Min	**1017.66**	311.49	1836.59
	Ave	1093.89	323.34	1898.91
	Max	1127.44	**330.62**	1951.70

Table 4.13:Assessing the Impact of the Vehicle Reduction Techniques

As this table clearly shows, the distances for the problematic dataset have been significantly reduced (though not quite to the level that they were at prior to the inclusion of the extended roulette wheel). However, the inclusion of these VRTs need to be assessed across all three datasets, and unfortunately the results are somewhat mixed.

Further improvements have been found with regards to f71, but both techniques seemed detrimental to the success on tai75a. However, due to the particularly poor performance of the ERWACS-DVRP on c100 prior to their inclusion, if one is to continue with the extended roulette wheel, it is necessary that one of these VRTs is chosen. The CU factor seems to represent a better compromise across all thee datasets than the DDV and is therefore included in the ERWACS-DVRP.

Note that it does not seem sensible to implement both methods in conjunction with each other, as the number of vehicles deployed in c100 has been reduced to beneath the level that was achieved prior to the inclusion of the extended roulette wheel. This

concludes the adaptations that are made to the construction of solutions and consideration is given to the algorithms improvement phase.

4.6.3 Reintroduction of Improvement Phase

With many construction or population (or both) metaheuristics, it is possible to include an improvement phase which implements some form of local search upon solutions. The ACS-DVRP included a simple descent algorithm that attempted to improve the solution constructed by each ant (though the proportion of time dedicated to improvement was small). This section considers reintroducing this improvement phase (and adapting it slightly) to see if the performance of the ERWACS-DVRP can be further improved.

Traditionally, an improvement phase (by its very design) should not be able to have a negative impact upon solution quality. Whilst this remains true with regards to individual solutions within a specific timeslot, it is possible that a better solution in one timeslot may lead to a worse solution in a subsequent timeslot. Unfortunately this is symptomatic of the problem being solved and as such, there is little that one can do to prevent this from occurring.

The neighbourhood utilised by Montemanni et al. (2005) was the previously described 1-Opt neighbourhood (Section 3.3.4), consisting of the i:1,0 and x:1,0 moves. This neighbourhood is analysed along with another consisting of two different moves: i:1,1 and x:1,1 (see Figure 4.2). An amalgamation of the two neighbourhoods will also be assessed (i.e. λ-Opt where λ=1[8]).

This new neighbourhood effectively swaps the placements of two customers whether they are in the same vehicle (i:1,1) or in different vehicles (x:1,1). As with the 1-Opt neighbourhood (i:1,0 and x:1,0), a 50/50 choice is made as to which type of move to make (cross vehicle or intra). The final composite neighbourhood consists of four possible moves, each of which is assigned a 25% probability of being selected (each iteration).

[8] As λ=1 (i.e. only a single customer is considered) this neighbourhood could also be referred to as L-Opt (L=1).

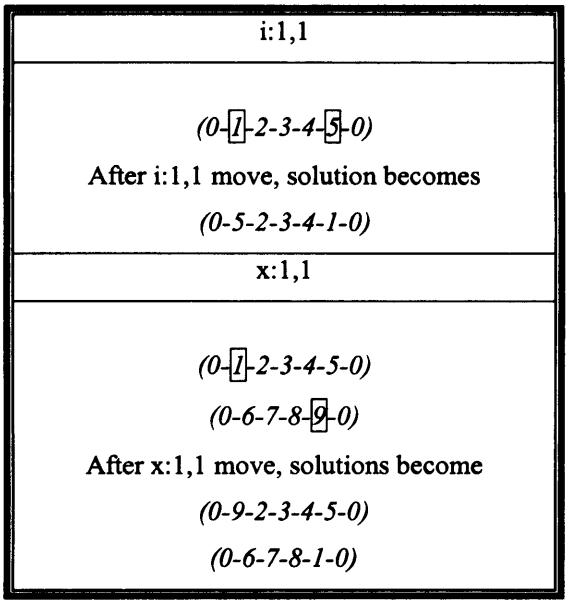

Figure 4.2: The Two New Neighbourhood Components

As one would hope that the ants will produce solutions of a higher quality than NN or PNN (as in GRASP), it seems sensible to reduce the number of iterations without improvement from the descent phase. This will allow for a greater number of cycles to be completed in the time-constrained variant of the problem. Preliminary analysis showed that reducing z=3000 to z=1000 had little impact on the quality of the results identified. It should be noted that Montemanni et al. (2005) devoted very little time to descent, preferring to rely on the ants to construct high quality solutions.

Results for the reintroduction of the descent (z=1000) with varying neighbourhoods can be seen in Table 4.14.

One can see immediately that all three of the neighbourhoods have increased solution quality across each of the three datasets. As for which performs best, it is clearly the original neighbourhood (i:1,0 and x:1,0) as utilised by Montemanni et al. (2005). Although the larger neighbourhood λ-Opt (λ=1) cannot perform to a worse standard within the individual timeslots (assuming a local optimum is being reached), these results suggest such an increase in neighbourhood size is not necessary.

The 1-Opt neighbourhood will also have less impact on the number of cycles that one will be able to complete in the time-constrained variant of the problem than the expanded neighbourhood λ-Opt (λ=1). Note that one will still have to consider

whether the neighbourhood is appropriate for the time-constrained variant of the problem. If the difference between result quality in the non time-constrained and time-constrained variants of the problem is sizable, then it is likely that insufficient cycles have been completed and one will have to reconsider the inclusion of such a sizable descent component.

Descent		c100	f71	tai75a
None	Min	1020.85	309.79	1878.46
	Ave	1063.15	318.14	1923.08
	Max	1114.99	334.57	1971.39
i:1,0 and x:1,0	Min	**963.07**	**305.59**	1866.27
	Ave	**999.35**	313.97	**1883.78**
	Max	**1029.64**	**319.60**	**1901.35**
i:1,1 and x:1,1	Min	1000.12	307.79	**1782.31**
	Ave	1064.06	322.51	1911.20
	Max	1311.99	372.82	1976.00
λ-Opt (λ=1)	Min	973.87	**305.59**	1862.62
	Ave	1019.09	**313.80**	1887.07
	Max	1069.44	**319.60**	**1901.35**

Table 4.14: Comparing Various Neighbourhoods for the ERWACS-DVRP Improvement Phase

4.7 Results and Conclusions

This section is divided into two distinct sections, both of which are concerned with producing results for each of the datasets belonging to the complete set of benchmark problems. The first section will continue to tackle the problem on which the previous analysis has been conducted i.e. the non time-constrained variant of the problem. The second contains the first time-constrained results presented in this thesis. Note that this section contains a slight alteration to the improvement phase, resulting in an increase in the number of cycles able to be carried out in the 12.5 minutes.

For both of these sections, the results used for comparative purposes are those produced via the ACS-DVRP (Montemanni et al. (2005)). It should be noted that comparing the non time-constrained results of the ERWACS-DVRP with the time-constrained results for the ACS-DVRP is not a like-for-like comparison, and as such, conclusions drawn from this should be treated with trepidation.

4.7.1 Results: Non Time-Constrained

In Table 4.15 the results achieved by the optimised EWACS-DVRP are compared with the results obtained by Montemanni et al. (2005). In addition, the average time taken (in minutes) for the EWACS-DVRP to produce the results is given. It should be noted that this figure does not represent T, due to the uneven distribution of time across the 25 timeslots, as the size and complexity of the timeslots constantly change (with new customers arriving and others being committed to). An extrapolated value estimating the minimum length that the working day would have to be (based on the maximum amount of time spent solving a single timeslot) is also presented.

Dataset	*ACS-DVRP*			*ERWACS-DVRP*			ERWACS Ave	ERWACS Working
	Min	Ave	Max	Min	Ave	Max	Time	Day
c100*	973.26	1066.16	1100.61	**963.07**	**999.35**	**1029.64**	30	58
c100b	944.23	1023.60	1123.52	**889.30**	**909.48**	**930.37**	20	38
c120	1416.45	1525.15	1622.12	**1256.76**	**1331.25**	**1447.21**	25	48
c150	1345.73	1455.50	1522.45	**1308.01**	**1353.48**	**1413.44**	55	106
c199	1771.04	1844.82	1998.87	**1626.91**	**1654.24**	**1679.76**	93	179
c50	631.30	681.86	756.17	**592.42**	**622.83**	**639.22**	14	27
c75	1009.38	1042.39	1086.65	**948.15**	**1008.83**	**1038.79**	19	37
f134	**15135.50**	16083.56	17305.69	15290.58	**15577.90**	**15827.90**	10	19
f71*	311.18	348.69	420.14	**305.59**	**313.97**	**319.60**	19	37
tai100a	2375.92	2428.38	2575.70	**2187.23**	**2248.92**	**2303.20**	26	50
tai100b	2283.97	2347.90	2455.55	**2184.73**	**2253.25**	**2301.74**	26	50
tai100c	1562.30	1655.91	1804.20	**1524.24**	**1622.36**	**1682.00**	22	42
tai100d	2008.13	2060.72	2141.67	**1896.84**	**1969.24**	**2080.30**	29	56
tai150a	3644.78	3840.18	4214.00	**3364.48**	**3446.05**	**3618.59**	50	96
tai150b	3166.88	3327.47	3451.69	**2872.18**	**2889.56**	**2906.17**	57	110
tai150c	2811.48	3016.14	3226.73	**2595.27**	**2751.67**	**2895.55**	47	90
tai150d	3058.87	3203.75	3382.73	**2952.70**	**3025.66**	**3071.65**	49	94
tai75a*	**1843.08**	1945.20	2043.82	1866.27	**1883.78**	**1901.35**	17	33
tai75b	1535.43	1704.06	1923.64	**1464.29**	**1575.99**	**1675.46**	16	31
tai75c	**1574.98**	**1653.58**	1842.42	1619.27	1657.94	**1708.02**	18	35
tai75d	1472.35	1529.00	1647.15	**1436.15**	**1444.75**	**1457.15**	15	29

Table 4.15: Computational Results for the Non Time-Constrained ERWACS-DVRP

It should be noted that as Montemanni et al. (2005) showed the benefits of a pheromone based technique over that of a multi-start algorithm (i.e. α=0) a similar analysis has not been presented here.

These results are only the second complete set that have been produced for these benchmarks, and as such, the conclusions that one can draw are somewhat limited. Furthermore, as this comparison involves comparing results that were produced on the time-constrained variant of the problem (ACS-DVRP) with some that were not, one has to be careful when drawing conclusions.

The most basic observation is that result quality has markedly improved, with 59 of the 63 metrics reporting better results for the algorithm developed in this chapter. Before consideration as to whether this is likely to be due to the increase in time (the working day calculation was greater than 25mins for all but one of the datasets), it seems sensible to report some basic summary statistics to quantify how much the result quality has improved by.

Considering the percentage change in each of the three metrics averaged across the 21 datasets one can see that the minimum distances (from the trial) have been reduced by 4.45%. The changes in the average and maximum are even more impressive, where distances have been reduced by 6.86% and 10.69% respectively. Although these results are not directly comparable, these changes are significant because they show how much scope for improvement there is (when one considers the time-constrained variant). Note that in some cases the ACS-DVRP operating on the time-constrained variant has outperformed the ERWACS-DVRP on the non time-constrained variant.

One can also see that the variability in the results (within the trial of ten) has been significantly reduced ($\approx$50% less), which is a good achievement given that the high levels of variability appears to be one of the biggest problems faced when solving the DVRP.

With it firmly established that these results are better than those attributed to the ACS-DVRP, one can give consideration to the time for which the algorithm has been run, and the intensification/diversification balance.

Run Time Observations

Without recreating the ACS-DVRP and running it with a similar exit criterion (i.e. giving it as much time as it requires), one will never be entirely confident (from these results) that the algorithm developed in this chapter is superior. One might consider the fact that Montemanni et al. (2005) would have been unlikely to select an exit criterion that did their algorithm a disservice, but this is purely speculative and one should not assume this to be true.

However, although these results would be beneficial with regards to stating whether the algorithm has been improved, this is not the primary concern of this set of results. Of more interest is whether these results are suitable for application within a real-life environment.

If one considers the average time that the algorithm took to run (and the extrapolated values for T) there are some interesting observations to be made. None of the run-times are particularly excessive, with the slowest requiring the working day to be less than three hours long. This does not seem an unrealistic figure if one considers the actual times and distances likely to be involved when routing a fleet of vehicles to 199 customers.

The run times are fairly well correlated with the size of the problems being solved (as one would expect). However, one of the larger datasets (fl34) was the quickest to be solved; requiring a working day of just 19 minutes for the results to be valid. This is an unusual dataset as there is considerable variation in demand (1-1126), which is likely to result in vehicles being declared as TTV even though many customers could still be allocated to them. This problem although not desirable, should not be overstated, as the algorithm still seems to be able to produce average results of a higher standard (than the ACS-DVRP), though it would be worth investigating if further research was to be conducted in this area.

Intensification/Diversification Balance

While drawing conclusions on solution quality are complicated by the time discrepancies, analysing the performance of the algorithm, as a single entity remains unaffected.

When one considers the four metrics that were not bettered, three of them were the minimum (of the trial). This may have been because the algorithm of Montemanni et al. (2005) diversified the search to a greater extent (which would also go some way to explaining the greater levels of variability in result quality). Although the algorithm being improved is likely to reduce the variability, it will be interesting to see if similar behaviour is observed when one considers the time-constrained variant of the problem.

A possible reduction in diversification is of particular interest, as one would have expected the reverse to be true; the greedier Pseudo-RPR was removed in favour of the more explorative RPR. However, any reduction to the intensification level that this caused could well have been counteracted by one or a collection of the other adjustments (e.g. the criterion for tentative schedule completion or the decrease in the trail initialisation value).

In Section 4.1.2 the ACS was shown to be the most aggressive of the ACO algorithms, though when one considers the performance of the ERWACS-DVRP improvements were often observed in the latter cycles. It appears that a good balance between intensification and diversification has been identified, and were one to consider further changes to the algorithm this may well be affected.

The algorithm has shown that it can produce results of a high quality, though one has to consider whether this is due to the increase in time, or whether the ACS-DVRP has simply been bettered.

4.7.2 Results: Time-Constrained

The time-constrained problem allows for a direct comparison with the ACS-DVRP, thus conclusions can be drawn with far more certainty than from the results presented previously.

As stated in Chapter Two, the 25 minutes that Montemanni et al. (2005) had available, has been reduced (in accordance with Dongarra (2002)) to 12.5 minutes due to the relative increase in computing performance since 1999. Given the run-times presented in Table 4.15, it seems clear that imposing a time limit of 12.5 minutes is likely to significantly reduce the number of cycles completed. It therefore seems sensible to consider if there is any way to reduce the run-times, whilst not reducing the quality of the solution.

As the improvement phase of this algorithm is particularly time consuming (not true for the ACS-DVRP), it seems sensible to concentrate on identifying possible time savings in this area. Obviously the parameter z could be considered, reducing this will decrease the run-times, but it will lessen the time in which the descent algorithm has to reach a local optimum.

However, due to the unique way (with regards to other ACO based techniques) that ACS updates influence the trail, there is another option available that will not significantly alter the workings of the algorithm. Instead of reducing z, one need not carry out a descent phase on every solution. Every cycle of the ERWACS-DVRP algorithm involves the creation of m (set at 3) distinct solutions, each with varying distance. As some of these solutions may be inherently poor, it could be considered an inefficient use of the now somewhat limited time, trying to improve these.

It is proposed that a better use of the time will be to descend just the best solution each cycle, thus reducing the amount of time spent descending by approximately two thirds (not all solutions will require the same amount of time to descend). This will increase the proportion of time spent on the construction phase, thus increasing the number of cycles able to be completed. This theory is considered on the test-set (see Table 4.16).

The result quality does not seem to have been adversely affected by the reduction in the number of solutions that are being descended. Somewhat unexpectedly, it appears that solution quality is slightly higher in the case where only the best ant is considered in the improvement phase, though this has to be due to the randomness of the solution technique. Although this could have been implemented in the non time-constrained

variant of the problem (and seemingly not have greatly impacted the quality of the solutions), given the fact that this cannot increase the quality of the solutions it was felt preferable to descend each constructed solution.

Ant		c100	f71	tai75a
All	Min	981.28	305.59	**1862.62**
	Ave	1008.49	**313.00**	1903.20
	Max	**1046.65**	320.65	1980.51
Best	Min	**970.34**	**304.80**	1872.93
	Ave	**1006.16**	314.34	**1900.97**
	Max	1049.01	**319.60**	**1977.00**

Table 4.16: Comparing Descending from All Ants and the Best Ant.

The algorithm developed (including this adjustment on the number of solutions descended) is now considered on the complete set of benchmark problems (see Table 4.17).

This analysis is primarily concerned with identifying whether the results produced by the ACS-DVRP have been bettered, and if so, by how much. The NN (with descent) results are presented as a reference point so that one can see the strength of both ACO algorithms (note that they are not presented for a full analysis).

It seems sensible to begin with the discussion regarding the difference in solution quality between the simple NN (with descent) algorithm and the results produced via the two ACO techniques. These algorithms are not distinguished between in this analysis, as this discussion will form the basis of the next subsection.

Although this algorithm has not strictly been conducted on the time-constrained variant of the problem, as a single construction and improvement phase requires considerably less time such a comparison is not entirely without merit (as they are both valid solutions to the time-constrained problem).

	NN-DVRP	*ACS-DVRP*			*ERWACS-DVRP*		
		Min	Ave	Max	Min	Ave	Max
c100*	1463.17	973.26	1066.16	1100.61	**970.34**	**1006.16**	**1049.01**
c100b	1065.48	944.23	1023.60	1123.52	**874.06**	**900.50**	**916.99**
c120	1599.55	1416.45	1525.15	1622.12	**1256.27**	**1344.37**	**1440.99**
c150	1767.91	1345.73	1455.50	1522.45	**1342.79**	**1388.71**	**1447.88**
c199	2266.37	1771.04	1844.82	1998.87	**1654.54**	**1731.12**	**1780.06**
c50	832.10	631.30	681.86	756.17	**604.32**	**629.09**	**644.79**
c75	1220.04	1009.38	1042.39	1086.65	**989.13**	**1014.91**	**1052.74**
f134	16676.90	**15135.51**	16083.56	17305.69	15190.45	**15672.58**	**16406.66**
f71*	341.00	311.18	348.69	420.14	**304.80**	**314.34**	**319.60**
tai100a	3005.11	2375.92	2428.38	2575.70	**2186.11**	**2254.20**	**2359.76**
tai100b	2700.87	2283.97	2347.90	2455.55	**2207.39**	**2253.39**	**2292.99**
tai100c	2002.05	**1562.30**	**1655.91**	1804.20	1566.52	1667.85	**1682.00**
tai100d	2362.28	2008.13	2060.72	2141.67	**1907.50**	**2012.42**	**2127.42**
tai150a	4452.54	3644.78	3840.18	4214.00	**3319.35**	**3437.58**	**3537.85**
tai150b	3558.03	3166.88	3327.47	3451.69	**2874.75**	**2936.70**	**3054.33**
tai150c	3054.46	2811.48	3016.14	3226.73	**2576.08**	**2788.00**	**2953.54**
tai150d	3717.59	3058.87	3203.75	3382.73	**3004.66**	**3075.90**	**3179.98**
tai75a*	2065.63	**1843.08**	1945.20	2043.82	1872.93	**1900.97**	**1977.00**
tai75b	1708.00	1535.43	1704.06	1923.64	**1508.99**	**1586.60**	**1642.92**
tai75c	2138.05	**1574.98**	1653.58	1842.42	1583.20	**1653.18**	**1712.82**
tai75d	1817.83	1472.35	1529.00	1647.15	**1438.92**	**1446.30**	**1453.99**

Table 4.17: Time-Constrained Comparison of the NN-DVRP, ACS-DVRP and ERWACS-DVRP

If one considers the worse average ACO solution (from either of the two algorithms) one can see that the NN solution is not better 20 of the 21 benchmarks analysed. It should be noted that it does produce better solutions than the worse maximum of the two algorithms for 6 of the 21 benchmarks, but one should not be overly concerned with this performance as these worse solutions all belong to the ACS-DVRP (see analysis below).

Consideration can now be given to comparing the performance of the two ACO algorithms.

ACS-DVRP vs. ERWACS-DVRP

If one considers comparing the metrics in Table 4.17 (in much the same way as they were for the non time-constrained variant of the problem), one can see that 58 of the 63 metrics have been bettered. This suggests that the ERWACS-DVRP is consistently outperforming the ACS-DVRP and that at its most basic level the alterations made to the algorithm are successful.

Although there are many other points to consider within this analysis, the importance of this simple comparison should not be understated. This chapter's aim was to see if one could improve upon the results produced by the ACS-DVRP and these results suggest that the work conducted has been successful in achieving that.

As with the non time-constrained variant, it seems sensible to continue the analysis by considering some simple summary statistics. However, as these algorithms are now directly comparable, this analysis can be more in-depth and conclusions can be drawn with greater confidence.

If one considers the percentage change in each of the three metrics (averaged across the 21 datasets), one can see that minimum distances (from the trial) have been reduced by 3.81%. This improvement is even more apparent in the average and maximum metrics, where reductions of 5.92% and 9.44% respectively, have been achieved. Although these reductions are not as sizable as was achieved when the non time-constrained variant was considered, they are more significant as those results were not produced with the same exit criterion. A discussion on the change in performance in the ERWACS-DVRP with the two termination criteria is presented on completion of this analysis.

There are some other interesting observations that can be made from the results that demonstrate just how much the solution quality has been improved. In the CVRP and VRPTW (and many other combinatorial optimisation problems) much is made of the best solution that has ever been produced for a particular benchmark. Although this problem (and hence the benchmarks) are relatively recent in their inception, it is hoped that eventually people will consider this problem in a similar light.

With that in mind, when a new best solution was achieved by this algorithm the result has been shaded grey. As one can see, new best-achieved results[9] have been achieved for 17 of the 21 datasets, and that some of the improvements are very sizable indeed (e.g. 325.43 for tai150a). Interestingly, 10 average results were better than the previous best-achieved solution (e.g. 230.18 for tai150b) and 5 maximums were of a higher standard (e.g. 112.55 for tai150b). This gives a good impression of the size of the improvements made to the ACS-DVRP, though one should remember that if the problem becomes more established, one would expect the size of subsequent improvements to be far smaller.

The variability of the results produced by a single algorithm was cited as a potential problem in the DVRP when the very first metaheuristic was implemented in Chapter Three. Although the results of the ERWACS-DVRP still exhibit this, the amount of variability has been significantly reduced (≈50% less, as was the case for the non time-constrained analysis) when compared with the ACS-DVRP.

Although variability can be a problem in more traditional routing problems, the amount of it is often far less. Furthermore, in a real-life situation one may well be able to select the best result from a number of runs (i.e. a trial) and implement that. In the DVRP, one only has a single attempt at solving the global problem (though multiple attempts can be made at the individual timeslots if time permits), and as such, any reductions in variability are much welcomed.

Solution Quality Discrepancy

This subsection is concerned with quantifying the change in solution quality associated with the alteration in exit criterion (from giving the algorithm the time to perform to the best of its capabilities to a fixed limit of T=12.5 minutes).

The change in performance is identical for each of the three metrics (when averaged across the 21 datasets), with a reduction in solution quality of 0.01% when the time-constraint is introduced. This is a somewhat negligible difference, and one has to assume that the extra cycles being implemented in the non time-constrained variant

[9] This is specifically within the time-constrained variant of the problem.

are not strictly necessary. This may well be the case as one was erring on the side of caution when selecting appropriate termination criterion.

Of the five metrics that were not improved in the time-constrained variant (ACS-DVRP vs. ERWAC-DVRP) three of the same were not improved in the non time-contained variant. This suggests that either the technique is having greater difficulty with these datasets or (and perhaps more likely) the ACS-DVRP was performing to a better standard on these datasets.

4.7.3 Analysis Summary and Chapter Conclusions

If one were to summarise the performance of the ERWACS-DVRP, one would have to consider its development a success. The improvements that have been made to the ACS-DVRP (though the algorithm still contains many of the key components of the algorithm produced by Montemanni et al. (2005)) have improved solution quality across all of the benchmark datasets considered.

There are still some issues regarding the level of variability in the results produced (it is likely to be unavoidable due to the nature of the problem and the timeslot solution strategy), but it has been significantly reduced. One is likely to have much greater confidence in results produced from a single run of the ERWACS-DVRP than they would have had in the existing algorithm.

Obviously the amount of quantitative analysis is somewhat stifled by the lack of other results with which to make comparisons. If one were to be solving a more traditional problem (such as the CVRP or VRPTW) one could compare with a variety of different metaheuristics to establish a better understanding of its capabilities.

This chapter has shown how the DVRP can be approached in a similar fashion to more traditional problems with regards to comparing results and trying to better those produced by other authors. It is hoped that as the wider research community discovers the potential in solving these types of problems, and that others use these results in the way in which this chapter benefited from those produced by Montemanni et al. (2005). As explained in Chapter Three, Chapter Five is going to be considering

solving the same problem with a TS algorithm, thus one will be able to use these results for comparative purposes.

With regards to further possible enhancements to the algorithm (aside from experimenting with different neighbourhoods in the improvement phase), one is likely to need to be fairly innovative. The algorithm developed makes use of many of the most successful ACS techniques that have been developed on other problems. However, with research constantly evolving and new ideas being considered this may not remain the case for long. It would also be interesting to see research pursued within one of the other ACO algorithms, with MMAS suggested as the most likely to be competitive.

Chapter Five

Tabu Search: Enhancing its Performance

5.1 Introduction

Chapter Three assessed the merits of two simple heuristics and four more complex metaheuristics with regards to their application to the DVRP. The decision was made to continue research in two of the most fruitful areas. The first of these areas (ACO) was considered in the previous chapter, within which, significant improvements to the early implementation were identified. The purpose of this chapter is similar (to improve an algorithm's performance), however this time the starting algorithm comes in the form of the other metaheuristic chosen for discussion, namely TS.

Unlike the previous chapter (ACO), there does not exist a TS algorithm for the DVRP (using the timeslot solution strategy). As a result of this, the development continues from the basic (and rather unsuccessful) TS algorithm presented in Chapter Three. A brief recap of this can be seen in Section 5.1.1

The TS algorithm was classified as an LS algorithm in Chapter Two (alongside SA) and therefore requires a neighbourhood, within which the search process can take place. The current 1-Opt neighbourhood (as used in the TS algorithm in Chapter

Three) is unlikely to offer the scope required to produce solutions of a competitive nature and therefore must be improved. Two further neighbourhoods were briefly assessed in Chapter Four (used in the improvement phase in the hybrid ACS algorithm) neither of which offered an improvement to the currently well performing 1-Opt. A better neighbourhood will need to be found.

Along with identifying a better neighbourhood, one must find a tabu criterion (TC) that can operate successfully in conjunction with that neighbourhood. This leads to an interesting situation as to which to develop first; given that some TC will work best with a particular neighbourhood (as seen in Chapter Three). Section 5.1.2 discusses the advantages and disadvantages of producing the TC before the neighbourhood and vice versa.

After a choice has been made as to which of these two methods represents the most promising route forward, two sections are presented: one identifying a suitable TC and one analysing various neighbourhoods. Obviously the order in which these are presented depends on the outcome of the discussion in Section 5.1.2. Note that these analyses will include any necessary parameter optimisation.

With a suitable TC and neighbourhood identified, a further section detailing various shortcomings and potential remedies is presented. Upon completion of this section, the competitiveness of this new algorithm will be scrutinised on both the standard and time-constrained variants of the problem.

The aim for this chapter is to sufficiently improve the result quality, such that it can genuinely be considered as a realistic alternative to the adapted ACS algorithm (as identified in Chapter Four).

5.1.1 A Brief Recap of the TS Algorithm

The TS algorithm implemented in Chapter Three was flawed in so much as the TC failed to prevent cycling (indirect) in the intra vehicle components of the neighbourhood. To allow for the production of solutions the neighbourhood was split into two, with only the cross vehicle moves subject to the TS ideology (whilst the intra vehicle moves operated in a simple descent fashion).

The TC stored the customer number, placement and vehicle for the previous and current positions each time a cross vehicle move was made. Upon the completion of *xt* cross vehicle moves, one would descend to a local optimum with the intra vehicle components of the neighbourhood. It was suggested that this split was the likely cause of the relatively poor performance of the TS metaheuristic (which has been shown to be one of the most successful at solving other routing problems).

Although it would be possible to adapt this TC so that one could prevent the need to split the neighbourhood (see workings of Osman (1993)) it would only be appropriate for this specific neighbourhood. It is hoped that during the workings of this chapter, a better performing neighbourhood will be identified, thus implementing this change is not necessary (unless no neighbourhood improvements are identified).

5.1.2 Should One Adapt the Neighbourhood or TC First?

As outlined in the introduction, when implementing a TS algorithm, one cannot simply begin by analysing various neighbourhoods. An improvement (in terms of complexity) to the neighbourhood will be somewhat redundant unless it is being used in conjunction with an appropriate TC[1]. Unfortunately, one could present a similar argument for beginning with the development of a TS algorithm with identifying a TC. Without the neighbourhood with which it is to be used in conjunction with being known, one is unlikely to know what TC will be compatible (i.e. if they will work well together).

Obviously one of these crucial components will need to be developed before the other, and as such, a choice needs to be made. Details regarding the two approaches, and how one would cope with the lack of knowledge regarding the component not being considered are presented below:

- TC then Neighbourhood

  Requires a very general TC to be identified i.e. one that is applicable to all/most potential neighbourhoods (with little or no modifications). Once this

[1] Failure to do so is likely to result in a poor level of performance, as seen in the TS implementation in Chapter Three.

suitable TC is chosen, various neighbourhoods can be assessed to identify a suitable one for this problem.

- Neighbourhood then TC
  Various neighbourhoods can be assessed outside of the TS framework i.e. within an algorithm that does not need adapting depending on the neighbourhood utilised (e.g. SA). Once a well performing neighbourhood has been identified, consideration can be given to identifying a compatible TC (this may still be a generalised TC).

There is little in the way of precedent upon which to base this decision, as this problem is often not addressed in literature. Either of these pathways would offer a realistic route with which to progress, and as such the decision is fairly subjective.

One would say that it is not usual to develop a neighbourhood within a TS algorithm, but this chapter is looking to implement variants of existing neighbourhoods rather than developing entirely new ones. However, given the lack of research into this DVRP it seems sensible to focus on the neighbourhood first. It will allow an easier transfer of well-performing VRP neighbourhoods (which seems the most sensible direction to pursue). This method also has the advantage that the neighbourhoods are being assessed in a simpler algorithm. This increases the likelihood of any differences in solution quality being down to the alteration in neighbourhood, rather than a variety of other (less controllable) factors.

5.2 Identifying a Suitable Neighbourhood

When the LS category of metaheuristics was first introduced, it was made clear that the selection of an appropriate neighbourhood is critical to the success (or failure) of techniques classified in this sub-group. As a member of this sub-group, TS will not be capable of high quality solutions unless one can be sure that the neighbourhood can successfully reduce the distance. Given its importance, the identification of a suitable neighbourhood will constitute a considerable volume of the work in this chapter.

As a decision has been made to assess the neighbourhood prior to identifying the TC, it has been proposed that one considers the neighbourhood within an alterative LS

algorithm. The frameworks for other LS algorithms were discussed in Chapter Two, and it seems sensible to make use of one or more of these.

5.2.1 Temporarily Abandoning TS in Favour of Descent and SA

The choice as to which algorithm(s) one should use (as a test-bed for the soon to be proposed different neighbourhoods) is fairly subjective. The comparison will need to be fair, and as such, one has to ensure that the only variable is the neighbourhood.

The obvious starting point for any local search technique is the standard descent algorithm (FI). This algorithm was detailed in Chapter Three, but was shown to produce rather uncompetitive results (as expected). Given this poor performance, it seems prudent to consider another algorithm. This second algorithm will ideally be more competitive, thus enabling one to consider the impact that the neighbourhood has on better solutions.

Given that the purpose of this section is to investigate various neighbourhoods that could later be introduced into a TS algorithm, it seems sensible to select an algorithm that shares various ideologies with TS.

The closest[2] of the previously implemented algorithms is clearly SA. Similarly to TS, SA allows increasing moves to be made in the hope of escaping a local optimum (a region of the search-space within which a standard descent algorithm would get trapped). However, in contrast to the existing TS implementation, SA does not require any adaptations to be made when implementing it with different neighbourhoods. One may need to consider the parameters (t_i, t_t and α) if they have been optimised for a specific neighbourhood, but given the desire to give the algorithm the best chance possible these have been set in such a way (high start temperature, low exit temperature, slow cooling rate) that this is unlikely to be an issue.

Analysing two different algorithms (descent and SA) will enable any conclusions to be made with greater confidence, as the impact that randomness can make, will be lessened.

[2] Classified into the same sub-group in Figure 3.1, Chapter 3.

Both of these techniques are clearly defined as improvement techniques, and as such require initial solutions to be produced. To ensure the best chance of a neighbourhood showcasing its strengths, two construction algorithms both of which were detailed in Chapter Three, are used:

- Random

  This is likely to produce a poor solution that all suitable neighbourhoods should be able to significantly improve upon. It is hoped that a higher quality neighbourhood will be able to identify further reductions thus enabling one to distinguish their performance.

- PNN

  These solutions are likely to be of a higher quality than the random construction, and are therefore useful in identifying a neighbourhood's ability to improve a good solution (this is more applicable to the descent algorithm than SA, see conclusions of this analysis). Note that PNN has been selected in favour of NN as one is conducting a trial, thus the added randomness in desirable.

Note that once one has completed the neighbourhood analysis i.e. established which neighbourhood one wishes to use for the remainder of this chapter, the descent and SA algorithms are to be discarded. One will revert back to the TS metaheuristic and consideration will be given to the establishment of a suitable TC (one that complements the aforementioned neighbourhood).

5.2.2 A Larger and More Complex Neighbourhood

The existing descent and SA implementations were implemented on a random initial solution, with the improvements coming from the previously defined 1-Opt neighbourhood (containing the i:1,0 and x:1,0 moves). However, it is well known that there exist more complex and better performing neighbourhoods than 1-Opt for routing problems such as the TSP and VRP. It seems sensible to assume that such improvements are likely to follow through to the DVRP.

Using the results produced under the 1-Opt neighbourhood as a base set of results, it is possible to compare the performance of various other neighbourhoods. It should be noted that as subsequent neighbourhoods will contain the existing 1-Opt neighbourhood, they are clearly more complex. Although a neighbourhood that can evaluate a great number of moves is desirable, as the local optimum will be better[3], one must remember that the time an algorithm requires can have a significant impact on performance. Were a more complex neighbourhood implemented in a BI strategy (which TS uses), each move would require an increased number of calculations, which could result in a significant increase in the algorithms run time or reduction in the number of moves made if one were considering the time-constrained variant of the problem.

In Chapter Four, the first alternative to the 1-Opt neighbourhood was proposed. It suggested two different customer exchanges, and tested their performance both independent of the existing neighbourhood and in conjunction. However, the moves (i:1,1 and x:1,1) were discarded in the improvement phase analysis of the ERWACS-DVRP, as they did not seem to improve the performance of the algorithm.

Given this poor performance (albeit on far better solutions than the random or PNN construction algorithms are likely to produce), it is not unreasonable to assume that this neighbourhood is unlikely to offer the level of improvement one is hoping for in this analysis. It is suggested that one expands the neighbourhood further before including it in descent and SA algorithms.

It should be noted that due to the complexities of the problem being solved, the decision has been made not to consider neighbourhoods that allow infeasible solutions (i.e. strategic oscillation) to be considered. These types of neighbourhoods (as discussed in Chapter Two) are likely to further complicate proceedings, whilst increasing the algorithm's run-time. Prior to considering any potential enhancements, some notes regarding the feasibility (and ways to increase the likelihood of feasibility) of moves are presented.

[3] Provided the smaller neighbourhood is a subset of the larger neighbourhood.

Feasibility

The current 1-Opt neighbourhood has been successful in reducing the distances within a variety of algorithms, namely descent, SA, GRASP, TS (in Chapter 3) and ACS (in Chapter 4). Given this success, it is clear that the 1-Opt neighbourhood is successfully identifying moves that reduce the distance of a particular solution. This is crucial to any neighbourhood, as no matter how many moves are considered, if it fails to identify feasible moves (with regards to the time or capacity constraints) the neighbourhood is inappropriate.

As it is desirable that one identifies as many feasible moves as possible (as this will give a better chance of improving the solution quality), it seems sensible to begin the neighbourhood expansion by attempting to retain the 1-Opt neighbourhood's success in this area. Some details are presented with regards to how one could design a neighbourhood that is tailored towards moves that are more likely to be feasible.

The 1-Opt neighbourhood contains two types of move (i:1,0 and x:1,0). The intra vehicle move does not alter the length (number of customers) or capacity of a route, thus these moves can only cause a route to become infeasible, if the duration of that route becomes longer than the working day. The cross vehicle move (which moves a customer from one vehicle to another) is more interesting with regards to its potential impact with regards to feasibility. The vehicle from which the customer is being taken cannot become infeasible as the load and distance will both be reduced (distance is reduced due to the triangle inequality, as all of the datasets are symmetric). However, the feasibility of the vehicle gaining the customer must be assessed, as both the load and the distance will be increased.

The vehicle load will increase by the demand of the customer being gained. As it has been suggested that there is a correlation between good solutions and those with fewer vehicles, it is reasonable to assume that in good solutions, the capacity utilisation is likely to be high. Were the demand of the customer being moved high, it seems that it is less likely that the move would be acceptable with regards to the capacity constraint. A similar argument can be considered for the time constraint, whereby in good solutions (with fewer vehicles), there will be less slack time in their tentative schedules. Were the increase in distance to be large (i.e. customer was located far

from the current vehicle route), it seems less likely that the move would be acceptable with regards to the distance constraints.

These ideas suggest that for a move to be likely to be feasible, one would hope for minimal increases in both load and distance (for the vehicle gaining the customer). Using this theory, one can assume that moves which are attempting to add a single extra customer to a vehicle (as in x:1,0), are more likely to be feasible than one where more customers are moved (e.g. x:5,0).

If one wished to continue with neighbourhood moves that have an increased likelihood of being feasible, it seems sensible to concentrate on customer exchanges that do not dramatically alter the length of a solution. One can generalise this theory by restricting the moves to those that either do not adjust the length of a solution (i.e. x:L_a,L_b where L_a=L_b), or those that only alter a vehicle's length by one customer (i.e. x:L_a,L_b where L_a-L_b=±1). Note that all four neighbourhood moves currently analysed adhere to these restrictions.

Some further moves (that meet these restrictions) are now considered.

Implementation and Results

As it has been proposed that for this first neighbourhood analysis, any extra moves that are added to the neighbourhood should adhere to one of the two conditions previously identified (with regards to the size of L_a and L_b), their selection is fairly straightforward. Consideration is given to two moves that exchange two customers for a single customer, and two moves that exchange two sets of two customers (these moves are subject to the restriction that the customers are required to be consecutive).

These new moves are appended to the existing expanded neighbourhood's four moves (i:1,0, x:1,0, i:1,1 and x:1,1) and can be seen in Figure 5.1. Note that the selection of which move to make is still made randomly.

This expanded neighbourhood now consists of 8 potential moves (each of which requires its own programming), and subsequently will be referred to as the 8-Move

neighbourhood. If the 8-Move neighbourhood was found to offer a noticeable improvement to the 1-Opt neighbourhood (when implemented in the descent and SA algorithms), then it would seem sensible to consider further neighbourhoods based on consecutive customer exchanges.

i:2,1	i:2,2
(0-1-2-3-4-5-0) After i:2,1 move, solution becomes *(0-5-3-4-1-2-0)*	*(0-1-2-3-4-5-0)* After i:2,2 move, solution becomes *(0-4-5-3-1-2-0)*
x:2,1	x:2,2
(0-1-2-3-4-5-0) *(0-6-7-8-9-0)* After x:2,1 move, solutions become *(0-9-3-4-5-0)* *(0-6-7-8-1-2-0)*	*(0-1-2-3-4-5-0)* *(0-6-7-8-9-0)* After x:2,2 move, solutions become *(0-8-9-3-4-5-0)* *(0-6-7-1-2-0)*

Figure 5.1: The Four Additional Components of the 8-Move Neighbourhood

Results presenting the performance of the 1-Opt and the 8-Move neighbourhoods (with the two different construction algorithms) are presented in Table 5.1 and 5.2 for descent and SA respectively. The parameters for the algorithms with which the previously described neighbourhoods are to be tested, are the same as was identified in Chapter Three, and are presented prior to each of the tables.

As one would expect, the neighbourhood containing the greater number of possible moves is performing to a higher standard than the now comparatively restrictive 1-Opt neighbourhood. It should be noted that this increase in performance could only be foreseen as there was no time-constraint and a high value was chosen for z. If one were to consider a more restricted exit criterion, one must consider the trade-off between expanding the neighbourhood and the resulting increase in the time required to fully utilise that said neighbourhood.

Descent Parameter: z=3000 (FI descent strategy).

N'hood	Build		c100	f71	tai75a
1-Opt	PNN	Min	1453.90	438.99	2428.73
		Ave	1690.21	552.80	2743.52
		Max	1850.98	669.47	3083.27
1-Opt	Rand	Min	1497.38	476.33	2468.02
		Ave	1678.00	581.63	3078.64
		Max	1811.06	694.70	3651.56
8-Move	PNN	Min	1412.89	431.10	**2083.52**
		Ave	**1471.26**	**501.49**	**2260.75**
		Max	**1562.32**	**624.75**	**2454.93**
8-Move	Rand	Min	**1291.12**	**363.19**	2215.65
		Ave	1504.48	507.18	2548.62
		Max	1748.02	710.19	2935.79

Table 5.1: Comparing the 1-Opt and 8-Move Neighbourhoods (in a Descent Algorithm)

The PNN build algorithm appears to represent a better starting point for the descent algorithm, which again is somewhat expected. The better the starting point the more likely one is to descend to a good final solution (this would be even more prevalent in the time-constrained variant of the problem).

With the descent algorithm clearly showcasing the greater level of success that the 8-Move neighbourhood had over the 1-Opt neighbourhood, consideration is now given to SA to see if the increase in performance is mimicked.

Here the results do not appear so clear-cut, with the lowest values achieved seemingly scattered throughout the table. However, one could argue that the 8-Move neighbourhood does appear to achieve lower results with a greater consistency, and can therefore be considered a success with regards to representing an improvement to the 1-Opt neighbourhood.

The construction algorithm appears to have little effect upon the resulting solution and given the particularly high starting temperature and low cooling rate, this behaviour could have been foreseen. Regardless of the initial solution (which is likely to be better in the PNN build than in the random build), one will accept many increasing moves in the initial phases of the SA algorithm. These are likely to move

the solution to a seemingly arbitrary position in the search-space, before the temperature is reduced to such a level that it begins to suitably restrict the moves.

SA Parameters: t_i=100, t_f=0.1, α=0.9 and l= 26060.

N'hood	Build		c100	f71	tai75a
1-Opt	PNN	Min	1041.06	316.10	2029.94
		Ave	1092.32	**320.80**	2227.34
		Max	1212.83	**331.55**	2487.11
1-Opt	Rand	Min	**1008.55**	317.90	2022.02
		Ave	1084.10	335.70	2292.71
		Max	1152.55	414.83	2793.47
8-Move	PNN	Min	1030.15	**304.28**	**1840.54**
		Ave	**1072.23**	326.79	2054.83
		Max	1148.39	386.14	2273.93
8-Move	Rand	Min	1026.98	304.69	1874.99
		Ave	1090.10	323.10	**1978.04**
		Max	**1131.19**	335.74	**2184.48**

Table 5.2: Comparing the 1-Opt and 8-Move Neighbourhoods (in an SA Algorithm)

In summary, the expansion of the neighbourhood (to include the six extra moves) has improved the performance in both the descent and SA algorithms. It seems sensible to assume that were an appropriate TC criterion identified, this improvement would also be apparent in TS. One could choose to revert back to TS now, but given the improvement, it seems sensible to consider whether a further neighbourhood expansion may be of benefit.

5.2.3 Generalising the Neighbourhood: The CROSS Neighbourhood

The neighbourhoods investigated thus far have relied on consecutive (when L_a or L_b>1) customer string exchanges, and have only considered the movement of two strings in a single move. Given the apparent success of the existing neighbourhoods, it seems sensible to continue with this type of neighbourhood, and consider whether there is a benefit from it being expanded further.

There is a popular VRPTW neighbourhood known as the CROSS Neighbourhood (introduced in Section 2.2.3), that one could view as a more generalised version of the 8-Move Neighbourhood. Although it does not include the restriction identified to encourage feasibility (with regards to vehicle routes not changing length by more than a single customer), if one were to achieve a local optimum, one would expect the performance to be of a higher standard. Note that one will have to consider the time that this neighbourhood requires to reach a local optimum and identify whether it is suitable for this work (i.e. could it be included in an algorithm that is to be run on the time-constrained variant of the problem?).

The CROSS Neighbourhood does not require one to predetermine values for L_a and L_b. It considers them to be variables with an upper bound of either the largest possible string that can be removed from that vehicle, or the user–controlled parameter L_{max}. A move consists of an exchange (between two vehicles) of two strings (of any suitable size). By allowing L_a and L_b to be variables, the neighbourhood can make moves of different string lengths in each iteration and is not restricted to a (relatively) small selection of moves.

Although this neighbourhood was developed for the VRPTW (as it preserves orientation, see discussion in Section 2.2.3) it has been used on the CVRP, and there is no reason why it should not perform to a high standard on the DVRP. Given the increased complexity of this neighbourhood it seems sensible to present some details regarding the implementation of this neighbourhood (specifically on this problem) before one presents the results and the accompanying discussion.

Implementing the CROSS Neighbourhood

The CROSS Neighbourhood was introduced to try to allow a more general move to be made; in contrast to the previously used rigid neighbourhoods (such as 1-Opt and 8-Move). This increased flexibility is the new neighbourhoods' strength, as one is significantly expanding the neighbourhood (i.e. there are many more potential moves).

Taillard et al. (1997) originally described the technique as a cross vehicle neighbourhood (which in itself offers a large number of potential moves), but within

the same paper it was proposed that it was expanded to consider intra vehicle moves. As this expansion will clearly enhance the technique, it seems sensible to include these moves in this implementation.

As with the 1-Opt and 8-Move Neighbourhoods, a (random) decision is made at each iteration as to whether one is to consider an intra or cross vehicle move. Although there are likely to be more improvements identified in the cross vehicles moves (as the neighbourhood contains many more of this type of move), a local optimum will still be found if z is set high enough.

When implementing any neighbourhood, certain complexities are likely to arise in the programming (note that they may appear trivial until one undertakes the task of programming them). In the previous neighbourhoods it was felt unnecessary to discuss these explicitly, as the rigidity of the neighbourhood meant such issues were minimised. However, the introduction of a variable based neighbourhood significantly increases the likelihood of encountering such problems, and thus it seems sensible to highlight a few implementation details.

Note that these details relate to the actual selection process i.e. how a programme decides which move to evaluate. Other difficulties will be encountered when one actually evaluates the move, but these are simpler to deal with due to the extra information (L_a, L_b etc.) being known. The intra and cross vehicle move selection processes are detailed in Figures 5.2 and 5.3 respectively.

V_a and V_b are vehicles, FL_a and FL_b the length of the fixed (committed) component of the routes, L_a and L_b are the length of the strings being exchanged (one of these can be 0), LV_a and LV_b are the total length of the vehicle routes and P_a and P_b are the placement of the first customer in the string being moved (if L_a=0 then one is considering the gap to the left of customer P_a, $L_b \neq 0$). Note that these flowcharts assume that no restriction has been placed on the maximum string length (i.e. L_{max} is set arbitrarily high).

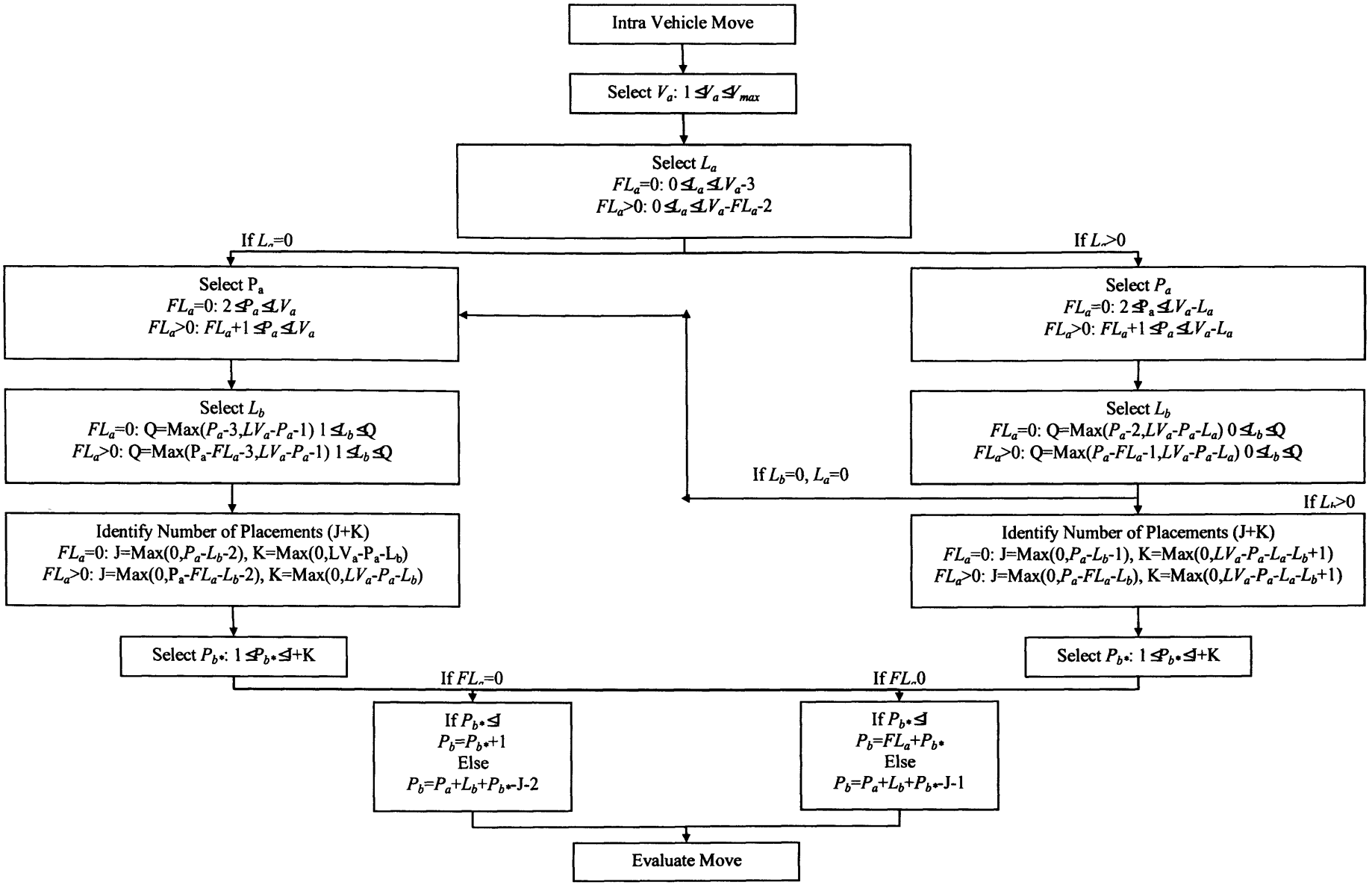

Figure 5.2: Identifying an Intra Vehicle Move in the CROSS Neighbourhood

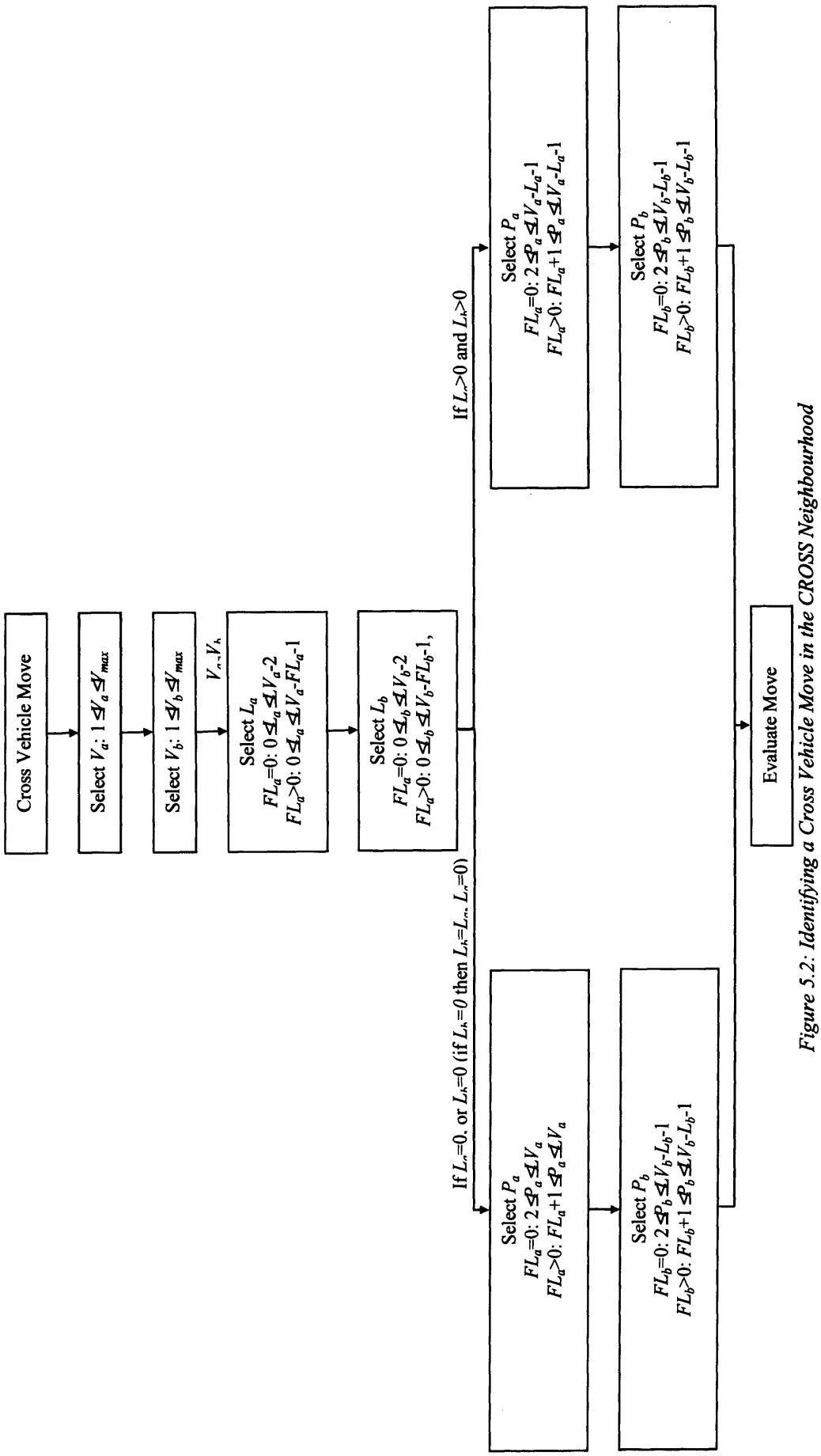

Figure 5.2: Identifying a Cross Vehicle Move in the CROSS Neighbourhood

These flow charts allow one to see how complex the neighbourhood becomes when L_a and L_b are variables, and demonstrates how one needs to account for the sections of the vehicle routes that have become fixed (committed to)[4]. It would have been possible to produce similar flow charts to detail the previous neighbourhoods, but when one is concerned with constants, the understanding and programming are greatly simplified.

