Algorithms for Stochastic Vehicle Routing Problems

Daron R. Roberts

The Management School
Imperial College of Science, Technology and Medicine
University of London
London, UK

Algorithms for Stochastic Vehicle Routing Problems

by

Daron R. Roberts

B.Sc. (Southampton University, UK) 1992

M.Sc. (London School of Economics, University of London, UK) 1994

A thesis submitted for the degree of

Doctor of Philosophy of the University of London

and the

Diploma of Imperial College

The Management School

Imperial College of Science, Technology and Medicine

University of London

London

UK

July 1998

This document was prepared with LaTeX.

Printed in the United Kingdom.

For Liz

Abstract

This thesis is concerned with the study of Stochastic Vehicle Routing Problems (SVRP). We first consider the Vehicle Routing Problem with Stochastic Demands (VRPSD) in which a set of vehicles with fixed capacity must be routed at minimum expected cost over a series of customers with stochastic demands. In a first stage, planned routes are designed. In a second stage, whenever demand exceeds capacity along a route, a recourse cost is realised as the vehicle returns to the depot to refill before continuing along its pre-defined route.

We develop a new exact algorithm, the Paired Tree Search Algorithm (PTSA), for solving the VRPSD. The problem is formulated within the framework of stochastic integer programming and lower bounds are obtained for both the first stage deterministic problem and the second stage stochastic recourse problem. This algorithm is used to optimally solve randomly generated test problems of medium size and to provide heuristic solutions of known quality to VRPSDs of larger size. In addition, a series of VRPSD extensions that incorporate different levels of demand-related information disclosure and route-related reliability are solved to optimality.

The remainder of the thesis is concerned with alternative SVRPs. Algorithms are presented for the multiple Travelling Salesman Problem with Stochastic Travel Times (m-TSPST) and the VRP With Stochastic Service Times (VRPSST). Moreover, a real life application of the VRPSST to a maintenance problem in the utilities sector is presented. The computational implementation of the planning model is described and results are obtained with reference to a pilot study.

Finally, we consider the issue of reoptimisation. An empirical study is completed that assesses the usefulness of a priori strategies (obtained by solving individual SVRPs) as a practical alternative to a posteriori solutions (obtained by successively solving deterministic VRPs). The computational results reported support the hypothesis that the SVRP is a good alternative to the reoptimisation strategy.

Algorithms for Stochastic Vehicle Routing Problems

Daron R. Roberts

The Management School

Imperial College of Science, Technology and Medicine

University of London

London, UK

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Eleni Hadjiconstantinou, for introducing me to the subject of vehicle routing and for her support, encouragement and guidance over the last three years. I would like to thank my former colleagues, Dr. Paul Chu and Dr. Stéphane Marchand-Maillet, for their intellectual insight and, most importantly, for being encyclopedic in their knowledge of UNIX and Latex.

I am particularly grateful to my family and my friends for supporting me in their own individual way during the past few years. I especially wish to thank Rob. His ability to be unstintingly omniscient, honest and intelligent, irrespective of time and circumstance, has always been a constant source of inspiration to me.

Finally, I wish to thank Liz without whom none of this would be worthwhile.

Chapter 1

Introduction

For a number of practical applications, in fields as varied as design, construction, maintenance and engineering, the technique of optimisation, i.e. the selection of a "best" configuration of parameters (decisions) that satisfy a number of specified requirements (constraints) to minimise or maximise the desired benefits (objectives), is of substantial importance. In route planning and scheduling, a large group of such problems are combinatorial in nature in that the decision-making process involves locating an optimal (ordered) combination of items, e.g. places to visit on a tour or commodities to manufacture on a machine. Such problems are referred to as Combinatorial Optimisation Problems (COPs).

This thesis describes, demonstrates and analyses the creation and implementation of a new algorithmic method for solving a class of complex stochastic combinatorial optimisation problems. These problems incorporate elements of uncertainty within their associated problem environments that, if not in conception then in interpretation, are ordinarily considered to be entirely deterministic. In this study, the specific COP for which stochasticity is introduced is the well-known transportation problem, the Vehicle Routing Problem (VRP).

1.1 The Vehicle Routing Problem

The vehicle routing problem is the generic name given to a large class of problems involving the distribution of goods, services, information or personnel. The standard interpretation of a VRP concerns a group of customers, geographically dispersed around a single depot, that require a certain amount of known demand. The solution to such a VRP involves determining how a fleet of vehicles, limited in capacity and stationed at the central facility, must be routed at minimum cost in order to serve the set of customers.

The vehicle routing problem has enjoyed an intensive and successful period of research over the past two or three decades, together with an associated proliferation of related articles. This interest has been fuelled by both academics, rising to the challenge of a complex VRP solution belying a simple original concept, and practitioners, profitably applying VRP modelling techniques to a large number of commercial and industrial applications. It is estimated that transportation costs account for approximately one sixth of the gross national product of the USA [167] and although many theoretical advances and associated algorithms lack the robustness to optimise all of these problems, economic motivation to attempt to do so is certainly more than justified. Moreover, the huge spectrum of practical transportation systems for which VRP analysis can be applied are not all directly related to the physical delivery of commodities. Applications including the transport of units of stock from a warehouse to particular retail outlets, the collection of cash by a security company from a number of geographically dispersed banks, the collection of mail from postboxes and the inspection of gas meters across a range of customer sites, are all examples of VRPs in which the deliveries can be interchanged with collections (or services) and in which the vehicles and commodities can vary in form. Clearly, due to the large number and variety of constraints and/or objectives that can arise in such cases, the existence of a standard, underlying version of the vehicle routing problem, together with a series of alternative practical interpretations of this basic definition, is of inevitable value to the researcher and practitioner; comprehensive classifications and surveys of the VRP can be found in a number of articles in the literature, e.g. [78, 34, 160, 35, 43, 141, 105, 33, 140, 82].