CROSS Neighbourhood Results

In a similar approach to when the comparison was being made between the 1-Opt and 8-Move neighbourhoods, the impact that this neighbourhood has on performance will be assessed in both descent and SA implementations (using the same parameter settings). The use of two different construction algorithms has also been retained (thus allowing one to compare the results directly).

The descent results are presented in Table 5.3 with the analysis of these results presented directly afterwards. Following this analysis, the SA results are presented (Table 5.4), and are again accompanied by the relevant analysis.

N'hood	Build		**c100**	**f71**	**tai75a**
8-Move	PNN	Min	1412.89	431.10	2083.52
		Ave	1471.26	501.49	2260.75
		Max	1562.32	624.75	2454.93
8-Move	Rand	Min	1291.12	363.19	2215.65
		Ave	1504.48	507.18	2548.62
		Max	1748.02	710.19	2935.79
Cross	PNN	Min	1065.20	314.10	2020.32
		Ave	**1109.43**	332.59	**2074.94**
		Max	**1177.98**	**342.57**	**2214.85**
Cross	Rand	Min	**1062.22**	**308.65**	**1863.68**
		Ave	1119.84	**329.14**	2104.59
		Max	1204.40	345.73	2260.35

Table 5.3: Comparing the 8-Move and CROSS Neighbourhoods (in a Descent Algorithm)

[4] These also needed to be accounted for in the previous neighbourhoods.

This further expansion of the neighbourhood appears to have significantly improved the quality of results achieved on each of the three test datasets. The quality of result that has been achieved (although not close to those produced by the ERWACS-DVRP) are of a standard high enough to suggest that when implemented in a technique more advanced than descent, some very good solutions could be produced.

In contrast to analysis of the 1-Opt and 8-Move neighbourhoods, it appears that the PNN construction algorithm out-performs the random construction algorithm (though this difference is not sizable). A possible explanation for this behaviour is that the quality of the initial solution is simply less important, given the size of the expanded neighbourhood. When the neighbourhood is small (i.e. 1-Opt) one would expect there to be a strong correlation between the quality of the initial solution and the final solution (due to the somewhat restricted movement within the search-space). However, this correlation is likely to diminish as more dramatic moves in the search-space are allowed (e.g. the CROSS Neighbourhood).

With the performance of the CROSS Neighbourhood seemingly of a very high standard, it is hoped that its inclusion in SA may produce results that may rival those produced by the ACS-DVRP (and even the ERWACS-DVRP). These results are presented in Table 5.4.

N'hood	Build		**c100**	**f71**	**tai75a**
8-Move	PNN	Min	1030.15	304.28	1840.54
		Ave	1072.23	326.79	2054.83
		Max	1148.39	386.14	2273.93
8-Move	Rand	Min	1026.98	304.69	1874.99
		Ave	1090.10	323.10	1978.04
		Max	1131.19	335.74	2184.48
Cross	PNN	Min	1015.26	312.81	1870.52
		Ave	1044.41	321.24	1991.74
		Max	1071.36	337.31	2311.15
Cross	Rand	Min	**996.17**	**302.16**	**1791.20**
		Ave	**1037.87**	**319.12**	**1936.29**
		Max	**1067.54**	**333.05**	**2074.07**

Table 5.4: Comparing the 8-Move and CROSS Neighbourhoods (in an SA Algorithm)

As with the descent algorithm, the inclusion of the expanded neighbourhood has significantly improved the quality of the results achieved. Comparing the performance with that of the ACS-DVRP (note that these SA results are non time-constrained) one can see increases in the average distance of just 3.85%, 1.64% and 2.79% (for c100, f71 and tai75a respectively). Furthermore, it betters the minimum solution produced throughout the ten runs, for two of the three test datasets. Unfortunately there is an increase in all three of the maximum solutions produced, highlighting the undesirable variability in result quality that this problem seems prone to.

As to which build algorithm performs best, looking at the results one would obviously state a preference for the random construction. However, due to the reasons identified earlier (high start temperature, low cooling rate etc.) this discrepancy could purely be due to randomness. One could investigate this further, but given that TS (not SA) is the primary focus of this chapter, such analysis is somewhat unnecessary.

Of interest is whether there are any details that can be taken from these analyses to help with the development of a successful TS algorithm. To make any such conclusions, some further details that will allow a greater understanding of what is happening when this well-performing neighbourhood is implemented are presented.

Move Selection Details (Descent Algorithm)

Given that the CROSS Neighbourhood is far larger than any of the previous neighbourhoods that have been considered, it seems sensible to try to understand which moves are being accepted (and with what frequency). This will allow one to assess the neighbourhood's suitability to this problem and whether one ought to consider a change in the descent strategy (currently FI).

Using the ideology that ts_0 represents one of the more difficult timeslots to solve, ten runs (across the three test datasets) were conducted, and some details about the success of the search were noted (attempted move percentage and absolute acceptance and average change in distance). Given that the neighbourhood was programmed in four subroutines (i:L,L, i:L,0, x:L,L and x:L,0), it seems sensible to produce details relating to the success of each of these.

The results presented (see Table 5.5) are averaged across the ten runs.

Move	Attempt	Accepted	Average δ
i:L,L	45.81%	73.9	-7.81
i:L,0	4.19%	18	-6.46
x:L,L	42.89%	41.8	-9.24
x:L,0	7.11%	11.6	-8.61

Move	Attempt	Accepted	Average δ
i:L,L	46.29%	39.8	-3.1
i:L,0	3.71%	9.2	-2.48
x:L,L	43.58%	19.1	-4.77
x:L,0	6.42%	4.9	-5.98

Move	Attempt	Accepted	Average δ
i:L,L	42.95%	52.3	-10.89
i:L,0	7.04%	14.2	-8.75
x:L,L	35.91%	31.9	-11.39
x:L,0	14.10%	13.2	-8.27

Table 5.5: Detailing the Successful Moves from the ts_0 of CROSS Neighbourhood in a Descent Algorithm with the Random Construction (c100, f71 and tai75a)

There is an obvious pattern across the three datasets, whereby the i:L,L and x:L,L moves are being attempted much more frequently than i:L,0 and x:L,0 respectively (note that given the 50/50 split between the intra and cross vehicle moves, one should not analyse these figures with each other). As the selection of L_a is made randomly between 0 and the maximum string length possible, this suggests that the neighbourhood is operating as expected.

The number of moves that are actually accepted is surprisingly low, given that one is beginning from a randomly constructed solution. However, given the use of an exit criterion (z=3000), which terminates when one has attempted z moves without improvement, one is likely to have reached (or got close to) a local optimum. The spread between the four moves seems similar for each of the datasets, though it is clear that fewer moves are accepted on the clustered dataset (f71). This behaviour

would appear to relate to the ease at which the neighbourhood has at reaching a high quality solution, rather than the neighbourhood's inability to deal with this dataset.

The average change in distance (for each of the four neighbourhood components) has been presented for completeness, and suggests that greater reductions are found (per move) by the moves that exchange two strings of customers as opposed to string relocation (i.e. L_a=0). As there are only a small number of moves being accepted (average=110), a change in the descent strategy may be beneficial.

Best Improvement

The problem of accepting so few moves that improve the distance is symptomatic of any large neighbourhood definition for this problem, and as such, cannot truly ever be suppressed. However, as one is actually accepting so few moves, it seems worth considering whether one should adopt the BI descent strategy (see Section 3.3.3). Note that as one is accepting the best move at each iteration, one would expect even fewer moves to be accepted (in order to reach a local optimum).

This analysis also allows one to identify the suitability of the CROSS Neighbourhood for inclusion in a TS algorithm (as TS makes use of the BI descent strategy). Were the time required to evaluate the entire neighbourhood be so sizable that one would not be able to make a sufficient number of moves (noting that TS requires a local optimum to be identified and then allows the acceptance of moves than increase the distance) one would have to consider reverting back to a smaller neighbourhood (e.g. the 8-Move Neighbourhood).

As explained in Section 3.3.3, the BI strategy is run until there are no moves in the defined neighbourhood that will further reduce the distance (i.e. one is sure that a local optimum has been reached). Results are presented in Table 5.6.

The first observation about these results is that there is little difference between the two descent policies. One would not immediately favour either, though this in itself is not an entirely undesirable conclusion. As the BI descent strategy ensures the algorithm is run until a local optimum is achieved, by the FI strategy performing to a

similar level, one is able to assume that z was set high enough for a local optimum to be achieved.

Strategy		c100	f71	tai75a
BI	Min	**1038.14**	**302.01**	1901.16
	Ave	**1103.98**	331.56	**2022.21**
	Max	**1166.38**	351.36	2291.92
FI	Min	1062.22	308.65	**1863.68**
	Ave	1119.84	**329.14**	2104.59
	Max	1204.40	**345.73**	**2260.35**

Table 5.6: Comparison of the FI and BI Descent Strategies

There was very little difference in the run-times for the two algorithms (the BI strategy appears perfectly suitable to be used in a TS algorithm), and as such, one would be likely to favour the BI strategy ahead of FI (given the certainty of a local optimum being achieved). Before one reverts back to considering TS and identifying a TC that can be used in conjunction with the CROSS Neighbourhood, some observations about the neighbourhoods limitations are presented.

CROSS Neighbourhood Observations

It is important to remember that no neighbourhood (that can be run in real time) is able to investigate every possible combination of customer placement (thus ensuring one identifies a global optimum). Neighbourhoods by their very definition need to be restrictive, and as such, will always be flawed. This in itself need not be a problem, as heuristics/metaheuristics are approximation algorithms, designed to achieve good solutions in an acceptable time period.

Although the CROSS Neighbourhood is both complex and time-consuming, there will still exist many situations (particular solutions) where the neighbourhood is unable to further improve the solution quality. Figure 5.4 presents solutions that were returned at the end of ts_{17} and ts_{18} in the c100 dataset. Although the number of customers has not been altered[5] (this is beyond the halfway stage, so there are no further arrivals),

[5] Technically it will have reduced but one will have only committed to parts of the solution from ts_{17}.

the solution quality has decreased. The differences between the two solutions are shown in black

ts_{17}	ts_{18}
0-94-60-93-98-85-91-14-15-100-38-43-42-0	0-94-60-93-98-85-91-14-15-100-38-43-42-0
0-76-77-79-78-81-9-71-65-29-34-35-51-33-3-0	0-76-77-79-78-81-9-71-65-29-34-35-51-33-50-0
0-88-62-82-7-10-90-63-66-20-32-64-11-19-31-0	0-88-62-82-7-10-90-63-66-20-32-64-11-19-31-0
0-58-57-87-97-95-13-6-89-36-49-47-46-83-0	0-58-57-87-97-95-13-6-89-36-49-47-46-83-0
0-53-73-72-4-55-67-25-54-24-80-68-12-26-0	0-53-73-72-4-55-67-25-54-24-80-68-12-3-0
0-2-21-56-23-75-39-41-22-74-40-0	0-2-21-56-23-75-39-41-22-74-40-26-0
0-18-5-61-16-17-45-8-48-52-0	0-18-5-61-16-17-45-8-48-52-0
0-96-99-59-92-0	0-96-99-59-92-37-44-86-0
0-27-69-70-30-1-50-28-0	0-27-69-70-30-1-28-0
0-84-86-44-37-0	0-84-0
Distance = 1211.60	Distance = 1231.26

Figure 5.4: Example of Solutions from Successive Timeslots

Consider the solution shown in ts_{18}; its distance has increased above that which was identified in ts_{17}. As there have been no new customers arrivals, one would hope that solution quality would not have deteriorated. It might have been expected that the CROSS Neighbourhood would have successfully identified the necessary moves to reduce the solutions distance to the level shown in ts_{17}, but the neighbourhood's limitations soon become clear.

There are only ever changes being made to a maximum of two vehicles, yet these solutions contain differences in six vehicles. In order to descend to the solution identified in ts_{17}, a number of moves would be required. However, unless each of the interim stages (i.e. each individual move) were both feasible and reduced the distance, the current neighbourhood would not successfully traverse the search space from the point in ts_{18} to that in ts_{17}.

As stated previously, such a restriction, although prohibitive, is necessary for any neighbourhood definition. The CROSS Neighbourhood appears to be capable of

producing high quality solutions, thus no attempts are made to expand the neighbourhood further.

This analysis has also identified the need for some form of feedback mechanism[6] to prevent what is essentially the acceptance of a solution worse than one that has already been identified. To implement this, one simply incorporates a check to see if the customers in the current timeslot are the same as the previous (i.e. no new customers have arrived). If this condition is met, at the end of the timeslot, the solution identified will be compared with that found in the previous timeslot and the best selected. Note that this will not be included in any of the results until the end of the analysis to ensure that the results are directly comparable.

5.3 Reintroducing Tabu Search: Generalising the Tabu Criterion

The primary focus of this chapter is to produce a TS algorithm that significantly outperforms the simple TS algorithm developed in Chapter Three. With a suitably complex (and clearly improved) neighbourhood identified, one needs to consider identifying a suitable TC with which this neighbourhood is able to operate.

As stated earlier in this chapter, the attribute storage system of Chapter Three is not suited to a more complex neighbourhood, and as such, a new TC must be introduced. The complexity of this task is that it must be able to handle the variable string lengths that one is exchanging in the CROSS Neighbourhood. Two new TC are considered:

- A Cost Based System
  Uses the cost of previous solutions as a way of identifying if a potential solution (i.e. one in the current neighbourhood) has been located previously.

- Customer Pair System
  Uses arcs (or customer pairs) that have belonged to previous solutions as a way of identifying if a potential solution may have been visited previously.

[6] This was not included in the ACO analysis, due to the high number of solutions produced and the extremely powerful global update rule (making repetition extremely likely).

A primitive aspiration criterion is also included in both of these implementations (and remains in all subsequent TS implementations). If at any point a move is analysed that results in a distance lower than any previously identified, one selects that move, regardless of its tabu status.

5.3.1 Generalised Type 1: Cost Based Tabu Criterion

When trying to identify whether a potential next solution has been visited previously, one has to rely on some way of distinguishing solutions from each other. Instead of relying on the customer's placements (as seen in Chapter Three), this system is going to use the cost (i.e. distance) of a solution. This technique is similar to that of Taillard et al. (1997), though one places the actual cost on the TL instead of the cost modulus T (where T is a user controlled parameter set to some arbitrarily high value).

Given that the cost of a particular solution will be appended to the TL, any subsequent move that results in a solution of equal cost will be excluded from the neighbourhood. Although it is unlikely that there will be a number of solutions with exactly the same cost (due to the complexities of the datasets), one should not assume the system to be infallible. Preliminary tests supported this stance i.e. showing that it is an infrequent occurrence (though an example of this will be discussed later in this subsection).

Given that this strategy is intended to prevent the repetition of solutions one need not consider the parameter LS (as one can include every previously visited solution i.e. $LS=l$). This should prevent any previously visited solution from being visited, which is important, as the local search component of the TS algorithm does not contain any form of randomness. Note that for very large values of l, this parameter setting may be too time consuming but this should not be a problem in this implementation.

Results

With the parameter LS being set equal to the value selected for l, one needs to identify a value such that the algorithm is given suitable time to explore the search-space, but does not require excessive computational effort. From the preliminary runs, it seemed that one could achieve good solutions with a value of l=1000. These results are supplemented with a larger value of l=5000 to establish whether the search could benefit from being run for a longer period of time.

It should be noted that these values are considerably less than the l=45000 utilised in the TS algorithm in Chapter Three. Whilst the considerable increase in the neighbourhood size is likely to identify more (and better) moves, given the BI nature of the TS algorithm, this will come at the cost of a significant increase in the time required to identify each and every move. The results for this analysis are presented in Table 5.7.

l	*LS*		**c100**	**f71**	**tai75a**
1000	1000	Min	**991.76**	302.53	**1801.52**
		Ave	1054.85	328.68	**1958.17**
		Max	1107.38	344.49	**2132.34**
5000	5000	Min	1006.73	**293.69**	1843.95
		Ave	**1039.11**	**321.1**	1999.68
		Max	**1093.48**	**335.08**	2169.64

Table 5.7: Analysing Various Values of l (and LS) in a Cost Based TS Algorithm

Before comparing these results with others that have been achieved, one ought to consider whether the extra iterations (in l=5000) result in an improvement to solution quality. It is certainly debatable, but given the improvements in two of the three test datasets, and the notion that by definition l=5000 has to perform at least as well, it seems sensible to select the longer running parameter setting. This decision is made easier by the lack of any time constraint in this analysis. Note that given the similarity between these results there is little evidence to suggest that any further values (i.e. l>5000) need be assessed.

The two best reference points with which to compare these results, are the TS implementation in Chapter Three (see Table 3.12) and the CROSS Neighbourhood implementation of the SA seen previously in this chapter (Table 5.4).

Firstly, the Cost Based TC (in conjunction with the CROSS Neighbourhood) significantly improves upon the performance of the previous best TS implementation (in this thesis). The average distance has decreased by 9.18% (when averaged across the three test datasets), thus there is obviously a benefit associated with this TC and neighbourhood.

However, the performance is not at the level that was achieved via the SA algorithm, with TS increasing the average distance by 1.34%. Given that the same neighbourhood is being used, one could argue the case for selecting SA instead of this TS algorithm (if these were the only viable options). However, before this decision is made, another TC is to be assessed (this will follow a short investigation into the possible problems associated with the Cost Based TC).

Possible Problems with the Cost Based TC

Although the result quality was clearly an improvement to the previous TS implementations, its inability to match earlier SA workings contradicts a common perception. Many authors believe that TS is the best performing of the local search metaheuristics for routing problems, with Gendreau et al. (2002) stating '*applying SA to the VRP does not yield results that are competitive with those produced by the best tabu search implementations*'. Two potential reasons for this anomaly (both related to the selection of the Cost Based TC) are now discussed.

Firstly, it is possible that the Cost Based TC is using the wrong principal, with regards to why it is declaring moves as tabu. At first this seems a rather unusual idea, as theoretically one assumes that the aim of the TL is to prevent the repetition of a move, and any such definition for a TC that does this will succeed. However, this is not always the case, with many TS algorithms benefiting from excluding other areas of the search space due to the lack of precision in the TC (i.e. identifying previously visited solutions).

Glover (1989) believed in this idea, stating *'experiments have shown that tabu lists thus designed to prevent repetition rather than reversal of moves typically do not work well'*. Assuming that this principal is partially responsible for the poor performance (relative to the SA implementation), it will be interesting to see how the Customer Pair TC performs.

Secondly, this technique relies on a solutions distance to identify a previously solution and unfortunately this will not always be unique. There are certain customers (especially in the clustered datasets) that have quite a regular pattern and could lead to multiple solutions existing with the same cost. This can lead to the exclusion of

solutions that have not actually ever been visited, and more importantly, may prevent one from accepting a solution that could be further improved in subsequent iterations.

An example of this behaviour can be seen in Figure 5.5, where two partial solutions to ts_0 for the c100 dataset are presented. These solutions have equal cost, yet differ with regards to not only the order in which the customers are visited, but also how they are distributed between the two vehicles.

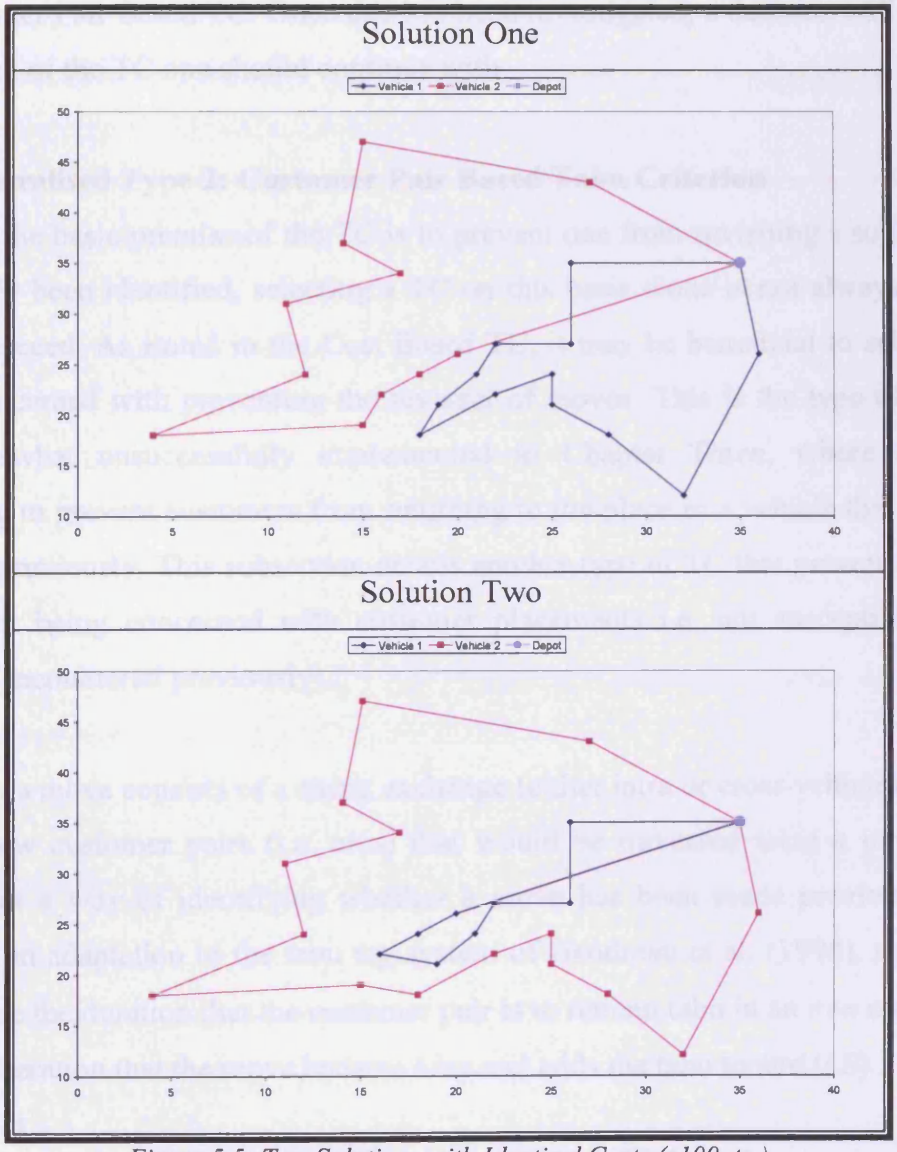

Figure 5.5: Two Solutions with Identical Costs (c100, ts_0)

If Solution One were found, one would add its cost to the TL and prevent any move that resulted in a solution with the same cost being produced. Unfortunately this

would prevent Solution Two from ever being identified, which would mean one would not have the chance to see if this solution could be bettered.
This example showcases the problem with this TC, as Solution Two contains a self-crossing route (which by definition cannot be optimal). If Solution Two were found, the next iteration would identify a move that would successfully reduce the cost; unfortunately this eventuality is never allowed to come to fruition.

This concludes the discussion of the Cost Based TC and consideration is now given to the Customer Pair Based TC. Once this has been investigated, a decision will be made as to which of the TC one should continue with.

5.3.2 Generalised Type 2: Customer Pair Based Tabu Criterion

Although the basic premise of the TC is to prevent one from revisiting a solution that has already been identified, selecting a TC on this basis alone is not always the best way to proceed. As stated in the Cost Based TC, it may be beneficial to select a TC that is concerned with preventing the reversal of moves. This is the type of TC that was somewhat unsuccessfully implemented in Chapter Three, where one was attempting to prevent customers from returning to the place in a vehicle that they had occupied previously. This subsection details another type of TC that prevents reversal (whilst not being concerned with customer placements i.e. not susceptible to the problems encountered previously).

Given that a move consists of a string exchange (either intra or cross vehicle), one can use the new customer pairs (i.e. arcs) that would be traversed were a move to be selected, as a way of identifying whether a move has been made previously. This system is an adaptation to the tabu tag system of Gendreau et al. (1994), and allows one to store the duration that the customer pair is to remain tabu in an $n \times n$ matrix; one notes the iteration that the move became tabu and adds the tabu tenure (LS).

When a neighbourhood move is considered, one simply checks whether the matrix entry is greater than the current iteration for any of the new customer arcs that would be traversed were the move accepted. If this is the case, one would deem the move tabu, otherwise it would be assessed according to the BI descent strategy.

As explained previously, one is able to break the CROSS Neighbourhood down into four generalised moves: i:L,0, i:L,L, x:L,0, and x:L,L. Figure 5.6 gives examples (for each of these) on how the Customer Pair TC checks whether a move is tabu and what information is stored i.e. updates to the $n \times n$ matrix to prevent repetition.

It is clear to see that the entries in the matrix will successfully prevent moves being made that will return a solution to one that has previously been identified (for *LS* iterations). However, the Customer Pair System is going to restrict the movement in the search-space considerably, not only preventing repetition, but also excluding areas that have not been visited. This is in contrast with the Cost Based System that primarily concerned itself with the prevention of repetition.

It seems sensible to match the initial value of l to the value that was first used in the Cost Based System (l=1000), thus allowing a direct comparison to be made between the previous results and those achieved here. The choice of *LS* is not as straightforward, as the restrictive nature of the TC requires moves to be removed from TL after a period of time i.e. allowing moves to be made back into those areas previously declared tabu. This idea is much more in keeping with the majority of TS algorithms and will require optimising in the usual manner.

Once a suitable value for *LS* has been identified, consideration will be given to whether the algorithm could benefit from further iterations (i.e. a higher value of l).

Results

The choice of *LS* can have a dramatic impact on the performance levels, and as such, it is important that consideration is given to a number of possible values. Given that there is no similar implementation to use to provide an estimate for this value, an initial analysis was required to identify a sensible range of values. This suggested that an initial range on which to base this analysis was 10-30.

i:L,0	x:L,0
0-[1-2]-3-4-5-[0]	0-[1-2-3]-4-5-0 0-6-7-8-[9]-0
Check Tabu Status 0-3 5-1 and 2-0	Check Tabu Status 0-4 8-1 and 3-9
If ok then new solution looks like 0-3-4-5-1-2-0	If ok then new solution looks like 0-4-5-0 0-6-7-8-1-2-3-0
Make Tabu 0-3 5-1 and 2-0	Make Tabu 0-4 8-1 and 3-9
i:L,L	**x:L,L**
0-[1-2]-3-[4-5]-0	0-[1-2-3]-4-5-0 0-6-7-[8-9]-0
Check Tabu Status 0-4 and 5-3 3-1 and 2-0	Check Tabu Status 0-8 and 9-4 7-1 and 3-0
If ok then new solution looks like 0-4-5-3-1-2-0	If ok then new solution looks like 0-8-9-4-5-0 0-6-7-1-2-3-0
Make Tabu 0-4 and 5-3 3-1 and 2-0	Make Tabu 0-8 and 9-4 7-1 and 3-0

Figure 5.6: How the Customer Pair Tabu Strategy Works With the Different Types of Move in the CROSS Neighbourhood

The results of this analysis can be seen in Table 5.8 where five different values of *LS* are assessed.

LS		c100	f71	tai75a
10	Min	1024.98	312.58	1842.81
	Ave	1059.99	323.15	1981.80
	Max	1101.06	**337.36**	2110.89
15	Min	1023.77	303.75	1854.10
	Ave	1056.93	326.68	1963.09
	Max	1080.17	346.83	2121.30
20	Min	**988.59**	**300.58**	1828.67
	Ave	1044.64	323.47	1968.47
	Max	1099.11	343.59	2101.17
25	Min	999.42	310.77	**1782.15**
	Ave	**1026.80**	**321.63**	**1919.97**
	Max	1060.45	340.55	**2065.96**
30	Min	995.05	305.94	1849.35
	Ave	1030.74	323.51	1974.19
	Max	**1057.40**	355.93	2126.55

Table 5.8: Analysing Various Values of LS in a Customer Pair TS Algorithm

Unlike many of the results produced so far, this table seems to offer a conclusive result (i.e. one without the need for compromise). It is clear *LS*=25 results in solutions of a superior quality and therefore should be selected. It produced the lowest average for each of the three test datasets whilst also remaining competitive among the other metrics.

In a similar fashion to the Cost Based System, it is important to consider if the Customer Pair System could benefit from further iterations. It seems sensible to increase *l* to the same value as the previous analysis (*l*=5000), to allow another direct comparison (results are presented in Table 5.9). Note that the algorithms will have similar run times given the matching values for the parameter *l*, though there will be some variability due to the different TC (though time is not a prominent concern as these results are being produced for the non time-constrained variant of the problem).

l		c100	f71	tai75a
1000	Min	999.42	310.77	**1782.15**
	Ave	1026.80	**321.63**	**1919.97**
	Max	1060.45	**340.55**	2065.96
5000	Min	**991.10**	**301.53**	1796.69
	Ave	**1013.06**	324.92	1924.84
	Max	**1050.55**	344.72	**2050.19**

Table 5.9:Analysing Various Values of l in the Customer Pair TS Algorithm

These results give little indication that quality will improve through increasing the number of iterations, therefore one can continue to use l=1000. To remain consistent a similar analysis to that which was performed for the Cost Based System (i.e. comparing with the Chapter Three TS algorithm and the SA implementation from this chapter) is now made.

The Cost Based System decreased the average distance (across the three datasets) by 9.18% whereas the Customer Pair System decreased it by 10.62%. With regards to the comparison with the SA implementation, the Cost Based system was unable to match (or better) the results, with an increase of 1.34%. In contrast, the Customer Pair System actually improved on the SA implementation, decreasing the average by 0.37%. These results not only further showcase the benefit of the CROSS Neighbourhood and revised TC; it confirms that the TS algorithm can be successfully implemented on this problem, and that the implementation in Chapter Three (which did not outperform SA) was indeed flawed.

To help with one's understanding of the movement in the search-space, Figure 5.7 shows the change in distance throughout the 1000 iterations (for each of the three datasets in timeslot ts_0). Two graphs are presented for each of these. The first includes the initial solution and gives a good indication just how quickly competitive solutions are identified, while the second focuses more clearly on the movement once the search has settled down. This second graph is a good indicator of the low level of diversification that is being achieved.

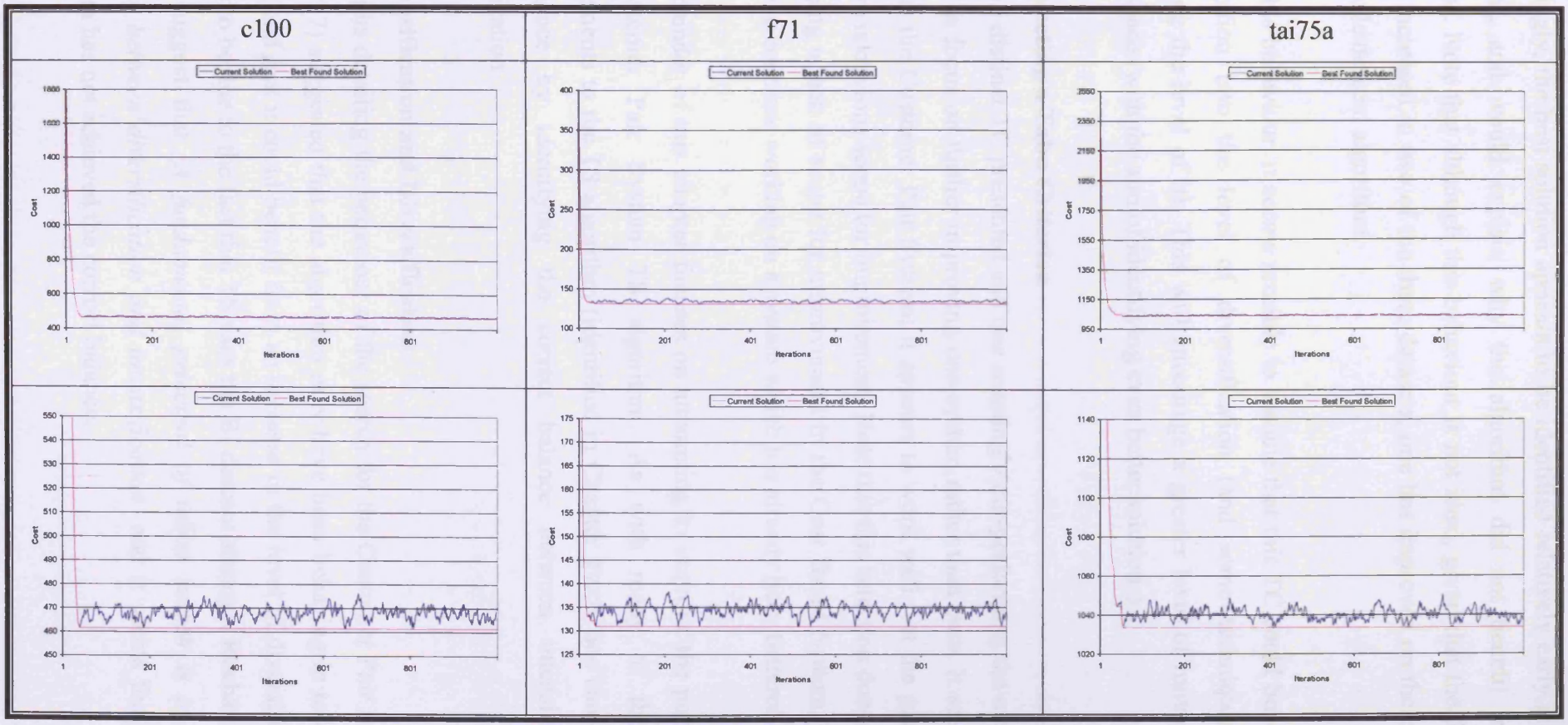

Figure 5.7: Tabu Search Behaviour with the Customer Pair System TC (ts_0)

Interestingly, the best solution appears to be identified relatively early into the 1000 iterations, and would explain why the algorithm did not benefit from further iterations. Note that although this behaviour is not ideal, given that the distance had already increased in two of the three datasets, one has improved on the performance of a simple descent algorithm.

Given this behaviour, it seems sensible to assume that this TC could benefit from an investigation into the level of diversification (and some techniques aimed at increasing the level of it). This will encourage a greater level of movement in the search-space (with the aim of identifying even better solutions).

5.3.3 Selecting a Tabu Criterion

With two distinct TC presented and one seemingly out-performing the other, it seems prudent to focus on further improving one system rather than both. It seems sensible to pursue the Customer Pair System; it appears to work well, yet the graphs suggest that there is obvious scope for improvement. Note that this selection does not relate to there being a lack of scope for improvement in the Cost Based System, but it is not sensible to continue working on a system which has already been bettered.

The remainder of this chapter focuses on attempting to improve the performance of the Customer Pair System TS algorithm. As with many of the suggested improvements to the TS algorithm (identified in Chapter Two), one can improve the performance by identifying the correct balance between intensification and diversification

5.4 Intensification and Diversification

The graphs detailing the behaviour of the search for the Customer Pair Based TC (in Figure 5.7) suggested that the algorithm may have been behaving in too intensive a fashion, and that it could benefit from an increase in the level of diversification. This is likely to be due to the fact that TS uses the BI descent strategy. Rochat and Taillard (1995) suggest that *'A fundamental principal of taboo search is to exploit the interplay between diversification and intensification'* and it seems that the current algorithm has not achieved the correct balance.

In Chapter Four (with regards to the ACO), it was discussed how the balance between intensification and diversification is often a difficult one to achieve and it seems reasonable to assume the same will be true for TS. The remainder of this chapter will focus on the inclusion of various intensification and diversification strategies into the Customer Pair System TS algorithm. Upon completion of this analysis a full set of results will be produced for both the standard and time-constrained variants of the problem.

5.4.1 Tabu Tenure

Choosing an appropriate value for the tabu tenure depends significantly on the context within which the TS algorithm is being used. Different types and sizes of problem will require information to be stored for varying amounts of time. Currently *LS* (the parameter denoting the tabu tenure) is set at 25, a user chosen parameter identified via a parameter optimisation. This constant has been shown to perform competitively and therefore represents a sensible choice.

However, given the apparent disparity in the intensification/diversification balance, it seems sensible to consider how varying the tabu tenure can alter this. Moreover, there is no reason why the tabu tenure need be a constant; this section considers the idea of dynamic tabu tenure (whereby it varies in value throughout the duration of the search).

Dynamic Tenure

The use of the term dynamic refers specifically to the proposed variability in the tabu tenure and has no relation to the fact one is solving a dynamic problem. To understand the principal behind dynamic tenure, one must consider the impact on the search if the tenure is either increased or decreased:

- Tenure Increase

  The search will diversify, as it will increase the period of time that a solution must move away from a previously visited area of the search-space before it can return.

- Tenure Decrease

  The search will intensify, as it will decrease the period of time that a solution must move away from a previously visited area of the search-space before it can return.

Given that a successful search will require elements of each i.e. periods of intensification and diversification, it seems sensible to have some mechanism in place to vary the tenure accordingly.

Glover and Laguna (1997) discuss two types of dynamic tabu tenure: those that recalculate the tenure for each and every move, and those that adjust the tenure periodically (i.e. after a user defined number of iterations). The idea of recalculating the tabu tenure after each and every move obviously offers greater flexibility and seems to be a more interesting area in which to focus. Two basic methods of dynamic tenure are now considered:

The first considered is sensibly referred to as Random Dynamic Tenure by Glover and Laguna (1997) and is the most basic form one could consider. The tenure (for that particular move) is calculated by randomly selecting from a range of two user defined parameters, known as $LS_{\min}$ and $LS_{\max}$. The more extreme the values chosen for these parameters, the greater the chance of the search being intensified or diversified. This method is similar to Robust Tabu Search, see Taillard (1991).

For this implementation, it seems sensible to not invoke particularly high levels of intensification or diversification in a purely arbitrary fashion; therefore the parameters have been set at $LS_{\min} = 20$ and $LS_{\max} = 30$ (centering on the previously optimised LS=25).

The second method is an amendment of the first. It uses a similar random principal when determining the tenure for a particular move and was developed in response to the apparent lack of diversification evident in Figure 5.7. The uniform distribution is again used (though limited for tenures between 20-24), and this is followed by a geometric distribution capped at 100. This distribution still heavily favours the 20-30

range, but does allow for longer tenures to be chosen. Obviously were this method to offer some level of improvement; other possible distributions would require consideration.

The decision process for the way in which the tabu tenure is altered for each of the two techniques are summarised below:

- Uniform

  Entry in $n \times n$ matrix= $Iteration + DU(LS_{\min}, LS_{\max})$

  (Where DU represents the discrete uniform distribution, Figure 5.8)

- Uniform/Geometric

  Entry in $n \times n$ matrix = $Iteration + x_1$

  (Where x_1 is sampled from the uniform/geometric distribution, Figure 5.8)

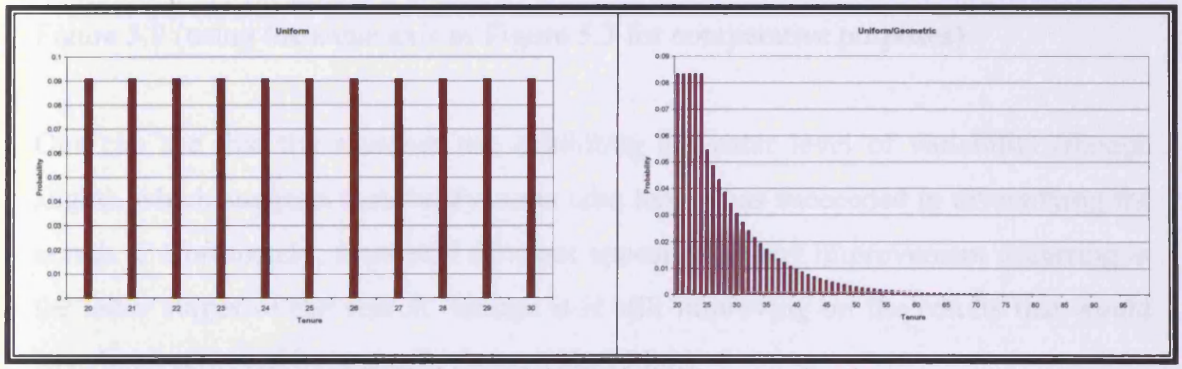

Figure 5.8: Uniform and the Uniform/Geometric distributions

The results achieved by the incorporation of these two dynamic tabu tenures can be seen in Table 5.10 (along with those using the previous constant LS=25). Note that no alteration has been made to the number of iterations (i.e. I=1000).

There does not appear to be any significant case with which to argue for the inclusion of dynamic tabu tenures of this random-like nature. The uniform implementation appears particularly unsuccessful with solution quality deteriorating across the three test datasets. The uniform/geometric implementation appears to have had less of a

negative impact (when averaged across the three test datasets), but one would still argue the merits of using a constant tenure.

Dist'n		c100	f71	tai75a
Constant	Min	999.42	310.77	**1782.15**
	Ave	**1026.80**	321.63	**1919.97**
	Max	**1060.45**	340.55	**2065.96**
Uniform	Min	994.65	302.76	1842.27
	Ave	1055.28	322.15	1925.78
	Max	1109.91	345.46	2147.60
Uniform/ Geometric	Min	**985.78**	**297.88**	1851.86
	Ave	1028.78	**316.19**	1998.96
	Max	1073.18	**338.69**	2149.17

Table 5.10: Assessing the Impact of Two Different Types of Dynamic Tabu Tenure

To see if one has successfully invoked a higher level of diversification, graphs depicting the behaviour in ts_0 for the uniform/geometric distribution are presented in Figure 5.9 (using the same axis as Figure 5.7 for comparative purposes).

One can see that the searches are exhibiting a greater level of variability (though slight), which suggests that the dynamic tabu tenure has succeeded in diversifying the search. Unfortunately, there still does not appear to be any improvement occurring in the latter stages of the search (though it is still improving on the results that would have been achieved by a simple descent algorithm).

As stated previously, results wise, using the uniform/geometric implementation does not represent an improvement to the use of a constant; though that is not to say the technique is without merit. It has shown that tenures can be used as an easily adaptable diversification measure, though for this problem they are seemingly not an acceptable one.

With the idea of random tabu tenures (those that do not give consideration to the success/failure of a move prior to assigning a tenure) showing little sign of improving the solution quality, a more intelligent system is investigated.

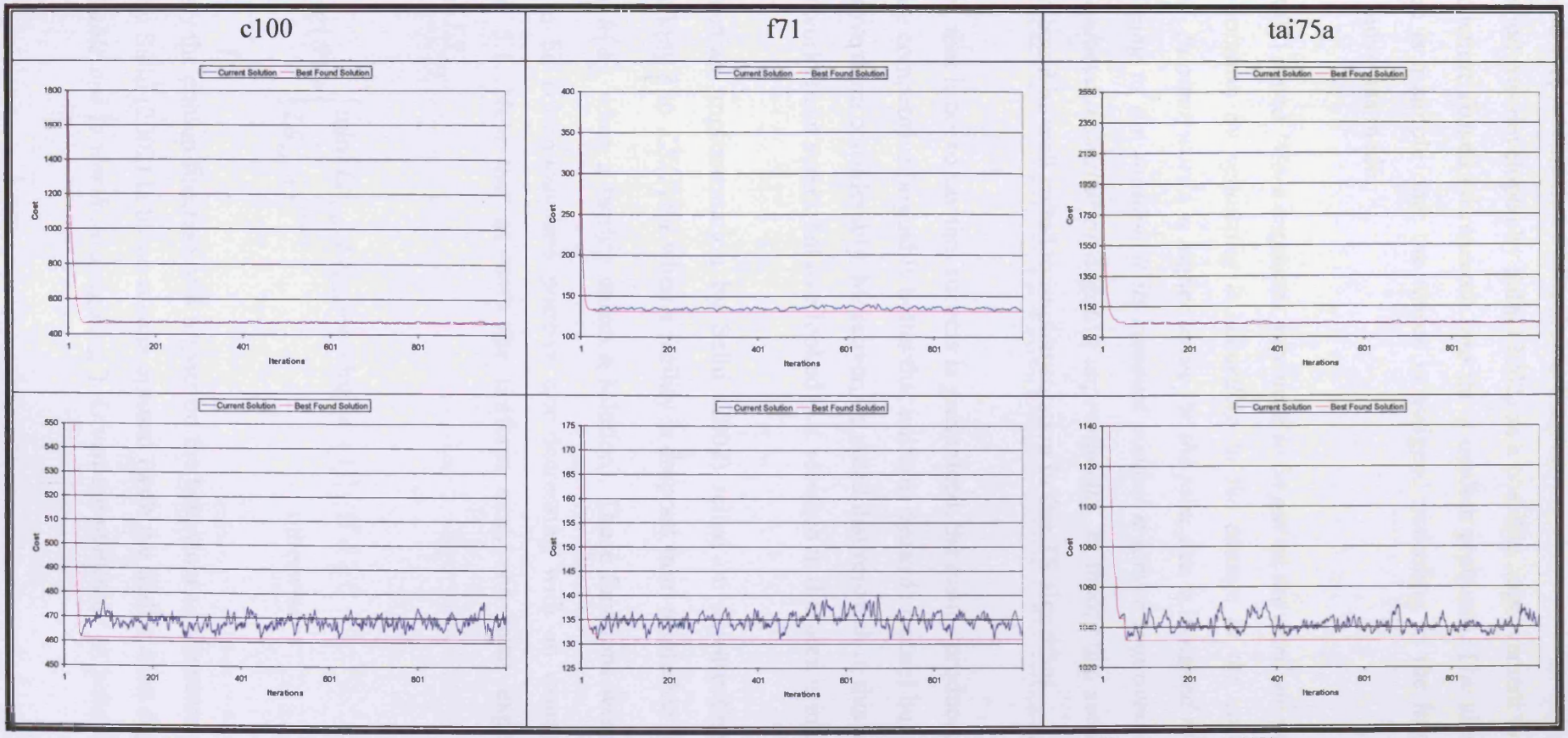

Figure 5.9: Tabu Search Behaviour with the Uniform/Geometric Dynamic Tabu Tenure (Customer Pair TC)

Functional Representation of the Tabu Tenure

This method was developed by Salhi (2002) as a possible improvement to the uniform dynamic tenure (based on research into the p-median problem). The idea behind the technique is a simple one; the tenure is assigned according to the impact that the specific move has made.

Salhi (2002) stated *'More emphasis may need to be put on the attribute which has just left the solution by penalizing it according to the change in the cost function it produced. In other words, a higher value for the tabu size is assigned to an attribute for returning to the solution if its removal yielded a larger improvement than one which produced a smaller or negative improvement'*. In theory this seems a sensible idea, and could be well suited to implementation in this TS algorithm.

However, that is not to say that success is guaranteed, the results produced by Salhi (2002) are competitive (arguably better that uniform dynamic tenure) but certainly do not improve them considerably. Moreover, he noted that research in this area has (up until his work) been somewhat overlooked (i.e. research in this area is in its infancy).

The p-median implementation by Salhi (2002) relied on constructing a mapping $g(\delta)$ linking δ to $LS(i)$ for when a facility is dropped from a solution (and a similar function $h(\delta)$ when a facility enters a solution). These functions were specifically chosen to be continuous and positive non-decreasing with an example given in Equation 5.1. Note that as with the uniform case, one must explicitly define $LS_{\min}$ and $LS_{\max}$.

$$LS(i) = g(\delta) = \begin{cases} \min\left(LS_{\max}, LS_{\min} + e + \log(\delta + 1)\right) & \text{if } \delta \geq 0 \\ LS_{\min} + e^{(1+\delta)} & \text{otherwise} \end{cases} \tag{5.1}$$

Obviously the chosen function will impact on the behaviour of the search and the one chosen by Salhi (2002) is by no means optimal (with the author even suggesting that *'any suitable one is worth investigating'*). Given the multitude of potential functions

to choose from, with no knowledge which are likely to perform well, an alternative discrete method is proposed.

Instead of relying on the user defining a suitable function, the proposed technique relies purely on the selection of $LS_{\min}$ and $LS_{\max}$. The way in which the tenure is distributed between these values relies on the relative success of the moves being chosen.

Consider the situation (as seen in Figure 5.7) whereby the search has successfully reduced the initial solution to a local optimum. The algorithm then begins accepting moves that increase the distance function, with results indicating that further improvements are often found relatively quickly. It is the period after this initial success that is interesting; the search settles down, and as a result, the range of values that δ takes is relatively small.

A user-defined boundary is selected, which divides the search into the initial success period and the remaining settled down search period (the current area of interest). Once this boundary is met, the tenure based diversification procedure is instigated.

The first move after the transition remains the same; a value of current iteration + LS is entered into the $n \times n$ matrix for the customer pair removed. However, at this point two new variables are introduced, $\delta_{\max}$ and $\delta_{\min}$, which are both initialised to the value of δ_{ij}. The next (and all subsequent moves) update the values of $\delta_{\max}$ and $\delta_{\min}$ according to the following rules:

If $\delta > \delta_{\max}$ then $\delta_{\max} = \delta$

If $\delta < \delta_{\min}$ then $\delta_{\min} = \delta$.

Once these variables have been updated (one of which will occur after the second iteration after the period transition) one can calculate $LS_{\max} - LS_{\min}$ bins, each with an equal width calculated according to Equation 5.2.

$$width = \frac{\delta_{max} - \delta_{min}}{LS_{max} - LS_{min}} \tag{5.2}$$

With the bins calculated, the number of the bin within which the current δ falls, is added to LS_{min} to calculate the tenure to be assigned to that particular move. This leads to the tenure being assigned based on its relative success to those moves which preceded it. This process is then repeated for all the remaining iterations.

For simplicity (and supported by Figure 5.7), the initial success period is defined as the first 250 iterations, with the remaining 750 constituting the period of interest. The results for two different ranges are presented in Table 5.11.

Dist'n	*LS*		**c100**	**f71**	**tai75a**
Constant	25	Min	999.42	310.77	**1782.15**
		Ave	**1026.80**	321.63	**1919.97**
		Max	**1060.45**	**340.55**	2065.96
Dynamic	20-30	Min	1003.57	**293.69**	1877.67
		Ave	1041.95	**318.35**	1950.14
		Max	1090.24	346.51	2060.71
Dynamic	15-35	Min	**973.99**	302.04	1829.35
		Ave	1028.27	325.15	1926.66
		Max	1067.33	341.69	**2038.86**

Table 5.11: Assessing the Impact of Functional Representation of the Tabu Tenure

Before interpreting these results (i.e. discussing whether any improvement has been identified), it is interesting to consider the frequency with which the different tenures have been selected. Analysing the larger (15-35) range, histograms and cumulative frequency plots have been produced for each of the three test datasets. These graphs, depicting the behaviour in ts_0, can be seen in Figure 5.10.

All of these graphs appear to be negatively skewed, indicating that there are a greater proportion of large tenures being assigned than small. The extent to which this is occurring can clearly be seen in the cumulative frequency plots, where even the least skewed dataset (tai75a) assigns over twice as many tenures >25 than it does ≤25.