The vehicle routing problem, which has also been studied under the auspices of the vehicle scheduling problem [49, 92, 78, 87, 211], the vehicle dispatch problem [56, 44, 102, 101] and the vehicle delivery problem [9, 202, 165], is a generalisation of the Travelling Salesman Problem (TSP) in which one vehicle of unlimited capacity is used to visit all the customers. Currently, optimal TSP solution methods exist for problems of a few hundred customers [45, 53, 208]. However, despite the amount of alternative vehicle routing formulations and solution methods, current VRPs can only be solved exactly for up to fifty customers in reasonable computational time [164, 81, 6, 114]. This follows solution algorithms for VRPs of up to twelve customers [102], twenty-five customers [46, 47] and forty customers [159, 153].

Needless to say, there exist a large number of heuristic solution methods that locate near-optimal VRP solutions in comparatively small computational times yet without typically providing more than an empirical guarantee on solution quality, e.g. [49, 87, 84, 37].

1.2 The Stochastic Vehicle Routing Problem

In the classical definition of the vehicle routing problem it is assumed that the associated parameters, concerning factors such as cost, customer demands and vehicle travel times, are deterministic. This conjecture is often too simplistic in today's dynamic environment where there exist increasing requirements on levels of productivity and service and a corresponding commitment to enlarged and more elaborate transportation systems. In parallel with the need to manage such a growing number of indeterminate systems there exists an increased amount of data augmentation and volatility. Moreover, although our tools to manage these systems are becoming more sophisticated, our ability to collect and use such incoming data is restricted by our ability to absorb, synthesise and analyse the information made available. Despite increasing computational power, the manual/deterministic methods that are still in use today remain ill-suited to respond in real time to last minute changes and disruptions. It is natural, therefore, that there should be an emerging emphasis on intrinsically allowing for uncertainty and consequently dealing with everyday variations that one encounters in a fast changing dynamic environment. An acceptance of uncertainty implies a movement towards dynamic and stochastic models, characterised by their versatility, which will provide useful and practical information for complex indeterminate systems. These models should be adept at coping with change as opposed to their deterministic counterparts which, apart from fixed mechanical processes, can only remain useful for a limited time-span.

The Stochastic Vehicle Routing Problem (SVRP) differs from the VRP by the introduction of some element of variability within the system in question. In modelling terms, this alteration is usually in the form of some parameter, which was deterministic and known in the VRP, being replaced with a random variable that describes the range of possible values available to the particular parameter. Examples include the Vehicle Routing Problem With Stochastic Demands (VRPSD), the Vehicle Routing Problem With Stochastic Service Times (VRPSST) and the Vehicle Routing Problem With Stochastic

Customers (VRPSC). Each type of problem has its own set of characteristics that are unique to the problem concerned however several concepts and properties overlap and can be applied to some or all of the individual SVRPs. For example, all SVRPs belong to a class of *a priori* optimisation problems for which it is impractical to consider an *a posteriori* approach that computes an optimal solution whenever the random variables are realised. Instead, prior to a "perfect information" state and the recognition of any single deterministic realisation, i.e. "scenario" or "state of nature", an a priori strategy attempts to obtain the "best" solution possible, over all realisations of the random variables involved. In fact, this inherent stability of a SVRP solution may provide an added benefit in certain practical cases due to the possible presence of inflexible labour assignments and/or a need to provide a seamless service in a customer-driven environment.

The SVRP, in all its guises, has seen relatively little research compared to its well known deterministic parent problem. Correspondingly, solution methods for such problems are scarce. Indeed, only a small number of both heuristic and optimal solution methods for Stochastic Combinatorial Optimisation Problems (SCOPs) of any kind can be found in the literature, e.g. the Stochastic Shortest Path (SSP) problem [89, 161, 188, 5, 90, 118], the SSP problem with recourse [52, 5], the stochastic bottleneck spanning tree problem [124, 122, 123], the probabilistic project evaluation and review technique [206, 11], the stochastic delivery man problem [7] and the stochastic location problem [88, 209, 158, 79]. The article that first considered a vehicle routing problem with stochastic elements appeared thirty years ago, [201], however only a very small number of similar SVRP studies have since been completed. For the most part, optimal solution methods have been designed for special cases of individual SVRPs of very small size and simple VRP-based heuristic approaches have been designed for practical applications involving SVRPs of medium size. Without doubt, the enormous advances experienced in the analysis of the deterministic VRP has not coincided with a similar furtherance in the study of its stochastic equivalent. Given the number of potential applications, this lack of parity is conjectured to be for a number of reasons. Firstly, there exists no classification of individual SVRPs, together with their unique and generic characteristics, with which to corroborate the few studies that have been completed. Secondly, there exists no considered outline concerning the possible implications of applying stochastic routing problems in practice. Consequently, there exist no concurrent series of extensions or interpretations of individual SVRPs that correspond to problems involving the various practical constraints and characteristics arising out of the presence of these different stochastic occurrences. Thirdly, and most importantly, the enormous complexity that the addition of a stochastic element brings to an already difficult COP has hindered research in terms of motivation and possibilities. This can be seen both mathematically, in terms of the scarcity of related theoretical approaches, and practically, in terms of the absence of any algorithms that can find optimal solutions to such problems in reasonable time and without experiencing problems with computer memory requirements.

1.3 Research Motives and Goals

This thesis is concerned with the study of stochastic vehicle routing problems. Initially, we introduce individual forms of the SVRP before considering one particular representation in more detail. Apart from creating a platform with which to test the principal SVRP algorithm presented in the thesis, this reduction in problem scope will lead to a greater understanding not only of the particular problem involved but also of how to classify and properly study any other SVRP. The final part of the research will involve the expansion of the methods and concepts established in the study to explore SVRP in a wider and more applicable context.

The SVRP variation which forms the basis for much of the work in the thesis is the vehicle routing problem with stochastic demands. This problem is the most researched problem of its kind and, during the initial survey and classification process, serves as an ideal framework with which to consider the fundamental structural changes that occur to a routing problem once a stochastic element has been added and the broad planning issues that need to be addressed when optimising such a stochastic problem. These characteristics, which are unique to stochastic routing problems, have never been formally classified in so thorough a fashion that ambiguity should not continue to cloud the associated mathematical modelling process. By considering the emblematic issues surrounding an individual SVRP and creating a series of appropriate substitute problems or extensions, this continual hindrance to research will be addressed.

The main objective of this research is to create a new algorithmic method for solving a series of stochastic routing problems. Initially, algorithm development will be directed towards solving a basic interpretation of the VRPSD, defined in the earlier classification process, before attempting to generalise the procedure in the latter half of the study. Importantly, during the development of the algorithm, priority will be given to the optimality of the derived solution, even at the expense of increased computational time. This is due to a number of factors given as follows.

- SVRP is a comparatively young COP and exact algorithms offer the best method to learn about the properties and characteristics of such problems for current and future research purposes. Sensitivity analysis can be carried out without concern for ambiguity and/or the presence of sampling errors and empirical conclusions hold more weight with the added guarantee of solution optimality. Furthermore, only one or two SVRP variations have ever been solved to optimality and no interpretations of an individual SVRP have ever been contrasted, hence an optimal solution method is also more beneficial than a near-optimal method from the point of view of uniqueness and originality. A standard group of test problems can, for example, be produced for future reference.

- An a priori SVRP solution involves a fixed routing strategy in a problem environment that contains a component which is usually stochastic over time. Therefore, since the duration of such a time interval will ordinarily not be insignificant due to the inherent uncertainty, "in most cases it is not necessary to solve [a stochastic] vehicle routing problem....in real time" [197].

- Optimisation for vehicle routing in general is increasingly considered to be a more practical approach for real problems since associated "algorithms offer the best promise for achieving robustness" [82]. This is because of three main reasons:

  (i) Rapidly decreasing costs of computation make higher quality solutions more obtainable.

  (ii) Even if an exact algorithm is not run to completion, the best available solution will often be better than what existing heuristics can provide.

  (iii) During the implementation of an exact algorithm, a precise bound between the best current solution and the theoretical optimum is maintained.

In addition to a pre-requisite of solution optimality, the new algorithmic procedure will be designed to be generic in nature due to a number of reasons given as follows.

- An attempt is to be made to extend the method to solve a full cross-section of VRPSD interpretations.
- An attempt is to be made to extend the method to solve a number of alternative SVRPs.
- The method is to be adapted for use in an applied setting that may require an additional number of unique constraints.

Following the design and empirical analysis of the modified algorithm in the cases of particular VRPSD extensions, alternative SVRPs and an applied example, the aim of the final part of the study is to consider the effectiveness of the SVRP approach in comparison to the "hypothetical" approach of perfect information, i.e. how close is the "stochastic" a priori solution to the "deterministic" a posteriori solution, obtained by averaging optimal VRP solutions for all possible "scenarios"/"states of nature" arising out of the uncertain problem environment. Such a conclusion to the thesis will provide a performance measure for the stochastic routing approaches studied previously.

In summary, our main goals in this research are fivefold:

(i) We intend to classify and survey the field of stochastic vehicle routing before deriving a group of practical extensions for one SVRP, namely the VRPSD.

(ii) We aim to create and implement a new algorithm designed primarily to find optimal solutions to the VRPSD.

(iii) We aim to extend the algorithm to solve different VRPSD extensions and alternative SVRP variations.

(iv) We hope to demonstrate the usefulness of studying stochastic routing problems by applying the algorithm to a practical case study.

(v) We hope to demonstrate the performance of the algorithm by empirically evaluating the quality of its associated solution against a theoretical "best".

1.4 Outline of the thesis

The goal of this chapter has been to introduce the general concept of vehicle routing as an important research field in operations research and to very broadly describe the state of the art. Moreover, the introduction of a stochastic emphasis to the orthodox deterministic vehicle routing approach has been discussed and the considerable practical justification for studying such problems has been highlighted. Specifically, the detailed motivation for embarking on this research, together with associated aims and objectives, has been outlined. The remaining chapters of the thesis will further explore the SVRP and will describe the creation of a new algorithmic technique for solving such a class of problems. The chapters are organised as follows.

In Chapter 2, a revised version of Roberts and Hadjiconstantinou [177], we provide an overview of the deterministic vehicle routing problem before reviewing SVRP in detail. Formulations and basic definitions are given for individual SVRPs before a thorough survey of associated methods, properties, concepts and applications. Particular attention is given to the VRPSD as the model for describing most of the algorithmic work in this thesis. There is an extensive discussion concerning the conflicting objectives, varied constraints and inherent subjectivity of the "basic" VRPSD. Concentrating on the broad planning issues involved, this discussion encourages the production of a more complete problem definition and, consequently, the attainment of a clearer VRPSD (and SVRP) methodological perspective. In addition, a group of standard VRPSD extensions is introduced and, together with the holistic framework given for the SVRP in general, this taxonomic process provides a suitable basis from which to structure the remainder of the thesis.