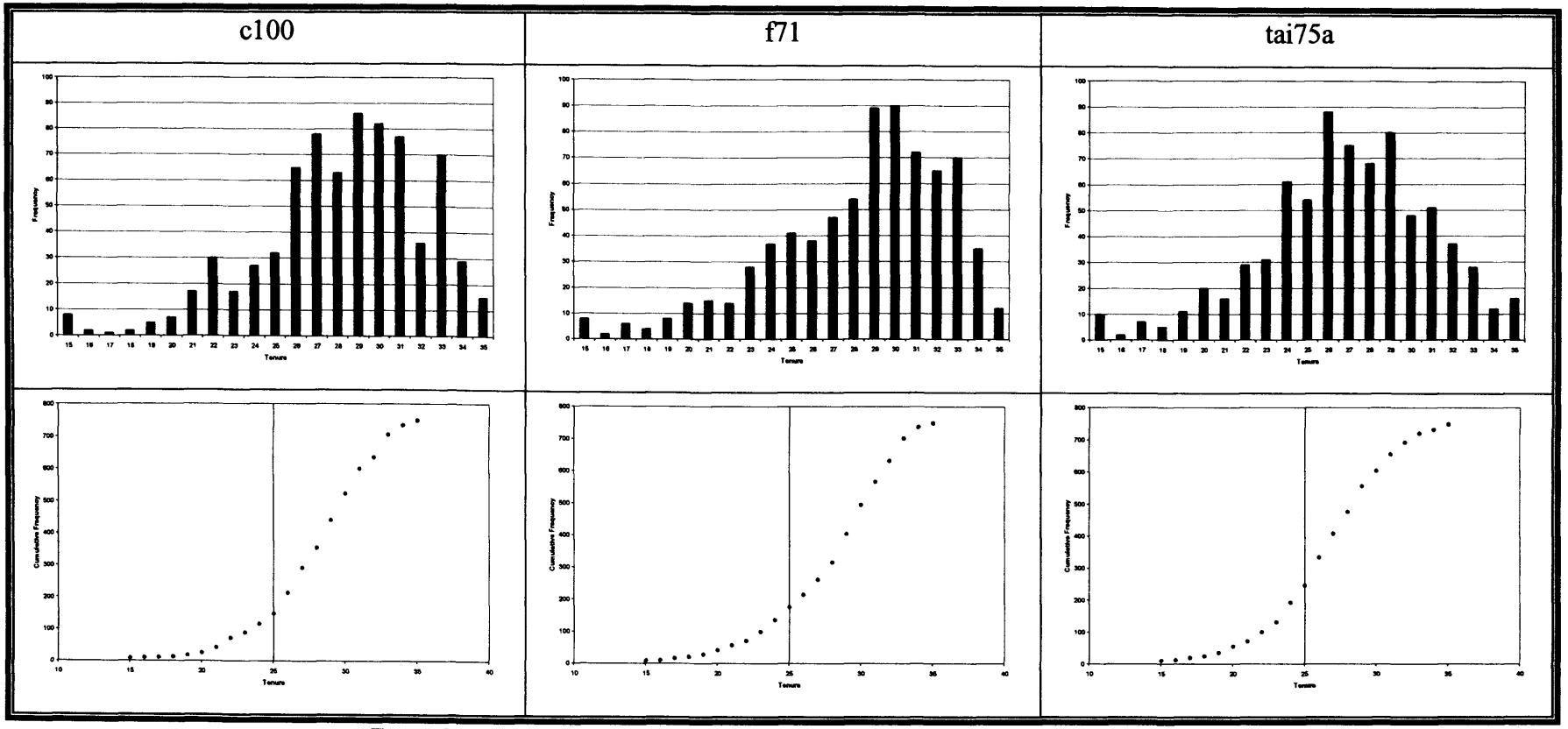

Figure 5.10: Frequency Plots for Functional Representation of the Tabu Tenure (15-35)

With regards to the actual performance, it is clear the larger range (15-35) seems to perform to a higher standard than the more restricted (20-30) case. Interestingly the (20-30) range here seems to marginally out-perform the (20-30) uniform case, much in line with the results on the p-median problem achieved by Salhi (2002).

Unfortunately, it seems clear that there is no improvement in solution quality found through the introduction of this dynamic tabu tenure. This method has produced one particularly good solution for the c100 dataset, but such behaviour cannot be directly attributed to the inclusion of the dynamic tenure. The constant tabu tenure (LS=25) is still the best performing (and is also the simplest to implement).

With no improvements found via the use of either the random or more intelligent dynamic tabu tenures, there is an obvious need to investigate other techniques that encourage diversification.

5.4.2 Further Diversification Techniques

In this section the aim is to encourage a greater level of diversification whilst not altering the tabu tenure, which has been shown to work well if it remains constant. With little control able to be exerted over the actual tabu mechanism, consideration is given to factors outside of this.

Two other forms of diversification are proposed: the first alters the selection of the moves made in the search, while the second is a simple adaptation that tries to utilise the strength of the initial success period. These techniques were chosen as they both attempt to address different shortcomings of the current algorithm.

Long-Term Memory: Customer Movement Constraining

The basic premise of the long-term memory structure was presented in Glover (1989) (see Section 2.5.3 for details). It outlined how the search could benefit from being forced to explore new regions of the search-space, and presented details of how one could implement it on the TSP. It was suggested that one stores the frequency with which each customer pair (arc) appears in the solutions generated and penalises those that occur frequently. It was suggested that one could view this as a '*frequency based tabu criteria*' as opposed to the more traditional '*recency based tabu criteria*'.

Although this technique could be included in the existing TS algorithm, an interesting adaptation to this technique has already been successfully included in two CVRP workings. Originally proposed in the workings of Taillard (1994) (though has since been successfully utilised by Gendreau et al. (1994)), was a method based on storing details relating to the way the search is being conducted (as opposed to the solutions identified).

The method relies on the assumption that certain customers will be moved within the search more frequently than others e.g. those with low demand. If this is the case (which it will be shown to be, see Figure 5.11), one can diversify the search by encouraging the more frequent movement of those customers that would otherwise have been likely to remain in their existing location.

Taillard (1994) suggests that the way to achieve this encouragement is to '*penalise moves that are frequently performed*'. The details of how this idea is incorporated into the search are now provided. These details are taken from the workings of Gendreau et al. (1994) (as their description of the method is more detailed).

It is proposed that one artificially alters the value of δ by '*adding... a term proportional to the absolute frequency of movement of the vertex*'. The selection of which move to make (BI) is then based on the new values (known as *Pseudo*δ). One can choose what components to include in the computation of *Pseudo*δ (though implementation is similar to that used by Gendreau et al. (1994)). Details for this implementation can be seen in Equation 5.3.

$$Pseudo\delta = \delta + \left(\delta_{\max} \times \sqrt{no.\ vehicles} \times g \times \left(\frac{movement_i}{current\ iteration} \right) \right) \tag{5.3}$$

where $\delta_{\max}$ is the absolute value of the maximum feasible move, g is a scaling factor (user controlled parameter) and $movement_i$ is the frequency with which customer i has been included in a string exchange. The square root of the number of vehicles has been included (as in the method of Gendreau et al. (1994)) as this is proportional to the size of the neighbourhood.

This technique is incorporated into the existing TS algorithm and the results are presented in Table 5.12. Note that as the technique is attempting to diversify the search, there is little benefit from implementing it until the search has begun to stagnate. To prevent the search from being unnecessarily diversified in the early stages of the search, the move selection remains based on δ (as opposed to *Pseudo*δ) for the first 250 iterations.

g		**c100**	**f71**	**tai75a**
	Min	999.42	310.77	1782.15
0	Ave	**1026.80**	321.63	**1919.97**
	Max	1060.45	340.55	2065.96
	Min	987.86	305.94	1801.29
0.01	Ave	1037.03	322.47	1948.18
	Max	1120.90	**339.16**	2072.92
	Min	1002.00	306.74	**1731.06**
0.5	Ave	1030.40	324.91	1943.65
	Max	**1060.42**	354.07	**2032.13**
	Min	996.41	303.16	1782.04
1	Ave	1028.12	323.52	1927.62
	Max	1077.35	343.17	2053.67
	Min	980.90	**299.70**	1805.38
5	Ave	1029.36	**316.67**	1953.27
	Max	1088.33	342.94	2179.95
	Min	**956.04**	307.31	1834.88
10	Ave	1035.70	317.16	1992.18
	Max	1076.07	341.00	2070.62

Table 5.12: Analysing the Inclusion of the Customer Movement Constraints

The results suggest that the inclusion of the customer movement constraints has not had a significant impact on the quality of the results produced. One could argue that there is no benefit associated with its inclusion, or if there is, it is minimal. Of the values for g assessed (g=0 excluded) it seems that one would favour the selection of g=5. The results are the most consistent across the three dataset even though g=0.5 contains more of the best performing metrics.

Given that one is diversifying the search, it was felt beneficial to see if the algorithm could benefit from an increase in the number of iterations (i.e. greater areas of the search-space will be visited). However, when the algorithm was rerun with z=5000, there appeared to be no increase in the solution quality.

Although g=0 is as good as any of the other results, it is proposed that one continues with the customer movement constraint to see if it can benefit from being used periodically (as suggested for another long-term memory in Glover (1989)). It may be that by permanently diversifying the search, one is preventing the search from focusing on the promising areas that are identified. However, before this investigation is considered, some details relating to the impact that the customer movement constraints have had on the search are presented.

It was noted that when this long-term memory was first proposed, it relied on the assumption that certain customers would be more frequently moved than others. To investigate whether this is true, and if the customer movement constraints are having the desired impact, Figure 5.11 shows the various frequencies with which each customer is being moved in the c100 dataset (for the various values of g assessed).

One can immediately see that there is a far from even distribution of movement between the 50 customers when the customer movement constraints are not considered (i.e. g=0). Customer number 51 has remained in the vehicle to which it was initially allocated and it has not directly been moved within that vehicle (indirect movement is not monitored); whereas customer number 59 has been moved in excess of 350 times. This supports the initial assumption, and suggests that if one can achieve a more even distribution of movement amongst the customers, one will diversify the search.

The other five graphs demonstrate how the distribution of customer movement is altered as the value of g is increased. As expected, the larger the value for g, the more even the distribution of movement appears to be between the 50 customers. When g=10 is selected there is only just over 100 moves difference between the most frequently moved customer and the least. One could further increase the value of g to hope for a more even distribution, but it is likely that any further diversification is

likely to remove the algorithms ability to focus in on the promising regions of the search-space.

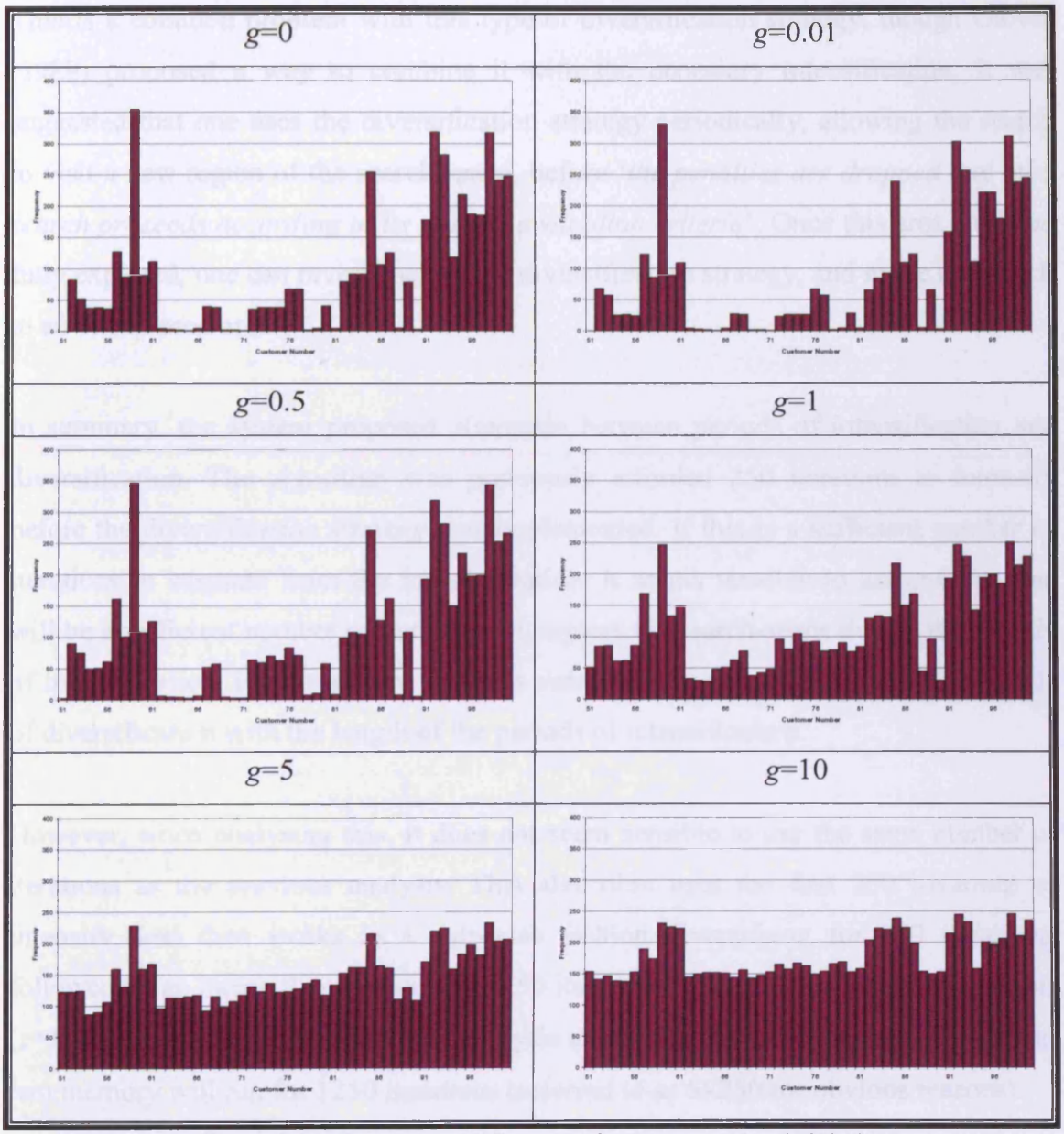

Figure 5.11: Frequency of Movement for Customers in c100 (ts_0)

With the technique seemingly working as expected, but the result quality not improved, one has to consider if the algorithm is diversifying the search too much.

Periodic Use of the Long-Term Memory

It seems that the long-term memory has had the desired impact of diversifying the search, but that the algorithm may now be lacking the necessary intensification to thoroughly explore the newly visited areas of the search-space (before the long-term

memory moves the search somewhere else). It seems that some form of adaptation will be required, if this technique is to be beneficial to the search.

This is a common problem with this type of diversification strategy, though Glover (1989) proposed a way to combine it with the necessary intensification. It was suggested that one uses the diversification strategy periodically, allowing the search to visit a new region of the search space, before '*the penalties are dropped and tabu search proceeds according to its normal evaluation criteria*'. Once this area has been fully explored, one can revert back to the diversification strategy, and move the search to an unexplored area.

In summary, the system proposed alternates between periods of intensification and diversification. The algorithm was previously afforded 250 iterations to intensify before the diversification strategy was implemented. If this is a sufficient number of iterations to stagnate from the initial solution, it seems sensible to assume that this will be a sufficient number of iterations to explore the search-space during the periods of intensification. For simplicity, it seems sensible to match the length of the periods of diversification with the length of the periods of intensification.

However, when analysing this, it does not seem sensible to use the same number of iterations as the previous analysis. This algorithm uses the first 250 iterations to intensify, and then works in a pair-wise fashion diversifying for 250 iterations, followed by an intensification period of 250 iterations. To match the previous analysis (z=1000) as closely as possible, the analysis containing the periodic use of the long-tem memory will run for 1250 iterations (referred to as 5×250 for obvious reasons).

The results of this analysis (see Table 5.13) are compared to those produced with the customer movement constraint included. If these results are bettered then a comparison will be made with the results where g=0 to see if an overall improvement has been identified.

It seems that the solution quality has improved in each of the three datasets, and that one would favour periodic use of the long-term memory as opposed to it being

constantly utilised. This pair-wise system is a simple technique, that seemingly ensures the desired balance between intensification and diversification (if one has confidence in the two phases individually). Furthermore, the results are of a higher quality than those achieved prior to the customer movement constraint being introduced (i.e. g=0) and it therefore seems sensible to continue to use g=5 (but only periodically).

z		c100	f71	tai75a
1000	Min	**980.90**	299.70	1805.38
	Ave	1029.36	316.67	1953.27
	Max	1088.33	342.94	2179.95
5×250	Min	991.14	**294.36**	**1782.15**
	Ave	**1022.51**	**314.54**	**1895.96**
	Max	**1082.38**	**340.98**	**1985.63**

Table 5.13: Assessing the Impact of the Periodic Use of the Long-Term Memory (g=5)

As the algorithm has already been tested with z=5000 and no improvement been found, it is safe to assume that this increase in performance is not down to the increase in iterations. As this is being considered on the non time-constrained variant of the problem, it seems sensible to assess whether the algorithm could benefit from further iterations. The algorithm was run with z=21×250 (i.e. z=5250) but no benefit was found. The behaviour of the search can be visualised in Figure 5.12 (same y-axis as Figures 5.7 and 5.9).

As one can see the search begins with a period of intensification, identifying a best solution, but then is unable to locate a higher quality solution. After 250 iterations have passed, the long-term memory is activated and more diverse moves (as indicated by the more extreme changes in distance) are selected. Upon reaching a new region of the search-space (i.e. after 500 iterations have been completed), the search again intensifies and locates a higher quality solution that was previously identified. The same procedure then occurs again, though no further improvement was identified (in this example).

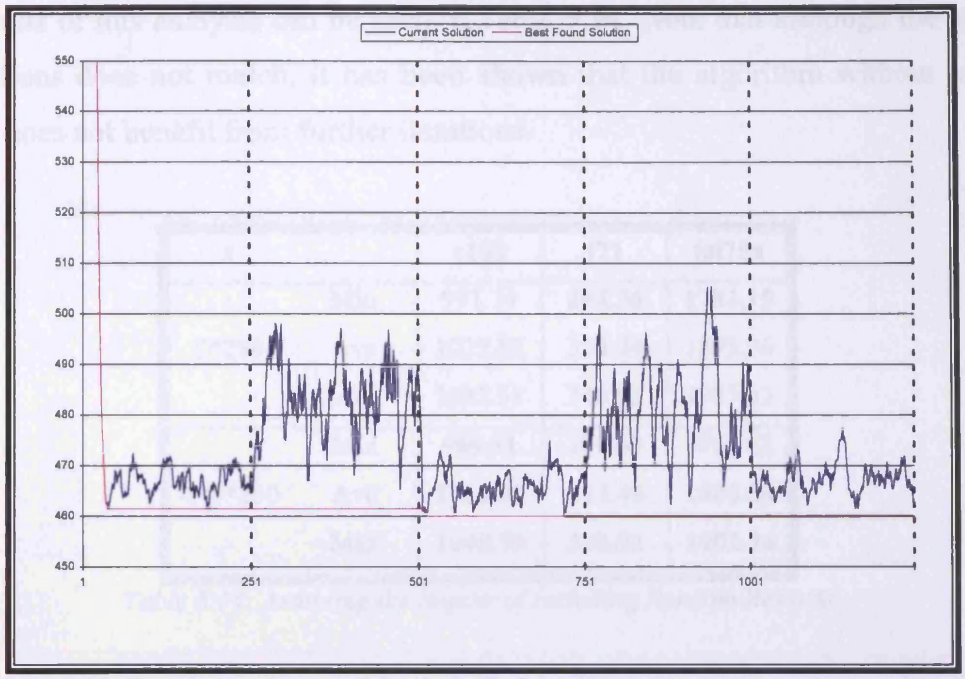

Figure 5.12: Visualising the use of the Periodic Long-Term Memory Function on c100 (ts_0)

With the success of this example, and the results supporting its inclusion, consideration can now be given to a more random form of diversification.

Further Diversification (Random Restarts)

The diversification strategy currently employed is implemented after a period of intensification and seeks to direct the search towards new previous unexplored regions of the search-space. However, it is likely that the solution (i.e. that found at the end of the diversification phase) will continue to exhibit common components with that which it is trying to get away from, as one retains an interest in the distance of the solution (it is included in the calculation of *Pseudo*δ).

As it has been shown that the algorithm does not benefit from an increase in the number of iterations, a more extreme form of diversification is proposed. Currently the search terminates after the 5×250 iterations have been completed. It is now proposed that one randomly moves to a new region of the search space and begins another set of 5×250 iterations i.e. runs the same algorithm again but from a new initial solution. This technique can be employed numerous times, though four has been selected, so as to limit the increase in run-times.

The results of this analysis can be seen in Table 5.14. Note that although the number of iterations does not match, it has been shown that the algorithm without random restarts does not benefit from further iterations.

z		c100	f71	tai75a
5*250	Min	991.14	**294.36**	**1782.15**
	Ave	1022.51	314.54	1895.96
	Max	1082.38	340.98	1985.63
4*5*250	Min	**969.31**	299.48	1862.62
	Ave	**1014.49**	**311.49**	**1886.60**
	Max	**1045.79**	**320.92**	**1902.36**

Table 5.14: Assessing the Impact of Including Random Restarts

The results suggest that the system does benefit from this type of increase in iterations (theoretically one cannot produce a worse result in each individual timeslot), and as such, one would favour the inclusion of random restarts in the algorithm.

The result quality has improved across all three of the test datasets, with particularly significant improvements being made to the worst solution of the trial (Max). This is beneficial as it means that one is reducing the variability in the results being produced, something that is very important if one is solving the problem in real-life (i.e. has only one attempt at producing a solution). All subsequent non time-constrained analysis will continue to use four random restarts.

Introducing Check-Back Mechanism

As explained in Section 5.2.3, it is useful to consider whether the customers in the current timeslot are the same as those that were in the previous (less those that have been committed to). If this is the case, it seems prudent to compare the result achieved in the previous timeslot with that produced in the current. If the solution quality has worsened, then it seems sensible to replace the solution with the higher quality one that is already known.

As explained when this issue was first considered, this has been left to the end of the algorithms analysis, so that all of the other results are directly comparable. Results including the check-back mechanism can be seen in Table 5.15.

CB		c100	f71	tai75a
No	Min	**969.31**	**299.48**	1862.62
	Ave	1014.49	**311.49**	**1886.60**
	Max	1045.79	**320.92**	**1902.36**
Yes	Min	976.01	299.59	**1849.32**
	Ave	**1005.15**	320.61	1891.45
	Max	**1044.57**	342.26	1993.73

Table 5.15: Introducing the Check-Back Mechanism

Although the results that include the check-back mechanism are clearly not of a higher standard than those that do not, with regards to the solution found within a single timeslot they have to be of a higher standard. Any decrease in quality has to be down to the randomness of the initial solution or the fact that a good solution in one timeslot can lead to a worse one in subsequent timeslots. For these reasons, one has to favour the inclusion of the check-back mechanism even if these results do not support it.

This concludes the development of this TS algorithm and consideration can be given to the production of a full set of results (both non time-constrained and time-constrained).

5.5 Results and Conclusions

As with the results (and analysis) of the ERWACS-DVRP, this section is divided into two distinct sections. The first section will continue to tackle the problem on which the previous analysis has been conducted i.e. the non time-constrained variant of the problem, while the second considers the time-constrained variant.

For both of these sections, the primary set of results used for comparative purposes are those produced via the ERWACS-DVRP (as produced in Chapter Four). However, both sections will continue to make reference to the results of the ACS-

DVRP of Montemanni et al. (2005) so as to ensure that one judges the Tabu-DVRP against a set of results not produced in this thesis. As before, this comparison is not like-for-like (i.e. TABU-DVRP results are non time-constrained), and as such, one cannot draw conclusions with absolute certainty.

5.5.1 Results: Non Time-Constrained

These are the second complete set of results that have been produced with the philosophy of giving the algorithm all the time that is necessary to perform to the best possible standard. As a result of this, one can make a direct comparison between these results and those produced via the ERWACS-DVRP. However, one must remember that the non time-constrained results of the ERWACS-DVRP were far superior to the time-constrained results of the ACS-DVRP, and thus making this comparison may show the Tabu-DVRP in a worse light (than had it been compared to the ACS-DVRP).

To compensate for this possibility, some summary statistics relating to the difference in performance of these algorithms will be provided to supplement the main analysis.

As with the results of the ERWACS-DVRP, it is useful to see the run-times that the algorithm required (and the extrapolated values for T). This will enable one to establish whether either of the two algorithms developed has a time benefit over the other. The results are presented in Table 5.16.

As one is comparing two algorithms that have been produced on the non time-constrained variant of the problem, one can draw conclusions with greater certainty than in the similar non time-constrained analysis in Section 4.7.1 (where one was comparing across problems).

The most basic observation is that solutions are of a similar quality to that produced by the ERWACS-DVRP. The metrics seem to be fairly evenly distributed across the two algorithms, with the Tabu-DVRP reporting improvements in 30 of the 63 metrics. This suggests that one has produced another algorithm that outperforms the time-constrained ACS-DVRP (though once again, one cannot state with absolute certainty given the time disparity). Had one made the comparison between the ACS-DVRP and

the Tabu-DVRP one would have seen that 60 of the 63 metrics had been improved (which is even more than for the equivalent ERWACS-DVRP analysis).

Dataset	*ERWACS-DVRP*			*Tabu-DVRP*			TABU Ave	TABU Working
	Min	Ave	Max	Min	Ave	Max	Time	Day
c100*	**963.07**	**999.35**	**1029.64**	976.01	1005.15	1044.57	15	29
c100b	889.30	909.48	930.37	**885.94**	**906.35**	**923.74**	6	11
c120	**1256.76**	**1331.25**	**1447.21**	1376.75	1460.94	1536.66	18	35
c150	**1308.01**	**1353.48**	**1413.44**	1313.82	1366.40	1429.87	24	46
c199	1626.91	**1654.24**	**1679.76**	**1620.84**	1666.19	1702.46	31	58
c50	**592.42**	**622.83**	**639.22**	**592.42**	629.33	658.88	3	5
c75	948.15	1008.83	**1038.79**	**928.50**	**996.68**	1056.47	3	5
f134	15290.58	**15577.93**	**15827.86**	**15220.03**	15841.54	16926.06	7	13
f71*	305.59	**313.97**	**319.60**	**299.59**	320.61	342.26	19	37
tai100a	**2187.23**	**2248.92**	**2303.20**	2205.56	2276.40	2335.96	7	14
tai100b	2184.73	2253.25	2301.74	**2156.06**	**2212.64**	**2271.81**	7	13
tai100c	**1524.24**	1622.36	1682.00	1529.05	**1576.47**	**1651.93**	6	11
tai100d	1896.84	**1969.24**	**2080.30**	**1880.80**	1987.46	2099.74	11	21
tai150a	3364.48	3446.05	3618.59	**3302.80**	**3391.33**	**3508.45**	23	44
tai150b	2872.18	**2889.56**	**2906.17**	**2869.36**	2923.54	3012.86	18	34
tai150c	2595.27	2751.67	2895.55	**2522.44**	**2708.30**	**2893.79**	19	36
tai150d	2952.70	**3025.66**	**3071.65**	**2947.46**	3030.72	3127.58	17	31
tai75a*	1866.27	**1883.78**	**1901.35**	**1849.32**	1891.45	1993.73	3	7
tai75b	1464.29	1575.99	1675.46	**1438.84**	**1502.82**	**1618.47**	2	4
tai75c	1619.27	1657.94	**1708.02**	**1527.96**	**1635.32**	1724.20	3	6
tai75d	1436.15	**1444.75**	**1457.15**	**1431.62**	1456.17	1488.83	3	5

Table 5.16: Computational Results for the Non Time-Constrained Tabu-DVRP

Note that had all three sets of results been used in this analysis, the best metrics would have been spread 2, 33, 28 across the ACS-DVRP, ERWACS-DVRP and Tabu-DVRP algorithms respectively.

When one considers the percentage change in each of the metrics (averaged across the 21 datasets) one can see just how similar the level of performance is. The Tabu-DVRP has resulted in an increase in the minimum distance of 0.40%, while it reports

decreases of 0.29% and 1.64% in the average and maximum values respectively. These figures suggest that there is less variability in the results of the Tabu-DVRP than there was in the ACS-DVRP, though this is marginal and may simply be due to randomness in the run rather than an inherent difference in performance. Summary statistics for a comparison between the ACS-DVRP and the Tabu-DVRP have not been produced, as they will be of a similar scale to those reported for the ERWACS-DVRP.

Consideration is now given to the time that the algorithm was run for and some details regarding the behaviour of the search.

Run Time Observations

The run times for these results ranged from 2-31 minutes, which gives extrapolated T values of 4-58 minutes. These values are less than the respective times for the ERWACS-DVRP, so one need not argue their suitability for the real-life environment (as it was felt that the ERWACS-DVRP times were acceptable). Some various details regarding the run-times are noted, before consideration is given to whether one should consider the Tabu-DVRP to be superior to the ERWACS-DVRP given the similar performance and lower run-times.

Given the low values for T (seven of the datasets produced results with an extrapolated T value of less than 12.5 minutes), one may well encounter less of a difference between the non time-constrained and time-constrained results in the next analysis. However, although these datasets will actually receive a greater amount of time when the time-constraint is introduced, if one has successfully run the algorithm long enough to showcase its ability, there should be no change in performance (other than through the randomness of the initial solutions).

As with the ERWACS-DVRP there appears to be a strong correlation between the number of customers and the time taken for the algorithm to run. This suggests that the problem size is a fairly good indicator of the neighbourhood size (though there are exceptions such as f134).

The Tabu-DVRP has produced results that are seemingly the equal of the ERWACS-DVRP in less time. However, it would be unfair to judge one of these algorithms as better than the other on the basis of time, because much of the parameter selection was made whilst erring on the side of caution (longer run times were accepted to try to ensure the best level of performance). Furthermore, the problem being tackled (the non time-constrained DVRP) does not require results to be produced within as little a time as possible; both of these methods can easily be used in a real-life working day.

The Behaviour of the Search

The search is comprised of four separate searches, each containing three periods of intensification interspersed with two periods of diversification. As expected, there appeared to be a fairly even distribution between the four separate searches as to which identified the best solution. This is due to the short-term memory (*TL*) being so short-term i.e. the values relating to one search will have little impact on the other searches. Note that although the best solution is often located in the fourth search (as often as in any other search) it was felt that further random restarts would be of little benefit.

Were one to have considerably more time and no desire to further alter the algorithm, one could continue to run with more random restarts, though it is suggested that one could further improve the search by considering a more appropriate use of this extra time (as detailed later in this subsection).

To help gain an understanding of the behaviour of the search, the results during the run were graphed (for a number of runs). It was noted that the best solutions were usually identified in the periods of intensification (i.e. when the long-term memory is not activated). This suggests that the search is operating as one would expect, and supports the use of the long-term memory periodically (it also confirms why the solutions that included the constant use of the long-term memory were not of a higher standard).

It is suggested that were one to continue research into this area, one ought to consider making the periods of intensification more intense. The one currently being utilised was originally proposed as a general search (i.e. not used in this two phase manner)

and has not been specifically tailored to this new objective. It may also be of benefit to include a more adaptive system of alternating between the two phases, based on the quality of the results being achieved. One may also wish to consider the use of an intensification procedure such as an intermediate-term memory to influence the new initial solutions i.e. a non-random restart (see Page 54).

5.5.2 Results: Time-Constrained

These are the third set of results that have been produced for this specific problem (the ACS-DVRP and ERWACS-DVRP have preceded it). In traditional combinatorial optimisation problems one would usually compare the results of a new algorithm with the best performing algorithm (across all of the datasets) and the best-achieved results (usually produced by a variety of algorithms). As one is hoping that this thesis will go someway to establishing the DVRP as a commonly researched problem, it seems sensible to continue with this policy.

Clearly the ERWACS-DVRP outperforms the ACS-DVRP and thus this algorithm is considered as the best performing algorithm (and is used for the majority of the analysis). The best-achieved results will be a combination of these algorithms (17 from the ERWACS-DVRP and 4 from the ACS-DVRP), see Section 4.7.2 for details. Note that at the end of Chapter Three ACO and TS were both considered worthy of further research. The fact that the ACO investigation preceded the TS investigation was a personal choice and thus it seemed sensible to present a short analysis (at the beginning of this subsection) of its performance, had the research been conducted the other way round (i.e. comparing the Tabu-DVRP with the ACS-DVRP).

In the development of the ERWACS-DVRP it was felt important to adapt the algorithm developed for the non time-constrained variant of the problem (as it was felt that the algorithm in its current guise was not using the time allocation appropriately). However, given that the run times for the Tabu-DVRP were lower than those for the ERWACS-DVRP (i.e. much closer to the time constraint value of T=12.5 minutes), it did not seem necessary to make any further alterations to the algorithm. The algorithm will continue to run in blocks of 1250 iterations (three intensification periods and two diversification) but instead of defining the number of random restarts, one just continues to instigate random restarts until the time limit has

been met. Note that the algorithm will terminate in the middle of a run if the time has expired.

The results for the Tabu-DVRP (time-constrained) are presented in Table 5.17.

	Best	ERWACS-DVRP			Tabu-DVRP		
		Min	Ave	Max	Min	Ave	Max
c100*	970.34	**970.34**	**1006.16**	**1049.01**	988.39	1020.16	1071.50
c100b	874.06	**874.06**	**900.50**	**916.99**	903.52	917.50	941.65
c120	1256.27	**1256.27**	**1344.37**	**1440.99**	1394.22	1482.73	1577.86
c150	1342.79	1342.79	1388.71	1447.88	**1334.05**	**1387.84**	**1412.56**
c199	1654.54	1654.54	1731.12	1780.06	**1622.85**	**1701.87**	**1770.58**
c50	604.32	**604.32**	**629.09**	**644.79**	620.58	637.18	658.88
c75	989.13	989.13	1014.91	**1052.74**	**942.87**	**1000.62**	1053.92
f134	15135.51	**15190.45**	**15672.58**	**16406.66**	15603.40	16144.30	16980.96
f71*	304.80	304.80	314.34	**319.60**	**299.91**	**311.26**	332.86
tai100a	2186.11	**2186.11**	**2254.20**	**2359.76**	2213.71	2269.19	2368.17
tai100b	2207.39	2207.39	2253.39	2292.99	**2137.46**	**2218.81**	**2280.40**
tai100c	1562.30	1566.52	1667.85	**1682.00**	**1521.71**	**1598.24**	1699.56
tai100d	1907.50	1907.50	2012.42	2127.42	**1857.19**	**1976.45**	**2060.33**
tai150a	3319.35	3319.35	**3437.58**	**3537.85**	**3313.85**	3489.84	3670.47
tai150b	2874.75	**2874.75**	2936.70	3054.33	2881.65	**2912.19**	**2956.03**
tai150c	2576.08	2576.08	2788.00	2953.54	**2521.87**	**2684.99**	**2846.47**
tai150d	3004.66	**3004.66**	3075.90	3179.98	3012.06	**3027.96**	**3047.07**
tai75a*	1843.08	**1872.93**	1900.97	1977.00	1874.45	**1890.64**	**1908.69**
tai75b	1508.99	1508.99	1586.60	**1642.92**	**1491.98**	**1579.29**	1655.24
tai75c	1574.98	1583.20	1653.18	**1712.82**	**1510.58**	**1562.60**	1720.44
tai75d	1438.92	1438.92	1446.30	**1453.99**	**1430.81**	**1444.72**	1478.41

Table 5.17: Time-Constrained Comparison of the ERWACS-DVRP and the Tabu-DVRP

As stated previously, a short analysis comparing the performance of the Tabu-DVRP with the ACS-DVRP is now provided. Upon completion of this analysis, the more traditional comparison with the best performing algorithm (the ERWACS-DVRP) will be provided.

ACS-DVRP vs. Tabu-DVRP[7]

As the objective of investigating the ACO and TS metaheuristics was to see if one could produce an algorithm that outperforms the ACS-DVRP, it seems sensible to begin the analysis by considering whether this TS implementation has achieved this. For the sake of this analysis, one is (for the most part) ignoring the results of the ERWACS-DVRP (a comparison with this algorithm will be presented in the next subsection).

If one were to compare the metrics of just these two algorithms one would see that 59 of the 63 metrics have been bettered. This suggests that the Tabu-DVRP is consistently producing solutions of a higher quality than the ACS-DVRP, and that the original target of this chapter has been met. Of the four metrics that were not bettered (3 minimum and 1 average) two were also not bettered by the ERWACS-DVRP.

It seems sensible to conduct a similar analysis to that which was conducted in Section 4.7.2 i.e. considering the percentage change in each of the three metrics (averaged across the 21 datasets). One can see that minimum distances (from the trial) have been reduced by 3.99%, while the average and maximum have decreased by 6.17% and 9.04% respectively. These figures highlight just how significant the improvement in results is, and that one would always favour the use of the Tabu-DVRP algorithm ahead of the ACS-DVRP algorithm. A discussion on the variability of results will be presented in the next subsection.

Note that had the Tabu-DVRP algorithm been developed before the ERWACS-DVRP, one would have reported that 18 new best-achieved solutions had been produced. This level of performance is similar to the ERWACS-DVRP, which reported 17.

ERWACS-DVRP vs. Tabu-DVRP

This is the central analysis in this section, as it compares the current best performing algorithm on the time-constrained DVRP, with one that has just been developed. As with much of the analysis, it seems sensible to begin by making an observation based

[7] Please note that the results for this analysis have not been presented, though one can compare those shown in Table 4.17 with those shown in Table 5.18.

on the metrics that have been presented. It appears that the Tabu-DVRP has bettered the ERWACS-DVRP in 34 of the 63 metrics, which suggests that the algorithm is performing to a very high standard. Note that the ERWACS-DVRP produced better results for the remaining 29 metrics.

If one considers the percentage change in each of the three metrics (averaged across the 21 datasets), one can see that minimum distances (from the trial) have been reduced by 0.12%. Although this improvement is somewhat marginal, given that the ERWACS-DVRP is the best performing algorithm for this problem, the fact that any improvement has been identified constitutes a success. The improvement is more apparent in the average, where a reduction of 0.20% was identified. As with the minimum distances, one can argue that this level of performance is of a high quality as it has slightly bettered the best performing algorithm. Unfortunately, there is an increase of 0.53% in the distance when one considers the maximum distance (from the trial). Although small, this increase, coupled with the decrease in the minimum, suggests that there will be slightly more variability in the results produced.

This is in contrast to the results of the non time-constrained analysis (where the minimum was increased and the maximum decreased) and it was suggested that the ERWACS-DVRP was the more variable algorithm. Although this change could be as a direct result of the change in termination criterion, one must appreciate that the size of the change is very small, and that it could easily be attributed to the randomness inherent in the algorithms. As with the analysis of the ERWACS-DVRP, a discussion on the difference in performance due to the change in termination criterion will be presented on completion of this analysis.

As with the previous analysis, if a new best-achieved solution has been produced, it has been shaded in grey. The Tabu-DVRP has produced 12 new best solutions, including improvements to ten of the solutions that had already been bettered by the ERWACS-DVRP. In summary, this means that between the two algorithms, new best solutions have been identified for all but 2 of the 21 datasets (i.e. only two best achieved results can now be credited to the ACS-DVRP: f134 and tai75a*). The fact that one has not improved upon the best-achieved solution for one of the test datasets is slightly unexpected, as the algorithm has been optimised for its performance on this

dataset. However, the fact that this has not occurred (and the performance level is high on the other datasets), suggests that one has managed to produce a suitably robust algorithm.

In summary, it seems sensible to assume that these two algorithms are operating at a very similar standard and that there is little to choose between them. However, if one were required to make a choice between them, it would depend on personal preference regarding whether one was prepared to accept the possibility of a longer distance (in the slightly more variable Tabu-DVRP), for the possibility of a slightly better solution.

Solution Quality Discrepancy

As for the ERWACS-DVRP, it seems sensible to quantify the change in solution quality associated with the alteration in exit criterion (from giving the algorithm the time to perform to the best of its capabilities to a fixed limit of T=12.5 minutes).

The change in performance is marginal for each of the metrics, with the distances increasing by 0.94%, 0.51% and 0.27% for the minimum, average and maximum metrics respectively (when averaged across the 21 datasets). It is possible that this change in performance is due to certain datasets receiving further time in the non time-constrained variant, but given it is so marginal it does not appear that there is a significant difference in performance whichever termination criterion is chosen.

This suggests that one has successfully managed to balance the two different problems (non time-constrained and time-constrained), and that a single algorithm capable of high quality solutions for both has been produced.

5.5.3 Analysis Summary and Chapter Conclusions

If one were to summarise the performance of the Tabu-DVRP, one would have to consider its development a success. An algorithm that seemingly equals the performance of the best performing metaheuristic implementation (the ERWACS-DVRP) has been produced. As both of these metaheuristics perform to a standard considerably higher than the ACS-DVRP, the objectives for the work on the DVRP have been met.

There is still a level of variability in the results (similar to the ERWACS-DVRP), though this would appear to further support the assumption that this is the result of the timeslot solution strategy. It should be noted that one would expect the kind of consistent results achieved on traditional routing problems to be unachievable due to the uncertainty inherent in the problem.

The results of the ERWACS-DVRP have provided one with a suitable set of results to offer a like-for-like quantitative analysis. The fairly even spread (between these two algorithms) of best-achieved results, supports the earlier assumption that one would not expect a single algorithm to be the best performing on every problem instance. The Tabu-DVRP appears to be better suited (than the ERWACS-DVRP) to the datasets adapted from the Taillard CVRP instances; though the reverse can be stated with regards to those adapted from the Christofides.

It should be noted that the Tabu-DVRP produces results of a higher standard (for the test set of benchmarks) than the SA implementation that utilised the same neighbourhood (see Section 5.2.2). This is in keeping with the commonly held view, that TS is better suited to application within routing based problems.
It is hoped that these results further support the conclusions of Chapter Four, in so much as giving further indication that one can now treat the DVRP as a solvable problem (in much the same way as the more traditional routing problems have been). The benchmark datasets are a suitable set of problem instances, and there is considerable scope for improvement.

With regards to further improvements to the algorithm, one could consider changes to either the neighbourhood, or the intensification/diversification balance. It would be interesting to consider cyclic transfers (i.e. moving customers in a λ-Opt fashion between b vehicles in a single move) and some form of intermediate-term memory. The development of a more bespoke intensification period may also be of further benefit.

Chapter Five (Addendum)

Varying the Degree of Dynamicity

5a.1 Introduction

This addendum details the first steps of an investigation into assessing the impact that varying the DOD has on the quality of the solutions one can produce. Although this research does compliment that which has preceded it (it is concerned with solving a similar problem), it approaches the problem from a different perspective, and thus does not fit well within the main chapters.

When Kilby et al. (1998) first developed the timeslot solution strategy; they were presenting a new technique that could be used to solve the DVRP by creating a series of static CVRP-like problems. Upon the establishment of this technique, it was used to investigate how the quality of solution achieved (by a simple insert and improve algorithm) altered depending on the DOD. This addendum is concerned with the same problem, but makes use of the more complex ERWACS-DVRP and Tabu-DVRP algorithms (developed in Chapters Four and Five respectively).

This addendum contains two sections:

Firstly, some details about the benchmark datasets are provided. A discussion is presented regarding the upper limit one can place on the T_{co} (which controls the DOD), if one wishes to ensure that feasible solutions are achievable.

Secondly, the results are produced (in graphical form). These are accompanied by a basic analysis that considers the impact that varying T_{co} has on the quality of the solutions, and whether either of the two algorithms developed is better suited to more (or less) dynamic problems. It should be noted that this work is not intended to be of a similar depth to the main chapters, as it is only suggesting a possible direction that one could pursue.

5a.1.1 Varying Dynamicity in the Benchmarks

Varying the DOD is not a complex task, as Kilby et al. (1998) showed the forethought to create the benchmark datasets with arrival times for every customer (and assumed those arriving after $T \times T_{co}$ had arrived the day before). This enables one to alter the DOD by simply adjusting the value of T_{co} prior to the algorithm being run. However, before consideration can be given to producing results for these datasets (with varying DOD), one must discuss why one cannot simply select any value for T_{co}.

Note that no changes are made to n_{ts} or T_{ac} from the values in the previous chapters (25 and 0.01 respectively). This prevents one from comparing the results directly with the workings of Kilby et al. (1998), but it does allow one to view these results in the context of the work in this thesis.

Limits on the Degree of Dynamicity

Although at first glance it may seem that one ought to be able to select any value for T_{co}, certain (particularly high) values will lead to one not always being able to produce a feasible solution. This upper limit is due to the timeslot solution strategy, and the way in which it artificially delays the arrival of customers till the end of the current timeslot (and does not allow them to be scheduled until the one after that ($+T_{ac}$)).

However, it is not possible to solve these problems with a DOD of 100% (i.e. T_{co}=1) as certain customers will become known at such a late time, that one cannot schedule

them to a vehicle before the end of the working day. To ensure that this situation is not encountered, one can calculate an upper dynamic threshold, and only assess the problem with a DOD up to this point.

If one calculates the time at which each customer would first be considered for inclusion in a schedule (i.e. the end of the timeslot in which it arrives + T_{ac}); one can establish the earliest time a vehicle could be despatched to that customer (another T_{ts} seconds). Given that one cannot make any assumptions about current vehicle positions, the only way to ensure feasibility is to assume that the vehicle will be dispatched from the depot. As one is aware of the distances and service times, one can establish whether a new vehicle would be able to travel to the customer, service it, and return to the depot before the end of the working day.

If the new vehicle cannot return to the depot in time, this customer is considered to be one that is unable to become known during the problem (i.e. it cannot be dynamic and must be known in advance). The latest time before the arrival of the earliest of these customers represents the upper dynamic threshold (as one must be aware of this customer in advance). For the three test datasets (these are the only datasets considered in this analysis) the upper dynamic thresholds are 78%, 75% and 68% for c100, f71 and tai75a respectively. Note that there is no lower limit, when the DOD=0% (i.e. T_{co}=0), one is faced with the static variant of the problem (i.e. the CVRP).

5a.2 Results

Given that this is a supplementary analysis to the main body of work in this thesis, the results (and subsequent analysis) are of a more observatory nature. Consideration is given to the overall pattern of the results (i.e. any trends observed, see Figure 5a.1.) and some simple summary statistics regarding the performance levels of the two algorithms.

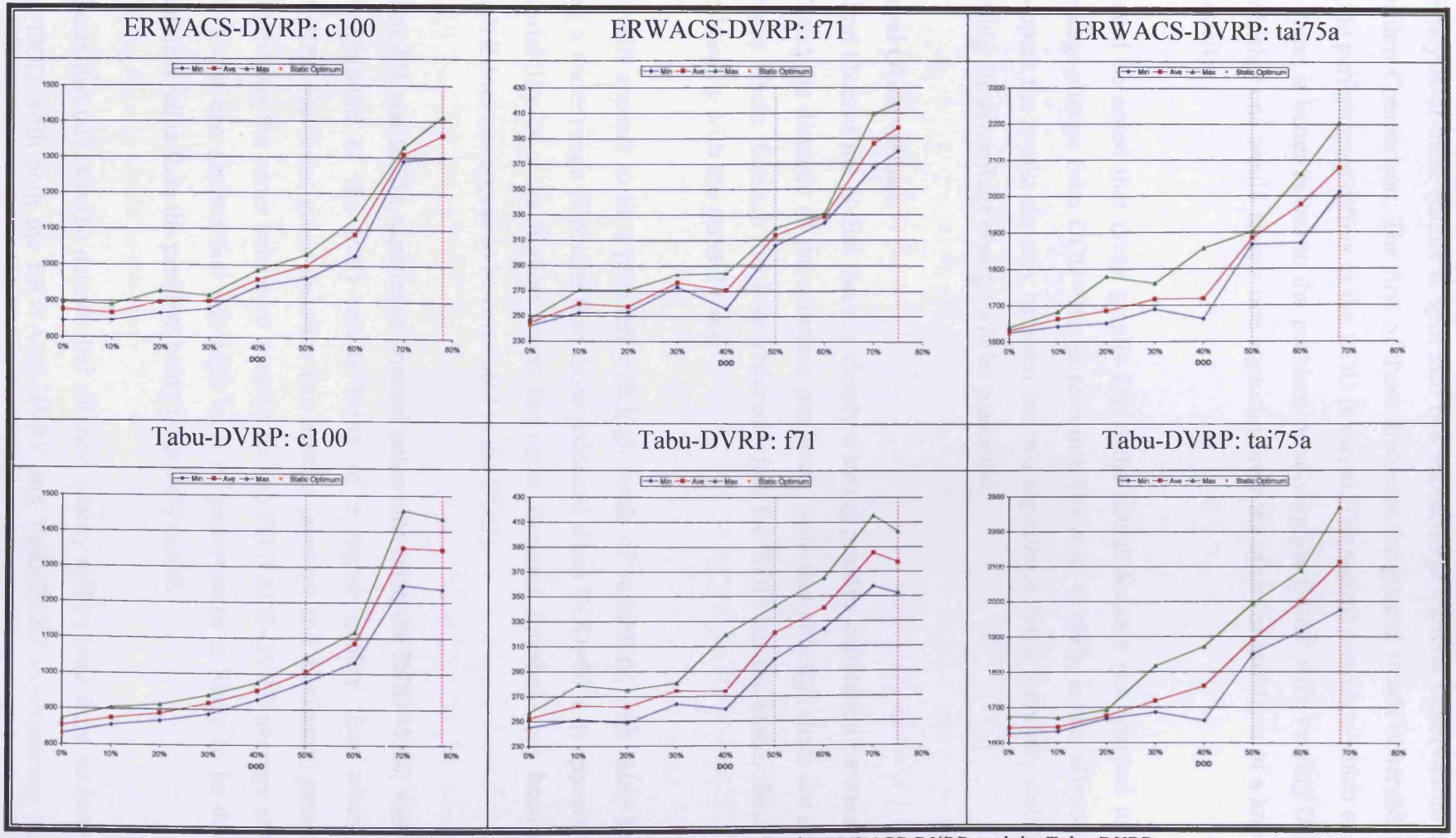

Figure 5a.1: Assessing the Impact of Varying DOD for the ERWACS-DVRP and the Tabu-DVRP

5a.2.1 Analysis

The analysis of these results is split into two subsections: General Observations and Algorithm Comparison. The first of these discusses the general trends observed and how the performance differs as the DOD is varied. The second considers which of the algorithms is better suited to the problem of solving the DVRP with varying DOD, and whether one would favour one algorithm over the other for problems of a known dynamicity.

It should be noted that these graphs depict the actual distance (as opposed to the percentage change from DOD=0%, as seen in Kilby et al. (1998)), as this allows one to compare the results directly between the two algorithms. Some summary statistics regarding the percentage change will be presented.

General Observations

The first observation is that there is clearly a strong positive correlation between the DOD and the distance of the solutions produced. Note that although there are some examples where distance has been reduced when the DOD has increased; these are not in keeping with the general trend.

There still appears to be a problem with high levels of variability, with many DOD having a wider range of results than those produced when DOD=50%. It appears that the variability is at its highest when the upper dynamic threshold has been met (though it does not appear to be correlated to the DOD).

Datasets f71 and tai75a experience unusual behaviour when the DOD=40%; with the best (minimum of the trial) results being of a higher quality than when the DOD=30%. However, given the other two metrics (average and maximum) generally do not display the same behaviour (except for the ERWACS-DVRP average results for f71), one can assume that this high level of performance is likely to be due to randomness rather than the problem being inherently easier.

It appears that tai75a is the dataset that solution quality suffers least from an increase in the DOD, with both the ERWACS-DVRP and Tabu-DVRP experiencing ≈28%

increase in distance when comparing the static result (DOD=0%) with the upper dynamic threshold (68%). The other two datasets (c100 and f71) experienced ≥50% increase in distance when making a similar comparison. Note that although these datasets both had higher upper dynamic thresholds, the performance levels were not of a comparable nature (to that in tai75a) when one considered the distance at DOD=70%.

Algorithm Comparison

If one were to conduct a metrics based analysis (similar to the previous chapters), and sum the values over all of the different DOD, one would favour the ERWACS-DVRP with 46 best (Min, Ave and Max) values to the Tabu-DVRP (that achieved 32). However, as the high levels of variability in the results was an issue when the DOD=50%, and it was observed how it was even more of an issue when the upper dynamic threshold has been met, it seems sensible to give greater precedence to the average result metric, than one did in the analysis of Chapters Four and Five.

If one were to do a similar analysis (but considered just the average result), one would be faced with a similar conclusion. The ERWACS-DVRP produced 16 best averages, compared with 10 for the Tabu-DVRP. These results would suggest that if one required as robust an algorithm as possible (with regards to customer dispersion and DOD) one would be likely to favour the ERWACS-DVRP.

If one considers the least dynamic case (i.e. the static problem, when DOD=0%), the ERWACS-DVRP outperforms the Tabu-DVRP on the f71 and tai75a datasets. The quality of the solutions produced for the static problem are of a sufficient standard that one would not be adverse to consider the algorithms developed as suitable for the traditional CVRP. Although only one best-achieved solution was produced (ERWACS-DVRP for the f71 dataset), average performance for the ERWACS-DVRP was only 6.03%, 1.04% and 0.74% higher than the best-achieved (for c100, f71 and tai75a respectively). Note that the average performance of the Tabu-DVRP was only 3.31%, 4.14% and 1.44% higher than the best-achieved.

If one considers most dynamic case (i.e. at the upper dynamic threshold), the Tabu-DVRP outperforms the ERWACS-DVRP on the c100 and f71 datasets. However, with lower levels of variability one would still favour the ERWACS-DVRP if required to implement the algorithm within a real-life environment.

5a.2.2 Analysis Summary and Addendum Conclusions

This investigation was conducted so one could see how the algorithms developed in Chapters Four and Five were suitable for solving problems of varying dynamicity. Although neither algorithm seemingly outperformed the other (with regards to average distance), the results produced by the ERWACS-DVRP were less variable (and would thus be favoured).

It is hoped that further work in this area will be pursued in the future, and that in time, there will be algorithms specifically developed to deal with problems of a known diversity. Note that using DOD=50% seems to represent a good level for the development of a robust (with regards to various DOD) algorithm; high levels of performance were noted for when the DOD=50% and on the static problem.

Chapter Six

ACO for the DVRPTW

6.1 Introduction

The VRP continues to receive substantial attention due to its continual evolution. This evolution can take many forms, the most common of which is to introduce new, interesting and challenging constraints that have to be adhered to. Referred to as variants of the VRP, they are responsible for keeping the interest levels high, along with enabling further realistic elements to be considered. It is within one of these variants (or more specifically a dynamic version of one of these variants) that this chapter focuses; the Dynamic Vehicle Routing Problem with Time Windows (DVRPTW).