In Chapter 3, a revised version of Roberts and Hadjiconstantinou [180], we describe a new algorithm, known as the Paired Tree Search Algorithm (PTSA), that has been designed to find exact solutions to a number of stochastic vehicle routing problems. The PTSA, which has its foundations in a dynamic stochastic decision tree approach, operates with the use of two linked trees. The nested branching scheme adopted by the PTSA is fully described, together with its associated notation. The new algorithm, in its crudest form, is tested on a small number of randomly generated problems and issues concerning computational complexity and sensitivity analysis are addressed.

In Chapter 4, VRPSD lower bounds that can be applied to the PTSA are formulated, see Roberts and Hadjiconstantinou [178, 181]. Computational results are given for a large number of randomly generated test problems and the efficiency and performance of the algorithm is highlighted in comparison with the only similar work in the literature. In addition, the PTSA is used to provide heuristic solutions of known quality for VRPSDs of larger size.

In Chapter 5, we solve a number of extensions of the VRPSD that are first defined in Chapter 2. A number of issues are discussed including different levels of demand-related information disclosure and route-related reliability. Emphasis is also placed on empirically testing a number of properties that were conjectured earlier using a number of randomly generated test problems.

In Chapter 6, a condensed version of which is to appear in Hadjiconstantinou and Roberts [115], the relevant modifications of the PTSA for other SVRPs is discussed. Algorithms based on the PTSA are presented for the multiple Travelling Salesman Problem with Stochastic Travel Times (m-TSPST) and the Vehicle Routing Problem with Stochastic Service Times (VRPSST). Moreover, a real life application of the VRPSST to a maintenance problem in the utilities sector is presented, see Roberts and Hadjiconstantinou [179]. The computational implementation of the planning model is described and results are obtained with reference to a pilot study.

In Chapter 7, we consider the issue of reoptimisation, see Roberts and Hadjiconstantinou [182, 183]. An empirical study is completed that assesses the usefulness of a priori strategies (obtained by solving individual SVRPs) as a practical alternative to a posteriori solutions (obtained by successively solving deterministic VRPs). The necessary VRP sampling procedures are detailed and a thorough testing process, using test problems from the literature, is described. This final part of the study is used to highlight the validity of the hypothesis that the SVRP approach is a good alternative to the reoptimisation strategy.

In Chapter 8, we provide a summary of the entire thesis. The main contributions derived from this thesis are highlighted. Some issues related to this research, including current limitations of this work, are discussed. Finally, we provide several suggestions for future research.

Chapter 2

The Stochastic Vehicle Routing Problem

2.1 Introduction

In this chapter we formally define the basic version of the vehicle routing problem before reviewing existing heuristic and exact VRP solution methods. Then, following the introduction of a number of formal "basic" SVRPs, stochastic vehicle routing is comprehensively reviewed. The vehicle routing problem with stochastic demands is considered in particular detail, together with a classification of the issues that arise in its associated practical problem environment and the introduction of a concurrent set of substitute problems known as VRPSD extensions.

2.2 The Basic Vehicle Routing Problem

The standard interpretation of the VRP which has developed over the years to be known as the Basic Vehicle Routing Problem (BVRP) is the problem of designing a minimum cost set of routes for a fleet of homogeneous delivery vehicles of limited capacity in order to satisfy a set of given customer demands. The routes must be designed so that:

(i) Each customer demand location is served exactly once by one vehicle.

(ii) Total demand on any one route is less than or equal to vehicle capacity.

(iii) Each route begins and ends at the central depot.

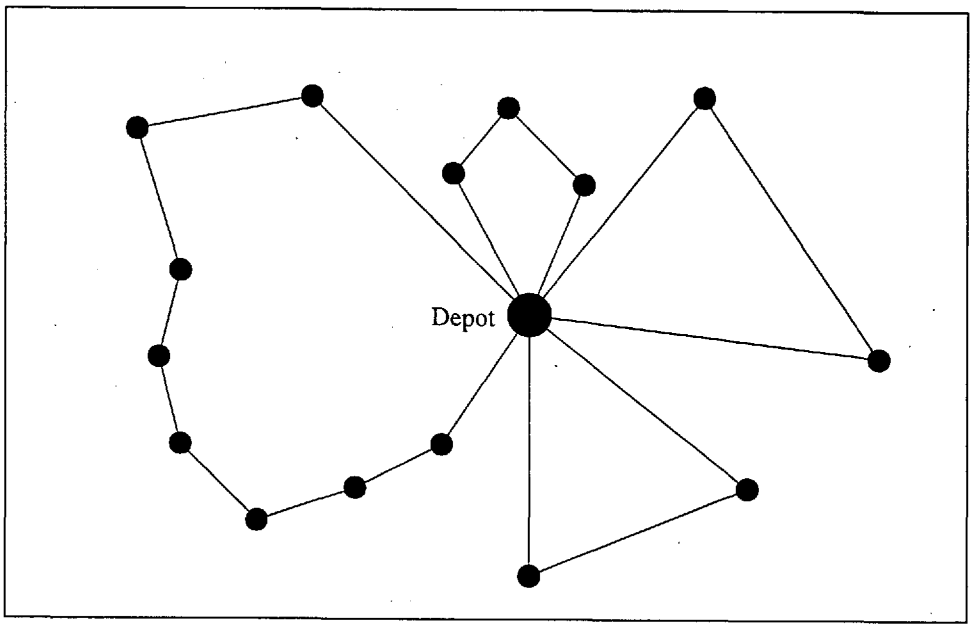

Figure 2.1: An Example of a VRP Solution

2.2.1 Definition

The BVRP is defined on a graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ corresponds to the set of vertices and $E = \{(v_i, v_j) : v_i, v_j \in V\}$ corresponds to the set of edges. The vertices have known and fixed locations and every edge (v_i, v_j) has an associated non-negative distance, or travel cost, c_{ij}. It is assumed that the graph is symmetrical and the matrix $C = (c_{ij})$ satisfies the triangular inequality, i.e. (v_i, v_j) is only defined for $i < j$ and $(c_{ik} + c_{kj} \geq c_{ij} \ \forall \ i, j, k)$. Vertex v_1 represents a depot, indexed by $i = 1$, at which a homogeneous fleet of m vehicles, each of capacity Q, is based. The remaining vertices, $V \backslash v_1$, correspond to a set of customers, indexed by $i \in \{2, \ldots, n\}$, with known demands q_i. The objective is to design a minimum cost-set of routes such that all demands are satisfied, each route starts and ends at the depot, each customer is visited exactly once by one vehicle and total demand on any individual route does not exceed capacity Q. Figure 2.1 shows the shape of a typical BVRP solution; in this case there are four vehicle routes and fifteen customers, i.e. $m = 4$ and $n = 16$.

2.2.2 Practical Interpretations

There are a large number of practical VRP applications in which the definition of the BVRP fails to satisfy particular requirements. This has resulted in the development of a number of practical interpretations of the BVRP that incorporate additional constraints, alternative objectives and/or applied assumptions. For instance, objectives of the BVRP that may be more applicable in practice include minimising the sum of fixed and variable routing costs or simply minimising the total number of vehicles used. Such BVRP interpretations are referred to as fleet size and mix problems [106, 10]. Moreover, a whole new class of problems that are usually referred to as Vehicle Scheduling Problems (VSPs) have arisen following the addition of temporal constraints into the original BVRP definition, see [63]. A number of these and other practical conditions are further specified as follows.[1]

1. Drivers (vehicles) may have overall time (distance) restrictions where the duration (length) of a route must not exceed a given time (distance) limit T, e.g. T equates to an operating time limit where there exist travel times (t_{ij}) between customers. This problem is referred to as the distance constrained routing problem [155].

2. Customers may have particular time intervals within which a service must take place. This problem is known as the VRP with Time Windows (VRPTW) and is comparatively well-researched in the literature [136, 61, 191, 60, 63].

3. Precedence relations may exist between sets of customers, i.e. vertex v_i must be visited before v_j, see [65, 173, 174, 175].

4. Customers may have to be serviced periodically, i.e. a customer that requires a delivery once every t days should be visited on T/t occasions during a period T. The latter interpretation of the BVRP arises a great deal in practice and is known as the Period Vehicle Routing Problem (PVRP), see [185, 93, 40, 113].

5. Vehicles may be based at a number of depots, i.e vehicles can begin and end at different locations, and so each depot cannot be considered in isolation. This problem, known as the Multiple Depot Vehicle Routing Problem (MDVRP), is another well-researched BVRP interpretation [101, 151, 156].

6. Vehicles may be heterogeneous in nature, i.e. each vehicle has a distinct capacity. This problem is often referred to as the Vehicle Routing Problem with Multiple Vehicle Types (VRPMVT). Alternatively, there may exist multiple vehicle capacity

[1]The citations given are not exhaustive and correspond in the main to recent papers and/or surveys.

restrictions where each vehicle has a number of distinct compartments. This can occur when vehicles are delivering different commodities or there exists a number of attributes relating to one commodity, e.g. weight and volume.

7. Deliveries may be allowed to be split between vehicles, i.e. customers can be serviced by more than one vehicle. This problem is known as the Vehicle Routing Problem with Split Deliveries (SDVRP), see [73, 74, 71].

8. Mixed customer collections and customer deliveries may be allowed, i.e. the Pickup and Delivery Problem (PDP), see [65, 75]. Alternatively, if vehicles must complete a series of collections before completing a series of deliveries or vice versa then the problem is known as the Vehicle Routing Problem with Backhauls (VRPB), see [204, 163].

9. Customer demands may be homogeneous [116, 3].

Table 2.1 displays a complete list of these practical characteristics and is based on similar classifications given in the literature. The characteristics of the TSP, the TSP variation that employs m vehicles of unlimited capacity, known as the multiple Travelling Salesman Problem (m-TSP), and the BVRP are also highlighted.

2.2.3 Computational Complexity and Solution Approaches

Like the majority of COPs, the BVRP and all its practical interpretations belong to a set of NP-complete problems, see [134, 91, 154, 169]. Problems in this set share the characteristic that all currently known algorithms for finding optimal solutions to these problems require a number of computational steps that, in the worst case, grows as an exponential (nondeterministic polynomial[2]) function of the "size" of the given problem. In the case of the TSP and the VRP, problem "size" generally refers to the number of geographical locations that must be visited, i.e. n.

The members of the set of NP-complete problems are equivalent in the sense that, if a solution algorithm can be developed that performs in a polynomial number of steps then such an algorithm can be developed for all the members of the set. It is, however, conjectured that no algorithm will ever be developed that is actually capable of solving

[2]NP is an abbreviation for nondeterministic polynomial.