This appears to be the very first implementation of a VRPTW containing a dynamic element pertaining to customer arrivals. With this in mind, one must be prepared for the research to contain a certain amount of unsuccessful ideas. Obviously this is due to the lack of precedent upon which ideas can be based (even less than the DVRP), but this should not detract from the validity of the work. One is able to gain significant insights into a problem from their failings, just as one can from their successes.

This chapter will focus on the solving of this problem using a modified version of the ERWACS-DVRP identified in Chapter Four. Moreover, it will seek to incorporate knowledge and experience gained from the TS implementation in Chapter Five. The investigation is structured as follows:

Firstly, the traditional VRPTW is recapped. This discussion will be supplemented by details regarding the inclusion of a dynamic element, and what effect this is likely to have on the problem's complexity (including how one might approach solving it).

Secondly, given that this will be the first such implementation of this problem (as detailed in the Chapter Two), a set of benchmark datasets are required. These will allow various techniques to be assessed. There are two sensible methods of producing such datasets, both of which are discussed, prior to a decision being made. After deciding as to which method lends itself best to this thesis, the datasets are constructed. Following this, a set of basic results are produced, providing a reference point with which other adaptations can be compared.

Thirdly, consideration is given to how the algorithm can be improved, given that the VRPTW (and hence the DVRPTW) objective function differs from that of the CVRP (explained in Chapter Two and again later in this chapter). After the adaptation to the objective function is made, several components of the ERWACS-DVRP are investigated to see if one can tailor the algorithm's performance to this new objective function.

Fourthly, the idea of tailoring the algorithm's performance towards the new objective function is taken a step further, with the introduction of a two-phase approach (the two components of the new objective function are treated in a disjoint fashion). Two key methods for how the phases should be split are considered, with a decision between them being made via the standard optimisation process.

Fifthly, the improvement phase is reintroduced to see if a similar increase in performance (as to that which was achieved on the DVRP) can be achieved here. Both of the descent strategies (BI and FI) will be assessed along with consideration given to the interplay between the time spent descending and the obvious impact this will

have on the number of solutions that can be constructed in a finite time (i.e. in the time constrained variant).

The chapter concludes with a full set of results for the DVRPTW, produced using the technique that was established in the previous section. After these results have been assessed, a discussion is presented ascertaining as to whether this problem could become a more mainstream research topic.

6.1.1 The VRPTW

Developed through a series of real-life examples, the VRPTW was a very natural extension to the CVRP. It was discussed in some detail in Section 2.2.3 (where it was first introduced), along with a selection of work that cemented its status as one of the most popular variants of the VRP. To reacquaint oneself with the problem, the two key differences between the VRPTW and the CVRP are presented:

- In the most popular variant (and that which is considered in this thesis), the VRPTW (hard time-windows), each customer is assigned a time window $[e_i,l_i]$ within which its service must begin. Any solution that does not respect these additional constraints is not feasible.

- Along with the inclusion of time windows comes a change in the objective function. The CVRP (and thus in turn, the DVRP) concerned itself with the minimisation of the distance that the vehicles travel. The VRPTW (and hence the DVRPTW) is primarily concerned with minimising the number of vehicles being deployed, whilst minimising distance is retained as a secondary objective.

Consideration is now given to the dynamic variant of this problem (DVRPTW), the solution of which is the focus of this chapter.

6.1.2 The DVRPTW

This problem was not formally introduced in the Literature Review (in the way in which the VRPTW and DVRP were), because prior to this chapter, this problem did

not exist. However, with the VRPTW and the DVRP discussed in some detail, introducing this problem is a fairly simple task.

The DVRPTW simply includes time windows $[e_i,l_i]$ and the adjusted objective function of the VRPTW into the DVRP. It will continue to be solved using the timeslot solution strategy, which will create a series of VRPTW-like static problems. All other details remain the same as the DVRP, though any areas of particular interest will be discussed when they are encountered.

Note that although describing the DVRPTW is a relatively simple task when one is familiar with the DVRP and the VRPTW, solving the problem is still a complex affair. The VRPTW is NP-Hard, as such, the DVRPTW will also be NP-Hard (as it is solving a succession of static VRPTW-like problems).

When one introduces a new problem (or significantly alters an existing one), one ought to consider whether the situation being suggested is something that one might be faced with in real-life or whether it just represents an interesting mathematical problem. This will not affect the validity of the work being conducted, but it may be useful to know the environment that one might encounter this particular problem.

At its most basic level, one could concede that there are probably very few instances of DVRPTW that are currently being solved on a daily basis. However, that is not to say that they will not become more frequent, as people realise that the computational effort required to solve a problem of this nature is now available. The problem is as valid from a mathematical perspective as any other routing problem and will provide a considerable challenge to produce high quality results for.

6.2 Datasets and Initial Results

With any new problem, one is faced with the need to create benchmarks datasets, upon which, one can test the performance of various algorithms. Creating benchmarks for such a complex problem is a far from simple task. Coupling the need to retain as much realism as possible with a desire to ensure that feasible solutions exist; one is faced with two obvious ways in which to proceed:

- One can create brand new datasets. This requires all of the data (e.g. customer positions, time windows, working day etc.) to be generated for each and every dataset.

- One can adapt existing datasets (currently acting as benchmarks for similar problems) such that they become representative of this new problem.

It is this second technique that Kilby et al. (1998) utilised when creating the benchmarks for the DVRP (they were adapted from a collection of existing datasets for the CVRP). Given the obvious success of this method, it seems sensible to continue in a similar fashion. There exist multiple reasons for making this decision, some of which are now considered.

It simplifies the creation process, as it only requires extra information to be appended to the existing datasets. Furthermore, it provides an initial reference point for the quality of the solutions produced (though one could state with near certainty that the inclusion of an extra constraint will lead to a reduction in solution quality). Note that one must be careful when comparing results across two problems, as results feasible for one problem may not be feasible for the other. However, if used carefully, information regarding a solution's quality may be available; along with an understanding of how this problem compares to that which the datasets were initially created for.

With a decision made regarding the adaptation of existing benchmarks, one must identify the set of benchmarks with which to begin the process. There seems to be two problems (for which benchmark datasets exist) that are closely related to the DVRPTW that is being considered, namely the VRPTW and the DVRP.

Before considering which of these to adapt, it is first necessary to consider what constitutes a feasible dataset for the DVRPTW. Obviously they must contain all the necessary data, but further to that, the arrival times and time windows must complement each other in such a way that all the customers can always be serviced (i.e. feasible solutions will exist).

A DVRPTW dataset will be referred to as feasible if, when a customer arrives, it can always be serviced within its time window. Given that one cannot ever be certain of the location of the vehicles that have already been deployed; this certainty can only be provided by always allowing for the deployment of a further vehicle.

Consideration is now given to what is required to adjust these datasets into ones applicable for the DVRPTW:

- VRPTW: Adding Dynamism
  Adapting the existing VRPTW datasets requires arrival times to be computed for each and every customer. These times will need to be created in such a way as to ensure that feasible solutions are always achievable. This process will need to be a complex adaptation of that used by Kilby et al. (1998) and Montemanni et al. (2005) (who created DVRP datasets from classic CVRP benchmarks).

- DVRP: Adding Time Windows
  Adapting the existing DVRP datasets (as used in Chapters 3, 4 and 5) requires time windows to be computed for each and every customer. These time windows will have to be constructed in such a way as to not make the existing arrival times seem illogical.

Both of these techniques have merit, and as such deciding between them is fairly subjective. The creation of either arrival times or time windows is complex, given the existence of the other.

However, with the desire to have the DVRPTW share a similar relationship to the VRPTW, as the DVRP does to the CVRP, it seems natural to pursue the idea of adding dynamism. It should be noted that this method also benefits a simple numerical advantage, as one need only to create one piece of information for each customer (as opposed to the two for time window creation).

The VRPTW Benchmark Datasets[1]

The existing VRPTW benchmark datasets appear to have a much more mathematical/structured feel to them than the CVRP (and DVRP) datasets that were considered previously. However, this should not detract from the fact that as a collection, one should be able to find a dataset that represents the majority of real-life situations.

These benchmarks are a set of 54 problems, each of which consists of 100 customers. Some authors have bypassed the consistent problem size by running their algorithms on subsections of these datasets, but such issues are not considered in this analysis, as a more complex scenario will provide a better test environment.

These datasets were created in such a way as to represent a variety of scenarios that one might be faced with when solving a VRPTW in real-life. The customers are dispersed in one of three ways, either random, clustered or a combination of the two. The lengths of the time windows also vary, with short time windows referred to as Type 1 problems, and those with longer time windows are referred to as Type 2. The benchmark datasets cover each combination of these scenarios (r1, r2, c1, c2, rc1 and rc2). These problem scenarios (with the numbers 01-11 or 01-08 appended, depending how many are in each group) are the names given to the datasets, so one is aware of the nature of the problem that is being solved.

With such varying problem scenarios, one must consider the possibility that maybe one cannot produce an algorithm that is optimised for all of these benchmarks. The different customer dispersions were tricky to deal with in the DVRP (when the ERW was first considered) and further complicating this with the varying time window sizes may be a step to far.

Coupling this idea, with the obvious practicality issues of solving 54 problems, a decision is made to focus on the Type 2 problems. They have far greater flexibility in their possible schedules than the Type 1 problems, something that fits nicely with the ethos of dynamic routing problems.

[1] Available on Marius M. Solomon's website
http://web.cba.neu.edu/~msolomon/home.htm

The various customer dispersions will still result in a robust algorithm being produced, whilst 27 datasets still represents a significant number of benchmarks to work with. To avoid confusion, whilst still allowing one to understand the nature of the problem being solved, once a dynamic element has been included, the datasets will have the letter d appended to the beginning of their name.

As before a test set of three datasets will be chosen to optimise the algorithm on (dr201, dc201 and drc201), before a complete set of results on the time-constrained variant of the problem are produced (static results showing the number of vehicles and distance for the three test datasets are presented in Table 6.1).

Dataset	VRPTW Best
r201	4 1252.37
c201	3 591.56
rc201	4 1406.94

Table 6.1: Static Results for the Three Test Datasets that have been made Dynamic

Note that one cannot make a direct comparison between results produced in this chapter and these results as they are for a different problem (hence why they are shown in red). However, they do give an indication of the quality of results that were achievable prior to the inclusion of dynamicity and allow for some interesting observations to be made when no other results are available for comparative purposes.

6.2.1 Inclusion of Dynamic Element into Benchmark Datasets

With the decision made to add the dynamism to the existing VRPTW benchmarks, one must ensure that the datasets are amended in such a way that feasible solutions can always be achieved. By considering the technique used by Kilby et al. (1998) (the additional details added by Montemanni et al. (2005) are to remain the same i.e. $T_{ac}=T\times0.01$ and $T_{co}=T\times0.5$) one can appreciate how a dynamic dataset is constructed.

Note that although T_{co} relates purely to time, it is hoped that datasets with ≈50% of customers arriving throughout the course of the day can be produced.

The first clear difference to the process (which also gives a good indication of how closely the VRPTW and DVRPTW are linked) is that there already exist the concepts of duration and the working day. As stated previously, one only need calculate the available time, in order for the datasets to be complete. However, how these times are created is what determines whether feasible results can actually be produced.

Kilby et al. (1998) simply '*...used a uniform random distribution to generate available times for each task*' but such a straightforward technique will not be appropriate for the time window variant. One must avoid the possibility of nonsensical datasets, whereby one could be faced with the possibility of a customer arriving after its required service time has passed. Clearly another method is required.

Creating Arrival Times: A New Technique

The chosen method is based on the existing technique, and begins with the exact same process. However, all of the times produced that involve a customer arriving after its time window has closed are allocated for time reassignment. Furthermore, of those not affected by this, those who arrive after T_{co} are still deemed to have arrived the previous day, and are therefore known from the start. All other customers are then sorted into those that will be added to this group, and those that will arrive (dynamically) throughout the course of the working day.

In order to divide the remaining customers into these two groups, the latest possible arrival times are calculated for all remaining customers (using a new technique, described shortly). Once this is done, those customers that were previously assigned arrival times later than their latest possible time are also allocated for time reassignment. Those that are allocated a nonsensical latest possible arrival time (i.e. negative) are deemed impossible to allocate a sensible arrival time to, and are added to those customers known from the beginning.

New arrival times for all remaining customers are then computed using a discrete uniform distribution between 0 and their own latest possible arrival time. Again, if this new time happens to be after T_{co}, then these customers are added to those that are known from the beginning.

The complexity of this task is in the identification of the latest possible arrival time (denoted ϕ_i). Initially, one might assume that by setting $\phi_i=l_i$, one is able to ensure that a customer has arrived before the end of its time window. Strictly, this is true, but if an arrival time equal to l_i were to be selected, it would require a vehicle to be at the customer's location the moment the customer is known. Obviously this is unlikely, and as such, an earlier upper limit (for ϕ_i) must be established.

There are three components that are to be discussed, two of which are critical (i.e. without them a feasible dataset will not always be produced) and one that is entirely subjective but is included to make the solving of these problems more interesting.

Starting with the non critical component, these issues are now considered:

- Solution Flexibility (Non Critical Component)
  As stated earlier a potential starting point for the setting of an upper bound (ϕ_i) is the latest point at which a customer can be serviced (l_i). However, if decisions were to be based on this, one can envisage a situation whereby a customer would only be guaranteed to be reachable at the latest possible moment. This does not allow for any flexibility in the results (one of the reasons for only using the Type 2 datasets) and therefore the decision was made to allow all customers to be reachable by the beginning of their time window (e_i).

With an initial value for ϕ_i established, one must reduce this value (to compensate for travel times and the timeslot strategy) in order to prevent the possibility of infeasible datasets being constructed. It should be noted that the first of these components is a variable based on each particular customer, whereby the other is a constant, known in the problem definition:

- Travel Time (Critical Variable Component)

  If a customer were to become known at the earliest point at which it can be serviced (e_i), one would be unlikely to be able to travel to that customer before (l_i), thus missing the time window. Obviously this is not a workable system, so some notion of travel time must be introduced. To ensure a new vehicle can always be dispatched in time, the decision is made to subtract the distance from that customer to the depot i.e. $c_{0,i}$ giving a value of $\phi_i=e_i-c_{0,i}$.

- Timeslot Technique (Critical Constant Component)

  The timeslot technique allows the global problem to be partitioned into a series of smaller sub-problems each of which is solved subsequently. Unfortunately, a by-product of this system is that a customer is not considered until the end of the existing timeslot; upon which the schedule for the next timeslot becomes fixed (plus T_{ac}). Therefore a customer must be known at least two timeslots and the advanced commitment time before a vehicle would need to be deployed, giving $\phi_i=e_i-c_{0,i}-(2\times T_{ts})-T_{ac}$.

When implementing this technique it is clear that it is impossible to ensure that exactly 50% of customers arrive during the working day. However, due to the random component in the dataset construction technique, one can often continue to reproduce datasets, until ≈50% of customers are not known in advance (as desired, in this chapter). Note that any datasets that cannot be produced with >40% dynamic customers (due to their customer's locations and values of e_i) will be produced to be as dynamic as possible and will have a star appended to their name.

An important characteristic of the rather unusual dataset creation technique is that the customers will not arrive in a uniform manner, as per the DVRP benchmark datasets. This is due to the values of e_i chosen when these datasets were developed for the VRPTW; if one wants to preserve feasibility and high levels of dynamicity (whilst solving using the timeslot solution strategy), an uneven arrival distribution must be accepted.

It is important (in terms of maximising the distinction between the DVRPTW and the VRPTW) that there is some form of spread amongst the 13 timeslots. The specific distribution of arrivals for the three test datasets are shown in Figure 6.1.

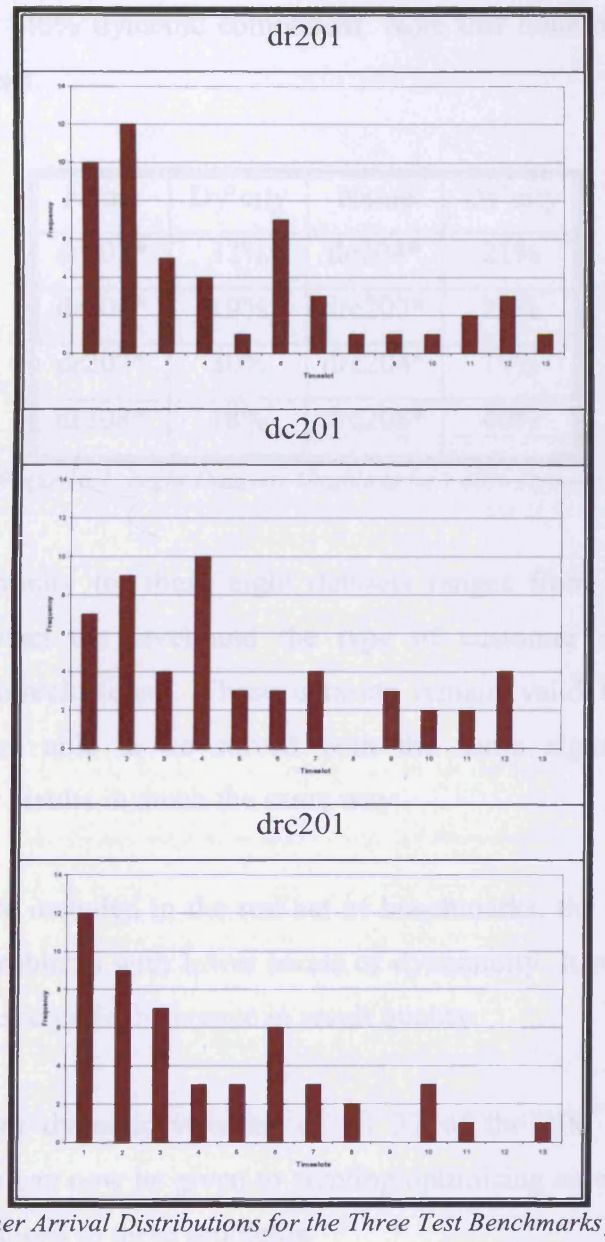

Figure 6.1: Customer Arrival Distributions for the Three Test Benchmarks for the DVRPTW[2].

As one would expect, there is a concentration of customers in the earlier timeslots (unavoidable due to the inclusion of time windows), but importantly, there are still customers arriving throughout the later timeslots. Clearly there are not as many as for

[2] Note that these graphs relate specifically to the arrival distributions, hence beginning at ts_1, not ts_0.

the DVRP, but they will still have the desired impact with regards to making it a more difficult problem (when compared to the non-dynamic variant, the VRPTW).

As expected, some of the datasets (as seen in Figure 6.2) were unable to be constructed with a >40% dynamic component. Note that none of these datasets are included in the test set.

Name	Dy'city	Name	Dy'city
dr203*	32%	dc204*	21%
dr204*	19%	drc203*	29%
dr207*	30%	drc204*	19%
dr208*	18%	drc208*	40%

Figure 6.2: Eight Datasets Unable to be >40% Dynamic

The level of dynamicity for these eight datasets ranges from 19%-40%, with no apparent link between the level and the type of customer dispersion (random, clustered and random/clustered). These datasets remain valid benchmarks for the DVRPTW; they are able to be solved with the same algorithm, and will be comparable to other results in much the same way.

As none of these are included in the test set of benchmarks, the final algorithm will not be tailored to problems with lower levels of dynamicity. It will be interesting to see if there is any discernible difference in result quality.

With the creation of dynamic versions of all 27 of the 100 customer problems completed, attention can now be given to creating/optimising an algorithm capable of producing good solutions to these problems.

6.2.3 Creating Feasible Solutions

This chapter intends to modify the ERWACS-DVRP (as created in Chapter Four) into a closely related ERWACS-DVRPTW. This is a complex procedure that begins with making the necessary adaptations to the existing algorithm in order to produce feasible solutions. Upon completion of this, one can then begin to consider the quality

of the solutions produced (see final component of this section and Sections 6.3 and 6.4).

Please note that an adjustment has been made to the ERWACS-DVRP, based on the improvements found in Chapter Five. The descent algorithm no longer operates via the FI descent policy, favouring the BI methodology inherent in the TS algorithm. Moreover, the best performing of the neighbourhoods analysed in Chapter Five (the CROSS Neighbourhood) is utilised. This neighbourhood was specifically designed for the VRPTW (as explained in Section 2.2.3); it retains route orientation which greatly improves the chance of moves being feasible (with respect to the time window constraints).

Incorporating Time Windows into the ERWACS-DVRP

The algorithm cannot simply be run in its existing form, as it will return solutions that do not respect the need for customers to be serviced within their time window. This obviously needs to be amended and it is clear that there are two different places in which one must make adjustments, namely the construction and improvement phases:

- Construction

  When one is making an RPR selection, one is making an informed choice from the predefined set of customers, denoted F^k. Obviously it is important that feasible customers are selected (with the obvious exception of those that declare a vehicle as a TTV), and as a result of this, it seems sensible to place a further restriction upon the set F^k; those customers whose time window cannot be reached (by that specific vehicle) are not included.

- Improvement

  When each descent move is considered, the current system checks its feasibility with regards to capacity and the length of the working day. One is able to consider the impact that the move will have on the other customers affected[3], simply by considering the residual impact to capacity and time.

[3] The customers affected are those that follow customers that have been moved.

However, with the introduction of the time window constraints, many extra calculations are going to be required when each move is considered. Every customer affected by the move needs to be considered individually, checking that it adheres to its strict time window constraints $[l_i,e_i]$. Obviously these extra calculations will significantly increase the computational effort required to analyse the same neighbourhood, but unfortunately this is unavoidable.

With these changes incorporated, some results, which will enable one to get a feel for the problem, are produced.

Initial Results

If the algorithm were to be run in its existing form, it is clear that feasible solutions would be produced. As to what the quality of such solutions is likely to be, it would be fairly safe to assume that they would be poor (although with no existing results, this is purely speculative). One of the reasons why performance is likely to be so poor is considered immediately.

There is a further component of the ERWACS-DVRP (and hence this adapted ERWACS-DVRPTW), which although does not affect the feasibility of a solution, is completely nonsensical with regards to the inclusion of time windows.

The current local and global updates rules (which control the diversification and intensification, within a particular timeslot) are operated in a symmetrical manner. The existing update rules in the ERWACS-DVRP update both τ_{ij} and τ_{ji} regardless of whether it is the arc (i,j) or (j,i) that was chosen via the RPR (in the case of the local updating) or belongs to the best solution produced so far (in the case of the global updating). This is because of the fact that the DVRP is a symmetrical problem (i.e. the distance matrix is symmetrical) and thereby solutions can be traversed in either direction with no change to the distance.

However, once the time windows are introduced, although the distances remain symmetrical, a completed route must be traversed in a particular order. This a feature of the VRPTW (and hence the DVRPTW) whereby the arc (i,j) has different attributes

to that of (j,i) and they should not be grouped together in the way in which they were previously.

Table 6.2 shows the difference between using the symmetrical update rules (as per the ERWACS-DVRP) and the non-symmetrical ones, whereby only the specific arcs chosen (or belonging to the best solution so far produced) have their trail values adjusted.

Update		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
Sym	Min	17	1446.80	**3**	**591.56**	17	1597.42
	Ave	18.3	1507.01	4.4	665.32	18.7	1666.08
	Max	20	1570.70	6	733.39	22	1791.16
Non-Sym	Min	**11**	**1330.10**	**3**	**591.56**	**10**	**1373.23**
	Ave	**12.4**	**1342.54**	**3.8**	**630.55**	**11.4**	**1423.96**
	Max	**14**	**1388.76**	**5**	**693.11**	**14**	**1491.14**

Table 6.2: Analysing the use of Symmetrical and Non-Symmetrical Update Rules.

With the number of vehicles being the primary objective it is clear that the algorithm has behaved as expected, with a notable improvement in the solution quality when the non-symmetrical update rules are used. Non-symmetrical update rules will be used for the remainder of the work in this chapter.

Interestingly, one can begin to see how the result quality compares to those that are achievable on the static variant of the problem (see Table 6.1). The solution quality is markedly worse, with significant increases in the number of vehicles being deployed in datasets dr201 and drc201. The quality of the solutions produced for dc201 are of a higher standard, though this may be because clustered problems are likely to be easier to divide the customers between the vehicles. The changes in distance are somewhat minimal, although they are also unimportant given the significant difference in the vehicle deployment.

As this algorithm still retains a number of features specifically tailored to the DVRP, it is sensible to give consideration to how the algorithm could be adapted so as to be

better suited to this new problem. Clearly, accommodating the change in the objective function is going to be central to these modifications, and is thus considered next.

6.3 Minimising the Number of Vehicles

The adjustment to the objective function is crucial to the DVRPTW, since solutions will be primarily judged on the number of vehicles scheduled for deployment (with distance being the secondary objective). In order to compensate for this alteration there are various adjustments that can be made to the algorithm. This section concerns itself with three key components, now described:

Firstly, the current objective function does not account for this change in the primary objective. Clearly the comparison of two solutions on anything other than what they are to be finally judged on is counter-productive, thus a new objective function is required.

Secondly, with an obvious nod to the workings in Chapter Four, the vehicle reduction techniques are revisited. These were initially introduced to encourage the use of fewer vehicles; clearly with the new primary objective, such encouragement is more prevalent than it was before. Note that the current algorithm already includes a low level CU factor of $\lambda=1$, which was introduced when these parameters were first assessed.

Finally, the current definition of the visibility is investigated. It is currently related to the distance between two customers, encouraging those close together to be visited in quick succession. However, with the introduction of time windows, such decisions could be counter-productive and as such an in-depth investigation into potential visibility definitions (which are designed to account for the inclusion of the time windows) is conducted.

6.3.1 A New Primary Concern

As has been stated previously, the DVRPTW is primarily concerned with minimising the number of vehicles. Given that there is often a correlation between less vehicles and lower distances, one could rely on this relationship to try to minimise the number

of vehicles. However, a much more sensible technique would be to formalise the relationship between the two objectives, such that the objective function reflects this.

An alternative objective function that accounts for this change in objective is now considered. The new objective function is defined as $\xi v + Distance$, where ξ is a new parameter that one can set to an arbitrarily high value, such that it is dominant when comparing solutions involving different numbers of vehicles.

Solutions with an equal number of vehicles will still be distinguished according to distance. The results comparing ξ=0 with ξ=5000 are shown in Table 6.3.

ξ		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
0	Min	11	1330.10	**3**	**591.56**	10	1373.23
	Ave	12.4	1342.54	3.8	630.55	11.4	1423.96
	Max	14	1388.76	5	693.11	14	1491.14
5000	Min	**8**	**1333.46**	**3**	**591.56**	**8**	**1375.33**
	Ave	**9.3**	**1376.52**	**3.4**	**609.29**	**9.2**	**1443.36**
	Max	**11**	**1463.13**	**4**	**646.89**	**11**	**1481.88**

Table 6.3: Analysing the Impact of ξ in ERWACS-DVRPTW

As one would expect, by including the actual number of vehicles into the objective function, a considerable improvement has been made with regards to the number of vehicles deployed. Interestingly, the improvement is not seen in the distance (across all three datasets), suggesting the correlation between lower numbers of vehicles and lower distances may not be as prevalent in the DVRPTW as it was in the DVRP.

Given the significant level of improvement achieved via this new objective function, all further analysis within this chapter will include this modification.

With the new objective function (specifically tailored to the problem) improving the performance, one must consider whether any other features of the ERWACS-DVRP belong in the algorithm for the DVRPTW. The most obvious remaining area to

consider is the criterion for tentative schedule completion that was introduced in Chapter Four, namely the TTV.

The TTV was introduced for two reasons: to prevent the depot being selected soon after a vehicle departed and to lessen the computational effort required when building a solution. However, although these reasons are still valid, one needs to consider whether any technique that readily allows a vehicle to be removed from consideration when customers can still be allocated is a sensible one for the DVRPTW.

With this in mind, it seems sensible to consider whether it is worth removing the TTV when solving the DVRPTW. Referring to the criterion employed in the ACS-DVRP (Section 4.4), one can see that only feasible customers (and importantly the depot) were included in the set F^k. The criterion for tentative schedule completion came through the selection of the depot, so whilst a vehicle may be removed while customers can still be allocated, it may not be so quick to do so.

The two methods are both assessed and Table 6.4 shows the results of this analysis.

Criterion		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
TTV	Min	**8**	1333.46	**3**	**591.56**	**8**	**1375.33**
	Ave	**9.3**	**1376.52**	**3.4**	**609.29**	**9.2**	**1443.36**
	Max	11	1463.13	**4**	**646.89**	**11**	**1481.88**
No TTV	Min	**8**	**1316.68**	**3**	**591.56**	9	1381.71
	Ave	9.5	1362.28	3.7	636.16	9.6	1466.71
	Max	**10**	**1427.14**	5	691.93	**11**	1498.94

Table 6.4: Analysing the Impact of Removing the TTV Tentative Schedule Completion Criterion.

What is immediately apparent is that the TTV is certainly not having a negative effect; result quality is higher when it is included. Given that this is being assessed on the non-time constrained variant of the problem, one cannot associate the improvement to an increase in the number of cycles that are being completed (though it is likely that this will further widen the difference in performance).

It is likely that the algorithm has been optimised with the TTV component included, and as such, the parameters are working together in such a way that they complement each other. The inclusion of the TTV criterion for tentative schedule criterion is not a simple decision (hence why it was considered so early in the analysis in Chapter Four, and again here); it significantly alters the way in which the algorithm operates.

With no obvious benefit attached to removing the TTV and reverting back to the system employed by Montemanni et al. (2005), the criterion for tentative schedule completion is not altered.

6.3.2 Vehicle Reduction Techniques

These were first encountered in Chapter Four, where they were presented as a response to the apparent correlation between distance and the number of vehicles deployed (in the DVRP). These methods were required to be flexible; merely encouraging the use of fewer vehicles as the objective function was entirely based on distance.

Two distinct methods (Capacity Utilization and Decreased Depot Visibility) were presented, with both significantly improving the algorithms performance. The CU appeared to outperform the less computationally expensive DDV, and was thus selected.

The CU factor remained in the ERWACS-DVRP and has therefore been included in the ERWACS-DVRPTW results already presented in this chapter. However, given the rather conservative parameter setting (λ=1), and the new primary objective, it seems sensible to re-optimise it for this problem. Furthermore, given that problems involving time windows are more time consuming than those that do not, the performance of the considerably less computationally expensive DDV is also to be reconsidered.

For all analysis, until signified otherwise, the improvement phase (descent) has been removed. This allows the results to be indicative of the construction procedure, in a similar way to their assessment in Chapter Four.

Capacity Utilisation

First described in Section 4.3.3 (and implemented in 4.6.2), the Capacity Utilisation factor encouraged the selection of customers that better filled the vehicles. The current setting (optimised on DVRP) is λ=1, and has been included in the earlier implementations on the DVRPTW.

This section concerns itself with identifying if the performance can be improved by further increasing the greediness in the construction procedure (via the RPR) towards those customers that will better fill the vehicles. Implementation details are exactly the same as described previously. The results can be seen in Table 6.5.

λ		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
1	Min	11	1330.10	**3**	**591.56**	10	1373.23
	Ave	12.4	1342.54	3.8	630.55	11.4	1423.96
	Max	14	1388.76	5	693.11	14	1491.14
2	Min	7	1356.06	**3**	**591.56**	**6**	**1402.13**
	Ave	7.7	1365.14	**3.2**	**602.80**	7.4	1471.55
	Max	10	1346.53	**4**	656.31	8	1557.51
3	Min	6	1346.90	**3**	**591.56**	**6**	1536.12
	Ave	6.6	1405.56	3.6	632.01	7.2	1523.88
	Max	8	1388.27	**4**	724.99	8	1609.24
4	Min	6	1362.57	**3**	594.32	**6**	1524.66
	Ave	**6.2**	**1417.70**	3.6	625.91	7.4	1555.32
	Max	**7**	**1490.78**	5	677.92	9	1593.25
5	Min	**5**	**1483.62**	**3**	597.88	**6**	1518.80
	Ave	**6.2**	1462.21	3.4	614.26	**6.7**	**1556.75**
	Max	**7**	1502.09	**4**	**649.65**	**7**	**1713.46**

Table 6.5: Analysing Various Values of λ in ERWACS-DVRPTW

There appears to be an obvious trend, with vehicle numbers decreasing as λ increases; a decrease of 60.64% in the average number of vehicles deployed (when averaged across all three test datasets). Clearly this is a useful tool in reducing the number of vehicles and should be retained in some capacity, but one must also remember not to excessively intensify the search; a clear possibility when the Capacity Utilisation bias parameter is set particularly high.

To allow the DDV the best possibility of performing well, the CU factor is reduced back to 1 for the subsequent analysis. On completion of this a decision can be made as to whether to pursue with either in the longer term.

Decreased Depot Visibility

The idea of DDV was described and implemented in sections 4.33 and 4.6.2 respectively. Designed as a less computationally expensive and very simplistic method for encouraging the deployment of fewer vehicles, it was rejected in favour of the CU factor due to the result quality not being as competitive. It should also be noted that the decrease in the number of vehicles was not as significant as for the CU factor (even though this was not an objective in the DVRP).

The parameter θ, which controlled the level of visibility decrease, was optimised at θ=0.5. This halved the a priori desirability of selecting the depot via the RPR. As the number of vehicles deployed is not the primary objective, it seems prudent to analyse smaller values of θ, thus reducing its desirability further. The results can be seen in Table 6.6.

θ		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
1	Min	11	1330.10	**3**	**591.56**	10	1373.23
	Ave	12.4	1342.54	3.8	630.55	11.4	1423.96
	Max	14	1388.76	5	693.11	14	1491.14
0.5	Min	7	1318.31	**3**	**591.56**	**7**	1407.95
	Ave	7.9	1334.70	3.6	631.05	7.7	1483.91
	Max	**8**	1435.83	5	746.49	**8**	1542.66
0.33	Min	7	1319.46	**3**	**591.56**	**7**	1490.47
	Ave	7.8	1349.69	3.7	630.09	7.7	1506.54
	Max	**8**	**1416.38**	5	689.34	**8**	1558.64
0.25	Min	**6**	1436.23	**3**	**591.56**	**7**	**1394.84**
	Ave	**7.2**	**1364.24**	3.5	622.85	**7.2**	**1473.34**
	Max	9	1357.64	**4**	690.45	**8**	**1490.21**
0.2	Min	**6**	**1322.30**	**3**	**591.56**	**7**	1482.32
	Ave	**7.2**	1374.09	**3.1**	**601.06**	7.5	1513.25
	Max	**8**	1438.10	**4**	**654.94**	**8**	1577.40

Table 6.6: Analysing Various Values of θ in ERWACS-DVRPTW

There is a definite trend, with vehicle numbers decreasing as θ is reduced, though the performance levels are not comparable to that which was achieved via the high increases in the CU bias parameter. It is quite possible that the selection process is acting too greedily, never selecting the depot, even when it would represent the best choice.

If one were not to have the required time or knowledge to implement the CU factor then the DDV would be a useful tool, but on these results it would be foolhardy to argue for its inclusion. CU (λ=5) seemingly offers the best level of performance of any of the vehicle reduction techniques.

However, there also exists another obvious method of altering the number of vehicles that is to be investigated; altering the visibility definition. The CU factor will remain at λ=1 for the analysis of this technique; though it will be increased if this analysis does not result in an even greater level of improvement.

6.3.3 Problem Specific Visibility

To further adapt the way in which the customers are chosen, one can alter the component parts of the RPR. Obviously CU and DDV do this, but there is also freedom to incorporate other factors, or further adjust those that already exist.

The inclusion of further components can be discounted on account of the already overtly complex and time consuming nature of the algorithm in its current form. With alterations to the existing components representing the most practical route forward; the definition of the visibility stands out as a component that clearly does not compensate for the existence of time windows.

The current visibility ($\eta_{ij} = 1/d_{ij}$), is that which was used by Montemanni et al. (2005) for solving the DVRP. This visibility can be described as the classic definition for routing based problems, with it appearing in the majority of ACO routing implementations (including the very early AS for the TSP workings of Dorigo et al. (1996)).

However, within the MACS-VRPTW, Gambardella et al. (1999) were aware that this visibility does not lend itself well to problems that need to accommodate time windows. Instead they proposed a new definition (which will be investigated shortly). However, prior to this, the reason for the existing visibility's failure to aptly deal with time windows is considered.

The Problem with a Distance Based Visibility

When selecting the next customer to visit, the existing visibility favours customers that are geographically close. This was an entirely sensible definition for the TSP as one is purely concerned with the minimisation of distance. It was able to work well for the CVRP (and hence DVRP) but often relied on other factors (descent, VRT) to ensure that solutions were not being produced in an excessively large number of vehicles (although the objective was still purely distance based).

However, the need to reduce vehicles is further enhanced, with it now constituting the primary objective. Remedies such as the CU and DDV may not be sufficient. The problem being solved differs from the DVRP with regards to when a vehicle can begin to service a customer; one must wait for the time window to open (if the vehicle arrives early). The distance between two customers no longer constitutes the time between finishing servicing one customer and beginning the next.

It is when discussing ideas such as this, that the complexities of the time windows come to prominence. They have the potential to delay the start of a customer's service to such an extent that one must be particularly careful when committing a customer to a vehicle.

The idea of delay is central to producing a solution in as few vehicles as possible, as it often constitutes wasted time. An example of when a distance-based choice would clearly represent an inappropriate choice can be seen in Figure 6.3.

The customers that are known in advance (i.e. those that arrived yesterday) are displayed, and a choice must be made from them, as to which should be the first customer to visit.

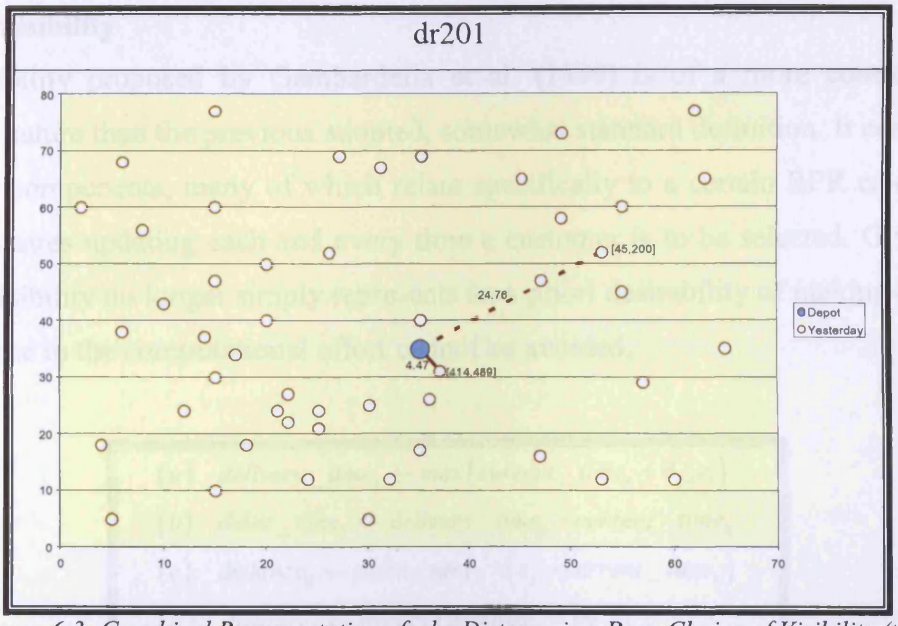

Figure 6.3: Graphical Representation of why Distance is a Poor Choice of Visibility (ts_0)

Utilising the existing visibility definition the customer chosen (as highlighted by the bold line) is one close to the depot, as expected. However, due to the lack of consideration given to the customer's time window, that vehicle is required to wait until t=414 until it can begin servicing that customer (a delay of 409.53).

A better choice of customer would be one whose time window opens close to the time at which the said vehicle would arrive (thus reducing the amount of delay). An example of this is shown (indicated by the dashed line) in the same figure. If this customer were to be scheduled first, servicing could begin at t=45, resulting in a delay of only 20.24.

Clearly the existing visibility does not represent an appropriate choice (and nor should it, given it was developed for another problem), and as such, one must give consideration to a method specifically designed to accommodate the existence of time windows. This is exactly the problem that was faced by Gambardella et al. (1999), which given the competitiveness of their solutions (on well known VRPTW benchmarks), they were clearly successful in solving. Consideration is now given to their proposed visibility.

A New Visibility

The visibility proposed by Gambardella et al. (1999) is of a more complex and variable nature than the previous adopted, somewhat standard definition. It consists of multiple components, many of which relate specifically to a certain RPR calculation i.e. it requires updating each and every time a customer is to be selected. Given that such a visibility no longer simply represents an a priori desirability of making a move, an increase in the computational effort cannot be avoided.

$$
\begin{aligned}
&(a)\quad delivery\_time_j \leftarrow max\left(current\_time_k + d_{ij}, e_i\right)\\
&(b)\quad delta\_time_{ij} \leftarrow delivery\_time_j - current\_time_k\\
&(c)\quad distance_{ij} \leftarrow delta\_time_{ij} \times \left(e_j - current\_time_k\right)\\
&(d)\quad distance_{ij} \leftarrow \max\left(1, \left(distance_{ij} - IN_j\right)\right)\\
&(e)\quad \eta_{ij} \leftarrow 1/distance_{ij}
\end{aligned}
$$

Figure 6.4: The Visibility Function as Defined in the MACS-VRPTW.

The visibility is calculated for each customer pair, by progressing through the steps a to e (see Figure 6.4). Each of the steps is going to be considered in turn, with a description of why this particular step is required. Although complex in appearance this visibility is fairly intuitive (with the possible exception of step d, though this will be discussed in due course).

As shown in Figure 6.3, the problem with the current visibility is that the distance no longer represents the time between a service finishing and the next beginning. In order to compute this one must progress though Step a and Step b.

Step a identifies the times at which the time window opens and the vehicle would arrive if it were to travel there; whichever is the later represents the earliest that the customer could be serviced (by that vehicle). Step b simply subtracts the current time, so that one is just left with the increase, this results in a figure that behaves in much the same way as the distance did in the ERWACS-DVRP.

This calculation represents a sensible value (to take a reciprocal of), were one requiring a basic visibility for the VRPTW (and hence DVRPTW). Given that one is

required to recalculate the visibility each time an RPR selection is to be made, it seems sensible to assess this simplistic visibility to see if it can perform to a similar standard as the more complex method of Gambardella et al. (1999).

Step c takes the current calculated value (not the reciprocal) and includes a sense of urgency into those customers whose windows are due to open shortly. One must schedule these customers, and logically the earlier they are scheduled the greater the choice one will have over their placement. The smaller the value of $e_j - current\_time_i$, the smaller the multiplier; this will result in a larger value when the reciprocal is taken (Step e). The reciprocal of this value represents the other visibility definition that is to be assessed. Note that Step d is not required in this algorithm, the reasons for which are now discussed.

The algorithm of Gambardella et al. (1999) was a two colony composite procedure that allowed for the construction of infeasible solutions in both colonies. Within this algorithm, there was the distinct possibility that the search would stagnate on an infeasible solution, and as such, they introduced a further diversification measure. They introduced the variable IN_j, which counted the number of times each customer had not been selected for inclusion in a solution. Step d was to subtract this figure, from the result of Step c, thus favouring customers that were frequently being excluded. As the ERWACS-DVRPTW cannot build infeasible solutions, including this parameter into the visibility definition was felt unnecessary.

In summary, two different visibility definitions based on the working of Gambardella et al. (1999) are assessed:

Max: $(a) \rightarrow (b) \rightarrow (e)$[4]

Adjusted: $(a) \rightarrow (b) \rightarrow (c) \rightarrow (e)$

These should provide knowledge pertaining to the interplay of the various components as well as giving a broader understanding as to why this specific visibility was chosen. The results can be seen in Table 6.7.

[4] Reciprocal is taken of $delta\_time_{ij}$ not $distance_{ij}$

Visibility		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
Distance	Min	11	1330.10	**3**	**591.56**	10	1373.23
	Ave	12.4	1342.54	3.8	630.55	11.4	1423.96
	Max	14	1388.76	5	693.11	14	1491.14
Max	Min	**4**	**1395.46**	**3**	**591.56**	**4**	1604.17
	Ave	**4**	**1428.89**	**3**	**591.56**	4.6	1589.60
	Max	**4**	1463.11	**3**	**591.56**	**5**	**1612.39**
Adjusted	Min	**4**	1418.53	**3**	**591.56**	**4**	**1552.71**
	Ave	**4**	1441.28	**3**	**591.56**	**4.3**	**1615.69**
	Max	**4**	**1457.94**	**3**	**591.56**	**5**	1694.56

Table 6.7: Analysing Various Visibility Definitions in ERWACS-DVRPTW

The quality of these results is superior to any of the others that have been produced for these datasets thus far. They negate the need to make use of the higher value of λ, as identified in the analysis of the VRTs (Section 6.3.2).

The theory behind the new visibility definitions is such that one can understand from its description why it has had such a significant impact on the quality of solutions produced. The new visibility represents a big stride in the transition of the algorithm into a true DVRPTW algorithm (as opposed to a converted DVRP algorithm).

With regards to analysing the two newly proposed visibility definitions, one would favour the use of the Adjusted visibility definition. It significantly outperforms the existing definition, while appearing marginally better than the Max definition.

The lack of variability in the vehicle deployment for datasets dr201 and dc201 demonstrates that the algorithm is suitable for solving problems with various customer dispersions. Though the performance can still be improved in the composite case (drc201), it does suggest that one can produce a suitably robust algorithm.

With regards to distance, these results are not noticeably different to those achieved via the increased CU factor. However, given the secondary nature of this objective, it is somewhat meaningless making such a comparison when the vehicles deployed are markedly different. The decrease in the variability of the distance is interesting, often

cited (in Chapters Three, Four and Five) as a problem with the DVRP; it seems to be slightly less of a concern here. This may be due to the fact that the DVRPTW is a slightly less variable problem (fewer results exist, due to the extra constraints), it could also be down to the fact that there is greater consistency in the number of vehicles deployed (as this was not controlled in the DVRP, the scope for variability would be greater).

During the analysis of these results some interesting behaviour (with regards to where one should pursue this research) was observed.

Observation

Although the algorithm was being run with t=15000 (as in Chapter Four), the algorithm locates its best solution (with regards to vehicle deployment) very early in the search process. Although this is not a problem if that solution is of a high quality, if it is poor (i.e. drc201 when 5 vehicles were deployed), the algorithm settles for building solutions with that number of vehicles and does not aggressively try to reduce it further. Note that distance does continue to benefit from the high number of cycles, with improvements located throughout the search. However, with this being a secondary objective it does seem that one could make better use of the time available.

In order to concentrate on the vehicle deployment (without creating a bespoke algorithm) it seems sensible to adopt the policy utilised in MACS-DVRP, which makes use of multiple ant colonies.

6.4 Two-Phase Approach

With the number of vehicles deployed being significantly reduced through the introduction of a new visibility definition, it is not unreasonable to think that any further improvements will not come easily. The vehicle deployment figures are optimal[5] for two of the three datasets (across all ten runs), with the other achieving optimality on seven out of ten runs. Obviously further improvements can be made to drc201 (and will be considered), but one must also give the necessary consideration to the secondary objective (distance).

[5] Regarding the static variants of the problem (VRPTW).

If one were to consider the secondary objective, it is clear that the obvious technique for reducing the distance is the reintroduction of the descent algorithm. However, were the descent phase to be introduced now, it would certainly constitute a computationally expensive procedure.

Within the time-constrained variant, such a procedure would significantly reduce the number of cycles that would be completed, which could easily impact upon the number of vehicles deployed. Therefore, it seems sensible to try to get the best possible performance (for both the primary and secondary objectives) from the ant construction phase.

In order to achieve this, a new approach is considered, whereby a second ant colony is introduced. The use of multiple colonies to solve the VRPTW was introduced in MACS-VRPTW by Gambardella et al. (1999). It allows one to consider the problem as the composite of two phases, each with differing objectives. One phase (P_0) focuses on minimising the number of vehicles deployed, the other (P_1) minimises the distance (subject to no increase in the number of vehicles deployed in the solution of P_0).

Two different methods of this two phase approach are considered: The first, referred to as consecutive, optimises P_0 before consideration is given to P_1. The second, referred to as interlaced, considers P_0 and P_1 alternately.

6.4.1 Implementation Details (Consistent with both Techniques)

Before specifics regarding the different approaches to implementing the two-phase composite procedures are discussed, there are certain practical and theoretical characteristics that remain constant in both implementations. To prevent the need for repetition, these details are now presented.

Details for both P_0 and P_1

Trails are not going to be reinitiated if any improvement is found in the number of vehicles needed for a solution. This is in contrast to Gambardella et al. (1999) who treat the identification of a lower number of vehicles as cause for the problem to be considered afresh. This technique has the potential to discard considerable amounts of

information that has been learnt, and has therefore been excluded. The existing trail matrices will remain, benefiting from their current values, which obviously were such that they managed to locate an improvement.

Details for P_0

After the first solution has been constructed, a variable (A_0) is set equal to the number of vehicles that would need to be deployed were that solution to be chosen. Upon commencement of the next P_0 cycle, an artificial limit of A_0-1 vehicles is placed on the ants new build attempt. If all the customers can be allocated to A_0-1 vehicles then A_0=A_0-1 and the process begins again. If all the customers cannot be allocated, then the ants will try again in the next cycle. This process continues until some predetermined transition criterion is met.

Note that the number of unscheduled customers in P_0 for each solution will be stored as the variable U_0^k, while the number of unscheduled customers in the best solution so far will be stored as U_0^{bs}.

Consideration is now given to the three-update rules and how they will work theoretically. Actual implementation details differ between the two tested techniques and will thus be described when required:

- Local Updating

  Designed as a diversification procedure in ACS implementations, it continues to do just that. If a solution is produced in A_0-1 vehicles, then further solutions are not required in that number of vehicles, hence the need to diversify. If a solution is not produced in A_0-1 vehicles, then the system will operate as per a typical ACS algorithm, diversifying and then relying on the global update rule for intensification.

- Global Updating

  The purpose of the global update is to intensify the search around the best solution currently identified; however the previous notion of best (minimal distance) no longer seems applicable. In P_0, one is trying to construct solutions

in as few vehicles as possible, thus the best solution is that which is closest to scheduling all customers in A_0-1 vehicles (conversely this can be thought of as minimising the number of unscheduled customers[6] i.e. U_0). A solution need not be feasible to constitute the best solution in P_0.

- Transition Updating

  When the existing timeslot has concluded and the next begins it has been shown to be beneficial to retain some proportion (defined by γ_t) of the previous timeslot's learning. This still represents a prudent ideology, as it will help the customers to be scheduled into fewer vehicles when P_0 begins again.

Details for P_1

When the algorithm enters P_1 it reverts back to the DVRP-like objective of minimising distance. However, this can now be done with the added knowledge learnt in P_0 i.e. the value of A_0. Any attempts to reduce distance are made subject to the solution requiring no greater than A_0 vehicles (fewer would be fine, though unlikely). As before, this process continues until some predetermined criterion is met.

Consideration is given to the theory behind the three update rules roles in P_1:

- Local Updating

  The local update rule continues to act as a diversification procedure, though its benefits can be divided into two sub-groups, depending on the success of a particular ant constructing a solution in A_0 vehicles. If the construction process is successful (i.e. a feasible solution is produced in A_0 vehicles) then the original theory behind the local updating remains, and movement in the search-space is encouraged. If the construction process is unsuccessful, then one obviously wants to move away from that solution (solutions which require more than A_0 vehicles are not beneficial) thus diversification seems prudent.

[6] This can be 0 if a feasible solution in A_0-1 vehicles was constructed during the cycle.

- Global Updating

  The need to intensify the search is particularly prevalent in P_1 as it serves two purposes. Not only does it act in the traditional way of helping to minimise the distance by focusing the search on the successful regions of the search space, it encourages the construction to use the required A_0 vehicles (as the best solution will always have). No infeasible solution will ever constitute the best solution in P_1.

- Transition Updating

  As in P_0, this remains a useful tool and does not require adaptation in the two-phase approach. Solutions that resulted in low distances in previous timeslots remain likely to offer benefits in subsequent phases.

With the details regarding the theoretical workings of the update rules within the vehicle minimisation phase (P_0) and the distance minimisation phase (P_1) presented, attention is now given to the implementation details. As mentioned previously, two techniques of using the two-phase approach are considered (consecutive and interlaced). The first of these is now considered with the other following on directly. Upon the successful implementation of (and possible improvements to) the two techniques, a comparison will be presented.

6.4.2 Consecutive Phases

As the name suggests, the phases are completed in a consecutive manner i.e. P_0 is run to completion, after which P_1 begins. The final result returned is the best achieved in P_1, and will use the number of vehicles from P_0 (or less). It should be noted that the best feasible solution in A_0 vehicles from P_0 will be used as the best solution, should no feasible solutions be produced in the first cycle of P_1.

Initial analysis suggested that the ants require more time to reduce the distance than they do the number of vehicles. Such a conclusion seems sensible, especially since the construction of a solution in one vehicle less than has previously been managed can often be impossible.

Consider the situation when one has finally managed to construct a solution in A_0-1 vehicles. If one were to try to construct in one less again, there is the potential for a large number of unscheduled customers. Obviously, sometimes such a build is just not possible and all subsequent work in P_0 is fairly redundant (though one could argue its worth when it comes to the transition updating).

Preliminary testing showed that a 30/70 split (in cycles) between P_0 and P_1 performed best, and as such all analysis within the consecutive phase work involving two colonies will utilise this.

As the description regarding the update rules was purely theoretical, a few details are required prior to implementation. The actual formulae do not differ for the local and transition update rules (regardless of phase), however changes are involved in the global update formula (see Equations 6.1 and 6.2)[7].