Characteristics	Options	TSP	m-TSP	BVRP
Objectives	Minimise variable routing costs	●	●	●
	Minimise fixed and variable costs			
	Minimise total vehicles required			
Depot types	Single depot	●	●	●
	Multiple depot (MDVRP)			
Fleet sizes	Single vehicle	●		
	Multiple vehicle		●	●
Fleet types	Homogeneous	●	●	●
	Heterogeneous (VRPMVT)			
Vehicle types	Unlimited capacity	●	●	
	Single compartment			●
	Multiple compartments			
Vehicle operations	Deliveries (collections)			●
	Split deliveries (SDVRP)			
	Mixed deliveries and collections (PDP)			
	Backhauls (VRPB)			
Vehicle properties	Maximum route times			
	Maximum customer visits			
Customer types	Precedence relations (priorities)			
	Definitive service times			
	Time windows (VRPTW)			
	Multiple services (PVRP)			
	Demands-equal (complete)			
	Demands-unequal (complete)			●
	Demands-unequal (not binding)			

Table 2.1: The Characteristics of Practical Vehicle Routing Problems

NP-complete problems in a polynomial number of steps. For this reason, a considerable number of polynomial-bounded heuristic (near optimal) algorithms have been developed for a range of NP-complete problems including the VRP. Loss of solution quality in the use of such heuristics is countered by increased computational efficiency. Nevertheless, exact approaches are far from redundant for solving such problems. Although restricted to solving problems of modest size, optimal methods may be more beneficial as a solution approach, especially for a comparatively new problems (see Section 1.3). In addition, it should be noted that NP-completeness is a worst-case phenomenon and the average-case may be polynomial, e.g. the Simplex algorithm for linear programming problems. Indeed, even though they are usually treated separately, exact and heuristic approaches can often complement each other, see Section 2.3.2. Many exact algorithms contain intuitive/heuristic rules that shorten the search for an optimum and the attempt to construct optimal solution methods can yield a series of insights into a problem that can lead to an efficient heuristic.

2.3 VRP Formulations and Solution Methods

The connection between theory and practice in vehicle routing began in 1959 when an article was written that described an application of VRP to the delivery of gasoline to gas stations [56]. This study outlined the first generic formulation of the VRP and suggested a possible heuristic solution method for the particular case study involved. Five years later, a general extension of this algorithm was used in another practical application involving the delivery of nondurable commodities to wholesalers in the UK [49]. The greedy heuristic presented in this study has since become known as the Clark-Wright Savings Algorithm and is still in use today.

Before reviewing the solution methods available for tackling the vehicle routing problem[3], we will introduce a number of mathematical formulations for the VRP that form the basis for some of the heuristic and exact approaches to be discussed in the next section.

2.3.1 Formulations

The following review of VRP formulations is not exhaustive and many other versions can be found in the literature, e.g. the vehicle flow-related integer program introduced by Golden [107], the commodity flow based formulation presented by Gavish and Graves [94, 95] and the modified assignment formulation presented by Laporte [148].

A Three-Index Vehicle Flow Formulation

Fisher and Jaikumar [83, 84] consider a three-index vehicle flow VRP formulation where x_{ijk} variables indicate whether or not the arc (v_i, v_j) is traversed by vehicle k. The variables can be defined as follows:

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ travels directly from node } v_i \text{ to } v_j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

[3]In the following discussion, the "vehicle routing problem" can be taken to refer to the BVRP unless stated otherwise.

A compact formulation for the VRP can then be given as follows.

$$\min_x \quad z = \sum_{i,j} \sum_k c_{ij} x_{ijk} \tag{2.2}$$

subject to

$$\sum_i q_i x_{ijk} \leq Q \qquad (k = 1, \ldots, m), \tag{2.3}$$

$$x = [x_{ijk}] \in S_m. \tag{2.4}$$

where S_m corresponds to the set of feasible solutions to the m-TSP and constraints (2.3) do not allow vehicle capacity to be exceeded along any one route. Moreover, by introducing binary y_{ik} variables that indicate which customers are assigned to particular vehicles, i.e.

$$y_{ik} = \begin{cases} 1 & \text{if node } v_i \text{ is visited by vehicle } k, \\ 0 & \text{otherwise,} \end{cases} \tag{2.5}$$

Fisher and Jaikumar expand (2.2)-(2.4) to explicitly formulate the VRP as follows.

$$\min_x \quad z = \sum_{i,j} \sum_k c_{ij} x_{ijk} \tag{2.6}$$

subject to

$$\sum_k y_{ik} = \begin{cases} 1, & (i = 2, \ldots, n), \\ m, & (i = 1), \end{cases} \tag{2.7}$$

$$\sum_i q_i y_{ik} \leq Q \qquad (k = 1, \ldots, m), \tag{2.8}$$

$$\sum_i x_{ijk} = y_{jk} \qquad (j = 1, \ldots, n;\ k = 1, \ldots, m), \tag{2.9}$$

$$\sum_j x_{ijk} = y_{ik} \qquad (i = 1, \ldots, n;\ k = 1, \ldots, m), \tag{2.10}$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \qquad (S \subset V;\ |S| \geq 2;\ k = 1, \ldots, m), \tag{2.11}$$

$$y_{ik} \in \{0, 1\} \qquad (i = 1, \ldots, n;\ k = 1, \ldots, m), \tag{2.12}$$

$$x_{ijk} \in \{0, 1\} \qquad (i, j = 1, \ldots, n;\ k = 1, \ldots, m). \tag{2.13}$$

These constraints specify that every customer is assigned to one vehicle (and that the depot is visited by all vehicles), constraints (2.7), that every vehicle which enters a node will also leave the same node, constraints (2.9 - 2.10), and that subtours are eliminated, constraints (2.11). The latter constraints, which work since a cycle over the set of nodes

S must contain $|S|$ arcs, can be represented in a number of ways, see [55, 162, 107].

Note, by replacing Q with the variable Q_k in (2.8), this construction allows a heterogeneous vehicle interpretation of the BVRP to be modelled.

A Two-Index Vehicle Flow Formulation

Laporte *et al* [150] present a two-index vehicle flow formulation that is obtained by removing index k from the variables introduced in (2.1). More specifically, variables x_{ij} correspond to how many times an edge (v_i, v_j) is traversed by a vehicle and can be defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \text{ is used in the solution and } 1 \le i < j \le n, \\ 2 & \text{if } (v_i, v_j) \text{ is used as a return trip and } i = 1, j > 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.14}$$

If $v(S)$ is a lower bound on the number of vehicles required to service all the vertices of S in the optimal solution, given by:

$$v(S) = \left\lceil \frac{\sum_{i \in S} d_i.}{Q} \right\rceil \tag{2.15}$$

where $\lceil * \rceil$ represents the smallest integer not less than $*$, then the VRP can then be formulated as follows.

$$\min_x \ z = \sum_{i<j} c_{ij} x_{ij} \tag{2.16}$$

subject to

$$\sum_{j=2}^{n} x_{1j} = 2m, \tag{2.17}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad (k = 2, \ldots, n), \tag{2.18}$$

$$\sum_{i,j \in S} x_{ij} \le |S| - v(S) \qquad (S \subset V \backslash \{v_1\};\ 2 \le |S| \le n-2), \tag{2.19}$$

$$x_{1j} \in \{0, 1, 2\} \qquad (i = 2, \ldots, n), \tag{2.20}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j = 2, \ldots, n). \tag{2.21}$$

The constraints (2.17) specify that m vehicles enter and leave the depot, the vertex degree constraints (2.18) specify that a customer receives a visit exactly once and the classical connectivity constraints (2.19) specify that individual routes disconnected from the depot are prohibited. The latter constraints are obtained by observing that for any $S \subset V\backslash\{v_1\}$, $3 \leq |S| \leq n-3$, we must have:

$$\sum_{\substack{i\in S, j\in \overline{S} \\ \text{or } i\in S, j\in \overline{S}}} x_{ij} \geq 2v(S), \tag{2.22}$$

and that the following identity holds:

$$\sum_{k\in S}\left(\sum_{i<k} x_{ik} + \sum_{k<j} x_{kj}\right) = 2\sum_{\substack{i,j\in S \\ i<j}} x_{ij} + \sum_{\substack{i\in S, j\in \overline{S} \\ \text{or } j\in S, i\in \overline{S}}} x_{ij}. \tag{2.23}$$

A Set Partitioning Formulation

Balinksi and Quandt [9] introduce a set partitioning formulation for the VRP. Let $R = \{1,\ldots,\hat{r}\}$ represent the family of possible feasible routes in the VRP and let the index set of the customers in route r be N_r. Let d_r be the cost of a route and let Q_r be the total load of a route given by:

$$Q_r = \sum_{i\in N_r} q_i. \tag{2.24}$$

Now, if M_i corresponds to the index set of routes visiting customer i and the binary variables y_r take the value 1 if and only if route r is used in the optimal solution, then the VRP can be formulated as follows.

$$\min\ z = \sum_{r\in R} d_r y_r \tag{2.25}$$

subject to

$$\sum_{r\in M_i} y_r = 1, \qquad (i = 2,\ldots,n), \tag{2.26}$$

$$\sum_{r\in R} y_r = m \tag{2.27}$$

$$y_r \in \{0,1\} \qquad (r \in R). \tag{2.28}$$

The constraints (2.26) express the fact that every customer must be served by exactly one vehicle (route) and the constraint (2.27) expresses the fact that the number of vehicles used is fixed and equal to m.

A Dynamic Programming Formulation

The Dynamic Programming (DP) formulation for the VRP was first presented by Eilon *et al* [78]. Let $X' = \{2, \ldots, n\}$ represent the set of customers and for any $T \subseteq X'$:

(i) let $f(k,T)$ be the minimum cost of servicing the customers in T using only $k \leq m$ vehicles,

(ii) let $v(T)$ be the minimum cost of a solution to the TSP defined by the depot and the customers in T, and,

(iii) let $q(T) = \sum_{i \in T} q_i$.

The minimum cost VRP solution can then be obtained through the following recursion:

$$f(k,T) = \begin{cases} v(T) & (k = 1) \\ \min_{S \subset T} [f(k-1, T - T^*) + v(T^*)] & (k > 1) \end{cases} \tag{2.29}$$

subject to

$$q(T) - (k-1)Q \leq q(T^*) \leq Q \qquad (k = 1, \ldots, m), \tag{2.30}$$

$$\tfrac{1}{m-k} q(X' - T) \leq q(T^*) \leq \tfrac{1}{k} q(T) \qquad (k \neq m), \tag{2.31}$$

$$q(X') - (m-k)Q \leq q(T) \leq kQ \qquad (k = 1, \ldots, m). \tag{2.32}$$

The solution cost is equal to $f(m, X')$ and the optimal solution corresponds to the optimising subsets T^* in (2.29).

2.3.2 Solution Methods

In the last thirty years, three main branches of study have developed in the research of vehicle routing problems. Fisher [82] refers to three separate generations of methodologies. The first relates to the development of simple heuristics to obtain computationally fast VRP solutions of varying quality. The second applies the tools of Mathematical Programming (MP), together with associated formulations, to find optimal solutions of sub-problems to the VRP which can be used as a foundation for good quality heuristics to the overall problem. The third involves two very different solution approaches: the development of expert systems and Artificial Intelligence (AI) search techniques, such as simulated annealing and tabu search, to obtain near-optimal solutions of good quality, and the development of optimisation algorithms, based on polyhedral combinatorics and relaxation techniques, to locate exact solutions to VRPs of medium size. These two disparate approaches are grouped together since they are both in active development today.