The global update rules utilised are as follows:

$$P_0 : (1-\rho)\tau_{ij} + \left(\frac{\rho}{U_0^{bs}}\right) \tag{6.1}$$

$$P_1 : (1-\rho)\tau_{ij} + \left(\frac{\rho}{L_{bs}}\right) \tag{6.2}$$

Furthermore, with two separate colonies, two trail initialisation values must be identified. Two new parameters (referred to as τ_{00} and τ_{01}, for phases P_0 and P_1 respectively) are introduced (each of which must be defined). Identifying a suitable definition can be complex (as in the DVRP where an initial solution is required) or one can simply use a constant. Table 4.10 (in Chapter Four) identified a certain level of robustness with regards to setting this parameter; though the extreme value tested showed that one can seriously effect performance if an entirely inappropriate value is chosen.

[7] No change made to global update rule in P_1.

With P_1 acting in the more DVRP-like manner (i.e. minimising distance) it seems sensible to utilise the already optimised definition of τ_0 as the initialisation for τ_{01}. However, as P_0 is a significantly different problem (with a new objective function and global update rule), its trail initialisation vale (τ_{00}) warrants investigation. This becomes the first parameter to be analysed (see Table 6.8). Note that only a small range of constants are considered, this was due to the apparent robustness of τ_0 in some preliminary runs.

τ_{00}		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
0.02	Min	**4**	**1408.87**	**3**	**591.56**	**5**	1531.75
	Ave	**5.1**	**1418.24**	**4**	**643.27**	**5.5**	**1555.28**
	Max	**6**	1457.57	**5**	723.41	**6**	**1572.70**
0.002	Min	5	1395.58	**3**	**591.56**	**5**	1497.23
	Ave	5.6	1425.28	4.7	674.03	5.8	1548.86
	Max	7	1477.37	6	704.93	**6**	1600.79
0.001	Min	**4**	1496.12	**3**	**591.56**	**5**	**1469.22**
	Ave	5.3	1428.63	4.7	672.87	5.7	1532.22
	Max	7	1454.74	6	753.95	7	1519.79
0.0001	Min	5	1387.37	4	622.91	**5**	**1523.01**
	Ave	5.3	1405.76	4.4	662.38	6	1563.78
	Max	**6**	**1437.90**	**5**	**720.36**	7	1639.04

Table 6.8: Analysing Various Values of τ_{00} in Consecutive Two-Phase ERWACS-DVRPTW

From the results it is clear that one would select a τ_{00}=0.02 as the best performing of the tested values. Further results suggested that a lower value would not improve things further (higher values were not assessed as Table 4.10 showed that too high a value will have a negative effect).

When considering the results, it is abundantly clear that the result quality has worsened significantly from those that were produced prior to the introduction of the two-phase approach. This should not be of an immediate concern as there is an obvious shortcoming in the existing implementation that is now going to be considered.

Problem Customers: Pheromone Top-Up Procedure

The biggest concern with the results presented in Table 6.8 is the increase in the number of vehicles deployed i.e. P_0 is failing to identify solutions of sufficient quality. Given the failings of P_0 (better solutions have been shown to exist), it seems sensible to consider some form of improvement. Investigating the process in its current form, it is apparent that there is a shortcoming, relating to the global update rule in P_0.

The global update rule is responsible for the intensification of the search, favouring the regions which have previously been explored and found to be promising.

However, given the objective of P_0 (minimising the number of unscheduled customers); those customers that were unable to be scheduled in the best solution so far do not receive any pheromone increase. Given that this process acts in an autocatalytic manner, those customers are again unlikely to be scheduled (and so on). Clearly as they were difficult to schedule they need to be considered earlier in the construction phase.

To encourage those customers that were not scheduled in the previous cycle (best solution) to be scheduled in the next, they are to have their pheromone values artificially[8] increased. Obviously one cannot know the best place for those customers to be included in the schedule, so pheromone is added to all customer pairs where one or both of the customers were failed to be scheduled.

This will increase the likelihood of these customers being selected, while still leaving the placement up to the RPR. If they are scheduled in the next cycle (best solution) they will receive pheromone as per the global update rule.

As this process falls outside the scope of the usual pheromone update rules, one cannot be certain as to what form (and size) this increase should be. It seems logical that it needs to complement the amount being added via the global update rule, and as such, it seems sensible to consider multiples of this.

[8] Outside the confines of the standard trio of update rules.

A new parameter κ is introduced to control the amount of pheromone added to those customers that were unscheduled in the previous cycle (best solution). This new update rule (referred to as pheromone top-up) can be seen in Equation 6.3.

$$P_0 : (1-\rho)\tau_{ij} + \left(\frac{\kappa \times \rho}{U_0^{bs}}\right) \tag{6.3}$$

It should be noted that there is no change to the global update rule for those customers that were scheduled. Results are presented in Table 6.9.

κ		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
0	Min	**4**	**1408.87**	**3**	**591.56**	5	1531.75
	Ave	5.1	1418.24	4	643.27	5.5	1555.28
	Max	6	1457.57	5	723.41	**6**	**1572.70**
1	Min	**4**	1418.75	**3**	**591.56**	**4**	**1553.18**
	Ave	5	1413.80	**3**	**591.56**	5.3	1589.17
	Max	6	1441.92	**3**	**591.56**	**6**	1634.07
2	Min	**4**	1421.97	**3**	**591.56**	5	1549.02
	Ave	**4.6**	**1439.62**	**3**	**591.56**	**5.2**	**1592.69**
	Max	**5**	1528.04	**3**	**591.56**	**6**	1587.62
3	Min	**4**	1448.71	**3**	**591.56**	5	1520.93
	Ave	4.9	1441.12	3.2	598.85	5.3	1581.51
	Max	**5**	**1491.95**	4	629.52	**6**	1627.32

Table 6.9: Analysing Various Values of κ in Consecutive Two-Phase ERWACS-DVRPTW

Firstly, the technique has indeed improved the result quality, with κ=2 seemingly the best performing. However, the consistency of this technique is notable, with all of the values of κ assessed improving the result. Given that the best performing is not an extreme value, it seems sensible to not consider further values. By setting κ=2, one is laying twice the level of pheromone on the arcs concerning the unscheduled customers, as they are on those that belong to the best solution.

Secondly, it is clear that none of the results produced could be considered competitive; however, when considering these results, it is important to remember

that this technique is in its infancy. It is likely that further improvements exist, and that these could be explored if one was convinced that the technique had the desired potential.

However, given the need to assess the interlaced two-phase approach, such a decision is delayed. Regardless of which method is most successful, the details identified here are likely to be applicable to the interlaced case too, and as such, are beneficial outside the constraints of this implementation.

6.4.3 Interlaced Techniques

The basic premise behind interlacing the phases is that it represents the first step of moving towards a situation whereby one can allow information to move between the phases. This information transfer, referred to as phase interaction[9], requires a system different to the consecutive technique previously discussed.

This interlaced technique (which is similar to that employed by Gambardella et al. (1999)) does not complete P_0 before P_1 begins. Instead one alternates between P_0 and P_1, such that, upon completion of each cycle of P_0, P_1 will be implemented for a single cycle. This process is then repeated until some predetermined exit criterion is met (t=15000).

This alternating technique will alter the split between the two phases, with it being 50/50 as opposed to the 30/70 split previously employed. One is still able to alter this ratio if results are poor e.g. one could complete two P_1 cycles for every P_0, but the level of control one can exert is obviously less than was available in the consecutive phase approach.

The initial benefits of this technique could be considered somewhat limited; though one can hope that P_1 identifies reductions in the number of vehicles that P_0 does not manage to locate. To further increase the interest in the results, two different global update rules for P_0 are going to be considered:

[9] Gambardella et al. (1999) referred to phase interaction by the less intuitive moniker double update.

- The level of pheromone deployed on arcs belonging to the best solution in P_0 is based on the number of unscheduled customers in particular solution (as used in the consecutive two-phase approach), see Equation 6.1

- The level of pheromone deployed on arcs belonging to the best solution in P_0 is based on the distance of that particular solution. This is the technique that was successfully implemented by Gambardella et al. (1999) and can be seen in Equation 6.4.

$$P_0 : (1-P)\tau_{ij} + \left(\frac{\rho}{L_{bs}}\right) \tag{6.4}$$

For both of these techniques, the number of unscheduled customers will remain as the way in which one measures solution quality, this assessment specifically relates to whether this value should be used in the global update rule (P_0), or whether one should continue to use the distance of the infeasible solutions (as seen in Gambardella et al. (1999)).

This unscheduled customer global update rule (P_0) will allow a direct comparison with the consecutive phase approach. Results can be seen in Table 6.10.

Update		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
Customer	Min	**4**	**1375.73**	**3**	**591.56**	**4**	1626.13
	Ave	4.8	1452.53	4.7	677.78	5.6	1613.97
	Max	7	1459.42	6	781.25	7	1707.17
Distance	Min	**4**	1377.09	**3**	**591.56**	**4**	**1518.22**
	Ave	**4.2**	**1449.59**	**3**	**591.56**	**4**	**1622.27**
	Max	**5**	**1446.91**	**3**	**591.56**	**4**	**1711.02**

Table 6.10: Analysing Various Global Update Rules in Interlaced Two-Phase ERWACS-DVRPTW

It is immediately apparent that the use of unscheduled customers in the global update rule (P_0) (within the ACS framework, and more specifically these parameter settings) is uncompetitive. Clearly this has implications within the consecutive phase approach,

as it was the only update rule tested, and issues relating to this will be discussed shortly. However, prior to this, consideration is given to why there is such an obvious discrepancy in performance.

One would be foolish to just discount the use of the number of unscheduled customers in the global update rule (P_0), as this is what measures how good a solution is (in P_0). It also seemed to theoretically work well with the trail ideology, with greater levels of pheromone deposited the better the solution.

However, it seems that the robustness of the algorithm was overestimated, with the change from dividing by distance (likely to be in the 00's or 000's) and unscheduled customers (likely to be in single figures) seemingly too much. This is likely to be due to the way in which the algorithm has been tailored to the problems which it was solving. It has continuously evolved; starting on the VRPTW it was adjusted to the DVRP, and then on to DVRPTW. Through all of these evolutions distance has always been used in the global update rule, and it clearly works well.

If one wanted to continue to use the number of unscheduled customers in the global update rule (P_0), one could adjust the algorithm accordingly. Changes would be required for many of the parameters, but one would expect to be able to improve on the current level of performance.

However, one should not forget that decisions need to be based on the standard of the results produced. The classic-like global update rule (distance based), employed by Gambardella et al. (1999), is producing results of a standard higher than any other results produced for these datasets. It would be foolish to disregard these and as such a decision is made to continue to use distance in the global update rule (P_0).

With regards to the consecutive approach, given the quality of the results produced via the (distance based) interlaced technique, a decision has been made to not continue with this area of research. The decision to proceed with the interlaced approach is further supported by the fact that when comparing the results using the unscheduled customer based global update rule (P_0), there is little difference in the level of performance.

At first glance it seems that one might favour the consecutive approach, but if one considers the results in the context of producing a robust algorithm, the interlaced has its advantages. The poor performance in the interlaced approach was experienced in what has shown to be the easiest of the test datasets (dc201), while in the more complex dataset (dr201) it performed better. Note that there is little to distinguish between the two algorithms on drc201.

Although the consecutive approach offers plenty of scope for improvements, one has to accept that no research can be exhaustive, and a decision to proceed with the more promising results seems logical.

Interaction between Phases

Phase interaction is the obvious next step, it is contained in the MACS-VRPTW and given the competitiveness of that algorithm (on the VRPTW benchmarks), it seems sensible to consider it within this setting. Its basic premise is to blur the divide between the phases, allowing information to be transferred from one colony to the other.

This technique will utilise information learnt by the P_1 colony in P_0. This will encourage results with lower distances, and thus a likely better allocation of customers in P_0.

Gambardella et al. (1999) stated, '*Numerical experiments have shown that a double update greatly improves the system performances*'. This obviously suggests that such a double update warrants consideration, but they were also keen to point out a further reason for its inclusion. If one were to proceed without such interaction, the current update rules in P_0 are '*... not increasing the trails towards the customers that are not included in the solution*'.

This shortcoming was identified in the consecutive phases approach and compensated for via the introduction of the pheromone top-up procedure. One should note that although phase interaction will help, one should not consider it a solution in the way in which the pheromone top-up procedure could be. This is because customers belonging to the best solution identified in P_0 (i.e. those that do not include all

customers) will receive two pheromone updates, once via the standard update and again via the phase interaction. Those that remain unscheduled will only receive a pheromone update from the interaction (Equation 6.5), and thus are still not encouraged for inclusion in the way in which they might need to be. Once the performance of the phase interaction is evaluated, the pheromone top-up procedure will be tested.

$$P_0 : (1-\rho)\tau_{ij} + \left(\frac{\rho}{L_{bs}}\right) \tag{6.5}$$

where L_{bs} relates to the best solution from P_1 from the previous cycle.

Results are presented in Table 6.11.

Inter'n		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
Without	Min	**4**	**1377.09**	**3**	**591.56**	**4**	1518.22
	Ave	**4.2**	1449.59	**3**	**591.56**	**4**	1622.27
	Max	**5**	1446.91	**3**	**591.56**	**4**	1711.02
With	Min	**4**	1407.51	**3**	**591.56**	**4**	**1511.84**
	Ave	**4.2**	**1444.49**	**3**	**591.56**	**4**	**1592.87**
	Max	**5**	**1446.04**	**3**	**591.56**	**4**	**1697.46**

Table 6.11: Analysing the use of Interaction in Interlaced Two-Phase ERWACS-DVRPTW

There is no change in the number of vehicles deployed in any of the three test datasets, so any conclusion as to whether one should include phase interaction in the algorithm will have to be based entirely on the distance of the solutions produced.

There does appear to be an improvement to the distance when the interaction is included, though its impact could not be described as to have 'greatly improved' the performance. However, given that these are different problems and there exist multiple differences between this algorithm and that which Gambardella et al. (1999) use, such discrepancies in performance is not overly surprising.

Concerning oneself with the simple decision as to whether the interaction belongs in this algorithm, one would argue that its inclusion is worthwhile, as it has improved solutions (albeit only marginally). As stated previously, with a decision to include phase interaction made, consideration is now given to whether the algorithm could benefit from the pheromone top-up procedure, in much the same way as the consecutive phase algorithm did.

Pheromone Top-Up Procedure

The inclusion of the phase interaction, although beneficial, did not have any impact on the number of vehicles being deployed. With this being the primary objective, it seems sensible to revisit the pheromone top-up procedure, which was specifically designed to do just that.

The theory behind the top-up procedure was described in Section 6.4.2, when it was implemented in the consecutive phase approach. In this framework (with κ=2) it had a positive performance, reducing the number of vehicles deployed in 18 of the 30 runs conducted across the three test datasets. Note that a similar level of improvement is impossible, given the current performance of the interlaced phases. However, given that it could still improve the performance, it still warrants investigating.

With the justification for its inclusion in the interlaced phase algorithm described when the idea of phase interaction was considered, only implementation details are required. Its implementation will be identical to that seen in the consecutive approach (Equation 6.3).

It seems sensible to reconsider various values of κ, as the interlaced and consecutive phase approaches are so different. The level of bias towards the unscheduled customers could well be different (likely to be lower) due to the fact that they are already receiving some level of pheromone increase due to the inclusion of the phase interaction. Results can be seen in Table 6.12.

As expected the difference between the results with the pheromone top-up and those without are much closer than they were before. Significantly, the first set of DVRPTW results with the optimal number of vehicles (as indicated by the VRPTW

datasets) identified in all 30 runs, have been produced. This occurred when κ=1, which is in keeping with the theory that a lower level of pheromone top-up might be required due to the presence of phase interaction.

κ		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
0	Min	**4**	1407.51	**3**	**591.56**	**4**	**1511.84**
	Ave	4.2	1444.49	**3**	**591.56**	**4**	**1592.87**
	Max	5	1446.04	**3**	**591.56**	**4**	**1697.46**
1	Min	**4**	1381.44	**3**	**591.56**	**4**	1551.54
	Ave	**4**	**1449.19**	**3**	**591.56**	**4**	1642.91
	Max	**4**	**1523.71**	**3**	**591.56**	**4**	1736.59
2	Min	**4**	**1371.57**	**3**	**591.56**	**4**	1555.22
	Ave	4.1	1419.79	**3**	591.91	**4**	1633.26
	Max	5	1468.92	**3**	595.12	**4**	1781.43
3	Min	**4**	1403.98	**3**	**591.56**	**4**	1532.24
	Ave	4.1	1463.81	**3**	591.83	**4**	1605.08
	Max	5	1569.50	**3**	594.32	**4**	1722.56

Table 6.12: Analysing Various Values of κ in Interlaced Two-Phase ERWACS-DVRPTW

Clearly these are the best results produced thus far, and as such all further implementations will contain κ=1.

6.4.4 P_1 Feasibility

With such promising levels of performance achieved by the interlaced phase algorithm (including the phase interaction and the pheromone top-up procedure), it is time to give further consideration to the secondary objective, namely distance.

It is currently the responsibility of P_1 to reduce distance; this is done via the repeated construction of solutions benefiting from the learning provided by the trail. However, before looking to improve the performance (see Section 6.5), it seems prudent to check how P_1 is operating.

Prior to the introduction of the two-phase approach, it was possible to state with a certain amount of confidence that given the opportunity, the algorithm could

successfully reduce the distance of solution. However, this is the first such implementation where the algorithm trying to reduce distance has the potential to produce infeasible solutions. One should investigate to see if P_1 is behaving appropriately i.e. if it is producing an acceptable number of feasible solutions (see Figure 6.5).

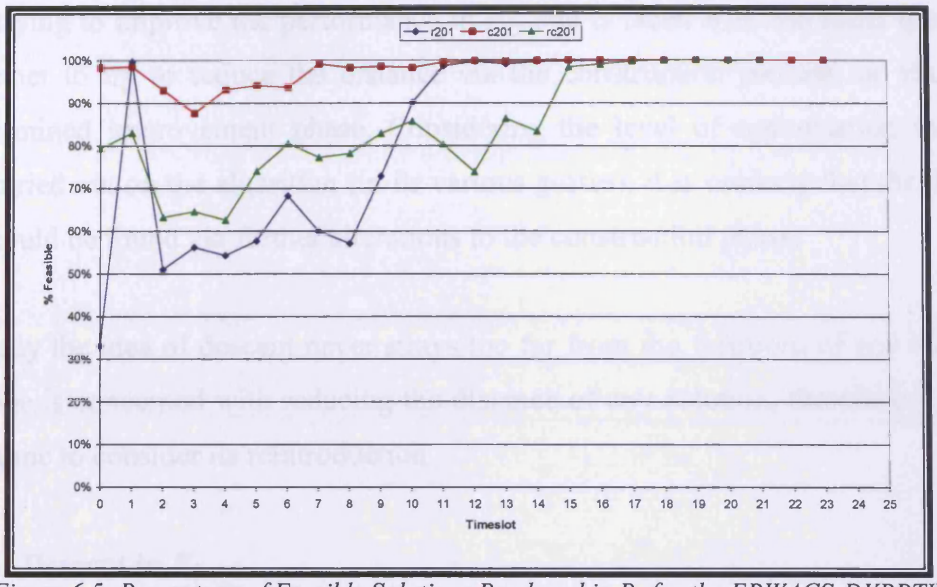

Figure 6.5: Percentage of Feasible Solutions Produced in P_1 for the ERWACS-DVRPTW

It is clear that high levels of feasibility are maintained for the majority of the algorithms run. As one would expect, there is an increase in feasibility as the number of uncommitted customers decreases. There is an interesting spike occurring in the two more difficult datasets (dr201 and drc201) during ts_1. This is understandable due to the fact that ts_1 is actually very similar to ts_0 (n_1=n_0 minus those that have been committed too). The other sharp increases indicate an increase in the number of vehicles being used, hence making scheduling a far simpler task.

Once new customers begin to arrive the feasibility level begins to drop, as the algorithm adjusts for their presence. Obviously the change in feasibility will be further affected by the change in the customers being dealt with, but it will also relate directly to the performance in P_0. The ease in which the customers can be scheduled into A_0 vehicles will be of significant importance. If P_0 (or P_1) had difficulty in achieving the solution in A_0 vehicles, then there is likely to be a far greater proportion of infeasible solutions produced in P_1.

Thankfully none of the three test datasets seems to be consistently producing a large number of infeasible solutions and as such, one can have a certain amount of faith in the performance of P_1. Consideration is now given to improving the final solution returned by P_1; the improvement phase is reintroduced.

6.5 Reintroduction of the Improvement Phase

When trying to improve the performance in P_1, one is faced with the usual quandary of whether to try to reduce the distance via the construction process, or via some predetermined improvement phase. Considering the level of optimisation that has been carried out on the algorithm (in its various guises), it is unlikely that the desired effect could be found via further alterations to the construction phase.

Obviously the idea of descent never strays too far from the forefront of any thoughts when one is concerned with reducing the distance of any solution, therefore it seems an apt-time to consider its reintroduction.

6.5.1 BI Descent in P_1

The descent phase will (as before) consist of the CROSS Neighbourhood (adapted from Chapter Five to accommodate the inclusion of time windows) and will be implemented under the BI descent strategy[10]. Obviously descending to a local optimum via steepest descent (which requires a complete neighbourhood evaluation prior to every move) will result in a significant increase in the run-times. Although this is not currently being assessed on the time-constrained variant of the problem, it seems sensible to try to compare like-with-like.

With this in mind, a precursory run was made to identify how many cycles of the ERWACS-DVRPTW including descent could be conducted in a similar time to t=15000 cycles without descent. It soon became evident that there would need to be a significant decrease (t=500) in order for the run times to be comparable.

With such a decrease, one would expect there to be an impact on the success that the ants are having, in both reducing the number of vehicles and the distance (in the

[10] The FI descent strategy will also be tested later in the chapter for comparative purposes.

construction phase). Obviously, one would hope that this second problem is superseded by the descent phase, negating the need to rely on the ants so heavily to reduce the distance, but what of P_0?

As the descent algorithm is simply operating on the feasible solutions produced in P_1, it is only going to affect P_0 through the phase interaction. One has to consider if this (hopefully significant) increase in solution quality impacts P_0 enough to negate the decrease in cycles. Results are presented in Table 6.13.

Descent	t		dr201		dc201		drc201	
			Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
None	15000	Min	**4**	1381.44	**3**	**591.56**	**4**	1551.54
		Ave	**4**	1449.19	**3**	**591.56**	**4**	**1642.91**
		Max	**4**	1523.71	**3**	**591.56**	**4**	**1736.59**
Cross	500	Min	**4**	**1276.84**	**3**	**591.56**	**4**	**1442.94**
		Ave	**4**	**1317.09**	**3**	**591.56**	4.3	1471.02
		Max	**4**	**1334.93**	**3**	**591.56**	5	1458.81

Table 6.13: Analysing the Impact of Reintroducing Descent

With the number of vehicles being deployed as the primary concern, it is clear to see that the introduction of descent has had a negative impact on dataset drc201. Datasets dr201 and dc201 remain unaffected with regards to vehicle deployment, but significantly, the average distance reduced by 10.34% for dr201. One must consider the performance in dataset drc201 carefully, before deciding whether descent should constitute such a significant part in proceedings.

This significant improvement in distance is obviously important, but the increase from 4 to 4.3 vehicles in dataset drc201 is a worry. It has been shown (in Chapter Four), that to achieve competitive results a descent phase is required. Thus a decision is made to retain the descent phase, but to attempt to limit its effect on P_0 i.e. reduce the time which it requires.

Note that it would be possible to introduce a descent phase into P_0, with an objective of further minimising the unscheduled customers. However, this would lead to

another increase in computational effort (hence a further substantial decrease in the number of cycles able to be completed), thus further impacting on what one could already consider a well performing algorithm.

Before considering reducing the impact of descent on vehicle deployment, consideration is given to how it has affected the feasibility of solutions produced in P_1.

What has the Reintroduction of Descent Done to P_1 Feasibility?

It is important that when trying to minimise the distance in P_1 (subject to the number of vehicles in P_0) one is maintaining a reasonably high level of feasibility. This (as in Section 6.4.4) is critical to the success of the algorithm, but it also provides two interesting pieces of information about the impact that the reintroduction of descent has had:

- Given that only the feasible solutions are descended; the greater the proportion of feasible solutions, the longer is going to be spent descending. Fewer feasible solutions will result in a reduction of the time required for the 500 cycles to be completed.

- It may offer an insight into the earlier workings of the algorithm, identifying whether the trail values need time to settle down before one starts to see a consistent number of feasible solutions in P_1. It will also help to understand the impact of the mass reduction in the number of cycles.

The feasibility details are presented in Figure 6.6.

Obviously any conclusions drawn from this figure are merely observations, but it appears to offer some insight into the algorithm's behaviour. There are some interesting similarities between this figure and Figure 6.5 (in Section 6.4.4), most obviously the increase in feasibility as one begins to approach the latter timeslots and the obvious spike located at ts_1.

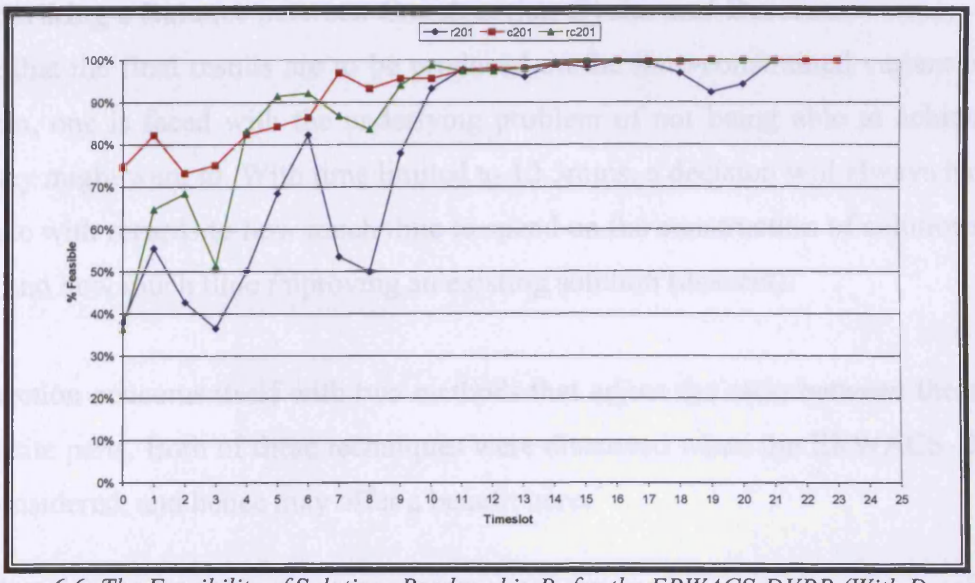

Figure 6.6: The Feasibility of Solutions Produced in P_1 for the ERWACS-DVRP (With Descent)

However, there are also some differences; most notably the markedly lower feasibility ratios in ts_0 for two of the three datasets (the one that was low without descent remained that way). This is likely to be due to the trail values taking a while to settle down (the significant reduction in the number of cycles increases the proportion of the run that this settling down period constitutes).

Interestingly, drc201[11] reaches a plateau in ts_{10} whereas previously (without descent) this did not occur until ts_{16}. As for why this occurred, it is possible to speculate that the descent algorithm is likely to result in more solutions containing common characteristics thus reinforcing them with greater levels of pheromone (and starting the autocatalytic process).

Overall, the feasibility levels are maintained at a reasonably high level, and as such, do not represent a cause for concern. Again, it should be noted that the sharp increases in feasibility are often due to an increase in the number of vehicles deployed, hence making the scheduling a simpler task.

Consideration is now given to how one can increase the number of cycles without compromising the performance increase provided by the reintroduction of descent.

[11] This was a solution that deployed four vehicles, which is important as this is the number in the solution without descent.

6.5.2 Striking a Balance between Construction Cycles and Descent

Given that the final results are to be produced on the time-constrained variant of the problem, one is faced with the underlying problem of not being able to achieve all that they might want to. With time limited to 12.5mins, a decision will always have to be made with regards to how much time to spend on the construction of solutions (via ACS) and how much time improving an existing solution (descent).

This section concerns itself with two methods that adjust the ratio between these two composite parts. Both of these techniques were discussed when the ERWACS-DVRP was considered, and hence may offer a benefit here.

The first of these techniques concerns itself with only descending selected feasible solutions, thus reducing the number of descent moves made (and hence increasing the number of cycles that can be completed). The second is likely to offer less of a change, and simply considers whether the descent algorithm should operate under the BI (as is now) or FI descent strategy.

Is it Necessary to Descend Every Feasible Solution in P_1?

This issue was discussed and deemed beneficial to the DVRP in Section 4.7.2. The selection of which solutions to descend was made purely on the distance, with only the best solution constructed each cycle descended. This enabled a greater number of cycles to be completed and as such, resulted in an overall increase in the level of performance.

Clearly, with the new objective function, increasing the number of cycles is even more prevalent than it was in the DVRP. Given that the current results presented are on the non time-constrained variant of the problem, a decrease in result quality is a very real possibility. However, the significance of this decrease (if it occurs) is what is important, and clearly this is subjective. For the first time in this chapter, the time taken for the algorithm to run is going to be presented.

With considering the change in run times, one should understand that the impact will be influenced by a number of factors. It does not simply reduce the run time by a

factor of $((m-1)/m)$, as the m solutions are still constructed. Furthermore, with infeasible solutions occurring in P_1, there is no saying that m descents would have happened were all solutions being descended. Lastly, as the solutions being descended are those of the highest quality, less moves may be required to reach a local optimum.

Results and run-times are presented in Tables 6.14 and 6.15 respectively.

Descent		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
All	Min	4	**1276.84**	3	**591.56**	4	1442.94
	Ave	4	**1317.09**	3	**591.56**	4.3	1471.02
	Max	4	**1334.93**	3	**591.56**	5	**1458.81**
Best	Min	4	1288.13	3	**591.56**	4	**1396.08**
	Ave	4	1325.31	3	**591.56**	**4.2**	**1457.87**
	Max	4	1385.56	3	**591.56**	5	1532.05

Table 6.14: Analysing the Impact of Only Descending from the Best Ant each Cycle

What is immediately clear is that the average number of vehicles deployed in drc201 has decreased by 0.1. It is important to be aware that as this is the non time-constrained variant of the problem, this is not as a result of an increase in the number of cycles (which remains at the user defined t=500). Any decrease here is simply due to the randomness of the algorithm. Were an increase observed, this could be due to worse solutions being transferred via the phase interaction. However, with no such impact, decisions can be based purely on the distance of the solutions produced.

Interestingly, there is an increase (albeit small) in the Min, Ave and Max. This is the kind of alteration to performance that would be expected, and is unfortunately unavoidable. Dataset dc201 offers no details as to whether this is a sensible avenue to pursue, while the performance level on drc201 (average distance) has actually improved (again one would have to assume due to randomness). With such varied results it comes down to the size of the impact, and that is clearly small.

There is no significant decrease in performance when only the best ant each cycle is descended, thus one would certainly be happy to utilise this approach.

As explained previously, it is the reduction in the time spent descending solutions that is the reason for wanting to introduce this. The greater the time devoted to solution construction, the more likely one is to reduce the number of vehicles being deployed. Descent has the ability to remove a vehicle (x:L,0 when L is equal to the length of a vehicle, less the depot), but from observing multiple runs this does not happen frequently enough to rely on the technique alone.

Descent		dr201	dc201	drc201
All	Ave	23.23	14.37	17.09
Best	Ave	8.59	4.34	6.14

Table 6.15: Analysing the Run-Times for All Ants and Best Ant Descent Techniques

The change has significantly altered the run-times of the algorithms, with a reduction of between 63-70% observed. This will allow considerably more cycles to be completed in the 12.5 minutes available, when solving the time-constrained variant of the problem. These extra cycles will increase the potential for reducing the number of vehicles, and will also give the ants a greater chance to reduce the distance in ways that an improvement algorithm cannot.

This policy is clearly a sensible one to adopt, with little (if any) change in solution quality and such a notable reduction in run-times. Consideration is now given to the descent strategy, and to whether the BI or FI would be more suitable.

The ERWACS-DVRPTW Descent Strategy

This issue as to which descent strategy one should use when solving the DVRP has not been discussed directly. In Sections 4.6.3 the EWRACS-DVRP employed a FI strategy, (with an appropriate exit criterion), while in Chapter Five the TS implementation used its traditional BI strategy.

Provided the exit criterion is set sufficiently high for the FI strategy, both of these techniques should obtain a local optimum. With that in mind, there should be no recognisable difference between their performances with regards to distance. However, it is possible that one is able to locate that optimum in less time than the

other, and as such, would be preferential, as it would lead to the possibility for more cycles to be completed in the time-constrained variant of the problem.

With any implementation details well documented in previous assessments, one merely needs to set the (exit criterion) before producing the relevant results, see Tables 6.16 and 6.17 for results and run-times respectively.

Descent		dr201		dc201		drc201	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
FI	Min	**4**	**1276.84**	**3**	**591.56**	**4**	**1442.94**
	Ave	**4**	**1317.09**	**3**	**591.56**	4.3	1471.02
	Max	**4**	**1334.93**	**3**	**591.56**	**5**	1458.81
BI	Min	**4**	1288.13	**3**	**591.56**	**4**	**1442.94**
	Ave	**4**	1325.31	**3**	**591.56**	**4.2**	**1457.87**
	Max	**4**	1385.56	**3**	**591.56**	**5**	**1411.30**

Table 6.16: Analysing the Descent Strategy used in P_1

As expected, there does not appear to be a noticeable difference between the results produced via the two descent strategies. The difference in the average number of vehicles deployed in drc201 is negligible, and furthermore cannot be directly related to the change in descent strategy.

With these results showing no clear difference in performance, any distinction (if there is in fact one) will need to relate to the run-times of the algorithm.

Strategy		dr201	dc201	Drc201
FI	Ave	8.59	4.34	6.14
BI	Ave	8.92	4.06	6.84

Table 6.17: Analysing the Run-Times for the FI and BI Descent Strategies

There does not appear to be a significant run-time difference between the two descent policies, which although not immediately apparent, makes the selection between them simple. With all of the run-times within 42 seconds of each other, and no clear trend as to which is slower, the selection of the BI strategy is made. This guarantees that a local optimum is identified, which one cannot be sure of with the FI strategy.

Time-Constrained Variant

These results are being produced to enable a comparison to be made between the time-constrained and non time-constrained variants of the problem. This will provide evidence as to whether there is any improvement identified by this alteration to the exit criterion.

The results (see Table 6.18) produced in this section use the ERWACS-DVRPTW in its most evolved state, including all of the performance enhancements identified throughout this chapter. The exit criterion is adjusted from t=500 to T=12.5 with an equal distribution of time throughout the n_{ts} timeslots i.e. 30 seconds. Given the low run times in the previous results produced, it is likely that this new criterion will increase the number of cycles in most timeslots[12].

		dr201		dc201		drc201	
Descent		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
	Min	4	1284.48	3	591.56	4	1457.60
Best	Ave	4	1327.14	3	591.56	4.3	1466.83
	Max	4	1358.48	3	591.56	5	1527.32

Table 6.18: The ERWACS-DVRPTW on the Time-Constrained Variant of the Problem

For a reference point, one is likely to want to compare these results with Table 6.16 (the non time-constrained results produced with the same algorithm). These results have not been presented here; given the alteration to the problem variant, making such a direct comparison is inappropriate.

However, general observations can be made, the most significant being that the result quality has not altered much. The number of vehicles deployed is identical in both cases, with, one could argue, slightly better distances produced in the non-time constrained variant. This lack of sizable change in performance suggests that the optimisation that was carried out is applicable to the time-constrained variant, and as such, results for the complete set of 27 benchmark problems can be produced.

[12] This ambiguity is due to the uneven distribution of time in the non time-constrained variant of the problem.

6.6 Results and Conclusions

These results will be produced using the algorithm that has been developed throughout the course of this chapter and as such one would hope to be performing to a high standard. However, understanding the quality of the results is not simple, due to the lack of other results with which to make a comparison.

As one would expect on new datasets, quantifying an algorithm's performance is a challenge. To aid in this process, a further two sets of results are presented along with those produced by the ERWACS-DVRPTW:

- These are results produced on the same dynamic datasets, but with construction decisions made in a greedy fashion (q_0=1). The trail has no influence over decisions (β=0) and the two-phase approach is abandoned as only one solution is produced. However, all other components of the ERWACS-DVRP remain: objective function, criterion for tentative schedule completion, CU factor, and visibility.

- These are the current best solutions identified for the VRPTW. These results were produced by a multitude of authors on the datasets in their original form i.e. before the added of dynamism in Section 6.2.1. These results may or may not be achievable in the dynamic case, so should not be considered in a like-for-like comparison.

These should allow one to gain an understanding as to the quality of the results produced via the ERWACS-DVRPTW. One should note that the results produced via the greedy technique benefit from many of the improvements identified throughout this chapter and as such, represent a fairly complex implementation in their own right. It is hoped that the ERWACS-DVRPTW will out perform this, but were it not to, its development need not be viewed as a failure.

The results are presented in Table 6.19 (note that the results for the VRPTW will be shown in red, highlighting the fact that they should not be used for direct comparison purposes). As the VRPTW results are the best currently produced, they have been

shown in bold. It is seems sensible to assume that none of the results DVRPTW results will better those achieved on the VRPTW, but any that match, will also be shown in bold.

	Greedy & Descent	ERWACS-DVRPTW Min	Ave	Max	VRPTW Best
dr201	5 1417.03	**4** 1284.48	**4** 1327.14	**4** 1358.48	**4** **1252.37**
dr202	4 1227.55	4 1139.04	4 1186.92	4 1265.05	**3** **1197.66**
dr203*	**3** 1300.65	**3** 995.16	3.1 1039.27	4 1015.88	**3** **942.64**
dr204*	3 955.14	3 784.15	3 805.84	3 821.4	**2** **849.62**
dr205	4 1218.19	**3** 1042.74	**3** 1074.05	**3** 1126.13	**3** **998.72**
dr206	4 1151.18	**3** 929.95	**3** 982.49	**3** 1028.66	**3** **912.97**
dr207*	3 1156.15	3 868.02	3 905.46	3 937.42	**2** **914.39**
dr208*	3 810.78	**2** 764.01	**2** 790.75	**2** 818.75	**2** **731.23**
dr209	5 1065.6	**3** 958.06	3.2 995.08	4 939.05	**3** **910.55**
dr210	**3** 1171.45	**3** 992.53	3.6 1013.75	4 1023.07	**3** **955.39**
dr211	3 857.74	3 819.82	3 852.23	3 868.19	**2** **910.09**

dc201	**3** **591.56**	**3** **591.56**	**3** **591.56**	**3** **591.56**	**3** **591.56**
dc202	4 724.87	**3** **591.56**	**3** **591.56**	**3** **591.56**	**3** **591.56**
dc203	4 828.15	**3** 744.73	**3** 771.81	**3** 827.91	**3** **591.17**
dc204*	4 932.97	**3** 623.76	**3** 639.06	**3** 688.03	**3** **590.6**
dc205	**3** **588.88**	**3** **588.88**	**3** **588.88**	**3** **588.88**	**3** **588.88**
dc206	**3** **588.49**	**3** **588.49**	**3** 590.72	**3** 592.21	**3** **588.49**
dc207	**3** 617.02	**3** **588.29**	**3** **588.29**	**3** **588.29**	**3** **588.29**
dc208	**3** 650.28	**3** **588.32**	**3** **588.32**	**3** **588.32**	**3** **588.32**
drc201	5 1521.72	**4** 1457.6	4.3 1466.83	5 1527.32	**4** **1406.94**
drc202	5 1366.02	4 1214.32	4 1264.57	4 1397.67	**3** **1389.57**
drc203*	5 1147.2	**3** 1138.3	3.8 1104.45	4 1130.97	**3** **1060.45**
drc204*	**3** 1019.42	**3** 839.47	3.2 880.08	4 848.07	**3** **799.12**
drc205	6 1454.77	**4** 1487.02	4.5 1444.12	5 1438.83	**4** **1302.42**
drc206	5 1368.57	**3** 1238.84	3.6 1219.36	4 1207.39	**3** **1156.26**
drc207	5 1191.24	**3** 1294.44	3.9 1129.43	4 1143.22	**3** **1062.05**
drc208*	**3** 972.93	**3** 886.82	3.5 880.91	4 882.89	**3** **832.36**

Table 6.19: Results for the ERWACS-DVPTW.

As explained previously, one cannot simply compare these results with other authors to gauge the level of performance. The required analysis is based more on a combination of comparison to a similar problem and conjectures. In order to simplify this process, this analysis is going to be divided into two independent sections, followed by a general summary.

The first of these sections will be of the more traditional type, whereby a comparison is going to be made on the performance of two algorithms on the DVRPTW. This will be conducted on the ERWACS-DVRPTW and the similar (but simpler) greedy-type implementation.

The second of these is the more uncertain analysis, the comparison with the best-produced VRPTW solutions. Any conclusions drawn from this analysis need to be treated with a certain amount of trepidation for obvious reasons.

Greedy Implementation vs. ERWACS-DVRPTW

Making a comparison between a complex metaheuristic and what initially appears to be a rather simple implementation may seem to be a somewhat foolhardy exercise. One would expect the metaheuristic to outperform the simpler technique (in this case a greedy algorithm) and if one were to summarise the performance, that is the conclusion that would be drawn. However, there are many interesting issues, which would benefit from some form of discussion.

Firstly, when comparing the ERWACS-DVRPTW Min values with the greedy implementation, the potential of the ERWACS-DVRPTW becomes clear. The number of vehicles deployed has decreased by 14.7% with 14 of the 27 benchmark datasets requiring less vehicles (no solutions required extra vehicles). This improvement in over half of the datasets is obviously significant, but one cannot simply consider the Min solutions.

When comparing the average number of vehicles deployed, one can observe that improvements were still evident in 14 of the 27 datasets, though solution quality deteriorated in three datasets (dr203*, drc204* and drc208*). All of these poor solutions were on datasets that were not created with the ≈50% dynamic customers. This may be significant, as greater numbers of customers are known at the start, thus one is actually solving a more complex problem in the earlier timeslots.

The maximum number of vehicles deployed is significant, as this represents the worst performance one might expect from the algorithm. If these solutions are very poor, one might consider the greedy algorithm to be preferable in certain circumstances. The number of vehicles deployed has decreased in 12 of the 27 benchmark datasets, with increases apparent in 5. Ideally one would want the worst ERWACS-DVRPTW solutions to be better than the greedy algorithm, but this is not always possible. The high level of variation in the results is always going to be problematic in such

dynamic datasets, but even in this worst-case scenario, one would be likely to favour the ERWACS-DVRPTW.

Comparing the distance is only possible in the cases when the number of vehicles deployed is identical, distance comparisons when the number of vehicles has changed are not that insightful (as distance is purely a secondary objective). In the 13 datasets where vehicle deployment (Min) is equal to the greedy algorithm, distance has reduced by 10.30%, a substantial decrease. When comparing the distance for the 8 Ave solutions with vehicle deployment equal to the greedy algorithm, a 6.88% decrease can be seen.

Overall, the solution quality achieved by the ERWACS-DVRPTW is far superior (when one considers the performance across all 27 datasets) to that which the greedy algorithm produced. This result, although expected, confirms the validity of the workings in this chapter. A comparison of these results with the best-produced VRPTW solutions is now provided.

ERWACS-DVRPTW vs. Best-Produced VRPTW

Comparing the results to the VRPTW, one is immediately aware that the result quality is markedly lower. This was to be expected, not only because the DVRPTW is a harder problem than the VRPTW, but also because these are the best solutions ever produced for these datasets. The solutions were achieved via years of research by a large numbers of authors.

The relative poor performance should not dishearten, it was never likely (or maybe even possible) to produce results that could compete. Interestingly, the number of vehicles deployed has not risen significantly; with results indicating only a 7.74% and 13.10 % increase in the Min and Ave values respectively.

Making a distance based comparison is more complex, as less vehicles does not always imply a lower distance (e.g. dr202 and dr204*). With this preventing a direct comparison, it seems sensible to initially focus on the solutions that did not result in an increase in the number of vehicles (as it has already been shown that these constituted the majority of the solutions produced). When the number of vehicles

deployed in the ERWACS-DVRPTW (Min) was equal to the best-produced solutions for the VRPTW there was only a 4.48% increase in distance.

In each of the five datasets where results were not so promising (i.e. those where extra vehicles were deployed), the distance (Min) was actually less (-1.44%) than the distance for the best-produced VRPTW solutions. This supports the assumption that the correlation between vehicles and distance may be less prominent in problems with time windows than it was in the CVRP (and the DVRP). Furthermore, it highlights that the algorithm may be suffering from its initial development as a distance-reducing algorithm. Attempts were made to accommodate for the new objective function, but it might be able to benefit from further adaptations.

If distance had remained the primary objective it would have been interesting to compare whether the results produced for the DVRPTW are closer to best-produced VRPTW results, than the DVRP results were to the best-produced CVRP results. However, such an analysis would not be fair, as the different objective functions would have too large an impact.

6.6.1 Analysis Summary and Chapter Conclusions

If one were to summarise the performance of the ERWACS-DVRPTW, it would be that it consistently outperforms the greedy implementation but does not produce results of a standard similar to the best-produced for the VRPTW. The amount of variability in the results is not ideal, but is likely to be unavoidable due to the nature of the problem and the timeslot solution strategy being utilised.

In truth one will not be able to genuinely offer substantial quantitative analysis until further research has been conducted on this problem. This chapter contains only one metaheuristic, and thus does not offer the necessary diversity amongst the various potential techniques to truly evaluate its performance. Any investigation into a newly developed problem is going to be faced with this problem (as Montemanni et al. (2005) were for the DVRP), but in time other authors may choose to further this investigation in much the same way as Chapters Three, Four and Five did for the DVRP.

In conclusion, this chapter has resulted in the development of a new problem; benchmark datasets were created and a set of what appear to be good results have been produced. Throughout the development of the ERWACS-DVRPTW much has been learnt about the problem's subtleties, how to best solve it and how one must respect it as a problem in its own right.

Chapter Seven

Problem Decomposition for the VRPTW

7.1 Introduction

The primary concern of Chapter Six was the establishment of the DVRPTW as a solvable problem through the utilisation of many of the techniques that had previously been developed in this thesis. The problem was suitably defined and what one can assume to be high quality solutions (for reasons outlined in Section 6.6) were produced. This chapter builds on a particularly interesting offshoot of those workings.

The current method was developed for solving the DVRPTW, but one need not concern themselves solely with one specific problem, if the method may have potential benefits in others. It is in one particular area that it appears that a significant development may be achievable, namely the VRPTW or more specifically Large Scale[1] VRPTW.

This chapter will focus on the adaptation of the existing DVRPTW technique into a suitable method for solving Large Scale VRPTW (time-constrained). The investigation will be structured as follows:

[1] Large Scale in this instance refers to problem sizes ≥200 customers.

Firstly, one needs to consider why one would be interested in such an adaptation. One might assume that a method specifically designed for the VRPTW would be better suited to the Large Scale VRPTW. This section is divided into two subsections with the first discussing the benefits of adapting the current technique while the second aims to support these claims with some preliminary results. The second subsection will conclude with the Large Scale VRPTW benchmarks being divided into two sub-groups based on the number of customers.

Secondly, utilising the first of the Large Scale VRPTW sub-groups (200-400 customers), a new dataset creation technique is considered. As the datasets are no longer dynamic, one can choose how to make them appear to be dynamic (which is required for the current algorithm to solve them). Building on the workings of Section 6.2.1 one will consider if the same technique should be used, or whether better techniques exist.

Thirdly, consideration is given to time management. One need no longer require each timeslot to be of equal length, with them existing purely as a solution technique rather than a component of the problem. The balance between the ACO construction and improvement phases also needs to be considered, as with the increase in problem size, the neighbourhoods will become significantly larger. This section will conclude with a full set of results for this sub-group of Large Scale VRPTW.

Finally, the second sub-group (with 600-1000 customers) of Large Scale VRPTW are considered using a greedy heuristic for construction, whilst retaining the complex improvement phase utilised for the other sub-group. Some adaptations are made to the now very basic construction phase with an aim of improving the initial solution (i.e. before the improvement phase begins). This section will conclude with a full set of results for this sub-group of Large Scale VRPTW.

7.1.2 Why use a Dynamic Technique on Static Problems

This is an interesting question and to provide an appropriate answer one is best considering one of the most successful methods for solving Large Scale CVRP. Although a different problem it offers up an interesting parallel which one might be able to use in the time window environment.

The work that is being considered is Reimann et al. (2004), which coincidently is an ACO technique (though this is not the reason why it is of significance). Entitled 'D-Ants: Saving Based Ants divide and conquer the vehicle routing problem' it operates in much the way one would expect from the title; dividing the problem into more manageable sub-problems (and then bringing these solutions together as a solution for the entire problem). This technique is known as problem decomposition and has allowed for better solutions to be produced for Large Scale CVRP.

Without getting tied down in the complexities of the workings of Reimann et al. (2004), one must be aware that the process of breaking the problem down into sub-problems is made geographically. This seems a sensible policy, with one rarely seeing a high quality solution containing long customer arcs being traversed. No such decomposition technique currently exists for the VRPTW (with hard windows[2]). Initially one may feel that such a similar idea could be implemented on the Large Scale VRPTW, but unfortunately this is unlikely to be the case. Prior to considering an alternative decomposition technique, the reasoning for this likely failing is discussed.

The VRPTW is not as concerned with distance as the CVRP. This can be seen from the need to incorporate a time element in to the visibility function (Section 6.3.3), whereby it could be seen that customers that are close together do not always make for good solutions in the VRPTW.

Examples of a good solution to a CVRP and a VRPTW are shown in Figure 7.1.

As one can see immediately the difference in appearance is significant. The CVRP exhibits a petal-like structure with customers in close proximity being visited subsequently. Furthermore, none of the routes are self-crossing, as these cannot be contained in optimal solutions and should not be apparent in good solutions[3]. Not all CVRP solutions will follow these trends as exactly as this example, but one will

[2] Geographical decomposition is used in Taillard et al. (1997) on the soft window variant of the problem.
[3] Otherwise one should uncross the self-crossing section and have an even better solution.

usually be able to identify these traits within solutions. The VRPTW solution can only be described as appearing much more arbitrary; there does not appear to be any clearly identifiable pattern (beyond a slight petal structure in dark blue and red routes). The routes are self-crossing, and there is no global structure to the solutions.

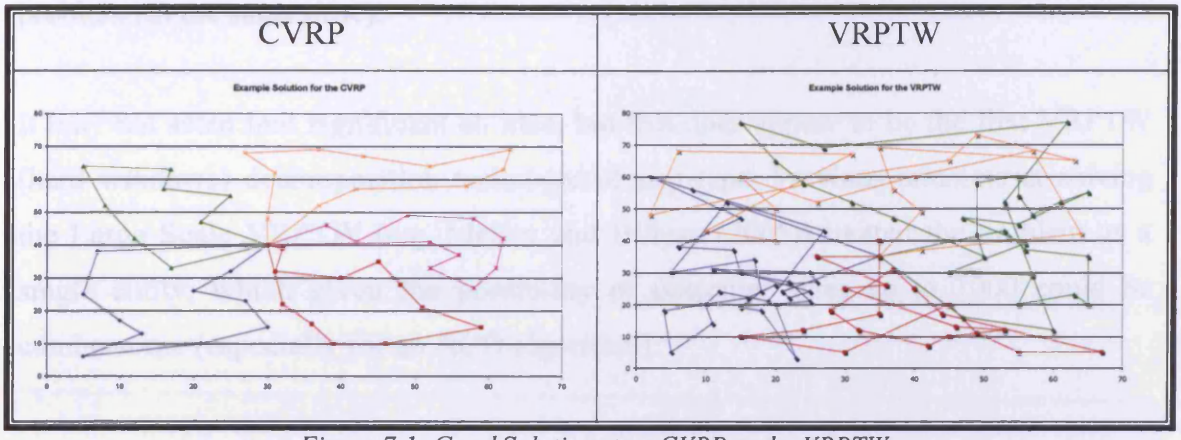

Figure 7.1: Good Solutions to a CVRP and a VRPTW

With a good VRPTW solution being so different from a good CVRP solution, a geographical decomposition is unlikely to reap the same level of benefits in the Large Scale VRPTW instances. If one were to want to divide the VRPTW (or more specifically the Large Scale VRPTW) problem into sub-problems, a more original technique ought to be considered.

Time Base Decomposition

The initial discussion concerning the creation of the DVRPTW stated that there were greater similarities between the VRPTW and the DVRPTW than there were between the CVRP and DVRP. The primary reason for this was the fact that customers had a time window within which they had to be serviced. The time that they became known was not the significant information (as it was in the DVRP), as when that customer had to be scheduled remained the same (regardless of when they became known).

Referring back to the analysis of the performance of the ERWACS-DVRPTW in Section 6.6, one can see that the average result quality increased by only 13.10% (when averaged across the 27 datasets) when compared to the best ever results for the VRPTW. If one were to view this level of performance as fairly encouraging, one could suggest a non-geographical decomposition method has already been developed.

If one were to implement this technique on the Large Scale VRPTW, one would have to artificially create the dynamic environment[4] by limiting the information they choose to use at certain times. By doing this, one would be able to break a large problem down into a series of smaller sub-problems, hopefully resulting in an improvement to performance level to that which can be achieved on the whole problem (in the same time).

It may not seem that significant an idea, but this does appear to be the first VRPTW (hard windows) decomposition technique of any type. Previous attempts at solving the Large Scale VRPTW (e.g. Mester and Bräysy (2005)) treated the problem as a single entity, which given the possibility of customer sizes up to 1000 could be cumbersome (especially for an ACO algorithm).

In addition to being cumbersome, they can be particularly time consuming, with even the production of a feasible solution by anything other than the most basic heuristic taking a significant amount of time. This chapter is only concerned with the time-constrained variant of the problem, and whether decomposition can help produce better solutions for Large Scale VRPTW given that time limit.

This offers up a significant real-life situation where one could make use of the workings in Chapter Six in a non-dynamic environment. When computing power is unable to deal with such large problems, it may allow results to be produced when previously they would have been unobtainable.

Introduction to Large-Scale Static VRPTW Datasets

The Solomon benchmarks contain three different types of datasets (those that were used in Chapter Six) denoted r, c and rc for random, clustered and random clustered respectively. However, although this offered up some level of variation between the datasets, in contrast to the datasets used in Chapters Three through Five, the number of customers was always 100.

[4] Given its artificial status one need no longer consider the parameter T_{ac} (i.e. T_{ac}=0)

The lack of variance in the problem size is an interesting quandary, with a potential for over-specialising the solution technique; resulting in the possibility that the optimised algorithm will not be significantly robust (for the more variable problem sizes likely to be encountered in real-life).

To prevent this from becoming a problem, and to offer up a greater challenge, Homberger and Gehring (1999) created similar datasets but with 200, 400, 600, 800 and 1000 customers. They adhered to Solomon's structure of creating datasets with different customer distributions (r, c and rc) as well as what Solomon referred to as benchmark type. Chapter Six detailed the two types of problem: Type 1 and Type 2, which have tight and loose time windows respectively.

For each customer size, customer distribution and type, ten datasets were created. In total this means that there exists 300 Large Scale VRPTW benchmarks. Given the NP-hard status of the VRPTW, these are significantly more difficult problems (than the 100 customer datasets) to produce good solutions[5] for.

As before, attention is only being given to the Type 2 problems, as the algorithm being adapted was created for these. A set of test benchmarks is made from the first dataset in each of the customer distributions for 200 and 400 customer sizes. These are made to appear like dynamic datasets by artificially creating an appearance time in much the same way as in Section 6.2.1 (though one no longer needs to include T_{ac} here either).

Although this may well not be the ideal way in which to make a dataset artificially dynamic (alternatives are considered in Section 7.2), it should offer an insight into whether this decomposition technique is indeed a realistic alternative to tackling the entire problem at once.

[5] The current best results achieved on these datasets are available at http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html

7.1.3 Comparing the Traditional and Timeslot Solution Techniques

As with any new idea, one must be assured that the technique has merit before conducting a significant amount of research into the area. Partially supporting this work is the high standard of result achieved on the 100 customer datasets in Chapter Six. However, one must also consider whether a similar level of performance will be achievable in the larger datasets that were introduced previously.

Initial focus is going to be on the 200 and 400 customer problems, as these are likely to be able to identify whether the method is worth pursuing; particularly within the ACO framework currently being considered. This is significant because as explained previously ACO, although very effective at producing good results, is one of the slower metaheuristics. The multiple time-consuming constructions, coupled with long improvement phases could be problematic when one has only 12.5 minutes within which to produce a solution.

Traditional Solution Technique

With a seemingly well performing DVRPTW method produced in the previous chapter (the ERWACS-DVRPTW), one need only produce a complementary algorithm for the VRPTW to be able to conduct the necessary comparison.

In Section 4.4.1 it was shown that when solving ts_0 of the DVRP, one was in fact solving a traditional CVRP. The case is similar when time windows are introduced; with ts_0 of the DVRPTW solved as it were a VRPTW. Therefore, were one required to solve the problem using the traditional technique (i.e. treating the problem as a single entity), one need only assume all customers are known from the start and solve according to ts_0 of the ERWACS-DVRPTW.

Results for the test datasets of 200 and 400 customers (produced using the traditional solution technique) can be seen in Table 7.1 and 7.3 respectively (with run-times shown in Tables 7.2 and 7.4). The algorithm was run with and without the improvement phase, as one will need to examine the balance between the number of solutions constructed and the improvement phase, when dealing with the larger datasets. To help gain an understanding of this balance, the implementations with the improvement phase were run to a local optimum instead of terminating the algorithm

when 12.5 minutes have elapsed (the number of cycles completed is also shown). Note that the best-achieved results for these datasets (non time-constrained) have also been presented.

		r2_2_1		c2_2_1		rc2_2_1	
Descent		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
	Min	6	3872.42	**6**	**1931.44**	7	2985.21
Yes	Ave	6.1	3945.11	6.1	1934.20	7.8	2983.56
	Max	7	3747.71	7	1959.01	8	3069.70
	Min	**5**	**5142.08**	**6**	**1931.44**	**6**	**4259.47**
No	Ave	**5**	**5502.48**	**6**	**1951.34**	**6.8**	**4458.80**
	Max	**5**	**5728.12**	**6**	**2033.93**	**7**	**4801.30**
Best		*4*	*4501.80*	*6*	*1931.44*	*6*	*3103.48*

Table 7.1: Traditional Solution Technique Applied to 200 Customer Problems.

	r2_2_1		c2_2_1		rc2_2_1	
Descent	Time	Cycles	Time	Cycles	Time	Cycles
Yes	13.21	35	12.55	427	12.80	32
No	12.50	16888	12.50	18830	12.50	16219

Table 7.2: Run Times for Traditional Solution Technique Applied to 200 Customer Problems.

		r2_4_1		c2_4_1		rc2_4_1	
Descent		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
	Min	12	8634.42	**12**	**4116.14**	16	6467.66
Yes	Ave	12.1	8816.12	13	4191.44	16.7	6559.13
	Max	13	8503.09	14	4242.80	18	6624.73
	Min	**10**	**15781.20**	**12**	4236.25	**14**	**13119.23**
No	Ave	**10.3**	**16382.84**	**12.4**	**4336.49**	**14.7**	**13449.34**
	Max	**11**	**16883.72**	**13**	**4641.59**	**15**	**13788.44**
Best		*8*	*9257.92*	*12*	*4116.05*	*11*	*6834.02*

Table 7.3: Traditional Solution Technique Applied to 400 Customer Problems.

The analysis of these results is fairly basic at this point as one only has the best-achieved solutions to use as a reference point. The significant comparison is the one that is to be made with the timeslot solution technique results (and this will follow this

analysis). However, that does not restrict one from gaining a greater understanding of the workings of this algorithm, particularly with regards to the improvement phase.

	r2_4_1		c2_4_1		rc2_4_1	
Descent	Time	Cycles	Time	Cycles	Time	Cycles
Yes	35.28	3	13.05	14	21.58	3
No	12.50	3020	12.50	3050	12.50	2944

Table 7.4: Run Times for Traditional Solution Technique Applied to 400 Customer Problems.

The result quality is clearly considerably lower than the best-achieved, and the algorithm in its current form (with the traditional solution technique) is not appropriate for producing competitive solutions. It is hoped that the timeslot solution strategy will be capable of producing higher quality results. Some more general observations about the performance are now noted.

It is immediately apparent that the better solutions (remembering the number of vehicles deployed is the primary concern) are achieved when the improvement phase is removed. The ants are requiring more cycles to achieve good solutions than they are allowed when the improvement phase is included. This is in contrast to the DVRPTW implementations in Chapter Six, and this is likely to be due to the significant increase in problem size. Not only has the global number of customers increased by 100% or 300%, but all of the customers are being considered at once (as opposed to a maximum of approximately half in the DVRPTW).

The distances of the solutions when the improvement phase is included are markedly shorter than when it is removed, which although using a different number of vehicles, is indicative of the fact that the solutions without an improvement phase are unlikely to be close to a local optimum.

Of perhaps most interest is the time the algorithm took to run when the improvement phase was included and the effect this had on the number of cycles that could be completed. This time impact is less felt in the 200 customer problems than it appears to have been in the 400 customer problems, though this is somewhat expected due to the associated exponential increase in the neighbourhood size (with the descent being

run to a local optimum). The increase in time is particularly large for the case of r2_4_1, where the final cycle, which includes a single improvement phase, has taken over 22.78 minutes to complete. Note that none of the solutions that included the improvement phase were suitable for the time-constrained variant of the problem.

With regards to the number of cycles (each consisting of three constructions) that were completed; one can see that the introduction of descent caused this to fall by an average of 99.11% and 99.78% (for the 200 customer and 400 customer problems respectively).

An analysis of a single ant-constructed solution for the r2_4_1 dataset being improved via the extended CROSS-Neighbourhood to local optimum is now presented.

Analysing the Improvement Phase (Traditional Solution Technique)

The impact that the inclusion of the improvement phase has had on performance is so significant that it seems sensible to try to present as many details as possible regarding its behaviour. To aid this investigation, a new, single improvement phase on the first cycle of the r2_4_1 dataset (the most affected) is to be investigated further.

In this example there were 251 iterations of the BI descent strategy completed before the local optimum was successfully identified. This improvement phase successfully reduced the distance of the solution by 11158.41. The composite moves of the neighbourhood are split, and the associated impact of each is presented in Table 7.5.

	x:L,L	i:L,L	x:L,0	i:L,0
No. Moves	74	12	131	34
Total Change in Cost	-4182.96	-212.29	-6297.32	-465.84
Average Change in Cost	-56.53	-17.69	-48.07	-13.70

Table 7.5: Neighbourhood Breakdown for an Improvement Phase on r2_4_1

One can see that the moves that occur within a single vehicle (intra) are less frequent and make less of an impact to the cost than those that move customers from one vehicle to another (cross). This is somewhat expected as there are many more potential cross vehicle moves in the neighbourhood. Given that the entire

neighbourhood is evaluated prior to each and every BI descent move, one can not only understand why it is so time consuming, but why each of the neighbourhood components is so important.

The order in which these moves are made is also of interest, as it may be possible to separate the two types of moves into disjoint phases. Some researchers (e.g. the sweep algorithm of Gillet and Miller (1974)) suggest using a clustering-first, routing-second approach when solving routing problems. Were such a divide to be naturally occurring, there may not be a need to evaluate the entire neighbourhood prior to every move. It should be remembered that evaluating all of the cross vehicle moves is much more time consuming (due to the higher number of potential moves) than the time required to evaluate the relatively limited number of intra vehicle moves.

The order in which the neighbourhood moves are made is shown in the upper part of Figure 7.2 (i:L,L, x:L,L, i:L,0 and x:L,0 from top to bottom). The lower part (which has a matching x-axis) shows the associated impact that each particular move had on the distance.

One can immediately see that the cross vehicle moves are more prevalent than the intra vehicle moves in the early iterations of the improvement phase. However, once some form of initial clustering has been identified, they appear to be rather arbitrarily interspersed among the remainder of the neighbourhood moves. With little evidence to suggest the behaviour is particularly disjoint, there does not appear to be any obvious changes that need to be made to how one evaluates the neighbourhood.

With regards to the size of the improvement, the moves follow the general trend that one would expect. Moves make less of a reduction to the distance, the greater the number of iterations that have been conducted. Given the size of the improvements found in the earlier iterations, one can see just how poor some of the decisions made in the RPR were, when the solution was constructed.

With the knowledge that the improvement phase is incredibly successful at reducing the cost, but painstakingly slow, one can turn to the timeslot solution technique to see if it offers any improvement to the traditional solution technique.

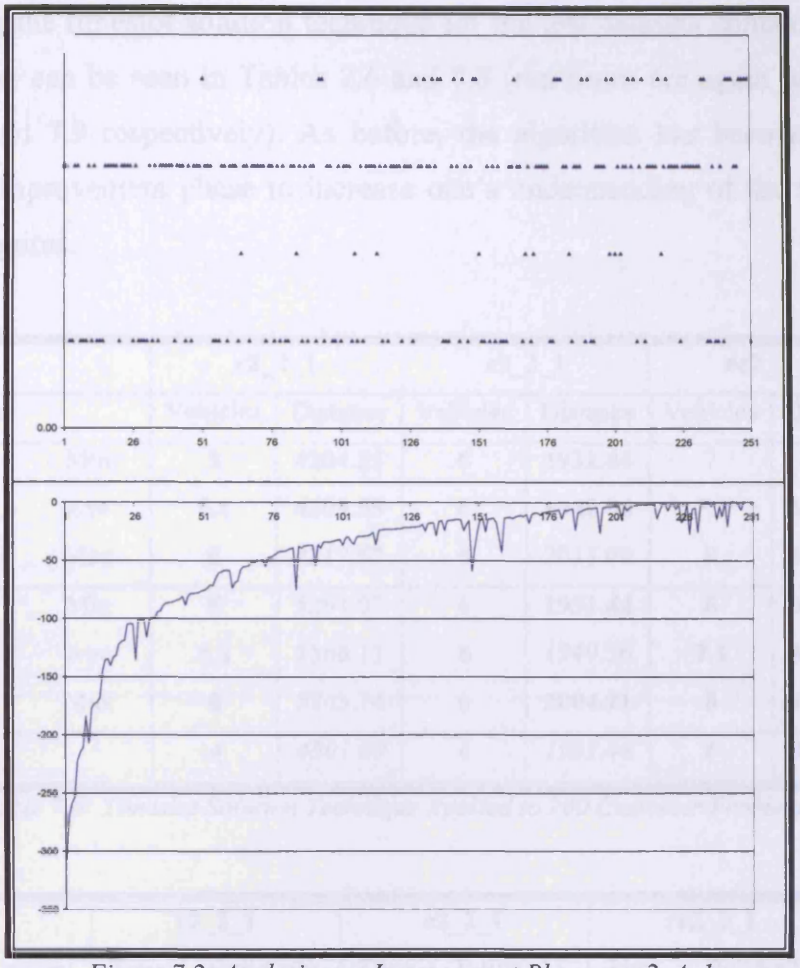

Figure 7.2: Analysing an Improvement Phase on r2_4_1

Timeslot Solution Technique

This section aims to solve the same problems as the traditional solution technique, using the algorithm that was developed in Chapter Six. As one is employing an algorithm for the DVRPTW on the VRPTW (or more specifically the Large-Scale VRPTW) one needs to transform these problems into artificially dynamic problems. It is important to realise that this is just being suggested as a strategy for solving the Large-Scale VRPTW i.e. it is not creating the Large Scale DVRPTW.

It has been discussed how one can make a non-dynamic dataset dynamic in Chapter Six, and given the quality of the results (compared to the equivalent static problem) it seems sensible to continue with the same technique of adding dynamism. The algorithm is exactly the same as that which was established in Chapter Six (and used to solve the entire problem as if it were ts_0 in the previous section).

Results using the timeslot solution technique for the test datasets containing 200 and 400 customers can be seen in Tables 7.6 and 7.8 (run times are again presented, see Tables 7.7 and 7.9 respectively). As before, the algorithm has been run with and without the improvement phase to increase one's understanding of the time that the algorithm requires.

		r2_2_1		c2_2_1		rc2_2_1	
Descent		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
Yes	Min	**5**	**4204.25**	**6**	**1931.44**	7	3081.26
	Ave	**5.1**	**4308.55**	**6**	**1940.06**	**7.1**	**3175.84**
	Max	**6**	**4217.57**	**6**	2017.60	**8**	**3188.05**
No	Min	**5**	5291.07	**6**	**1931.44**	**6**	**3909.70**
	Ave	5.2	5560.11	**6**	1949.36	**7.1**	4055.12
	Max	**6**	5743.74	**6**	**2004.21**	**8**	4259.89
Best		*4*	*4501.80*	*6*	*1931.44*	*6*	*3103.48*

Table 7.6: Timeslot Solution Technique Applied to 200 Customer Problems.

	r2_2_1		c2_2_1		rc2_2_1	
Descent	Time	Build(x3)	Time	Build(x3)	Time	Build(x3)
Yes	11.38	N/A	12.50	N/A	10.62	N/A
No	11.00	N/A	12.50	N/A	10.50	N/A

Table 7.7: Run Times for Timeslot Solution Technique Applied to 200 Customer Problems.

These solutions require comparison with two other sets of solutions. Firstly, one ought to consider how they compare to the best-achieved solutions, in much the same way as the traditional solution technique ones were. Secondly, and more importantly with regards to the direction of this chapter, one needs to make the comparison between these solutions and those produced by the traditional solution technique.

The results produced via the timeslot solution technique are of a considerably lower quality than the best-achieved results. If one were to wish to continue to work with this solution technique and algorithm, it is clear that significant improvements will be required to make it competitive. Some more general observations are now presented.

		r2_4_1		c2_4_1		rc2_4_1	
Descent		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
Yes	Min	11	9876.46	**13**	**4328.74**	15	7453.98
	Ave	12	9719.72	14.1	4471.43	16.1	7340.59
	Max	13	9774.44	**15**	**4823.12**	17	7213.67
No	Min	**10**	**13218.43**	**13**	5059.89	**14**	**10659.84**
	Ave	**10.2**	**13842.95**	**13.9**	**5590.69**	**15.4**	**11099.36**
	Max	**11**	**14155.30**	**15**	5916.95	**16**	**11702.62**
Best		*8*	*9257.92*	*12*	*4116.05*	*11*	*6834.02*

Table 7.8: Timeslot Solution Technique Applied to 400 Customer Problems.

	r2_4_1		c2_4_1		rc2_4_1	
Descent	Time	Build(x3)	Time	Build(x3)	Time	Build(x3)
Yes	17.52	N/A	13.93	N/A	15.96	N/A
No	11.02	N/A	12.03	N/A	11.52	N/A

Table 7.9: Run Times for Timeslot Solution Technique Applied to 400 Customer Problems.

One can see that the result quality is variable amongst the six datasets being analysed. In contrast to the traditional solution technique there is a discrepancy in performance between the 200 and 400 customer problems. The 200 customer problems appear to benefit from the inclusion of the descent phase, whereas for the 400 customer problems (in much the same way as the results produced via the traditional solution technique) it has a negative effect. This is an interesting observation as it suggests that the decomposition technique has reduced the problem size to such an extent that the neighbourhood does not become too time consuming (i.e. prevents a sufficient number of cycles from being completed).

As with the previous analysis, it is also important to see whether it was possible to produce the solutions within the time necessary for the solutions to be classified as for the time-constrained variant of the problem. One can see that without the improvement phase (as in the traditional solution technique) all of the solutions were constructed within the 12.5 minute time limit. When the improvement phase was included, the solutions for the 200 customer problems were successfully produced within time, though the 400 customer problems were not. This is likely to be due to the larger neighbourhoods in the 400 customer VRPTW-like sub problems.

In comparison with the traditional solution technique, one needs to consider if better results can be achieved via the problem decomposition. However, one is not simply judging which of the results are better, but also if there is enough potential in the decomposition results to warrant further investigation.

If one first considers the results without the improvement phase, there is little difference in the 200 customer results. The 400 customer results appear to be better when one considers the whole problem, which is in contrast to the type of behaviour one might expect to observe (as the larger problems are more complex to solve). However, one must also consider the fact that the artificially created VRPTW-like sub problems for the 400 customer problems will be larger than for the 200 customer problems, and that with the time restriction, one might be struggling to identify a good solution in some of the more complex timeslots.

When one considers the inclusion of the descent phase, the results are more interesting. The results for both the 200 and 400 customer problems are of a higher standard when the timeslot solution strategy is being utilised. The 400 customer problems are particularly interesting as one can see how the times (although above 12.5 minutes) are much closer to the imposed time limit than when one considers the whole problem. This suggests that if one wishes to include an improvement phase similar to the one currently being employed, one would favour the timeslot solution strategy (with its smaller sub-problems).

The important observation is that the technique may have some merits; some (relatively) good results have been produced. It appears possible to produce good solutions via time-based decomposition and that it is worthy of an investigation.

It should be noted that even with the timeslot solution strategy (in its current format), the inclusion of an improvement phase is still too time consuming to warrant inclusion in the 400 customer problems. It is temporarily removed (from all analysis) whilst a number of potential improvements are considered, after which, its inclusion will be re-evaluated.

Problem Size

Given the difficulty that the current ACO algorithm is having with the 400 customer problems, it seems that this may represent an upper limit (in terms of problem size) that this algorithm is suitable for. The use of a time-consuming metaheuristic like ACO on a problem where the time constraint is so restrictive (given the size of the problem) is never going to be ideal.

This chapter's primary function is to demonstrate the potential of the technique as a solution method in preference to the traditional solution strategy and one would hope that this is achievable on the 200 and 400 customer datasets. However, the 600, 800 and 1000 customer datasets should not be ignored, and as such, a single greedy construction and improvement phase will be assessed in Section 7.4.

7.2 Improved Artificial Dynamicity

Perhaps the most interesting area (in addition to the algorithm being used) that the user has control over when solving this problem, is the way in which the customers are chosen to appear to be dynamic. In the DVRP and DVRPTW the dynamicity was inherent in the problem definition and therefore was not a variable over which one had control. The same is not true when artificially creating a dynamic scenario as a way of solving a large scale VRPTW; the dynamicity can be added in whatever way the user sees fit.

7.2.1 Artificial Dynamicity: How Best to Create it

The dynamicity in the previous analysis (whereby one was investigating the potential of the technique) was added according to the same method outlined in Section 6.2. Though this did showcase the merits of the technique, there was nothing to suggest that this represented the best way of making the datasets appear dynamic.

It seems sensible to assess whether some other way of adding dynamicity results in solutions of a higher quality being achievable. There is no longer a need to try to ensure ≈50% of the customers are dynamic, but one must still ensure that customers are either known from the start or at the latest e_i-(2×T_{ts})-$c_{i,0}$ (to guarantee that feasible solutions are achievable).

Two new methods of dataset construction are now considered (along with the existing technique), to see if any improvements can be identified. The reasoning behind these methods selection, along with details regarding their implementation are now presented:

- A: As in Chapter Six (and previous analysis)
  This technique has been continued with, given the promising results that were achieved in Chapter Six. Although the results at the beginning of this Chapter were not as favourable as those, they represent a foundation from which one could build. The technique has been well discussed in Chapter Six and importantly, is still trying to achieve a DOD of ≈50%.

- B: random (0, $(e_i-2\times T_{ts}-c_{i,o})$).
  This technique retains certain similarities with Method A, but discards the first step i.e. 50% of the customers are not determined as being known in advance. The method simply assigns an arrival time according to the formula above, though if this figure is negative, the customer is considered immediately (i.e. it is assumed to be known from the start).

- C: $(e_i-2\times T_{ts}-c_{i,o})$
  This is the only method that does not include any form of random element. This method could be dubbed a 'last-chance' method, whereby each customer is only included for consideration (i.e. becomes known) at the latest possible point that still guarantees feasibility. The details as to why this calculation ensures this, has been well documented in Chapter Six.

These methods (and all the subsequent analysis, until the final results are presented) are only conducted upon the 400 customer test benchmarks (as opposed to the 200 and the 400 like previously). These should provide an adequate test-bed, given their greater level of complexity, whilst minimising the amount of results to be produced; the run-times prohibit one from conducting a more through analysis.

Given that these methods will alter the size of the problem tackled in each timeslot, it further supports the idea of conducting this analysis without the inclusion of descent. This will allow each of the algorithms to be concluded in the predetermined time limit (allowing a fair comparison), whilst it will produce results of a higher quality (given that they are to be primarily judged on the number of vehicles deployed).

The results achieved via these three construction methods are displayed in Table 7.10.

Creation		r2_4_1		c2_4_1		rc2_4_1	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
A	Min	11	9876.46	13	4328.74	15	7453.98
	Ave	12	9719.72	14.1	4471.43	16.1	7340.59
	Max	13	9774.44	15	4823.12	17	7213.67
B	Min	9	12089.88	**12**	4178.52	13	9607.14
	Ave	9.3	12535.11	12.5	4294.28	14.2	9857.44
	Max	10	13047.99	13	4483.89	15	10247.90
C	Min	**8**	**10896.5**	**12**	**4130.06**	**12**	**8467.35**
	Ave	**8.3**	**11325.7**	**12**	**4201.02**	**13.1**	**8932.51**
	Max	**9**	**11799**	**12**	**4274.69**	**14**	**9278.26**
Best		*8*	*9257.92*	*12*	*4116.05*	*11*	*6834.02*

Table 7.10: Analysing Various Ways of Adding Artificial Dynamicity to Large Scale VRPTW

It is immediately apparent that the method that removes the random element from the artificial appearance time (Method C) allows the ERWACS-DVRPTW the best chance of producing competitive solutions. This 'last-chance' method is consistently producing results of a higher quality than either of the other two methods and all subsequent analysis will make use of this technique. It should be noted that this improvement is likely to be less apparent in the 200 customer problems as one would expect the results for these datasets to be closer to the best-achieved across each of the three methods.

Comparing the best results of Method C with the best-achieved for these datasets (using a traditional solution technique), one can see that the same numbers of vehicles were deployed for r2_4_1 and c2_4_1, while rc2_4_1 only required one vehicle more. The distance values for r2_4_1 are still some way away (17.70%) but they are

remarkably competitive in c2_4_1 (0.34%). A distance comparison on rc2_4_1 is not possible due to the increased number of vehicles that have been deployed.

Obviously one cannot be solely concerned with the best results from the trial of ten, and consideration must be given to the other metrics presented. The algorithm achieved the best number of vehicles in seven of the ten runs for dataset r2_4_1, whilst it always achieved it on c2_4_1. Unfortunately rc2_4_1 (the one which the algorithm has performed worst on) also has the most variable results, with three vehicles more than the best-achieved, in the maximum of the ten runs.

However, this analysis was concerned with identifying which technique is the best for making the datasets artificially dynamic and it is clear that Method C (last-chance) is the best performing. Before looking at ways in which the algorithm can be adjusted to improve results, it seems sensible to investigate why this new way of creating the artificial appearance times has such a significant impact upon the quality of the solutions.

7.2.2 Customer Distribution

The marked improvement in solution quality due to the use of a different way of making the datasets artificially dynamic is particularly interesting, as the improvement is not associated to any alteration in the algorithm being used to solve the problem. Altering the problem's make-up is a tool that is not available in traditional combinatorial optimisation problems, as these details are usually inherent in the problem definition or benchmark datasets. It should be noted that the only other time in this analysis that one will again have control over this type of component is when the number of timeslots is analysed (given that one includes the length of the timeslot in the artificial appearance time).

It is important that the best is made of this opportunity to exhibit such a high level of control on the way in which the problem is solved. The selection of the best method was a fairly simple choice (given how consistently well it performed), but little reasoning for this performance has been suggested. This section seeks to redress this, and offer up an insight into why this method is likely to have performed so well.

One must remember that the central theme to this Chapter is time and the obvious lack of it (for tackling such sizable problems). Transforming the problem into an artificially dynamic scenario is proposed as a way to make the most of the time available by only considering smaller sub-problems (as the problem is NP-hard i.e. the relationship between problem size and complexity is not linear). However, the size of these sub-problems also needs to be considered, as one only has 30 seconds of computation time (as there are currently 25 timeslots) in which to produce the best solution possible.

Obviously all of the customers need to be dealt with at some point by the algorithm, but once a customer is committed to, they are removed from the problem and are not considered again. If one reduces the time between their artificial arrival time and the time that they are committed to, one reduces the complexity of the subsequent timeslots.

If time were not such an important factor, one would not suggest such a policy as it reduces the time available to identify the best placement for a particular customer. However, with strict time limits it seems sensible to try to minimise the complexity of each timeslot, thus ensuring the maximum number of cycles can be completed in the time available.

Consideration is now given to the size of the problem in each timeslot and the number of cycles that can be completed in the time available. See Figure 7.3 for details of Method A and Method C (so that a comparison can be made). Note that these graphs show the number of customers that need to be scheduled (bar chart, left y-axis) and the number of cycles that were completed (line, right y-axis) for each timeslot.

In the case of Method A, one can see how ≈50% (≈200) customers are known in ts_0 and that the size of the problem appears to remain somewhat constant for a few timeslots before it begins to tail off. Given the large number of customers known in the early timeslots, the low number of cycles completed is to be expected. When the problems sizes do begin to reduce, the number of cycles completed increases considerably.

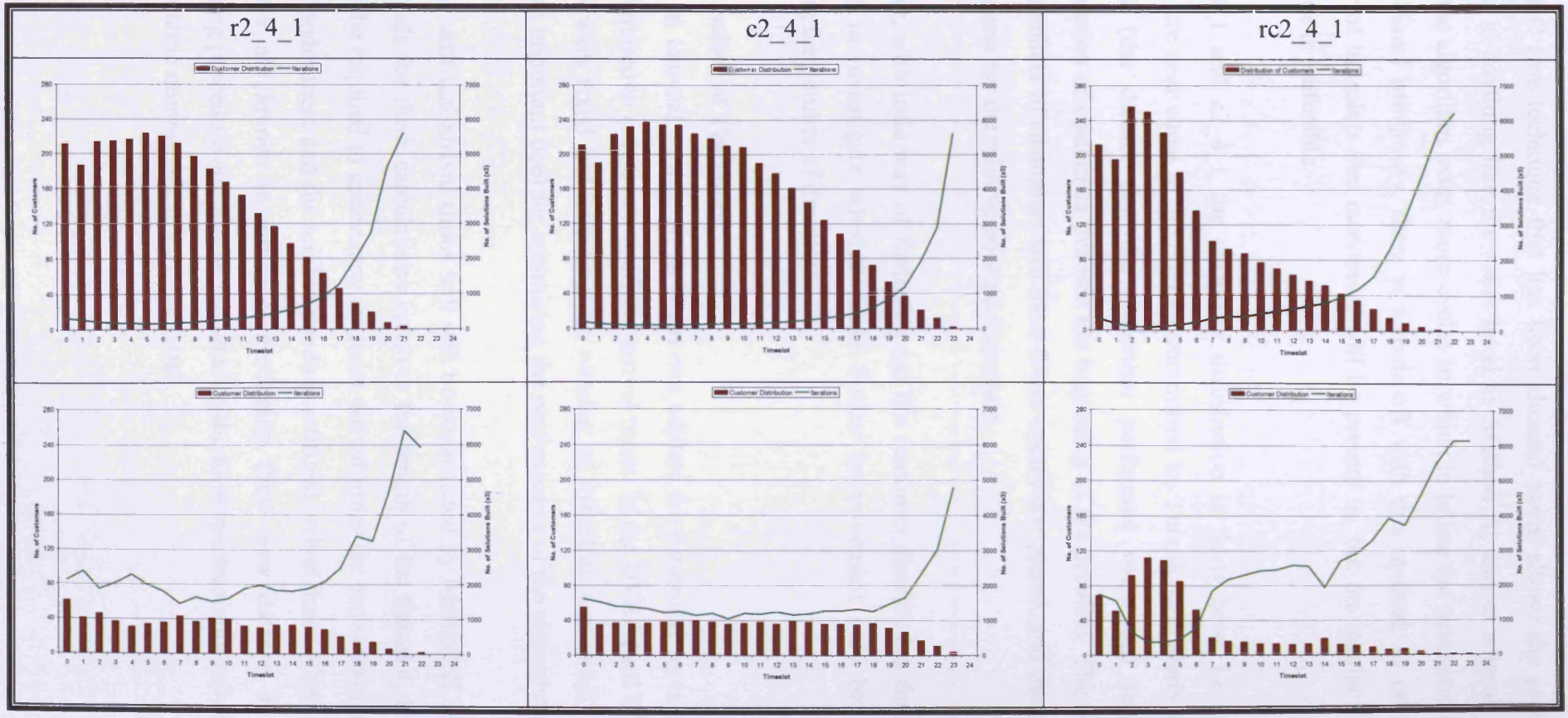

Figure 7.3: Customer Distributions for Artificial Arrival Times (Method A and Method C)

Method C (the technique that has been chosen) never allows the problem size to increase to anything like the same level as Method A, and as a result of that, one allows the algorithm many more cycles in which to locate the best solutions possible. As explained previously, there is a trade-off with this method, as one reduces the number of timeslots that customers will be present in, but the results show that this technique is preferable.

In r2_4_1 and c2_4_1 the customer distribution is fairly level i.e. the customer arrivals are near equal to the number committed to. This is particularly interesting as rc2_4_1 (the dataset that the algorithm performed worst on) still exhibits a concentration of customers towards the beginning of the problem. The timeslots with larger numbers of customers will have fewer cycles completed, and this may offer up some reason for the relative poor performance.

However, with little way of flattening out the customer distribution further, it seems sensible to investigate whether some further improvement may be available by adjusting the number of timeslots.

7.2.3 Number of Timeslots

This is an unusual parameter, as when one adjusts the parameter n_{ts} in this problem, one is required to create new artificial arrival times. In the DVRP (and DVRPTW) the datasets were fixed and adjusting the number of timeslots was simply a parameter (albeit an important one) for optimising the performance of the algorithm.

The new artificial arrival times will still be constructed by Method C, but given that the formula for these calculations involves the length of the timeslot, it is clear why one will be required to create new artificial arrival times for each customer. Given the lack of randomness and the need to produce artificial arrival times where ≈50% of the customers are known in advance, producing these new datasets is not a time consuming procedure and can be included in the time spent solving ts_0 with almost no impact on the number of cycles completed.

When considering which different values for n_{ts} to assess, it seems sensible to mirror those assessed by Montemanni et al. (2005). These values (10 and 50) will allow one to see if more or less timeslots are beneficial; note that given the time-constrained nature of this problem the computational time per timeslot will be 75 seconds and 15 seconds respectively.

The impact of these new artificial arrival times is going to alter the graphs that were seen in Figure 7.3, and if either of these values is selected, the associated graph will be produced. The results for this analysis can be seen in Table 7.11.

n_{ts}		r2_4_1		c2_4_1		rc2_4_1	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
10	Min	**8**	11354.95	**12**	4142.50	14	8880.73
	Ave	8.4	11510.44	12.1	4228.73	14.7	9285.79
	Max	**9**	11547.69	13	4380.04	15	9627.24
25	Min	**8**	**10896.51**	**12**	**4130.06**	**12**	**8467.35**
	Ave	**8.3**	**11325.73**	**12**	4201.02	**13.1**	**8932.51**
	Max	**9**	11798.97	**12**	4274.69	**14**	**9278.26**
50	Min	**8**	11202.41	**12**	4138.53	**12**	9239.68
	Ave	8.6	11230.34	**12**	**4196.89**	13.3	9193.75
	Max	**9**	**11111.56**	**12**	**4241.69**	15	9213.77
Best		*8*	*9257.92*	*12*	*4116.05*	*11*	*6834.02*

Table 7.11: Analysing Various Values for n_{ts} (New Dataset Creation Required)

It is clear that n_{ts}=25 is the best performing of the different parameter values assessed, though the variability across all three values is not that significant. The only particularly bad set of results produced was on the rc2_4_1 dataset when n_{ts}=10. This was the dataset that the algorithm was struggling on most when n_{ts}=25, and as discussed previously, the most probable reason for this was the high concentration of customers in the early timeslots (see Figure 7.3). Reducing the value of n_{ts} is likely to further accentuate this problem, as the number of customers in the early timeslots will be even larger.

Given the argument above, it may seem sensible to assume that by increasing the number of timeslots one would expect to see an increase in the quality of solution

produced for r2_4_1. This behaviour is not apparent in the solutions produced above and therefore warrants some discussion.

It is likely the decrease in solution quality is down to a short-sightedness in the timeslot solution technique that only becomes apparent as one tries to decompose the problem too far. The time spent on each timeslot will be very low, and although one should be able to locate high quality solutions (as the sub problems will be smaller), these sub problems will not reflect enough of the global problem for them to sufficiently represent it.

With n_{ts}=25 seemingly representing a good balance between the problem size and the necessary forethought for the entire problem, consideration is now given to how best to use the 12.5 minutes of computing time available.

7.3 Efficient Time Management

The importance of time in this chapter cannot be overstated, and with that in mind, it seems worthwhile considering how to best use the time that one has available. The first idea considered in this section is unique to this analysis, and concerns the distribution of time amongst the 25 timeslots.

Once the time distribution has been optimised, the main topic of this section can be considered, namely descent. The experiments with it in Section 7.1.3 were less than productive (the number of cycles was reduced too much), but with the alteration to the artificial arrival times having been made, it may prove more successful.

It is also worth considering if one can improve the balance between the descent phase and ACO construction phases by imposing some form of restriction on the CROSS neighbourhood. Although the neighbourhood is excellent at reducing the distance it may be too time-consuming in its current guise.

7.3.1 Distribution of Time

In this analysis, it is important to remember that one is no longer solving a real-life dynamic problem and that making these datasets appear dynamic is just a tool for solving the Large Scale VRPTW. In the time-constrained DVRP and DVRPTW the

30 seconds spent on each timeslot was a boundary imposed to allow a direct comparison with Montemanni et al. (2005). When solving these Large Scale VRPTW using the timeslot solution technique, one is no longer restricted to an even distribution of time across the 25 timeslots.

Given that no alteration has been made to the artificial arrival times, the graphs in Figure 7.3 are still valid and can be used to help explain the thinking behind this analysis. It was discussed how the complexity of the problems was greatly reduced (in comparison with Method A) and that the number of cycles one was able to complete in the time available was much higher than it was previously.

However, one still experienced a significant increase in the number of cycles completed when one approached the latter timeslots (as the problem complexity was greatly reduced). Dataset rc2_4_1 also experienced some particularly complex timeslots early in the analysis and the number of cycles that one could complete here was significantly lower than one would have hoped.

This section proposes two new time distributions that will increase the amount of time devoted to the early timeslots (i.e. the complex problems) and reduce the time spent on the latter ones (i.e. relatively simple problems). Although the artificial arrival times are simple to compute, when one is faced with a new problem they would not be privy to the timeslot problem sizes (as the committed customers cannot be known without producing a full and competitive solution). With that in mind, it seems sensible to suggest a general time distribution (See Figure 7.4), as opposed to tailoring it specifically to each dataset. Note that given the difference in the cycles completed across the three datasets, one will not be able to produce a distribution that levels the cycles completed in all cases.

Both of these methods increase the amount of time spent on the early timeslots in an attempt to increase the number of cycles completed when one is faced with a more complex problem. The first method (3 steps) increases the time spent on the ts_0-ts_{11}, while reducing that on ts_{13}-ts_{24} (time spent on ts_{12} remains at 30 seconds). The second method suggested (5 steps) is a more extreme version of this policy (see graph for details).

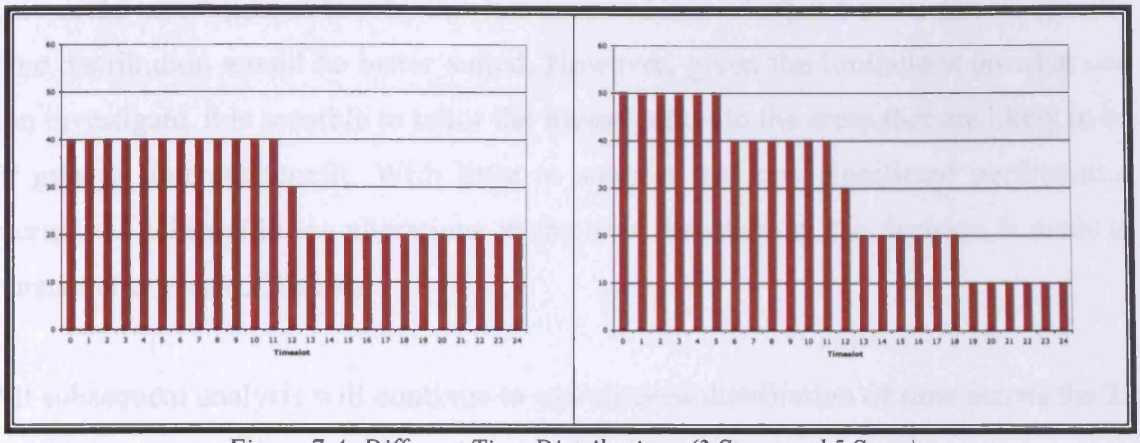

Figure 7.4: Different Time Distributions (3 Steps and 5 Steps)

If either of these two time distributions are successful in producing better solutions than the even distribution of time, then graphs similar to those in Figure 7.3 will be produced. The results of this analysis can be seen in Table 7.12.

Steps		r2_4_1		c2_4_1		rc2_4_1	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
1	Min	**8**	**10896.51**	**12**	**4130.06**	**12**	**8467.35**
	Ave	**8.3**	**11325.73**	**12**	**4201.02**	**13.1**	**8932.51**
	Max	**9**	**11798.97**	**12**	**4274.69**	**14**	**9278.26**
3	Min	8	11888.90	**12**	4182.24	13	9008.41
	Ave	8.6	11911.36	12.3	4330.94	13.3	9503.43
	Max	9	12542.48	13	4592.14	**14**	9849.62
5	Min	**8**	11596.51	**12**	4176.42	13	9136.78
	Ave	**8.3**	11896.36	12.1	4294.75	13.5	9319.95
	Max	**9**	11879.24	13	4456.56	**14**	9633.43
Best		*8*	*9257.92*	*12*	*4116.05*	*11*	*6834.02*

Table 7.12: Analysing Various Time Distributions

The introduction of either of the two new time distributions (3 Steps and 5 Steps) appears to have had a detrimental impact on the performance of the algorithm. The traditional even distribution of time (1 Step) is consistently better across all nine vehicle deployment metrics. The performance of the algorithm on these new time distributions was not significantly worse, and one need not disregard the idea as having any merit on the basis of these results.

One could investigate as to why this has occurred and whether a more dataset specific time distribution would be better suited. However, given the limitations on what one can investigate, it is sensible to tailor the investigation to the areas that are likely to be of greatest interest/benefit. With little to suggest that any significant performance increase is achievable via alterations to the time distribution, the decision is made to pursue other areas of interest.

All subsequent analysis will continue to use an even distribution of time across the 25 timeslots.

7.3.2 Reintroduction of Descent

Identifying the correct balance between constructing solutions (ACO cycles) and improving existing solutions (descent) has been central to much of the success that has been had on the DVRP and DVRPTW. Given the increase in problem size, this balance is likely to be even more crucial than it has been previously.

One cannot rely solely on the ants to reduce the cost sufficiently via multiple constructions, as the intensity of the algorithm is such that it would require considerably more time than one is able to devote to it. However, the descent algorithm is unlikely to reduce the number of vehicles being deployed and given that this is the primary objective one must be careful not to favour distance reductions if vehicle reductions are possible.

An initial analysis of the impact that descent has on the performance was conducted in Section 7.1.2. It was clear that for this problem, one would often not have sufficient time to run the algorithm to a local optimum. However, the new artificial arrival times created in Section 7.2.1 (which reduced the problem sizes in the individual timeslots), also significantly reduces the size of the neighbourhoods, and as such, one may be able to operate with the current neighbourhood.

It may be possible that the neighbourhood is still too large, and as such, it seems sensible to assess whether some restricted form of the neighbourhood would offer a better alternative.

When the CROSS-Neighbourhood was developed, a parameter (L_{max}) was included to control the size of the neighbourhood that one wanted to work with. The parameter represented the largest string that can be exchanged or moved (either within a route or across two routes). In the current form (i.e. the neighbourhood used at the end of Chapter Six) this parameter was set to the highest value possible (i.e. equal to the length of the longest non-fixed section of a vehicle), and as such, no restriction was in place.

This analysis considers reducing this parameter to lower values (e.g. L_{max}=1, as seen in Chapter Three). The lower the value selected, the more restrictive the neighbourhood and (assuming a local optimum is identified) the worse the solutions identified are likely to be. However, the speed at which the neighbourhood locates this optimum will be greatly increased, thus allowing for extra cycles (which are more likely to reduce the number of vehicle deployed) to be completed.

The results of this analysis can be seen in Table 7.13.

L_{max}		r2_4_1		c2_4_1		rc2_4_1	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
None	Min	**8**	10896.51	**12**	4130.06	**12**	8467.35
	Ave	8.3	11325.73	**12**	4201.02	13.1	8932.51
	Max	**9**	11798.97	**12**	4274.69	**14**	9278.26
1	Min	**8**	10412.12	**12**	**4122.75**	**12**	8132.23
	Ave	8.3	10761.68	**12**	4141.12	**13**	8168.25
	Max	**9**	**10517.49**	**12**	4164.35	**14**	8308.10
2	Min	**8**	10536.98	**12**	4123.65	**12**	8278.06
	Ave	8.5	10567.12	**12**	4149.86	**13**	**8135.69**
	Max	**9**	10912.64	**12**	4181.57	**14**	**8179.67**
Max	Min	**8**	**10381.39**	**12**	4123.65	**12**	**7854.51**
	Ave	**8.2**	**10812.50**	**12**	**4133.28**	13.3	8266.48
	Max	**9**	11014.56	**12**	**4144.47**	**14**	8700.00
Best		*8*	*9257.92*	*12*	*4116.05*	*11*	*6834.02*

Table 7.13: Analysing Various Values of L_{max} in the CROSS Neighbourhood

It is important to stress that for each of the values of L_{max}, the time limit did not prevent the algorithm from reaching the local optimum, and as such, each of these methods can be considered purely on result quality. Although the number of cycles that could be completed in the time (30 seconds) differed greatly, the quality of results (in terms of vehicles deployed) is remarkably consistent.

Although it may seem unusual that solutions with a similar number of vehicles were achieved in the runs that were unable to complete a similar number of cycles, the result is understandable. It should be remembered that the improved solution (i.e. after descent) is the solution that is used to update the values in the trail matrix; thus one will be reinforcing arcs from a better solution. This will result in better solutions being more likely in the next cycle, thus reducing the total number of cycles required.

The results clearly show that including some form of descent is a good idea and that this discussion should focus on the identification of the best value for L_{max}. Unfortunately there does not appear to be a consistent trend in the results, with both L_{max}=1 and L_{max}=max seemingly outperforming L_{max}=2.

However, the choice between these parameters is not a difficult one to make. In all of the analysis that has been conducted on this problem, one has been basing decisions on the performance across the three test datasets. Although it is hoped that these adequately represent the entire set of problems (and the performance between these two parameters will be similar on all of the datasets), it seems sensible to err on the side of caution and select the method that is most likely to reduce vehicles ahead of cost (L_{max}=1). This selection is further supported, as it is the best performing on the rc2_4_1 dataset, where minimising the number of vehicles deployed has been troublesome.

It has been shown that the inclusion of descent need not lead to the increases in the number of vehicles being deployed one might expect. Solutions are being produced within the time constraints (due to the smaller sub-problems and the restriction imposed on the neighbourhood) and result quality has been significantly improved.

An additional form of descent (that is unique to this problem) and is near guaranteed to improve results is now considered.

Post Completion Descent

This is a different type of descent that one was unable to apply to the problems that were tackled in the previous chapters. Once a customer has been committed to a vehicle in the DVRP and DVRPTW, that part of the schedule is never altered (due to the real-time environment which it is meant to be simulating). However, as the dynamic nature of this problem is purely a technique being used for problem decomposition, one is perfectly able to adjust the schedule after customers are committed too.

To allow the committed portion of the schedule to be adjusted from one timeslot to the next is going to complicate matters considerably; potentially undoing much of the benefit associated with problem decomposition. One would have to consider the multiple ways in which this could be done and assess the merits of increasing the problem complexity against the potential benefits. If suitably investigated, one may be able to identify some form of improvement in result quality, though it seems unlikely that it would make significant improvements. It goes against the workings of Section 7.2, whereby one was focussed on minimising the problem sizes.

However, the idea of altering the committed components is not without merit, and there does exist a way of implementing this that is perfectly logical. It seems sensible to give consideration to altering the committed customers once the entire schedule has been produced. This brings the problem back to the traditional solution technique but with a solution that was created through the timeslot solution technique.

As one rarely uses the entire 12.5 minutes (as there are often no customers that have not been committed too in the latter timeslots) it seems prudent to make use of this time rather than discard it (as is currently occurring)[6].

[6] In cases where customers are still being scheduled in ts_{24} this technique will not be implemented.

To give an example of this idea, the current best solutions to the r2_4_1, c2_4_1 and rc2_4_1 datasets are completed in 23, 24 and 24 timeslots respectively. This leaves 60 seconds worth of computing time for the r2_4_1 and 30 seconds for c2_4_1 and rc2_4_1 that can be used to improve the solution. It is important to note that this information is not available until a complete solution has been constructed, thus one cannot just spend extra time on each of the timeslots during the timeslot solution strategy.

These problems are going to be of a much larger size than the ones that are faced in the timeslot sub-problems, as they will include every customer (i.e. these datasets have n=400). With that in mind one has to consider how restrictive to be on the neighbourhood. It is unlikely that any but the smallest neighbourhood will be able to identify a local optimum in the time available but that may not be the best strategy to adopt.

Given that the BI descent strategy is being adopted, one may be better running a more complex neighbourhood for less iterations than a simple one for many. Although the solutions will not be local optima (with regards to the more complex neighbourhood) they may improve upon the solutions achieved via the simple neighbourhood.

As this technique is only going to be implemented if timeslots are unused, this post completion descent algorithm will always be run for blocks of 30 seconds. The results of this analysis can be seen in Table 7.14.

One is immediately drawn to the fact that the neighbourhood has not reduced the number of vehicles that have been deployed. This behaviour is to have been expected as the descent phase is concentrating on the reduction of distance rather than reducing vehicles. With regards to its success in reducing the distance, one has to say that it has been successful across all of the values assessed; significant reductions can be seen in r2_4_1 and rc2_4_1 (results for c2_4_1 have been improved but not by such a sizable amount).

L_{max}		r2_4_1		c2_4_1		rc2_4_1	
		Vehicles	Distance	Vehicles	Distance	Vehicles	Distance
None	Min	**8**	10412.12	**12**	4122.75	**12**	8132.23
	Ave	**8.3**	10761.68	**12**	4141.12	**13**	8168.25
	Max	**9**	10517.49	**12**	4164.35	**14**	8308.10
1	Min	**8**	**10115.63**	**12**	4122.75	**12**	7879.71
	Ave	**8.3**	10506.35	**12**	4140.98	**13**	7766.46
	Max	**9**	10310.37	**12**	4164.35	**14**	8017.94
5	Min	**8**	10120.61	**12**	4122.75	**12**	**7496.30**
	Ave	**8.3**	**10414.76**	**12**	4138.81	**13**	**7519.18**
	Max	**9**	**10116.48**	**12**	4164.35	**14**	**7566.21**
Max	Min	**8**	10308.29	**12**	**4116.14**	**12**	7978.85
	Ave	**8.3**	10621.21	**12**	**4127.01**	**13**	7996.05
	Max	**9**	10380.41	**12**	**4136.15**	**14**	8118.37
Best		*8*	*9257.92*	*12*	*4116.05*	*11*	*6834.02*

Table 7.14: Analysing Various Values of L_{max} in the CROSS Neighbourhood (Post Completion Descent)

The results are particularly interesting when one considers which value of L_{max} should be selected. As explained previously it is about getting a balance between neighbourhood size and the time available, however, the quality of the solution prior to this descent phase will have a significant impact upon which value is best suited.

If a high quality solution is produced via the timeslot solution technique, one is likely to want a complex neighbourhood, as identifying moves that improve the solutions are going to be harder to find. Furthermore, a high quality solution is going to require fewer moves to reach a local optimum and as such, the time constraint becomes less of a pressing issue.

It appears that the solution for c2_4_1 is of a higher quality (when compared to the best solutions identified) than the other two datasets, and as such, responds best when the neighbourhood is at its most complex (L_{max}=Max). The other two datasets (whose initial solutions are of a lower quality) seemingly require too much time to reduce the distance under this neighbourhood and are better suited to the restricted neighbourhood L_{max}=5.

It should be noted that for r2_4_1 with L_{max}=1, a local optimum was often identified (on nine out of the ten runs)[7]. A local optimum was not identified on any of the runs when L_{max}=5 or L_{max}=Max. For rc2_4_1 a local optimum was not identified for any of the values of L_{max} assessed (any runs), while for c2_4_1 a local optimum was always identified (for all values of L_{max} and all runs).

If one were faced with a problem that had not been solved numerous times before (i.e. a real-life scenario), immediately identifying the quality of the solution achieved at the end of the timeslot solution technique would be impossible. One would therefore be unable to choose how to restrict the neighbourhood for each particular instance. It would be possible to have some sort of adaptive neighbourhood that learns as this post completion descent phase is carried out, but with time at such a premium, introducing this is likely to be counter productive.

It therefore seems sensible to continue with the most robust setting possible (i.e. the one that will work the most well for all datasets), and that is clearly L_{max}=5. Although the reduced performance on c2_4_1 is undesirable, it seems worth rejecting this in favour of the improved performance in the datasets where the results are not as competitive. The selection of L_{max}=5 reduces the average cost by 3.74% (when averaged across the three datasets).

This concludes the analysis regarding the inclusion of descent, and with it, the work conducted on improving this algorithm. Consideration is now given to how this algorithm performs when one considers the entire set of benchmark problems (n=200 and n=400).

7.4 Large Scale VRPTW Results (200-400 customers)

With an algorithm developed over the course of this Chapter that is seemingly capable of producing high quality results on these test datasets, one needs to consider its performance on all of the benchmark problems.

[7] r2_4_1 has twice as much time as the other datasets to run the post completion descent.

The results for the full set of 60 datasets can be seen in Tables 7.15 and 7.16 (200 and 400 customer problems respectively), accompanied by the best-achieved on these problems[8]. Results have not been produced using the traditional timeslot technique; these are likely to be of such low quality that one need not consider it a worthwhile comparison.

It is important to note that improving upon these results was the focus of this chapter (not bettering the best achieved) and by accomplishing such dramatic advances via the timeslot solution technique (in the test datasets) the investigation has already proved successful.

As discussed previously, the principal reason for conducting this analysis was to identify whether time represented a suitable basis for problem decomposition. It was not in doubt whether the timeslot method used for the DVRPTW could be adapted for the Large-Scale VRPTW, but whether it could produce results of a sufficient quality. Realistically, the method has little merit (outside of it being an interesting theory) if one cannot use it to produce solutions that one might consider using.

In order for one to analyse the potential of the timeslot solution technique, one must have some existing results with which to make the necessary comparisons. Firstly, consideration is given to the results that one would ideally have access to (i.e. those that would allow for a suitably through analysis to be conducted). Secondly, each of these result sets is considered and then is either presented, or a discussion provided, detailing reasons for their absence.

- Traditional Solution Technique (Same Algorithm)
  As all other components of the algorithm would remain constant, these solutions would allow for a direct comparison between the traditional and timeslot solution techniques to be made.

[8] These results were not achieved with any form of time-constraint in place, and as such, comparison of results cannot be made in a purely objective manner.

	Vehicles			Distance			Vehicles	Distance
	Min	Ave	Max	Min	Ave	Max	Best	Best
r2_2_1	5	5	5	4340.93	4583.92	4870.55	4	**4501.80**
r2_2_2	4	4.7	5	3903.73	3852.67	3872.66	4	**3645.38**
r2_2_3	4	4.1	5	3109.29	3207.18	3194.33	4	**2883.16**
r2_2_4	4	4	4	2161.60	2358.62	2551.65	4	**1981.29**
r2_2_5	4	4.2	5	3812.42	4161.59	4112.66	4	**3367.55**
r2_2_6	4	4.2	5	3221.66	3449.73	3660.40	4	**2914.56**
r2_2_7	4	4.4	5	2702.09	3002.02	3242.98	4	**2453.62**
r2_2_8	4	4	4	2026.56	2215.25	2315.49	4	**1849.87**
r2_2_9	4	4.3	5	3469.41	3712.89	3735.77	4	**3111.41**
r2_2_10	4	4	4	3180.41	3349.25	3596.87	4	**2657.00**
c2_2_1	6	6	6	**1931.44**	**1931.44**	**1931.44**	6	**1931.44**
c2_2_2	7	7.2	8	2217.69	2366.95	2606.67	6	**1863.16**
c2_2_3	8	9.3	10	2306.62	2606.19	2860.55	6	**1775.11**
c2_2_4	8	8	8	2266.15	2399.05	2631.15	6	**1705.05**
c2_2_5	6	6	6	1889.70	1901.15	1925.03	6	**1878.85**
c2_2_6	6	6.8	7	2215.98	2310.86	2573.92	6	**1857.35**
c2_2_7	7	7	7	2117.27	2401.85	2833.56	6	**1849.46**
c2_2_8	6	6	6	1891.58	2043.70	2110.37	6	**1820.53**
c2_2_9	7	7	7	2069.56	2485.60	2735.28	6	**1830.05**
c2_2_10	6	6.8	7	1917.21	2110.29	2258.04	6	**1806.60**
rc2_2_1	6	6.1	7	3333.57	3468.61	3596.39	6	**3103.48**
rc2_2_2	5	5.9	7	3144.02	2987.52	2783.86	5	**2827.45**
rc2_2_3	4	4.7	5	2781.88	2728.57	2858.71	4	**2613.12**
rc2_2_4	4	4.3	5	2210.42	2268.45	2302.88	4	**2043.05**
rc2_2_5	6	6.3	7	2835.32	2895.52	2848.78	4	**2912.13**
rc2_2_6	5	5	5	2846.61	2923.42	2988.02	4	**2975.13**
rc2_2_7	4	4.9	5	2760.43	2741.54	2940.12	4	**2529.30**
rc2_2_8	4	4.1	5	2482.24	2603.08	2703.95	4	**2298.12**
rc2_2_9	4	4	4	2393.52	2595.99	2844.62	4	**2175.61**
rc2_2_10	4	4	4	2355.24	2618.62	2918.47	4	**2015.60**

Table 7.15: Solutions for 200 Customer Problems

	Vehicles			Distance			Vehicles	Distance
	Min	Ave	Max	Min	Ave	Max	Best	Best
r2_4_1	**8**	8.3	9	10120.61	10414.76	10116.48	**8**	**9257.92**
r2_4_2	**8**	8.3	9	8719.79	9127.07	9733.44	**8**	**7649.87**
r2_4_3	**8**	8.4	9	7207.07	7497.30	7512.79	**8**	**5988.02**
r2_4_4	**8**	8.5	9	5315.10	5462.07	5824.13	**8**	**4300.95**
r2_4_5	**8**	9	10	9593.90	9629.80	9379.49	**8**	**7143.55**
r2_4_6	9	9.3	10	7847.03	7954.91	8130.58	**8**	**6163.81**
r2_4_7	**8**	8.8	9	6494.54	6582.00	6976.39	**8**	**5082.10**
r2_4_8	**8**	**8**	**8**	4901.44	5150.72	5387.31	**8**	**4051.98**
r2_4_9	**8**	9.2	10	8269.24	8566.87	8470.95	**8**	**6493.13**
r2_4_10	**8**	8.1	9	7317.74	7637.91	7836.28	**8**	**5844.77**
c2_4_1	**12**	**12**	**12**	4122.75	4138.81	4164.35	**12**	**4116.05**
c2_4_2	**12**	12.3	14	4481.93	4870.66	5355.95	**12**	**3929.89**
c2_4_3	15	16.6	18	5450.69	5893.48	6506.40	**12**	**3739.72**
c2_4_4	**12**	13.5	14	4965.00	5193.48	5614.55	**12**	**3535.99**
c2_4_5	**12**	**12**	**12**	4073.72	4183.49	4402.99	**12**	**3939.42**
c2_4_6	**12**	12.8	13	4558.13	4870.38	5273.31	**12**	**3875.94**
c2_4_7	13	13.9	14	4909.58	5317.46	5652.08	**12**	**3894.13**
c2_4_8	**12**	**12**	**12**	4236.93	4412.09	4693.19	**12**	**3787.08**
c2_4_9	13	13.8	14	5004.06	5337.97	5972.83	**12**	**3876.10**
c2_4_10	13	13	13	4345.62	4527.99	4635.52	**12**	**3684.89**
rc2_4_1	12	13	14	7496.30	7519.18	7566.21	**11**	**6834.02**
rc2_4_2	12	12.5	13	6368.26	6611.91	6822.02	**9**	**6355.59**
rc2_4_3	11	11	11	5461.15	5742.20	6002.65	**8**	**5055.02**
rc2_4_4	**8**	8.3	9	4201.36	4393.01	4403.22	**8**	**3635.04**
rc2_4_5	11	11.7	12	6510.28	6466.06	6569.11	**9**	**6063.46**
rc2_4_6	9	10.1	11	6415.51	6513.93	6744.67	**8**	**5997.24**
rc2_4_7	10	10	10	5763.61	6039.50	6265.60	**8**	**5476.57**
rc2_4_8	**8**	8.8	9	5757.61	5688.66	5964.88	**8**	**4854.16**
rc2_4_9	**8**	**8**	**8**	5470.52	5877.61	6298.10	**8**	**4599.57**
rc2_4_10	**8**	**8**	**8**	5191.37	5632.08	5919.55	**8**	**4316.36**

Table 7.16: Solutions for 400 Customer Problems

- Alternative Decomposition Technique (Same Algorithm)

  This would allow one to compare the quality of the decomposition technique with an alternative, thus identifying whether time does serve as a suitable basis on which to base the decomposition.

- Best Known Solutions
  These would allow one to identify whether the solutions produced via the ACS algorithm (with the timeslot solution technique) are of a suitably high standard that one would consider using the technique.

Unfortunately, only one of these datasets is available for this analysis, though the reason for the absence of one of them is not entirely undesirable.

The traditional solution technique could be used to produce results for all of the datasets (similar to those in Table 7.1 and 7.3 for the 200 and 400 customer test datasets) but the results would be of such poor quality, that their production would be somewhat meaningless. The ACS algorithm being used (and the time-consuming descent component) would not allow for high quality solutions to be constructed in the 12.5 minutes that is available. This of course is not an entirely negative point, as one has succeeded in taking an algorithm that used to be inappropriate and too time-consuming to even be considered for implementation on this problem, and made it workable.

Unfortunately, there do not exist any alternative decomposition techniques for the VRPTW (hard-windows) with which one could make the desired comparison. As discussed previously, the lack of any other technique was partly responsible for inspiring the direction that this chapter has taken. As for proposing a second decomposition technique, identifying one was a complex procedure and attempting another with no foundation with which to base it upon, would be outside the remit of this chapter.

It was discussed in Section 7.1.2, how it would not be sensible to use a geographical technique (as seen in the CVRP), and that time represented the most sensible way in which one could proceed. With no other decomposition techniques available, one cannot make this type of comparison.

The best-known solutions are readily available[9] and are the only set of results that one has with which to make any sort of quantitative analysis. These will allow one to interpret the quality of the results, and whether the algorithm and solution technique are competitive. The analysis will be judging the algorithm and the potential of the timeslot solution technique, as without the other sets of results, one cannot separate these two components.

The analysis of the results will begin with a simple numerical analysis comparing the newly produced results from Tables 7.15 and 7.16 with the best-known solutions. Once the two sets of results have been compared, the reasons behind the algorithms performance will be investigated along with explanations offered for the varying levels of performance.

Numerical Analysis

It is immediately apparent that the solutions produced have not bettered any of the best solutions, and that some results are of a markedly lower quality. Although not ideal, this is somewhat to have been expected and the potential reasons for this will be considered in the next subsection. This subsection is purely concerned with investigating how much worse the solutions are, and whether the problems are due to the number of vehicles being deployed, the distance travelled or both.

Given the primary objective of the VRPTW is the number of vehicles that are being deployed; it seems sensible to base the majority of this analysis on this (though a short discussion on distances will be provided at the end of this subsection).

To enable one to interpret the results with greater ease, it seems sensible to consider the number of vehicles deployed above the desired amount (i.e. less the best achieved). Tables 7.17 and 7.18 present these figures (in the form of a count) for the minimum and the maximum solutions from the trial of ten runs.

By not considering the distances that the vehicles must travel (which on first appearances seemed poor), one is immediately given a better impression of the quality

[9] http://www.top.sintef.no/vrp/benchmarks.html

of the results that have been produced. Although still not competitive, they do show that many of the solutions that were produced did use a number of vehicles equal to the minimum that have been used for these datasets.

	0	1	2	3	4	5	6
Min	22	5	3	0	0	0	0
Max	8	17	3	1	1	0	0

Table 7.17: Additional Number of Vehicles from Best Achieved (200 Customers)

	0	1	2	3	4	5	6
Min	19	6	2	3	0	0	0
Max	6	10	8	4	1	0	1

Table 7.18: Additional Number of Vehicles from Best Achieved (400 Customers)

The solutions to the 200 customer problems appear to be of a higher standard (with less variability) than the 400 customer problems, though this might have been expected, as the time-constraint will become more prominent as the size of the problem increases. Considering the best run of the trial, one produced solutions in the desired number of vehicles for 22 of the 30 datasets (for the 200 customer problems) and 19 of the 30 datasets (for the 400 customer problems).

Crucially, this shows that it is possible to produce high quality solutions with this time-based decomposition technique, thus validating the principal behind this work. Although consideration still has to be given to why it manages it in some instances and fails in others (worst case is three extra vehicles required), one can be satisfied that the technique has potential and that with further research one may well be able to develop it further.

Obviously, one cannot focus entirely on the best solution (of the trial), as the user may not have the time to run the algorithm multiple times. In much of the dynamic routing work that has preceded this chapter, there has been a high level of variability in the results. By making these datasets artificially dynamic, one appears to have introduced this problem into the results for these Large Scale VRPs.

Considering the worst trial of the run, 25 of the 30 datasets (for the 200 customer problems) and 16 of the 30 datasets (for the 400 customer problems) have produced solutions that were within one vehicle of the best achievable and could be considered to not be particularly poor. The results that require multiple extra vehicles are more of a concern and do not reflect well on the technique. Were these solutions produced in a real-life scenario, the deployment of extra vehicles could be a great expense, and the possibility of requiring a further six vehicles (as in the worst result of c2_4_3) could be disastrous. Consideration will be given to why this is likely to have occurred in the next subsection.

This analysis concludes with some general figures about the quality of the algorithm when averaged across all of the datasets (though the 200 and 400 customer problems are still considered separately).

When considering the best result of the trial, the number of vehicles deployed is 7.69% higher (200 customer problems) and 6.67% higher (400 customer problems) than the best results that have been achieved. The average number of vehicles is 13.50% higher (than the best-achieved) for both the 200 customer and 400 customer problems.

These results are significant as they suggest that the performance on the 400 customer datasets is better than the 200 customer datasets. Such a conclusion is in direct contrast with the previous analysis, where one was considering the extra number of vehicles required.

As for which is a better measure of performance, it is subjective, and will depend on numerous factors that relate to the real-life constraints that one may have to operate within were they faced with these problems. However, in this analysis it is not necessary to identify whether the performance on one dataset is better than the other, nor is it necessary to identify a favoured measure of performance. As long as one is aware of the different measures of performance, they can happily coexist.

With regards to the distance that the solutions have been produced in, one cannot make much of a comparison between solutions unless they deploy the same number

of vehicles (as seen in Chapter Six). Of the best solutions (from the trial of ten), it has been shown that 22 of the 30 (for the 200 customers) and 19 of the 30 (for the 400 customers) did use the same number of vehicles. The distances for these results are 9.54% and 19.29% higher than the best achieved, for the 200 and 400 customer problems respectively. The reasons for this poorer performance on the 400 customer problems will be investigated in the next subsection, but it is likely to be due to the time-constraint reducing the number of cycles and the impact that the neighbourhood can have on the results.

If the best solution (from the trial of ten) did not produce a solution in the same number of vehicles as the best achieved, the distances were 8.18% and 14.76% higher (for the 200 and 400 customer problems respectively). These results are included for completeness and do not offer much of a further insight into the performance of the algorithm. An interesting observation from these results is that they differ from the results in Chapter Six (where the results that used a higher number of vehicles tended to have shorter distances). This is important as it suggests that the previous performance may not be due to the algorithms being unsuccessfully adapted from minimising distances to vehicles (with distance as a secondary objective).

Consideration is now given to trying to achieve a greater understanding of the reasons why the algorithm behaved as it did, and how one could positively influence its performance were further research to be conducted.

Understanding the Performance

This subsection accompanies the numerical analysis provided previously, and attempts to offer a more subjective look at the performance. Speculative details about why the algorithm performed as it did are presented, whilst some of the techniques potential shortcomings are also considered.

It has been discussed how one would ideally have had three different sets of results with which to analyse the performance of the timeslot solution technique, and how the absence of two of the sets has made understanding the performance a more difficult task. This analysis begins by considering whether the one set available (the best achieved) offers enough of a basis upon which to judge the quality of the results.

The best-achieved results (for the 200 and 400 customer problems) were produced by seven different algorithms, which demonstrates how one algorithm may not be capable of producing very high quality results for every instance in the benchmark datasets. This is a trend that is often apparent in routing problems, with no one algorithm out performing all others on every instance. With this in mind, the production of a single ACO algorithm (regardless of whether it was using the timeslot solution technique) and expecting it to challenge these multiple workings is somewhat unrealistic.

As the timeslot solution technique is the central focus of this work, the other sets of results that were desirable utilised the same ACO algorithm in order to isolate the solution technique. Although these results are not available, it is possible if further research were conducted upon the timeslot solution technique, one would not require this work to be done.

If multiple algorithms (i.e. a variety of metaheuristics) were implemented using the timeslot solution technique, it would enable a set of best-achieved results to be produced, which could then be compared with the performance of the traditional solution technique. This would require significant work outside the scope of this thesis, and as such, is only discussed as it represents a sensible direction for future efforts in this field.

If one were to temporarily forget about the problems associated with critiquing the performance of the algorithm and the solution technique (and the fact that one is not aware which of these components is most responsible for the quality of the solutions produced), one is faced with a further issue regarding the best-achieved results.

As explained previously, the best-achieved results were produced by a variety of algorithms. Unfortunately, this presents a problem with regards to comparing the computational effort required to produce the solutions (as each algorithm makes use of a different computer). In Mester and Bräysy (2005) the times for seven different algorithms are presented. For the 200 customer problems the run-times range form 2.4-182.1 minutes, while for the 400 customer problems, they range from 7.9-359.8

minutes. Details regarding the computers being used to produce these results can be seen in this publication.

One has to consider the issue of time, and whether the two sets of results being compared are for the same problem. By imposing the restrictive time limit of 12.5 minutes on the solution technique, one is actually altering the problem definition, and with it possibly the validity of these results as a like-for-like comparison.

Accompanying the time-constraint is the issue regarding the time-consuming nature of the ACS algorithm being used to produce the solutions. It is clearly not the most appropriate algorithm for solving a problem where one is acutely aware of the difficulty the time restriction is likely to have (for problems of this size). However, this chapter is focusing on the establishment of the timeslot solution technique (rather than the solution algorithm), and as such, it seemed sensible to progress with the ACS algorithm that had been developed in the preceding chapters. One may have to accept a lower level of performance, as one is implementing an entirely new solution philosophy without a bespoke algorithm to accompany it.

As one has seen, there are multiple issues associated with the fact that a single algorithm is unlikely to be competitive across all of the benchmark datasets and the self-imposed time-constraint (which alters the problem definition). However, it is still desirable to consider why the algorithm performed as it did, with particular interest in trying to establish potential reasons for the poor performance on some of the benchmarks.

One of the most obvious reasons for the poor performance is that given each benchmark is different; certain traits of certain problems may well not be suited to a particular algorithm. This is also true of the timeslot solution technique and there may just be problems for which it is not well suited.

With less interest in the algorithms suitability (which was used because it had been developed previously), consideration is given to what traits may make a problem unsuitable for the timeslot solution technique.

It is likely that certain problems may well not lend themselves to being divided into a series of smaller sub-problems because of the way in which the customers arrive, or the need for high vehicle capacity utilisation, which may be difficult to achieve without committing to inappropriate customers. The early timeslots may not be representative of the problem, and this could result in the commitment strategy removing customers from consideration when they are in unsuitable vehicles. Although this can be undone in the post completion descent, it may have resulted in extra vehicles being deployed in the construction, which are difficult to remove in this phase.

Consideration is now given to the dataset's DOD and whether this may have affected the quality of the results produced. This is considered as it relates directly to the dataset creation technique developed in Section 7.2.

The new method developed for making the datasets artificially dynamic was chosen due to the quality of the results that were produced on the test datasets. It appears that the test benchmarks all have relatively low DOD (while some of the datasets are much higher), and as a result, the technique may not have been sufficiently robust to allow the possibility of good solutions to be achieved on every benchmark.

Unlike the work in Chapter Six, one will be concerned solving a problem with too small a DOD. If this occurs, the problem in the first timeslot will be a large problem and possibly too complex to solve to a high standard (this will become a greater issue as the problem size increases). Unsuitable customers could be committed too, and they will adversely affect the quality of the solution produced. It should be noted that the timeslot solution technique does not include any technique for removing vehicles, and as such, poor performance in a single timeslot could ruin the chances of producing a good solution.

Figure 7.5 shows the DOD for each of the 60 benchmark datasets plotted against the number of vehicles deployed (average from the trial) above the best-achieved.

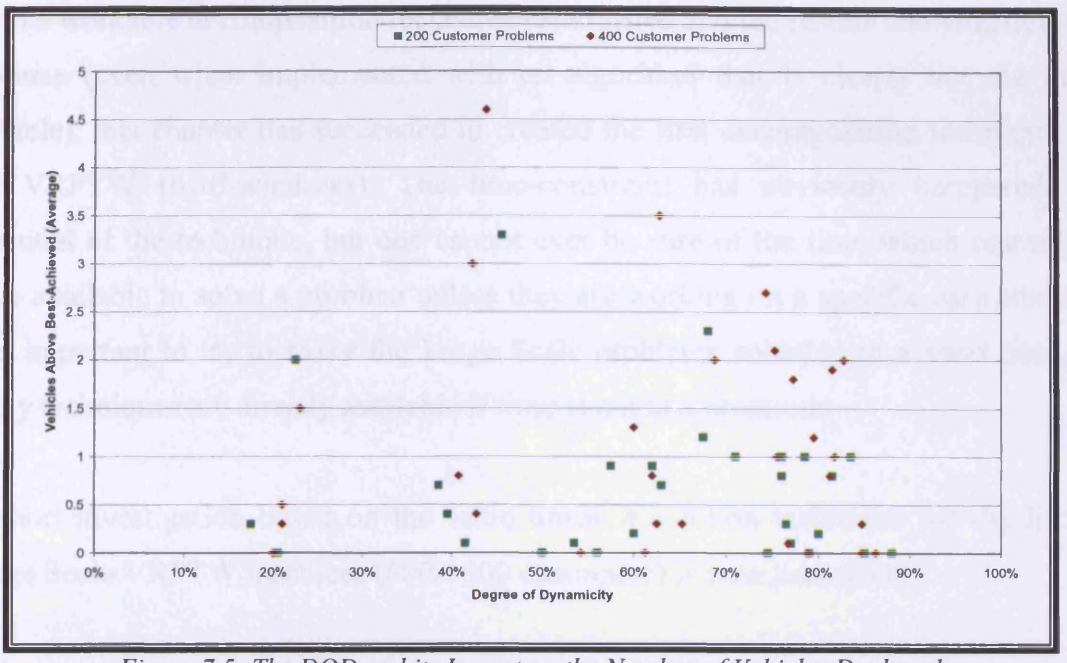

Figure 7.5: The DOD and its Impact on the Number of Vehicles Deployed

As one can see there is no immediately discernable pattern in the graph, with the particularly poorly performing datasets (r2_4_2, c2_2_3, c2_4_3 and rc2_4_2) having a DOD that varies between 42% and 62%. Of possible interest is the fact that 30 of the 60 benchmarks have a DOD >62%, but given the high levels of performance associated with some of the datasets with a DOD <21% there is little in these results to suggest that this warrants further research.

It may be that the datasets with the very low DOD are problems that are not as complex to solve but without delving into what constitutes a good dataset to break into sub-problems, this is little more than speculation.

As the poorer performance tends to be in those with higher DOD, one could suggest that the technique developed (for making the dataset appear artificially dynamic) may not be the best for those that have very wide time windows (i.e. those that are likely to result in a lot of customers being known from the start). It is suggested that were further work conducted in this area, consideration should be given to the Type-1 problems and the idea of basing the artificial arrival times on the closing time of the window for some customers.

With a workable decomposition technique established and the results showing definite promise (even when implemented with an algorithm that is clearly not the most suitable), this chapter has succeeded in created the first decomposition technique for the VRPTW (hard-windows). The time-constraint has obviously hampered the potential of the technique, but one cannot ever be sure of the time which one might have available to solve a problem unless they are working on a specific case study. It was important to try to make the Large Scale problems solvable in a short time, as many techniques are already available if time is not at a premium.

A short investigation based on the same timeslot solution technique for the Extra-Large Scale VRPTW instances (600-1000 customers) is now presented.

7.5 Extra-Large Scale VRPTW (600-1000 customers)

As discussed in the introduction to this Chapter, the Large Scale VRPTW datasets consist of either 200, 400, 600, 800 or 1000 customers. Sections 7.2 through 7.4 developed a suitable ACO solution technique for the 200-400 customer problems based on the work that was carried out in Chapter Six. This section looks to the development of a suitable technique for solving the 600-1000 customer problems.

Preliminary tests show that the ACO algorithm utilised in the previous sections of this chapter cannot be used to produce solutions of an acceptable quality. The number of cycles that one can complete prevents one from learning the way to a good solution. As a result of this, a decision has been made to begin this analysis with a simple LS method. Some improvements to the basic methodology are then proposed, before a complete set of results are produced. Note that this analysis is of a more concise nature (than for the 200-400 customer problems), as the time constraint is so restrictive.

7.5.1 Initial Results

Given the results of the preliminary tests, one is required to consider a less computationally expensive technique for producing solutions. The time limit of 12.5 minutes simply does not allow for any complex method to be considered, as one will often be solving a static VRPTW-like problem with in excess of a hundred customers.

As a result of this, a decision has been made to pursue with the simple NN and descent heuristics (first considered in Section 3.3).

The NN algorithm has been chosen (in preference to a roulette wheel like PNN technique) as it can be shown for a single run that the inclusion of a random element is unlikely to improve the solution quality. The descent algorithm has been a component of much of the ACO and TS work considered previously, and has been shown to improve solution quality.

Note that as the multiple constructions in the ACO algorithms were primarily responsible for minimising the number of vehicles being deployed, it is reasonable to assume that the results for these datasets will be of a noticeably poorer standard (than for the 200-400 customer problems). This is simply a result of the time limit being imposed; were one to consider these problems in a non-time constrained environment, one could continue to complete multiple cycles.

As with the previous analysis, it is sensible to base one's test set of problems on the most complex that one is likely to face (in this analysis it will be n=1000). One would expect a technique that is suitable for these problems to also be suitable for those containing fewer customers. Analysis will be conducted upon a test set of benchmarks created from the first problem in each of the different customer distributions.

With regards to producing an initial set of results using the technique described, one simply needs to select parameter values for the restrictions placed on the string lengths in the CROSS Neighbourhood (both within the timeslots and for the post completion descent). As values for these parameters were identified for use within the ACO algorithm, it seems sensible to continue to use these (L_{max}=1 and L_{max}=5) and assess whether any revision is necessary. Results are presented in Table 7.19, along with the best-achieved results[10] (for comparative purposes).

[10] One should be aware that the time used to produce these is considerably greater than 12.5 minutes.

	r210_1			c210_1			rc210_1		
	Vehicles	Distance	Time	Vehicles	Distance	Time	Vehicles	Distance	Time
Pre	23	60786.2	0.41	30	16992.2	0.1	25	39345.3	8.45
Post	23	58670.6	4.05	30	16992.2	0.16	25	35650.5	12.5
Best	*19*	*42467.9*	*NA*	*30*	*16879.2*	*NA*	*21*	*29754.1*	*NA*

Table 7.19: Pre and Post Completion Descent Results

The immediate observation is that four extra vehicles are being deployed (than the best achieved) in r210_1 and rc210_1. Clearly these are significant increases if one was to consider this as a real-life situation e.g. the costs associated with four extra vehicles and drivers, but this analysis is purely concerned with results achieved in 12.5 minutes. It is possible to consider many situations where the savings associated with the production of solutions in such a short period could outweigh the costs associated with the increased number of vehicles (e.g. if one were routing produce that spoils).

The result quality for the clustered dataset (c210_1) is encouraging as the result quality is (relatively) high. This suggests that good results can be achieved using a very simple algorithm, when it is used in conjunction with the timeslot solution strategy.

When one considers the distances, it is immediately clear that they are much higher for r210_1 and rc210_1. This is particularly interesting in r210_1 as the algorithm had sufficient time to achieve a local optimum in the post completion descent phase. One would have hoped were this occurring, the distances would not have been so uncompetitive. Although one was required to terminate the LS phase in rc210_1 before an optimum was reached, the moves being accepted at this stage were very small. As local optima have been identified well within the time limit for two of the three datasets, a decision to increase the size of the string length in the post completion descent is made.

However, although this will impact on the quality of the results, it is unlikely to alter the number of vehicles being deployed (the primary objective). As this is such a small

alteration, it seems sensible for it to be considered in conjunction with a further alteration.

7.5.2 Lexigraphical Objective Function

In Chapter Six, a significant portion of the algorithm's development concerned the change in the objective function; the number of vehicles deployed became the primary objective. This was achieved by multiplying the number of vehicles deployed by a large constant (ξ) and then adding the distance to the resulting figure. Although this successfully adapted the objective function, it did not encourage the algorithm to actively seek to reduce the number of vehicles deployed.

The construction of the solutions contained many features to try to minimise the number of vehicles being deployed (specifically the TTV and the CU), but the descent phase concerned itself purely with minimising the objective function. Although it has successfully removed vehicles from solutions, it only occurs if one is able to move all of the customers from the route being removed, into a single other vehicle route.

This situation occurs infrequently, as the vehicle routes often contain a sizable number of customers. Individual customers and strings of customers can be removed from the vehicle to others (making it easier to remove that vehicle in the future), but only if each move reduces the distance. However, given the primary objective is the number of vehicles deployed, it seems sensible to allow customers to be moved between routes, regardless of distance, if it has the potential to reduce the number of vehicles deployed.

This subsection considers whether the algorithm could benefit from the inclusion of an adapted version of the lexigraphical evaluation function of Bent and Van Hentenryck (2001). This technique splits the descent phase into a number of phases (they use three), each with a different objective.

They are concerned with minimising the number of vehicles, the distance and the periods of inactivity in the vehicle schedules. Each is assigned its own objective function and they are tackled in the order in which they receive precedence in the objective function. Although the sub-problems created by the timeslot solution

strategy could benefit from the inclusion of minimising the periods of inactivity the increase in computational time would not be welcome.

It is proposed that a two-phase lexigraphical objective function is included:

- Phase One

  This part of the descent phase encourages moves that are likely to lead to the possibility of vehicles being removed. It ignores the distance of the solution and is instead concerned with maximising the square of the vehicle lengths. This will encourage one to create some shorter vehicle routes. It is more likely that one will be able to move an entire vehicle's schedule into another vehicle if there are fewer customers (in the schedule). Moves that reduce the number of vehicles receive precedence.

- Phase Two

  This phase of the descent algorithm operates in the standard way for problems with time windows; with one selecting moves based on distance. Moves that reduce the number of vehicles can still be identified, and will receive precedence if found.

The solution identified at the end of Phase One is then used as an initial solution for Phase Two. This technique is included in the descent phase at the end of each timeslot as well as the post completion descent. Results for this technique are presented in Table 7.20 (only post completion descent results are presented).

	r210_1			**c210_1**			**rc210_1**		
	Vehicles	Distance	Time	Vehicles	Distance	Time	Vehicles	Distance	Time
Reg	23	58670.6	4.05	30	16992.2	0.16	25	35650.5	12.50
Lex	22	57572.2	6.42	30	17169.3	0.86	25	38259.2	12.50
Best	*19*	*42467.9*	*NA*	*30*	*16879.2*	*NA*	*21*	*29754.1*	*NA*

Table 7.20: Assessing the Impact of the Lexigraphical Evaluation Function

These results show a promising impact on the r210_1 dataset, with the schedule requiring one less vehicle to be deployed. It has also improved the distance of that

solution; though given the change in the number of vehicles deployed, this is a somewhat unnecessary observation. The number of vehicles deployed has not altered for either of the other two datasets and the distance has increased.

In c210_1 one is still locating a local optimum in Phase Two of the descent so one must assume that the Phase One of the descent makes a series of moves that cannot be undone if one requires the distance to be reduced at each interim step. The cost prior to the post completion descent was 16992.2, thus it is in this final descent that this problem occurs. In rc210_1 one was failing to achieve a local optimum before the inclusion of an extra descent phase. The algorithm still manages to identify a local optimum in the individual sub-problems, but the two-phase final decent is too time-consuming. All subsequent workings will make use of this two-phase lexigraphical evaluation function.

Before a remedy to this problem is presented, a more detailed look at the impact that this has on the algorithm's behaviour is presented. Figure 7.6 shows the behaviour of the lexigraphical evaluation function (two phases), the square of the vehicle lengths is denoted R^2.

These graphs show the two objective functions (distance and the square of the vehicle lengths) and how they impact upon each other in the post completion descent. In Phase One (to the left of the phase transition line) the value for R^2 is increasing, thus one is successfully creating the desired shorter vehicle routes. However, to the detriment of Phase Two, the distance can be seen to be increasing. This means that if a vehicle is not removed, one begins Phase Two with a solution that has a considerably worse distance than the one that would have been descended from, had the lexigraphical evaluation function has not been included.

This is less of an issue in r210_1 and c210_1 as one is obtaining a local optimum in the post completion descent, but in rc210_1 one is devoting what little time there is, to undoing moves that were counterproductive. A proposed remedy to this and two further improvements to the algorithm are now proposed.

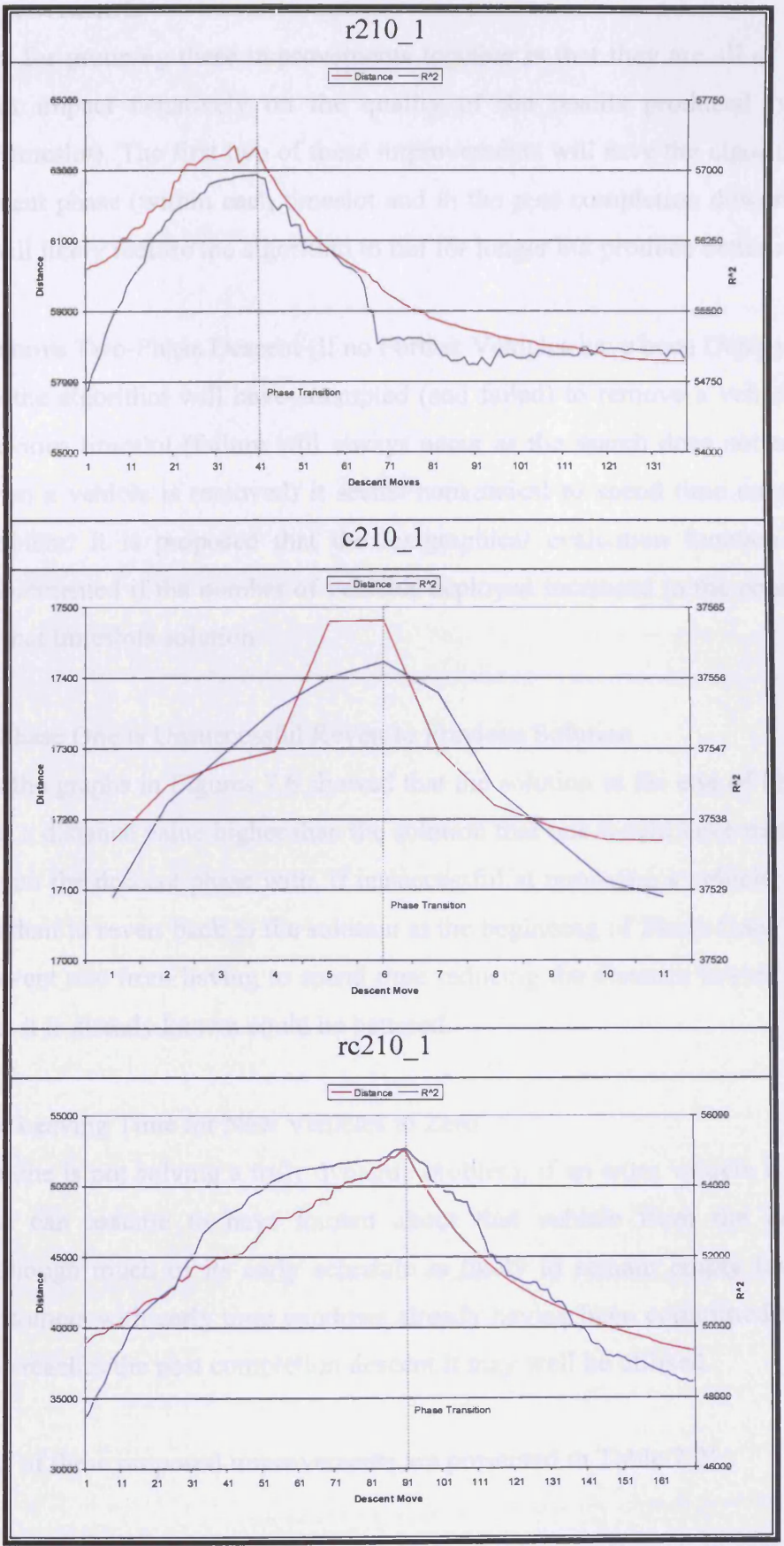

Figure 7.6: Understanding the Impact of the Lexigraphical Evaluation Function (Post Completion Descent)

Three Improvements

The reason for grouping these improvements together is that they are all of the type that cannot impact negatively on the quality of the results produced (within a particular timeslot). The first two of these improvements will save the algorithm time in the descent phase (within each timeslot and in the post completion descent) while the third will likely require the algorithm to run for longer but produce better results.

- Remove Two-Phase Descent (If no Further Vehicles have been Deployed)
  As the algorithm will have attempted (and failed) to remove a vehicle in the previous timeslot (failure will always occur as the search does not terminate when a vehicle is removed) it seems nonsensical to spend time on a similar problem. It is proposed that the lexigraphical evaluation function is only implemented if the number of vehicles deployed increased in the construction of that timeslots solution.

- If Phase One is Unsuccessful Revert to Previous Solution
  As the graphs in Figures 7.6 showed that the solution at the end of Phase One had a distance value higher than the solution that one would have traditionally begun the descent phase with, if unsuccessful at removing a vehicle, it seems prudent to revert back to the solution at the beginning of Phase One. This will prevent one from having to spend time reducing the distance towards a value that it is already known could be bettered.

- Set Leaving Time for New Vehicles to Zero
  As one is not solving a truly dynamic problem, if an extra vehicle is required one can assume to have known about that vehicle from the beginning. Although much of its early schedule is likely to remain empty (due to the customers with early time windows already having been committed to), when one reaches the post completion descent it may well be utilised.

The results of these proposed improvements are presented in Table 7.21.

	r210_1			c210_1			rc210_1		
	Vehicles	Distance	Time	Vehicles	Distance	Time	Vehicles	Distance	Time
Previous	22	57572.2	6.42	30	17169.3	0.86	25	38259.2	12.50
Improve	20	53904.1	12.50	30	16992.2	0.52	26	35567.4	12.50
Best	*19*	*42467.9*	*NA*	*30*	*16879.2*	*NA*	*21*	*29754.1*	*NA*

Table 7.21: Assess the Impact of the Three Proposed Improvements

The results for the inclusion of these improvements are generally positive, with significant improvements in r210_1. The number of vehicles deployed has been reduced (two less) and a significant reduction in the distance has already been identified. The increase in time is likely to be the result of the success of the third of these proposed improvements. One vehicle was removed during the main construction/descent phases, while a further vehicle was removed in the post completion descent. As there is greater freedom in the schedules (with the empty portions of the schedule now allowed to be utilised), one can assume that the number of feasible moves in the neighbourhood is greatly increased.

The improvement to c210_1 would have been achieved by the second of the three improvements, but no other impact can be observed from these results. The results for rc210_1 are somewhat disappointing, as the solution quality has worsened as a result of these proposed improvements. However, regardless of the results, these are still technically improvements, as there is little way they can have a negative impact on proceedings outside of just being unfortunate (with regards to the timeslot solution strategy). The distance (although secondary to the number of vehicles deployed) should be noted, as it has been significantly reduced. It is hoped that one will be able to remove the extra vehicle via some other form of alteration while keeping the reduction in distance.

All subsequent workings will make use of these three proposed improvements.

7.5.3 Undoing a Potential Poor Customer Selection

Given that one is making constructions in an entirely greedy manner, one will often make a poor selection, which until now, could only be undone within the descent phase. This subsection considers whether it is beneficial to include some form of

check-back mechanism to ensure that the customer selected is indeed the most appropriate.

The proposed adaptation will undo the last move and make an alternative selection if it feels that it will be beneficial in the longer term. In order for this technique to be applicable, one must be able to assess whether a customer selection is an appropriate one. Given that it is the number of vehicles deployed that is of primary concern it seems sensible to base this analysis on whether a move will result in another vehicle being deployed.

The NN algorithm is still making use of the extended roulette wheel, thus one is building multiple vehicle solutions at once. After a customer has been selected, it is proposed that one assess whether that move has resulted in any other customers being unable to be allocated to any of the existing vehicles. If this is the case, then the last move will have directly contributed to the need for a new vehicle. It is proposed that this move is then undone and the second nearest (according to the visibility definition in Chapter Six) is selected. If this results in less customers requiring new vehicles then this customer is selected, otherwise one reverts back to the initial selection. Results are presented in Table 7.22.

	r210_1			**c210_1**			**rc210_1**		
	Vehicles	Distance	Time	Vehicles	Distance	Time	Vehicles	Distance	Time
Previous	20	53904.1	12.50	30	16992.2	0.52	26	35567.4	12.50
Undone	21	53828.3	12.50	30	16992.2	0.52	25	35304.4	12.50
Best	*19*	*42467.9*	*NA*	*30*	*16879.2*	*NA*	*21*	*29754.1*	*NA*

Table 7.22: Assessing the Impact of Undoing an Potentially Poor Customer Selection

The impact of this adaptation is not overly significant; there is no change to the results on c210_1, while r210_1 has gained a vehicle and rc210_1 has lost one. Whether one should include this adaptation or not, is a fairly subjective choice (the impact on the distance is also negligible). However, throughout this thesis one has been aiming for the production of the most robust algorithm possible. Given the very poor performance on rc210_1 without it, one would have to favour its inclusion to maintain this policy.

This method was expanded to consider selecting and undoing each of the customers that would have required a new vehicle had the first move not been undone. However, the results were poor and it did not seem necessary to pursue this idea further than the single move.

All subsequent workings will consider undoing a move if it results in a customer requiring a new vehicle to be deployed (and testing the next best greedy selection).

7.5.4 Vehicle Bounds

A further adaptation is proposed, as the number of vehicles being deployed is still a concern (when compared to the best-achieved). When solving this problem, it seems prudent to make use of all the information that is available. As one is not solving a truly dynamic problem, one is aware of the demands of all of the customers, even if their artificial arrival time has not been reached.

As one effectively knows the total demand for the day in advance, one can estimate the number of vehicles that will be required to meet that demand. It is proposed that one calculates the minimum number of vehicles required (according to Equation 7.1), and one allows this number of vehicles to be deployed in ts_0. This may allow one to allocate more customers to the earlier parts of their schedules, as opposed to

$$Bound = 1 + int\left(\sum_{i=1}^{n} d_i \Big/ Q\right) \tag{7.1}$$

For reference, this gives bounds of 19, 28 and 18 vehicles for datasets r210_1, c210_1 and rc210_1 respectively. Note that NN selections will only be considered for undoing once all of the lower bound number of vehicles have had customers allocated to them. Results are presented in Table 7.23.

The analysis of this adaptation is somewhat simplified by the fact that there does not appear to be any benefit associated with including the vehicle bound. The algorithm is obviously not benefiting from the inclusion of this extra knowledge, and performs to a better standard without it. It may be a result of one being not as aggressive in trying to

minimise the number of vehicles in the earlier timeslots (before further vehicles are required) but given the poor performance, one need not consider this in further detail.

	r210_1			c210_1			rc210_1		
	Vehicles	Distance	Time	Vehicles	Distance	Time	Vehicles	Distance	Time
Previous	21	53828.3	12.50	30	16992.2	0.52	25	35304.4	12.50
Bounded	22	53921.2	12.50	30	16992.2	0.51	25	35794.1	12.50
Best	*19*	*42467.9*	*NA*	*30*	*16879.2*	*NA*	*21*	*29754.1*	*NA*

Table 7.23: Assessing the Impact of Including a Lower Bound on the Number of Vehicles Deployed

An adaptation was made to the algorithm which implemented multiple improvement moves in a single BI iteration. The best move was selected along with the next best move that concerned vehicles unaffected by the first move. This technique was continued either all vehicles were subject to a move or no further improvements were possible. Noticeable time savings were identified with P_0 and P_1.

This concludes the adaptations to the NN (BI descent) algorithm that was suggested as being appropriate for the Extra-Large Scale VRPTW. The performance of the resulting algorithm is now tested on the complete set of 90 benchmark datasets.

7.5.5 Extra-Large Scale VRPTW Results (600-1000 customers)

The preceding adaptations to the simple NN algorithm with a BI descent policy were made in the hope that one would be able to produce high quality solutions for the Extra-Large VRPTW within 12.5 minutes. This subsection presents the results that the aforementioned algorithm produced, along with a discussion comparing them to the best that have been achieved (in a non time-constrained environment).

The results for the full set of 90 datasets can be seen in Tables 7.24, 7.25 and 7.26 (600, 800 and 1000 customers respectively). As with the smaller datasets, results have not been produced with the same algorithm without the timeslot solution strategy, as their quality will be so poor. Again this demonstrates the strength of the proposed decomposition technique and in itself justifies the work in this chapter. Note that as the technique developed contains no randomness, there is no need to conduct a trial.

	Vehicles	Distance	Time	Vehicle Best	Distance Best
r2_6_1	12	21608.90	1	**11**	**18325.60**
r2_6_2	15	17566.94	3.7	**11**	**14997.65**
r2_6_3	15	18852.64	12.5	**11**	**11255.49**
r2_6_4	13	14537.74	12.5	**11**	**8126.87**
r2_6_5	13	17406.69	1.1	**11**	**15357.25**
r2_6_6	12	15192.97	8.7	**11**	**12803.83**
r2_6_7	13	12136.34	6	**11**	**10172.17**
r2_6_8	**11**	14441.91	10.1	**11**	**7752.78**
r2_6_9	13	15452.01	2.2	**11**	**13567.84**
r2_610	13	14446.90	1.2	**11**	**12513.45**
c2_6_1	**18**	7954.78	12.5	**18**	**7774.10**
c2_6_2	19	8738.36	12.5	**17**	**8799.38**
c2_6_3	19	10542.91	12.5	**17**	**7604.00**
c2_6_4	20	10491.26	12.5	**17**	**6993.77**
c2_6_5	**18**	7838.15	0.4	**18**	**7576.35**
c2_6_6	19	8750.63	0.7	**18**	**7478.63**
c2_6_7	21	11490.90	1.3	**18**	**7520.34**
c2_6_8	19	7994.01	0.9	**17**	**8579.89**
c2_6_9	20	9604.41	3.3	**18**	**7350.94**
c2_610	20	8600.01	2	**17**	**7523.34**
rc2_6_1	17	14232.37	2.1	**15**	**13163.03**
rc2_6_2	16	12190.48	4.6	**12**	**11853.72**
rc2_6_3	12	13683.46	12.5	**11**	**9816.47**
rc2_6_4	13	13744.05	12.5	**11**	**7197.11**
rc2_6_5	16	12903.40	1.9	**12**	**12560.43**
rc2_6_6	15	12685.46	1.8	**11**	**12282.52**
rc2_6_7	13	11940.59	2.2	**11**	**10929.52**
rc2_6_8	**11**	12001.64	2.9	**11**	**10474.95**
rc2_6_9	11	11977.75	3	**11**	**9821.39**
rc2_610	13	11697.75	3.4	**11**	**9339.41**

Table 7.24: Solutions for 600 Customer Problems

	Vehicles	Distance	Time	Vehicle Best	Distance Best
r2_8_1	17	33959.15	1.5	**15**	**28440.29**
r2_8_2	18	28383.22	4.9	**15**	**23274.22**
r2_8_3	16	22059.23	11.4	**15**	**17992.25**
r2_8_4	**15**	25734.44	12.5	**15**	**13413.79**
r2_8_5	19	29003.15	2.2	**15**	**24611.39**
r2_8_6	16	27163.36	8.2	**15**	**20697.06**
r2_8_7	16	29810.49	12.5	**15**	**16977.49**
r2_8_8	16	26540.04	12.5	**15**	**12945.22**
r2_8_9	18	27321.94	2.5	**15**	**22588.02**
r2_810	16	26975.14	2.1	**15**	**21092.27**
c2_8_1	**24**	11735.76	0.2	**24**	**11654.81**
c2_8_2	25	14827.78	12.5	**24**	**11422.34**
c2_8_3	27	18253.61	12.5	**23**	**11554.18**
c2_8_4	28	15513.14	12.5	**23**	**10963.49**
c2_8_5	**24**	**10886.32**	0.7	**24**	11432.92
c2_8_6	26	12166.55	2.3	**24**	**11357.86**
c2_8_7	27	11932.51	3.6	**24**	**11397.54**
c2_8_8	24	12007.93	2.1	**23**	**12927.45**
c2_8_9	27	14280.35	7.7	**24**	**11249.00**
c2_810	26	13029.50	4.8	**23**	**11284.46**
rc2_8_1	23	21952.31	3	**19**	**20954.95**
rc2_8_2	19	20234.72	9.4	**17**	**18032.89**
rc2_8_3	20	20441.51	12.5	**15**	**14800.78**
rc2_8_4	17	22841.41	12.5	**15**	**11312.68**
rc2_8_5	21	20574.34	4.8	**16**	**19105.75**
rc2_8_6	18	19856.10	3.5	**15**	**18882.30**
rc2_8_7	**15**	20496.98	4.9	**15**	**17327.53**
rc2_8_8	**15**	17992.26	6.6	**15**	**16203.18**
rc2_8_9	17	22055.66	5.4	**15**	**15622.52**
rc2_810	16	19980.20	7	**15**	**14892.29**

Table 7.25: Solutions for 800 Customer Problems

	Vehicles	Distance	Time	Vehicle Best	Distance Best
r210_1	21	53828.35	12.5	**19**	**42467.87**
r210_2	22	42727.58	12.5	**19**	**33589.08**
r210_3	24	48311.57	12.5	**19**	**25321.00**
r210_4	23	37383.56	12.5	**19**	**18222.30**
r210_5	23	45713.50	3.8	**19**	**36735.20**
r210_6	21	38365.12	12.5	**19**	**30261.75**
r210_7	22	44131.49	12.5	**19**	**23463.75**
r210_8	23	37095.54	12.5	**19**	**17705.20**
r210_9	22	39926.70	5.3	**19**	**33519.84**
r21010	20	40445.72	3.6	**19**	**30706.00**
c210_1	**30**	16992.20	0.8	**30**	**16897.25**
c210_2	31	20957.37	12.5	**29**	**17144.29**
c210_3	34	25937.19	12.5	**29**	**16367.59**
c210_4	33	22545.05	12.5	**29**	**15919.46**
c210_5	**30**	17339.25	1.5	**30**	**16561.70**
c210_6	32	17053.65	3.7	**30**	**16341.67**
c210_7	33	19115.94	8.6	**30**	**16435.10**
c210_8	31	14997.37	4.3	**29**	**16315.89**
c210_9	32	24013.51	12.5	**30**	**16161.74**
c21010	31	14518.85	9.7	**29**	**15885.41**
rc210_1	25	35304.40	12.5	**19**	**42467.87**
rc210_2	21	38290.64	12.5	**19**	**33589.08**
rc210_3	**19**	38290.64	12.5	**19**	**25321.00**
rc210_4	21	32112.41	12.5	**19**	**18222.30**
rc210_5	20	31458.18	6.3	**19**	**36735.20**
rc210_6	20	30999.55	5.5	**19**	**30261.75**
rc210_7	20	28393.68	8.4	**19**	**23463.75**
rc210_8	21	28317.36	9.8	**19**	**17705.20**
rc210_9	21	34494.20	7.1	**19**	**33519.84**
rc21010	**19**	**29978.38**	10.9	**19**	30706.01

Table 7.26: Solutions for 1000 Customer Problems

As stated in the introduction to this section, the investigation into the Extra-Large VRPTW is more concise than for the 200 and 400 customer problems (as the time is so restrictive). Although there are more results produced (90 instead of 60), this piece

of work is very much a subsidiary to the earlier work. The level of detail in the analysis will reflect this status.

The analysis of these results will be of a type similar to the numerical analysis in Section 7.4, though some details about the performance will be interspersed within this.

Numerical Analysis

Although the focus of this chapter was on the successful development of a decomposition technique that could be used in conjunction with multiple algorithms, it is interesting to know what quality of solution the simple NN descent algorithm produced. Although only the most basic of algorithms has been implemented, one is immediately drawn to the fact that two new best-achieved solutions have been produced.

The two datasets that have been improved are c2_8_5 and rc21010. Neither has seen any change in the number of vehicles deployed, though significant reductions in distance (4.78% and 2.37% respectively) have been achieved.

Identifying a new best-achieved solution is always desirable, and one should never discount the ability of any algorithm that has succeeded in this task. However, it is well documented that the NN (even when used in conjunction with descent) is uncompetitive when compared with more modern techniques (such as the metaheuristics discussed in Chapter Three) and one would never expect to find a new best solution with this type of algorithm. Examining the other techniques that have been used to produce solutions for these datasets, it seems clear that their algorithms are of a more complex nature, and that these new best-achieved solutions are likely to be the result of the timeslot solution strategy (as opposed to the algorithm producing the solutions).

The fact that new best-achieved solutions have been identified is even more unusual given the restrictive nature of the time-constraint. This further supports the theory that time based problem decomposition is a workable technique and that the benefits of employing it can be substantial.

Obviously producing two such high quality solutions with such a simple algorithm and a restrictive time constraint is significant; one may wish to consider what quality of solution is achievable if one were to use a more complex algorithm

These two high quality solutions present an interesting question as to what quality of solution one may be able to produce if the time limit was removed and a more complex algorithm implemented. However, this would constitute quite a considerable volume of research, and therefore remains unanswered in this thesis. It is suggested as a sensible avenue for further research, and it is hoped that it will be pursued in time.

As one has shown that two very high quality solutions have been produced, one might be inclined to expect that the result quality would be high across the board. However, it is often the case that certain techniques excel on certain datasets, while struggle with others; though one might expect less variability than can be seen here. This variability is likely to be as a result of the timeslot solution strategy, though as one cannot predict which datasets lend themselves best to this type of decomposition, it cannot be compensated for.

An analysis based on the increase in the number of vehicles deployed (similar to the one in Section 7.4) is now presented (Table 7.27). As there has not been a trial conducted, only one set of results is presented.

n	0	1	2	3	4	5	6
600	5	4	13	3	5	0	0
800	5	8	5	6	3	3	0
1000	4	4	11	4	4	2	1

Table 7.27: Additional Number of Vehicles from Best-Achieved

The first observation one would be inclined to make from this table, is that the result quality appears to be worse than those achieved by the ACO algorithm on the 200 and 400 customer datasets (minimum or maximum). The results also seem much more variable, with a far more even distribution of figures amongst the table. As expected (due to the consistent time restriction) the level of performance appears to have worsened as the problem size has increased.

Interestingly it is one of the datasets used to optimise the algorithm (rc210_1) that the algorithm performs worst on. This is unusual as one may often (inadvertently) over optimise the algorithm to the datasets which one is using to judge the performance (in this case the test set); clearly this is not the case here.

Some general figures about the quality of solution when averaged across all the datasets are now presented.

The average number of vehicles deployed (when compared to the best achieved) has increased by 14.71% (for the 600 customer problems), 11.60% (for the 800 customer problems) and 10.37% (for the 1000 customer problems). These figures support the observation that result quality is worse than for the best result produced for the 200 and 400 customer problems, though this observation is not upheld when one considers the worst performance (increases of 20.98% and 16.84% respectively).

As with the previous analysis there is little one can say with certainty about the distances of the solutions produced when the number of vehicles deployed differs. However, as a general observation, it seems that the NN (and descent) algorithm is struggling on the distance. The figures seem much higher (regardless of the change in vehicles), and this is an obvious cause for concern. The datasets where a local optimum was achieved (i.e. did not require the full 12.5 minutes) still seem to have poor results, which suggests that the neighbourhood may be inappropriate for problems of this size. Different values for L (post completion descent) may help for these datasets, but it is likely to have a negative effect on those that do not reach an optimum.

For the 600 and 800 customer problems only 8 of the 30 datasets (per problem size) required the full 12.5 minutes (i.e. a local optimum was not found). This figure rises to 15 for the 1000 customer problems, but even this demonstrates just how fast the algorithm can be run. Considerable attention should be placed on these figures as the speed represents one of the main strengths of this decomposition technique. Given the times reported by others (e.g. the various algorithms summarised in Mester and

Bräysy (2005) require 16.2-399.8, 26.2-512.9 and 39.6-606.3 minutes[11] to solve 600, 800 and 1000 customer problems respectively), it seems sensible to assume that one would find it difficult to produce solutions of this quality (given the time constraint) without this type of technique.

It would be possible to remove the time constraint and see if this method could be used to produce higher quality solutions (i.e. when it has not been artificially restricted), but this is outside the remit of this chapter. Note that when one does not require the full 12.5 minutes, the time required to produce the solution is positively correlated with the problem size.

In summary, these results tend to suggest that one will struggle to produce consistently competitive results with such a simple algorithm (and time restriction). However, the good results that were achieved demonstrate the technique's potential. In its current guise one would seemingly only favour this technique if there were a definite need to produce solutions quickly.

7.5.6 Analysis Summary and Chapter Conclusions

The chapter began by considering the idea of problem decomposition for the VRPTW. Arguments were presented as to why it was felt that problem decomposition based on geographical location (as was often used for the CVRP) was probably unsuitable for the VRPTW, and why one might favour a time-based approach. As a result of this, the work in this chapter was concerned with assessing whether the timeslot solution strategy could successfully be used as a problem decomposition technique for Large Scale VRPTW.

It was suggested that one should consider the 200-400 customer problems and 600-1000 customer problems separately; a summary of the work on each is now provided.

For the 200-400 customer problems, various adaptations were made to the datasets (creation of artificial arrival times), the solution methodology and the algorithm being used to produce the results (adapted from the algorithm produced in Chapter Six).

[11] See publication for details of the computers used to produce these result.

Given the quality of the results achieved (albeit only on certain datasets) one has to assume that it is indeed suitable, and that with further research could become a very competitive technique.

It was noted that the quality of result might have suffered slightly from the disparity between the time consuming nature of the ACO metaheuristic and the need to produce all of the solutions within the time constraint of 12.5 minutes. However, the timeslot solution strategy allowed for higher quality results to be achieved than had the same algorithm been run with the more traditional solution strategy (i.e. considering the whole problem at once). This showcases the time saving nature of solving the series of successive smaller sub-problems and suggests that decomposition is a workable technique for the VRPTW.

The unsuitability of the ACO algorithm was noted for the (600-1000) customer problems and a simpler NN construction (and descent) algorithm was proposed. The same artificial arrival times were used as for the 200-400 customer problems, whilst various adaptations to the algorithm were proposed. Results were not as promising as they were for the smaller problems (somewhat expected given the simple solution technique) but some results were of a high standard (again result quality was variable). Two best-achieved solutions were produced (c2_8_5 and rc21010), whilst other high quality results were also observed.

In summary it appears that the technique has great potential for solving Large Scale problems (it would be interesting for it to be assessed on even larger problems i.e. >1000 customers) and that glimpses of its promise were shown in this chapter. It would be interesting for a bespoke algorithm to be created (to complement the timeslot solution strategy), to see if very high quality results were achievable. Other directions for further work include attempting to identify what characteristic about a dataset lends itself well to this type of problem decomposition and if it is possible to adapt the commitment strategy to commit to more customers if one is happy with their current placement (given that the remaining customers are actually known).

Chapter Eight

Conclusions and Further Work

8.1 Introduction

This chapter seeks to offer an overview of the relative successes of the chapters that preceded it, and provide further discussions as to possible directions for future work. Note that as each chapter contained a sizable results and analysis section (which was concerned with presenting similar details) one is directed to those, if further details are required.

The work in this thesis can be considered as being divided into three distinct sections (based on the problem that was being tackled), and this allows for an obvious structure to these closing comments:

Firstly, the findings and suggested further work relating to the work on the DVRP are presented (from Chapters Three through Five). This section will detail further work relating to the further development of the two algorithms developed (the ERWACS-DVRP and the Tabu-DVRP) as well as more general observations about the DVRP itself. This section will also include a performance comparison of the two algorithms

produced with two further algorithms that have recently become publicly available, though remain currently unpublished.

Secondly, the findings and suggested further work relating to the work on the DVRPTW are presented (from Chapter Six). This includes details relating to the potential for the problems application within a real-life environment and a discussion regarding whether it will ever be researched in a similar manner to the DVRP.

Thirdly, the findings and suggested further work relating to the research that made use of the timeslot solution strategy as a decomposition technique for the Large Scale VRPTW is presented (from Chapter Seven).

The chapter concludes with a discussion of some more diverse directions for future work (i.e. outside the scope of these three problems), and some general closing comments regarding the research in this thesis.

8.2 The DVRP

The DVRP formed the primary focus of Chapters Three through Five, and provided a problem on which one was able to establish a number of new and interesting solution techniques. Chapter Three began with an important discussion relating to the idea of the time-constraint, and whether one ought to discard the idea (so the DVRP better represented the problem one would be likely to encounter in real-life) or to continue with it (so results were comparable in the mathematical community). Arguments were presented for both cases, and a decision to pursue both policies was made (the disadvantages of this dual strategy were also noted).

A metaheuristic classification scheme that did not rely on the high levels of subjectivity present in one of the existing methods was then presented. This allowed one to ensure that when the metaheuristics were implemented, one was making use of a varied number of techniques.

The work regarding the actual production of solutions began with two basic heuristic implementations (NN and descent), which enabled one to quantify the high quality of the solutions produced via the ACS-DVRP. Four basic metaheuristic implementations

were then presented, allowing one to select ACO and TS as the two algorithms for further research (in Chapters Four and Five respectively). Note that their selection was not entirely based on quality of solutions, but also the potential for further work and personal interest in the techniques.

8.2.1 Ant Colony Optimisation

The first technique that received a more detailed investigation was the ACO metaheuristic, in the form of the ACS algorithm (selection was made via a short investigation into the performance levels of the ACO algorithms on the TSP and the CVRP). Building upon the complex foundations of the ACS-DVRP, attempts were made to improve the performance via a series of alterations.

Alterations to the ACS-DVRP began with a parameter review (including an adaptation that removed the need for one to know the global problem size in advance), and slight improvements were identified. Consideration was then given to the two potential shortcomings of the existing algorithm that had been noted (and the implementation of proposed remedies):

- A potential problem relating to further vehicle deployment was identified, and a remedy in the form of a new criterion for tentative schedule completion was presented. This technique succeeded in reducing the number of vehicles deployed, whilst reducing the time required to produce the solutions (important in the time-constrained variant of the problem).

- A potential problem relating to the sequential nature of the ACS-DVRP construction technique was identified, and a remedy in the form of the extended roulette wheel was presented. This technique succeeded in improving solution quality in two of the three test datasets, but a problem regarding excessive vehicle deployment was observed in the final test dataset.

In response to the increase in the vehicles deployed (as a result of the non sequential construction technique proposed), two techniques that encouraged fewer vehicles to

be deployed were successfully implemented. Adaptations were also proposed to the neighbourhood definition, but no improvements were forthcoming.

The chapter concluded with full sets of results to the non time-constrained and time-constrained variants of the problem. The non time-constrained results demonstrated that there was significant scope for improvement in the results of the ACS-DVRP. Various observations were noted about the performance of the algorithm, before a slight alteration was made to the improvement phase. This adapted algorithm was then implemented on the time-constrained variant of the problem; 17 new best-achieved solutions were identified, along with reductions in the average distance (20 datasets) and the maximum distance (all 21 datasets).

Further details about the improved level of performance were noted; specifically a 5.92% average reduction in average distance and a ≈50% reduction in the variability of the solutions produced. The importance of this second observation should not be understated, as when solving a DVRP in real-life, one cannot select the best result of a trial.

Further Work

With regards to the current algorithm (the ERWACS-DVRP), the majority of potential avenues for future work focus on the intensification/diversification balance. The ACS algorithm is (by design) particularly aggressive and thus, may well benefit from further diversification. It would be interesting to develop an adaptive system, which diversified the search, as and when required (i.e. the algorithm would have to monitor the variability of the solutions produced, and turn on some diversification procedure if it was felt necessary). Note that numerous techniques exist for increasing the diversification e.g. smoothing of trail values or changing the bias parameters α, β, and λ.

It is also suggested that one could benefit from a more detailed investigation into the behaviour of the trail, understanding why decisions were being made, and if the search could benefit from adaptations to the update rules. With regards to the improvement phase, it would obviously be of interest to consider more complex

neighbourhood structures, though it seems sensible to discuss these within the future work subsection of the TS algorithm.

Outside of the current algorithm, it seems sensible to consider another of the ACO algorithms. It is suggested that an algorithm based on the MMAS (which has been shown to be one of the better performing, if the time is sufficient) would provide an interesting comparison for the ERWACS-DVRP.

8.2.2 Tabu Search

The second technique to receive a more detailed investigation was TS. It was selected as it had performed to a worse standard that SA (in Chapter Three); somewhat unexpected behaviour with regards to these technique's relative success on more traditional routing problems.

A discussion relating to the difficulties of assessing the quality of a neighbourhood in a TS algorithm was presented, and a decision was made to temporarily abandon TS in favour of descent and SA (which are more suitable environments for this analysis). The 1-Opt neighbourhood (employed in the ACS-DVRP and Chapter Three), was expanded from two potential moves to eight (referred to as the 8-Move neighbourhood). The quality of the results produced by this neighbourhood suggested that one ought to consider further expanding the neighbourhood to allow for the exchange of more variable length strings. The CROSS Neighbourhood was introduced (along with details relating to its complexity when applied to the DVRP); results supported its inclusion.

With a well-performing neighbourhood established, attention reverted back to the TS metaheuristic, and the development of a suitable TC (this was identified as the likely cause of the poor performance of the TS metaheuristic in Chapter Three). Two very general TC were presented:

- A TC that made use of the distance of a solution as a way of identifying previously visited regions of the search-space was proposed. It appeared that this technique might have suffered from trying to prevent repetition rather than

the reversal of moves. It was shown that distance did not always uniquely represent a solution.

- A TC that made use of the customer pairs (arcs) as a way of preventing the reversal of previously accepted moves was proposed. This technique did not appear to suffer the same failings as the distance based technique, and was thus selected.

The remainder of the adaptations to the algorithm came in the form of alterations to the intensification/diversification balance. Investigations into increasing the diversity of solutions through the use of a dynamic tenure (of both a random and more intelligent nature) were explored, and many interesting observations noted. Further diversification came in the form of a long-term memory that attempted to restrict the movement of customers that were currently being moved frequently. This method succeeded in diversifying the search, though it was not until it was implemented periodically that the result quality was noticeably improved. A series of random restarts was also proposed as a diversification technique, and further improved the quality of the solutions produced.

The chapter concluded with full sets of results to the non time-constrained and time-constrained variants of the problem. The result quality (in both problem variants) was remarkably similar to that achieved by the ERWACS-DVRP, thus the algorithm was successful in its objective of improving upon the results of the ACS-DVRP. On the time-constrained variant, 12 best-achieved solutions were produced (along with reductions in the average and maximum distances in 14 and 8 of the datasets respectively). The results appeared to be slightly more variable than the ERWACS-DVRP though no preference was expressed.

A short addendum was included at the end of Chapter Five that detailed the first steps of an investigation into the impact of varying the DOD. It was shown how the distance of solutions and the DOD were positively correlated, while the ERWACS-DVRP seemed to be the more robust algorithm (with respect to the DOD).

Further Work

With regards to the Tabu-DVRP, the majority of the changes one would be likely to consider are going to relate to either the neighbourhood definition or the intensification/diversification balance. Note that these are merely proposals, and as such, will not necessarily lead to improvements in solution (particularly in the time-constrained variant of the problem).

If one were to consider more complex neighbourhoods, it is suggested that moves of a cyclic nature (i.e. those that move customers between more than two vehicles in a single iteration) would be a sensible place for this analysis to begin. Obviously the time-constraint will hinder attempts to increase the neighbourhood size too much, and it may be necessary for the neighbourhood definition to adapt based on the complexity of the timeslot that one is solving.

The concept of strategic oscillation has frequently been applied to the TSP and CVRP and presents an interesting direction that one could pursue. One must be aware that this will lead to an increase in the computational effort, and that the neighbourhood may need to be restricted accordingly (time-constrained).

Further improvements are suggested in the form of an intermediate-term memory to influence the construction of new initial solutions (as opposed to the random restarts), and alterations to the current rigid system that alternates between the periods of intensification and diversification. It seems sensible to only continue to intensify the search, if one is producing high quality solutions (in relation to the best found so far).

8.2.3 General Comments: The DVRP

The work in Chapters Three through Six will hopefully help with the establishment of the DVRP as a more commonly accepted variant of the VRP family. The quality of solutions for the benchmark problems has been significantly improved, though it seems reasonable to assume that there remains scope for further improvement.

With regards to the time-constraint (and which variant of the problem is the most sensible for other authors to consider in the future), one has to consider whether the work is being conducted for implementation in real-life, or just to compare with other

researchers. The time-constraint allows for a direct comparison in solution quality to be made (if one is able to scale the time appropriately with regards to the computer being used), but is nonsensical with regards to the real-life application.

If the problem remains within the mathematical community it is unlikely to flourish as it becomes less worthwhile if it has no real-life application. The level of research able to be conducted will be restricted to the somewhat arbitrary computational time (which is likely to restrict the quality of results achievable). However, if one continues to allow algorithms to show their true ability (as was attempted in Chapters Three through Six) one will be able to achieve greater improvements, which in turn should increase the chance of the problem being implemented in real-life situations.

In the short-term it seems acceptable to continue research into the problem with or without the time-constraint, but as computational power continues to increase, one can envisage a situation where there is no benefit (outside comparing results with other researchers) of producing results for the time-constrained variant.

In subsequent analyses, it is suggested that one considers the variability of the results produced (e.g. range or standard deviation) as a further criterion for distinguishing between results. This is particularly important, as one cannot rerun the algorithm multiple times when one is considering a real-life DVRP (it is also proposed that one could investigate whether minimising the number of vehicles, as per the DVRPTW, could reduce variability).

The dynamic problems in this thesis have all focused on problems concerned with pick-ups (as they are better suited to the dynamic environment), but an investigation into deliveries may also be of interest.

8.2.4 Results Comparison

Upon completion of the DVRP work that was conducted in this thesis (i.e. Chapters Three through Five), a piece of work, Hanshar and Ombuki-Berman (2007), produced

two new algorithms for the same problem[1]. Although the algorithms produced (referred to as the DVRP-GA and the DVRP-TABU) produce results of a quality higher than the ACS-DVRP (and will thus lessen the apparent quality of the results produced in this thesis), one has to view the further research being conducted on the DVRP as beneficial.

It helps to validate the work in this thesis; it suggests that the DVRP is becoming an active area of research, and that in time, it may become a commonly accepted VRP variant. Note that as there remains considerable scope for development outside these four algorithms, further research is still encouraged.

Although not strictly necessary, it was felt beneficial to provide a simple analysis (see Table 8.1) comparing these algorithms with the performance of the ERWACS-DVRP and the Tabu-DVRP (results for the ACS-DVRP are also included).

As this is in the Conclusions and Further Work of this thesis, a through analysis is not conducted, though some basic observations are presented The result quality (as indicated by the Min and Ave metrics) seems to be fairly evenly shared across the four newer algorithms (i.e. not the ACS-DVRP). Hanshar and Ombuki-Berman (2007) chose to report the Total and Average distance (for the minimum and average distance across the 21 datasets), and thus the same figures have been reported for the ERWACS-DVRP and the Tabu_DVRP.

These figures suggest that the ERWACS-DVRP, Tabu-DVRP and DVRP-GA are producing results of a slightly higher standard than the DVRP-TABU. This is an interesting observation as it suggests that of the two TS implementations, one would favour the one developed in Chapter Five above the DVRP-TABU. Note that the DVRP-TABU seems to produce the highest number of best minimum solutions (seven), which when coupled with the overall worst average minimum distance, suggests that the algorithm is not particularly robust.

[1] Including the timeslot solution strategy, the reduction in time to match the computational effort with Montemanni et al. (2005) (i.e. time-constrained) and parameter settings for T_{ac} and T_{co}.

	ACS-DVRP		ERWACS-DVRP		Tabu-DVRP		DVRP-GA		DVRP-TABU	
	Min	Ave	Min	Ave	Min	Ave	Min	Ave	Min	Ave
c100*	973.26	1066.16	970.34	1006.16	988.39	1020.16	**961.10**	**987.59**	997.15	1047.60
c100b	944.23	1023.60	**874.06**	**900.50**	903.52	917.50	881.92	900.94	891.42	932.14
c120	1416.45	1525.15	**1256.27**	**1344.37**	1394.22	1482.73	1303.59	1390.58	1331.80	1468.12
c150	1345.73	1455.50	1342.79	1388.71	1334.05	1387.84	1348.88	**1386.93**	**1318.22**	1401.06
c199	1771.04	1844.82	1654.54	1731.12	**1622.85**	**1701.87**	1654.51	1758.51	1750.09	1783.43
c50	631.30	681.86	604.32	629.09	620.58	637.18	**570.89**	**593.42**	603.57	627.90
c75	1009.38	1042.39	989.13	1014.91	**942.87**	**1000.62**	981.57	1013.45	981.51	1013.82
f134	**15135.51**	16083.56	15190.45	**15672.58**	15603.40	16144.30	15528.81	15986.84	15717.90	16582.04
f71*	311.18	348.69	304.80	314.34	299.91	311.26	301.79	309.94	**280.23**	**306.33**

	ACS-DVRP		ERWACS-DVRP		Tabu-DVRP		DVRP-GA		DVRP-TABU	
	Min	Ave	Min	Ave	Min	Ave	Min	Ave	Min	Ave
tai100a	2375.92	2428.38	**2186.11**	**2254.20**	2213.71	2269.19	2232.71	2295.61	2208.85	2310.37
tai100b	2283.97	2347.90	2207.39	2253.39	**2137.46**	2218.81	2147.70	**2215.39**	2219.28	2330.52
tai100c	1562.30	1655.91	1566.52	1667.85	1521.71	**1598.24**	1541.28	1622.66	**1515.10**	1604.18
tai100d	2008.13	2060.72	1907.50	2012.42	1857.19	1976.45	**1834.60**	**1912.43**	1881.91	2026.76
tai150a	3644.78	3840.18	3319.35	**3437.58**	**3313.85**	3489.84	3328.85	3501.83	3488.02	3598.69
tai150b	3166.88	3327.47	**2874.75**	2936.70	2881.65	**2912.19**	2933.40	3115.39	3109.23	3215.32
tai150c	2811.48	3016.14	2576.08	2788.00	**2521.87**	**2684.99**	2612.68	2743.55	2666.28	2913.67
tai150d	3058.87	3203.75	3004.66	3075.90	3012.06	**3027.96**	2950.61	3045.16	**2950.83**	3111.43
tai75a*	1843.08	1945.20	1872.93	1900.97	1874.45	1890.64	1782.91	**1856.66**	**1778.52**	1883.47
tai75b	1535.43	1704.06	1508.99	1586.60	1491.98	1579.29	1464.56	**1527.77**	**1461.37**	1587.72
tai75c	1574.98	1653.58	1583.20	1653.18	1510.58	1562.60	1440.54	**1501.91**	**1406.27**	1527.80
tai75d	1472.35	1529.00	1438.92	1446.30	1430.81	1444.72	**1399.83**	**1422.27**	1430.83	1453.56
Total	50876.25	53784.02	49233.10	**51014.87**	49477.11	51258.38	**49202.73**	51088.83	49988.38	52725.93
Average	2422.68	2561.14	2344.43	**2429.28**	2356.05	2440.88	**2342.99**	2432.80	2380.40	2510.76

Table 8.1: Comparing the Performance of the ACS-DVRP, ERWACS-DVRP, Tabu-DVRP, DVRP-GA and DVRP-TABU

If one were pressed to express a preference between the three highest performing algorithms, one would probably discount the Tabu-DVRP, as it was discussed in Chapter Five how one might favour the less variable ERWACS-DVRP. As there is such a little difference between the ERWACS-DVRP and the DVRP-GA it seems unreasonable to distinguish between these algorithms performance. Note that Hanshar and Ombuki-Berman (2007) chose not to report the maximum result of the trial, which may have helped one to make a more definite decision regarding which algorithm performs to the highest standard (given the need to reduce variability as much as possible).

8.3 The DVRPTW

The DVRPTW was introduced in Chapter Six; it was suggested that as a problem closely related to the DVRP, it may come to prominence in the near future. This chapter began with a discussion on the reason for the problems development (and whether one was likely to be faced with this problem in real-life), and how the problem was less different from the associated static variant (i.e. the VRPTW) than the DVRP was.

As one was proposing a new problem, it was necessary to develop a set of benchmark datasets[2] on which one could assess the quality of the algorithms developed for producing solutions. This dataset creation was a much more complex task (with regards to ensuring that feasible solutions existed), than that which was faced when the DVRP was first developed. Upon the creation of a set of benchmark datasets (adapted from a set produced for the VRPTW), consideration was given to modifying the ERWACS-DVRP (with the CROSS Neighbourhood for its improvement phase), such that it produced feasible solutions for this new problem.

Once feasibility was achieved, focus switched to improving the performance, including adaptations to the update rules, objective function and visibility definitions. Although these changes significantly improved the quality of the solutions produced, it was felt that given the dual objective function (minimising vehicles and then distance) one ought to consider the use of multiple ant colonies.

[2] Available at www.cardiff.ac.uk/maths/people/postgraduate/wallace_dvrptw.html

The multiple ant colonies (one for each objective) were developed for use in a disjoint and interlaced approach; the interlaced technique appeared to produce solutions of a higher quality and was thus pursued. Given that the two-phase approach no longer required a feasible solution to be constructed by every ant (in the first colony), a technique was developed to encourage the customers less frequently selected by the RPR.

The chapter concluded with a full set of results for the 27 benchmark problems created previously. As a newly developed problem, quantifying the quality of the results produced was not a simple task; though one was able to make use of the best-achieved solutions for the VRPTW. The result quality was noticeably lower than for the static variant (13.10% increase in vehicles deployed), but not of such a poor quality that one would suggest the algorithm developed (the ERWACS-DVRPTW) was of a poor standard. This conclusion was further supported through the production of some NN (with descent) results, which were of a noticeably lower quality (the ERWACS-DVRPTW required 14.7% less vehicles on average).

8.3.1 General Comments and Further Work

Although there are many ways to improve the performance of the ERWACS-DVRPTW (many of which will be similar to the improvements proposed for the ERWACS-DVRP), as this is such a new problem, this section will focus on the development of the future of the problem rather than the solution algorithm.

Any newly developed problem variant will require other authors to consider it worthy of research, if it is (as hoped) to become more established. Although this may happen as a result of this work, one has to consider the short-term likelihood of this (given the current level of research on the problem which inspired it, the DVRP). It seems likely that until the DVRP becomes more widely known in the research community, few people will be likely to consider the DVRPTW. This is not in detriment to the DVRPTW as a problem in its own right, it is just that the DVRP is more established and thus more likely to receive attention.

It is hoped that if the DVRP continues to receive greater levels of attention, more people will begin to consider the DVRPTW, and produce algorithms that will allow one to further quantify the results of the ERWACS-DVRPTW.

As with the DVRP, in the short-term the time-constraint seems an acceptable policy to continue with (with regards to comparing performance with other researchers), but does not belong in the long-term definition of the problem. If the problem becomes more researched it may inspire its more frequent use in real-life environments. Note that if it does become commonly utilised in real-life, the time-constraint becomes nonsensical.

8.4 Large Scale VRPTW

The final problem considered in this thesis was the only one that did not contain a dynamic element. The VRPTW has long since been considered a routing problem worthy of research, and as such, has had a multitude of publications associated with it. Chapter Seven considered a problem decomposition technique for Large Scale VRPTW based on the idea of making the datasets appear artificially dynamic. This allowed one to solve a series of successive smaller sub-problems (to produce a solution to the whole problem).

Problem decomposition has been well developed for the CVRP (usually based on geographical location), but has received substantially less attention for the VRPTW. Note that all analysis was made on the time-constrained variant of the problem.

Preliminary results were presented for 200-400 customer problems (using the dataset creation technique from Chapter Six), showcasing the potential of the technique. Once satisfied that one could produce high quality solutions through this type of decomposition (and a decision made with regards to begin by investigating problems of this size), consideration was given to the establishment of a better way of creating the artificial dynamicity. Various techniques were considered, with a 'last chance' method appearing to offer the higher quality of solution. It was shown how this method was very effective at reducing the size of the sub-problems, which allowed one to complete a greater number of cycles (of an adapted ERWACS-DVRPTW) than one would have been able to under the previous dataset creation technique.

Consideration was then given to time, and whether one could make better use of it; as one was not required to spend an equal amount of time on each timeslot. Although no improvements were identified, this did represent an interesting investigation, and it is suggested that further consideration be given to this idea in the future (see Section 8.4.1). The improvement phase was also implemented after the entire solution had been produced (as one is free to alter the committed components).

Results were then produced for the 200-400 customer problems. These were shown to be of a slightly variable nature, but generally of a good standard (an average increase of 13.50% in vehicles deployed when compared with the best-achieved results). It was concluded that high quality solutions were definitely achievable with this technique, though one may have benefited from a bespoke algorithm to accompany the new solution technique (given how little time was available).

A short investigation into the 600-1000 customer problems, based on a simple NN (and descent) algorithm was then presented. Adaptations were made to this in the form of a lexigraphical evaluation function (to help utilise the improvement phase to remove vehicles) and a series of other (simpler) techniques aimed at improving the initial solution (in each timeslot) and the feasibility of moves in the neighbourhood.

Results were then produced for the 600-1000 customer problems. These were of a more variable nature, with increases of between 10.37%-14.71% in the number of vehicles deployed (when compared with the best-achieved solutions). Two new best-achieved solutions were produced, and it was noted that by achieving these results with such a simple algorithm the time-based decomposition technique had significant potential.

8.4.1 General Comments and Further Work

As with the discussion of further work for the DVRPTW, this description is not going to focus on the furthering of the algorithm developed but on the problems future development. The time-based decomposition technique was shown to be capable of producing high quality solutions, though its poor performance on certain datasets was noticeable. It is suggested that it may be beneficial to investigate if certain traits of the

datasets (distribution, time window size, ideal capacity utilisation etc.) lend themselves better to this type of problem decomposition.

The ACS algorithm utilised on the 200-400 customer problems is not known for being a particularly fast algorithm (even if it is the most aggressive of the ACO algorithms), and as such, it is proposed that one considers utilising a different technique (e.g. TS). As one is restricted to 12.5 minutes for solving problems of such a significant size, the associated time savings could be very beneficial.

The descent algorithm utilised on the 600-1000 customer problems may have been too simple (if one observes the run-times) for some of the datasets and some more complex form of local search may be beneficial.

The technique has numerous possibilities and it is hoped that they are pursued, allowing one to establish whether one can use this technique to produce further best-achieved solutions. Note that it may also be of interest to consider whether Type 2 problems would benefit from a similar type of decomposition.

8.5 More Diverse Applications

Although all of the work in this thesis related to the solving of routing based problems, there are a number of ideas and techniques discussed in this thesis that could be employed in problems of a more diverse nature. Some details of these possible applications are now presented.

Most combinatorial optimisation problems assume that all of the relevant information is known in advance, and that a solution needs to be produced prior to it being implemented. This is different to many real-life situations, where data evolves over time. As modern technology allows one to process this data in real time, many problems could make use of this information (i.e. become dynamic) so as to increase their realism (e.g. job-shop scheduling, project planning and workforce scheduling).

This thesis demonstrated how problem decomposition could be applied to the VRPTW, if the divisions are made on a time basis (as opposed to geographically as for the CVRP). This method is particularly logical for the VRPTW, but could also be

applied to scheduling problems such as job-shop scheduling and workforce scheduling (especially if certain jobs or employees are not always available). This technique could also be applied to large-scale non-dynamic timetabling problems. One could investigate if better solutions result from dividing the problem into time periods (i.e. solve individually and then combine the solutions) than if one solved the entire problem at once. For example, rather than solving the crew allocation problem in a single attempt, the problem could be solved in stages; the best allocation for each time period could be identified and then combined in such a way that the final crew schedules are feasible.

Obviously these only represent a selection of the alternative problems that could benefit from similar techniques to those implemented in this thesis, and as such, one is encouraged to consider similar ideas when tackling a broad range of combinatorial optimisation problems.

8.6. Closing Comments

This thesis has presented a considerable volume of research detailing an investigation into three different variants of routing problem, and proposed a variety of solution techniques. Many interesting details about these problems were identified, along with the production of seemingly competitive algorithms.

It is hoped that research into these types of problems will benefit from the work in this thesis and that as dynamic routing problems will often better represent the problems people will face in real-life, their further research is encouraged.

References

Aarts, E. H. L. and P. J. M. Korst (1989). Simulated annealing and boltzmann machines. Chichester, Wiley.

Altinkemer, K. and B. Gavish (1991). "Parallel savings based heuristic for the delivery problem." Operations Research **39**: 456-469.

Applegate, D., R. Bixby, V. Chvátal and W. Cook (1998). "On the solution of traveling salesman problems." Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung, International Congress of Mathematicians 645-656.

Archetti, C., A. Hertz and M. G. Speranza (2006). "A tabu search algorithm for the split delivery vehicle routing problem." Transportation science **40**(1): 64-73.

Baker, B. M. and C. A. C. Carreto (2003). "A visual interactive approach to vehicle routing." Computers & Operations Research **30**: 321-337.

Baker, E. and J. Schaffer (1986). "Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints." American Journal of Mathematical and Management Sciences **6**: 261-300.

Barán, B. and M. Schaerer (2003). A multiobjective ant colony system for vehicle routing problem with time windows. Proceedings of 21st IASTED International Conference, ACTA Press: 97-102.

Barbarosoglu, G. and D. Ozgur (1999). "A tabu search algorithm for the vehicle routing problem." Computers & Operations Research **26**: 255-270.

Bell, J. E. and P. R. McMullen (2004). "Ant colony optimization techniques for the vehicle routing problem." Advanced Engineering Informatics **18**: 41-48.

Bent, R. and P. Van Hentenryck (2002). Dynamic vehicle routing with stochastic requests, Technical Report: Brown University.

Bentley, J. L. (1992). "Fast algorithms for geometric traveling salesman problems." ORSA Journal on Computing **4**: 387-411.

Bitattari, M., L. Paquete, T. Stützle and K. Varrentrapp (2001). Classification of metaheuristics and design of experiments for the analysis of components, Technical Report: AIDA.

Blum, C. and A. Roli (2003). "Metaheuristics in combinatorial optimization: overview and conceptual comparison." ACM Computing Surveys **35**(3): 268-308.

Bramel, J. and D. Simchi-Levi (2002). Set-covering-based algorithms for the capacitated VRP. The Vehicle Routing Problem. P. Toth and D. Vigo. Philadelphia, SIAM - Monographs on Discrete Mathematics and Applications.

Brandão, J. (1998). Metaheuristic for the vehicle routing problem with time windows. Boston, Kluwer.

Bräysy, O. (2003). "A reactive variable neighbourhood search for the vehicle routing problem with time windows." INFORMS Journal on Computing **15**(4): 347-368.

Bullnheimer, B., R. F. Hartl and C. Strauß (1999a). "A new rank based version of the ant system: a computational study." Central European Journal of Operations Research and Economics **7**: 25-38.

Bullnheimer, B., R. F. Hartl and C. Strauß (1999b). Applying the ant system to the vehicle routing problem. Metaheuristics: advances and trends in local search paradigms for optimization. S. Voss, S. Martello, I. H. Osman and C. Roucairol. Boston, Kluwer: 285-296.

Bullnheimer, B., R. F. Hartl and C. Strauß (1999c). "An improved ant system for the vehicle routing problem." Annals of Operations Research **89**: 319-328.

Carlton, W. B. (1995). A tabu search approach to the general vehicle routing problem Austin, University of Texas.

Caseau, Y. and F. Larburthe (1999). "Heuristics for large constrained vehicle routing problems." Journal of Heuristics **5**(3): 281-303.

Cerny, V. (1985). "A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm." Journal of Optimization Theory and Applications **45**: 41-55.

Chaovalitwongse, W., D. Kim and P. M. Pardalos (2003). "GRASP with a new local search scheme for vehicle routing problems with time windows." Journal of Combinatorial Optimisation **7**: 179-207.

Chen, C.-H. and C.-J. Ting (2005). "A hybrid ant colony system for the vehicle routing problem with time windows." Journal of the Eastern Asia Society for Transportation Studies **6**: 2822-2836.

Christofides, N. and J. E. Beasley (1984). "The period routing problem." Networks **14**(2): 237-256.

Christofides, N., A. Mingozzi and P. Toth (1979). The vehicle routing problem. Combinatorial Optimization. N. Christofides, A. Mingozzi, P. Toth and C. Sandi. Chichester, Wiley.

Christofides, N., A. Mingozzi and P. Toth (1981). "Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxations." Mathematical Programming **20**: 255-282.

Clarke, G. and J. W. Wright (1962). "Scheduling vehicles from a central depot to a number of delivery points." Operations Research **12**(4): 568-581.

Colorni, A., M. Dorigo and V. Maniezzo (1992). Investigation of some properties of an "Ant algorithm". Proceedings of Second International Conference on Parallel Problem Solving from Nature, Elsevier**:** 509-520.

Cordeau, J.-F., G. Desaulniers, M. Solomon and F. Soumis (2002). VRP with time windows. The Vehicle Routing Problem. P. Toth and D. Vigo. Philadelphia, SIAM - Monographs on Discrete Mathematics and Applications.

Cordeau, J.-F., G. Laporte and A. Mercier (2000). A unified tabu search heuristic for vehicle routing problems with time windows. Montréal, Technical Report: Centre for Research on Transportation.

Croes, G. A. (1958). "A method for solving traveling salesman problems." Operations Research **6**: 791-812.

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

Dantzig, G. B., R. Fulkerson and S. Johnson (1954). "Solution of a large scale traveling salesman problem." Operations Research **2**: 393-410.

Dantzig, G. B. and J. H. Ramser (1959). "The truck dispatching problem." Management Science **6**: 80-91.

De Jong, K. A. (1975). An analysis of the behaviour of a class of genetic adaptive systems, University of Michigan.

Deneubourg, J. L. and S. Goss (1989). "Collective patterns and decision-making." Ethology, Ecology and Evolution **1**: 295-311.

Deneubourg, J. L., J. M. Pasteels and J. C. Verhaeghe (1983). "Probabilistic behaviour in ants: a strategy of errors." Journal of Theoretical Biology **105**: 259-271.

Desrochers, M., J. Desrosiers and M. Solomon (1992). "A new optimization algorithm for the vehicle routing problem with time windows." Operations Research **40**(2): 342-354.

Doerner, K., M. Gronalt, R. F. Hartl, G. Reinelt, H. Strobel and M. Stummer (2002). SavingsAnts for the vehicle routing problem. Proceedings of EvoWorkshops2002, Springer: 11-20.

Dongarra, J. J. (2002). Performance of various computers using standard linear equations software, Technical Report: University of Tennessee.

Dorigo, M. (1992). Optimization, learning and natural algorithms. Dipatimento di Electronica. Milan, Politecnico di Milano.

Dorigo, M. and L. Gambardella (1996). A study of some properties of Ant-Q. Proceedings of Fourth International Conference on Parallel Problem Solving from Nature, Springer-Verlag: 656-665.

Dorigo, M. and L. Gambardella (1997). "Ant colony system: a cooperating learning approach to the traveling salesman problem." IEE Transactions on Evolutionary Computation **1**: 53-66.

Dorigo, M., V. Maniezzo and A. Colorni (1991). Positive feedback as a search strategy, Technical Report: Politecnico di Milan.

Dorigo, M., V. Maniezzo and A. Colorni (1996). "The ant system: optimization by a colony of cooperating agents." IEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics **26**(1): 29-41.

Dorigo, M. and T. Stützle (2004). Ant colony optimisation. Cambridge, MIT Press.

Dror, M., G. Laporte and P. Trudeau (1989). "Vehicle routing with stochastic demands: properties and solution frameworks." Transportation Science **23**: 166-176.

Dueck, G. (1993). "New optimization heuristics - the great deluge algorithm and the record-to-record travel." Journal of Computational Physics **104**(1): 86-92.

Durbin, R. and D. Wilshaw (1987). "An analogue approach to the travelling salesman problem using an elastic net method." Nature **326**: 689-691.

Ellabib, I., O. A. Basir and P. Calamai (2002). An experimental study of a simple ant colony system for the vehicle routing problem with time windows. Proceedings of Ant Algorithms: Third International Workshop, Springer Berlin: 53-64.

Feo, T., G. C. Resende and S. H. Smith (1994). "A greedy randomised adaptive search procedure for maximum independent set." Operations Research **42**: 860-878.

Fisher, M. L. (1994). "Optimal solution of vehicle routing problems using minimum k-trees." Operations Research **42**: 626-642.

Fisher, M. L. and R. Jaikumar (1981). "A generalized assignment heuristic for vehicle routing." Networks **11**(2): 109-124.

Fleurent, C. and F. Glover (1999). "Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory." INFORMS Journal on Computing **11**: 198-204.

Flood, M. M. (1956). "The traveling-salesman problem." Operations Research **4**: 61-75.

Gambardella, L. M. and M. Dorigo (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. Proceeding of Twelfth International Conference on Machine Learning, Morgan Kauffman: 252-260.

Gambardella, L. M. and M. Dorigo (1996). Solving symmetric and asymmetric TSPs by ant colonies. Proceedings of IEEE International Conference on Evolutionary Computation, IEEE Press: 622-627.

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

Gaskell, T. J. (1967). "Bases for vehicle fleet scheduling." Operational Research Quarterly **18**: 281-295.

Gendreau, M., F. Guertin, J. Y. Potvin and E. Taillard (1999). "Parallel tabu search for real-time vehicle routing and dispatching." Transportation Science **33**(4): 381-390.

Gendreau, M., A. Hertz and G. Laporte (1992). "New insertion and postoptimization procedures for the traveling salesman problem." Operations Research **40**: 1086-1094.

Gendreau, M., A. Hertz and G. Laporte (1994). "A tabu search heuristic for the vehicle routing problem." Management Science **40**(10): 1276-1290.

Gendreau, M., A. Hertz, G. Laporte and M. Stan (1998). "A generalized insertion heuristic for the traveling salesman problem with time windows." Operations Research **43**: 330-335.

Gendreau, M., G. Laporte and J. Y. Potvin (2002). Metaheuristics for the capacitated VRP. The vehicle routing problem. P. Toth and D. Vigo. Philadelphia, SIAM - Monographs on Discrete Mathematics and Applications.

Gillet, B. E. and L. R. Miller (1974). "A heuristic algorithm for the vehicle dispatch problem." Operations Research **22**(2): 340-347.

Glover, F. (1977). "Heuristics for integer programming using surrogate constraints." Decision Sciences **8**: 156-166.

Glover, F. (1986). "Future paths for integer programming and links to artificial-intelligence." Computers & Operations Research **13**(5): 533-549.

Glover, F. (1989). "Tabu search, part 1." ORSA Journal on Computing **1**: 190-206.

Glover, F. (1991). Multilevel tabu search and embedded search neighbourhoods for the traveling salesman problem, Technical Report: University of Colorado.

Glover, F. and L. Laguna (1997). Tabu Search. Boston/Dordrecht/London, Kluwer Academic Publishers.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization and machine learning. Reading, Addison Wesley.

Golden, B. L., L. D. Bodin, T. Doyle and W. R. Stewart (1980). "Approximate traveling salesman algorithms." Operations Research **28**: 694-711.

Golden, B. L., T. L. Magnanti and Nguyen (1977). "Implementing vehicle routing algorithms." Networks **7**: 113-148.

Golden, B. L. and W. R. Stewart (1985). Empirical analysis of heuristics. The traveling salesman problem: a guided tour of combinatorial optimization. L. Lawler, Rinnooy Kan, and Shmoys (Editors). Chichester, John Wiley & Sons.

Guntsch, M. and M. Middendorf (2001). Pheromone modification strategies for ant algorithms applied to dynamic TSP. Proceedings of EcoWorkshops 2001. E. J. W. Boers, Springer-Verlag: 213-222.

Guntsch, M. and M. Middendorf (2002). Applying population based ACO to dynamic optimization problems. Ants 2002. M. Dorigo, G. Di Caro and M. Sampels. Berlin, Springer-Verlag.

Hansen, P. and B. Jaumard (1990). "Algorithms for the maximum satisfiability problem." Computing **44**: 279-303.

Hanshar, F. T. and B. M. Ombuki-Berman (2007). "Dynamic vehicle routing using genetic algorithms." Applied Intelligence **27**(1): 89-99.

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

Hoffman, A. J. and P. Wolfe (1985). History. The traveling salesman problem: a guided tour of combinatorial optimization. L. Lawler, Rinnooy Kan, and Shmoys (Editors). Chichester, John Wiley and Sons.

Homberger, J. and H. Gehring (1999). "Two evolutionary meta-heuristics for vehicle routing problem with time windows." INFOR **37**: 297-318.

Ichoua, S., M. Gendreau and J. Y. Potvin (2000). "Diversion issues in real-time vehicle dispatching." Transportation Science **34**(4): 426-438.

Jaillet, P. (1985). Probabilistic traveling salesman problems. Cambridge, Massachusetts Institute of Technology.

Johnson, D. S. and L. A. McGeoch (1995). The traveling salesman problem: a case study in local optimization. Local Search in Combinatorial Optimization. E. H. L. Aarts and J. K. Lenstra. London, John Wiley and Sons.

Kilby, P., P. Prosser and P. Shaw (1998). Dynamic VRPs: a study of scenarios, Technical Report: University of Strathclyde.

Kirkpatrick, S., C. D. Gelatt Jnr and M. P. Vecchi (1984). "Optimization by simulated annealing: quantitative studies." Journal of Statistical Physics **34**(5-6): 975-986.

Knight, K. and J. Hofer (1968). "Vehicle scheduling with timed and connected calls: a case study." Operational Research Quarterly **19**: 299-310.

Knudsen, K. G. (1989). Routing with time windows. Aarhus, Aarhus University.

Kolen, A. W. J., A. G. H. Rinnooy Kan and H. W. J. M. Trienekens (1986). "Vehicle routing with time windows." Operations Research **35**(2): 266-273.

Kontoravdis, G. and J. F. Bard (1995). "A GRASP for the vehicle routing problem with time windows." ORSA Journal on Computing **7**(1): 10-23.

Laguna, M. and J. L. González-Verlarde (1991). "A search heuristic for just-in-time scheduling in parallel machines." Journal of Intelligent Manufacturing **2**: 253-260.

Laporte, G., H. Mercure and Y. Norbert (1986). "An exact algorithm for the asymmetrical capacitated vehicle routing problem." Networks **16**: 33-46.

Laporte, G. and F. Semet (2002). Classical heuristics for the capacitated VRP. The vehicle routing problem. P. Toth and D. Vigo. Philadelphia, SIAM - Monographs on Discrete Mathematics and Applications.

Lenstra, J. K. and A. G. H. Rinnooy Kan (1981). "Complexity of vehicle routing problems." Networks **11**: 221-227.

Li, H. and A. Lim (2003). "Local search with annealing-like restarts to solve the VRPTW." European Journal of Operational Research **150**: 115-127.

Li, Z., S. Guo, F. Wang and A. Lim (2004). Improved GRASP with tabu search for vehicle routing with both time windows and limited number of vehicles. Proceedings of the 17th international conference on Innovations in applied artificial intelligence.

Lin, S. (1965). "Computer solutions of the traveling salesman problem." Bell System Journal **44**: 2245-2269.

Lin, S. and B. W. Kernighan (1973). "An effective heuristic algorithm for the traveling salesman problem." Operations Research **21**(2): 498-516.

Lundy, M. and A. Mees (1986). "Convergence of an annealing algorithm." Mathematical Programming **34**: 111-124.

Madsen, O. B. G. (1976). Optimal scheduling of trucks - a routing problem with tight due times for delivery. Optimization Applied to Transportation Systems. H. Strobel, R. Genser and M. Etchmaier, IIASA: 126-136.

Madsen, O. B. G. (1990). Lagrangean relaxation and vehicle routing, Working Paper, IMSOR, The Technical University of Denmark.

Malandraki, C. and M. S. Daskin (1992). "Time-Dependent vehicle routing problems: formulations, properties and heuristic algorithms." Transportation Science **26**(3): 185-200.

Mester, D. and O. Bräysy (2005). "Active guided evolution strategies for large-scale vehicle routing problems with time windows." Computers & Operations Research **32**: 1593-1614.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller (1953). "Equation of calculations by fast computing machines." Journal of Chemical Physics **21**: 1087-1092.

Mole, R. H. and R. H. Jameson (1976). "A sequential route-building algorithm employing a generalized savings criterion." Operational Research Quarterly **27**: 503-511.

Montemanni, R., L. M. Gambardella, A. E. Rizzoli and A. V. Donati (2005). "Ant colony system for a dynamic vehicle routing problem." Journal of Combinatorial Optimisation **10**(4): 327-343.

Naddef, D. and G. Rinaldi (2002). Branch-and-cut algorithms for the capacitated VRP. The Vehicle Routing Problem. P. Toth and D. Vigo. Philadelphia, SIAM - Monographs on Discrete Mathematics and Applications.

Or, I. (1976). Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking. Department of Industrial Engineering and Management Sciences. Evanston, Northwestern University.

Osman, I. H. (1993). "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem." Annals of Operations Research **41**: 421-451.

Osman, I. H. and K. Hindi (2004). "Editorial." Journal of Mathematical Modelling and Algorithms **3**: 179-181.

Perttunen, J. (1994). "On the significance of the initial solution in travelling salesman heuristics." Journal of Operational Research Society **45**(10): 1131-1140.

Pitsoulis, L. S. and G. C. Resende (2001). Greedy randomized adaptive search procedures, Technical Report: AT&T Labs Research.

Pitsoulis, L. S. and G. C. Resende (2001). Greedy randomized adaptive search procedures. Handbook of Applied Optimization. P. M. Pardalos and G. C. Resende, Oxford University Press: 168-181.

Potvin, J. Y., T. Kervahut, B. Garcia and J.-M. Rousseau (1996). "The vehicle routing problem with time windows - part 1: tabu search." INFORMS Journal on Computing **8**: 158-164.

Potvin, J. Y. and J.-M. Rousseau (1993). "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows." European Journal of Operational Research **66**: 331-340.

Prais, M. and C. C. Ribeiro (2000). "An application to a matrix decomposition problem in TDMA traffic assignment." INFORMS Journal on Computing **12**: 164-176.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. Vehicle Routing: Methods and Studies. B. L. Golden and A. Assad, North Holland.

Psaraftis, H. N. (1995). "Dynamic vehicle routing: status and prospects." Annals of Operations Research **61**: 143-164.

Pullen, H. G. M. and M. H. J. Webb (1967). "A computer application to a transport scheduling problem." Computing Journal **10**: 10-13.

Qili Zhu, K. and K.-L. Ong (2000). A reactive method for real time dynamic vehicle routing, Working Paper, Department of Computer Science, University of Singapore.

Rego, C. and C. Reucairol (1996). A parallel tabu search algorithm using ejection chains for the vehicle routing problem. Meta-Heuristics: Theory and Applications. I. H. Osman and J. P. Kelly. Boston, Kluwer: 661-675.

Reimann, M., K. Doerner and R. F. Hartl (2002). Insertion based ants for vehicle routing problems with backhauls and time windows, Technical Report: University of Vienna.

Reimann, M., K. Doerner and R. F. Hartl (2004). "D-ants: savings based ants divide and conquer the vehicle routing problem." Computers & Operations Research **31**: 563-591.

Reimann, M., M. Stummer and K. Doerner (2002). A savings based ant system for the vehicle routing problem. Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann: 1317-1325.

Reinelt, G. (1994). The traveling salesman. Berlin, Springer-Verlag.

Resende, G. C. (1998). Greedy randomized adaptive search procedures (GRASP), Technical Report: AT&T Labs Research.

Resende, G. C., L. S. Pitsoulis and P. M. Pardalos (2000). "Fortran subroutines for computing approximate solutions of MAX-SAT problems using GRASP." Discrete Applied Mathematics **100**: 95-113.

Robinson, J. B. (1949). "On a Hamiltonian way (a traveling salesman problem)." RAND Research Memorandum **RM-303**: 1-10.

Rochat, Y. and E. Taillard (1995). "Probabalistic diversification and intensification in local search for vehicle routing." Journal of Heuristics **1**: 147-167.

Rosenkrantz, D. J., R. E. Stearns and P. M. Lewis (1977). "An Analysis of Several Heuristics for the Traveling Salesman Problem." SIAM Journal on Computing **6**(3): 563-581.

Russell, R. A. (1977). "An effective heuristic for the m-tour traveling salesman problem with some side conditions." Operations Research **25**(3): 517-524.

Ryan, D. M., C. Hjorring and F. Glover (1993). "Extensions of the petal method for vehicle routing." Journal of Operational Research Society **44**(3): 289-296.

Salhi, S. (2002). "Defining tabu size and aspiration criterion within tabu search methods." Computers & Operations Research **29**: 67-86.

Salhi, S. and G. K. Rand (1987). "Improvements to vehicle routing heuristics." Journal of Operational Research Society **38**: 293-295.

Savelsbergh, M. W. P. (1985). "Local search in routing problems with time windows." Annals of Operations Research **35**: 254-265.

Solomon, M. (1986). "On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints`." Networks **16**: 161-174.

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

Solomon, M. and J. Desrosiers (1988). "Time window constrained routing and scheduling problems." Transportation Science **22**(1): 1-13.

Stewart, W. R. and B. L. Golden (1984). "A lagrangian relaxation heuristics for vehicle routing." European Journal of Operational Research **15**: 84-88.

Stützle, T. and H. H. Hoos (1997). Max-Min ant system and local search for the travelling salesman problem. Proceedings of IEEE International Conference on Evolutionary Computation, IEEE Press: 309-314.

Stützle, T. and H. H. Hoos (2000). "Max-Min ant system." Future Generations Computer Systems **16**(8): 889-914.

Taillard, E. (1991). "Robust taboo search for the quadratic assignment problem." Parallel Computing **17**: 443-455.

Taillard, É. (1994). "Parallel iterative search methods for vehicle-routing problems." Networks **23**(8): 661-673.

Taillard, É., P. Badeau, M. Gendreau, M. Guntsch and J. Y. Potvin (1997). "A tabu search heuristic for the vehicle routing problem with soft time windows." Transportation Science **31**(2): 170-186.

Thompson, P. M. and H. N. Psaraftis (1993). "Cyclic transfer algorithms for multivehicle routing and scheduling problems." Operations Research **41**(5): 935-946.

Toth, P. and D. Vigo (1998). The granular tabu search (and its application to the vehicle routing problem), Technical Report: Università di Bologna.

Toth, P. and D. Vigo (2002). Branch-and-bound algorithms for the capacitated VRP. The Vehicle Routing Problem. P. Toth and D. Vigo. Philadelphia, SIAM - Monographs on Discrete Mathematics and Applications.

Toth, P. and D. Vigo, Eds. (2002). An overview of vehicle routing problems. The Vehicle Routing Problem. Philadelphia, SIAM - Monographs on Discrete Mathematics and Applications.

Van Breedham, A. (1995). "Improvement heuristics for the vehicle routing problem based on simulated annealing." European Journal of Operational Research **86**: 480-490.

Volgenant, A. and R. Jonker (1983). "The symmetric traveling salesman problem edge exchange in minimal 1-trees." European Journal of Operational Research **12**: 394-403.

Watkins, C. J. C. H. (1989). Learning with delayed rewards. Psychology Department. Cambridge, Cambridge University.

Whitely, D., T. Starkweather and D. Fuquay (1989). Scheduling problems and the travelling salesman: the genetic edge recombination operator. Proceedings of Third International Conference on Genetic Algorithms, Morgan Kauffman: 133-140.

Wolpert, D. H. and W. G. Macready (1997). "No free lunch theorems for optimization." IEE Transactions on Evolutionary Computation **1**(1): 67-82.

Xu, J. and J. P. Kelly (1996). "A network flow based tabu search heuristic for the vehicle routing problem." Transportation Science **30**: 379-393.

Yang, W.-H., K. Mathur and R. H. Ballou (2000). "Stochastic vehicle routing problem with restocking." Transportation Science **34**(1): 99-112.