Here, we review the three methodologies in turn[4].

Generation 1: Simple Heuristics

Simple VRP heuristics can be categorised into three different methods: route building, route improvement and two-phase. Route building heuristics are straight-forward in that arcs along a vehicle route are selected sequentially until a feasible solution is found. There will usually exist some minimisation criteria on the selection of appropriate arcs and such arcs will be chosen so that they do not violate the vehicle capacity constraints. The simplicity of these algorithms can be illustrated by using the Clarke-Wright savings algorithm as an example. The algorithm begins with an infeasible solution that relates to the pairing of vehicles with customers. Single routes are then combined to use one less vehicle and to contribute a saving to the overall cost of the problem, denoted by s_{ij}. For two customers, i and j, served individually, s_{ij} is given by:

$$s_{ij} = (c_{1i} + c_{i1} + c_{1j} + c_{j1}) - (c_{1i} + c_{ij} + c_{j1}) = c_{1i} + c_{1j} - c_{ij} \tag{2.33}$$

The savings algorithm simply selects the arc (v_i, v_j) that provides the maximum value of s_{ij} given that the derived route is feasible, i.e. vehicle capacity is not exceeded. Then, customers i and j are considered to correspond to one individual customer k such that another customer l can be connected to k at a cost of $\min[c_{il}, c_{jl}]$. The route building process continues iteratively, either by completing one vehicle then the next or by adding a customer sequentially to each vehicle route. Recent modifications of this original method all retain the essential structure of the algorithm, see [92, 215, 107, 4].

Route improvement heuristics for the VRP involve selecting a random m-TSP tour and improving on the current solution by the deletion and insertion of arcs such that the capacity constraints are not violated. Christofides and Eilon [44] were the first to use such an approach for the VRP by adapting work by Lin [157] for the TSP. In contrast to Lin's 2-optimal approach, the authors derived a 3-optimal method which involves reaching a final

[4]For the third generation, AI and exact methods will be treated separately as 3A and 3B.

solution such that no improvement is possible by eliminating three links and replacing them by three others. Clearly, the probability that a r-optimal tour is minimal increases with r but the amount of computations required to produce such a tour increases rapidly with r. Setting r to 3 is usually preferred although considerable improvements can be obtained by a selective exchange of more than three arcs, see [184]. Recent research on similar methods have been completed by Thompson [199] and Savelsberg [186].

Two phase VRP heuristics involve assigning customers to vehicles (phase 1) before a travelling salesman algorithm is used to optimally route such assignments (phase 2). The most famous two-phase method for the VRP is the "sweep" algorithm presented by Gillet and Miller [102] following studies by Wren [212] and Wren and Holliday [213]. Customers are first represented in a polar coordinate system with the origin at the depot. A circular sweep then occurs and customers are assigned to a vehicle. Once vehicle capacity is exceeded, the most recent customer is assigned to a new vehicle and the algorithm proceeds. Another two-phase approach includes the use of user-controlled parameters in the construction of a least cost insertion criteria that can lead to different solutions in different trials, see [45].

In a comparison of these simple approaches, see [82], the heuristic that selectively extended Christofides and Eilon's 3-opt algorithm appeared to provide the best results but Clark and Wright's algorithm remains the most widely used method to obtain reasonable solutions in very short computational times. Certainly, one aspect that unites all these heuristics is the omission of an effective procedure for dealing with differences in customer demands; poor quality solutions can be obtained when capacity constraints are tight.

Generation 2: MP-based Heuristics

Mathematical programming based heuristics involve formulating and solving a VRP sub-problem to obtain a good near optimal solution of the overall problem. Indeed, MP-based approaches blur the boundary between exact and heuristic solution methods since the associated algorithms will often obtain optimal solutions if they are run to completion.

The MP-based two-phase method proposed by Fisher and Jaikumar [84] operates in a similar manner to the "sweep" algorithm given above. Customers are assigned to vehicles before the cost of an optimal travelling salesman tour is derived for each assignment. In phase 1 however, the optimal solution to a Generalised Assignment Problem (GAP) is utilised. This VRP sub-problem is a well known COP which involves finding the minimum cost assignment of n jobs to m agents such that each job is assigned to exactly one agent, subject to the agent's available capacity. GAP is well-researched in the literature, see [39], and has been solved optimally for problems of up to ten agents and fifty jobs, see [112]. The transfer of GAP to the assignment problem in the VRP is fairly intuitive. First, m customers are chosen to be the "seeds" of the customer clusters and a vehicle is allocated to each of them. The GAP is then optimised following the inclusion, for each customer i and each cluster k, an insertion cost, d_{ik}, relative to the cluster seed, i_k, where d_{ik} is given by:

$$d_{ik} = c_{1i} + c_{ii_k} - c_{1i_k}. \tag{2.34}$$

The mathematical programming nature of this approach can be highlighted using Fisher and Jaikumar's corresponding three-index vehicle flow formulation given in Section 2.3.1. Constraints (2.7), (2.8) and (2.12) actually define the generalised assignment problem described above. Furthermore, whenever the variables y_{ik} are fixed to satisfy these constraints, the constraints (2.9), (2.10), (2.11) and (2.13) define a TSP over the customers assigned to that vehicle. Although the two phase MP-based heuristic does not optimise the VRP based on this formulation, it does optimise the associated GAP, which uses an approximation of the cost of routing each assignment (using the d_{ik}'s), and the associated TSPs over each of the final assignments. In fact, Fisher and Jaikumar's heuristic, which is based on Bender's decomposition [17], operates in exactly this way however they do not run the algorithm to completion, i.e. they provide a heuristic solution to a heuristic interpretation of the VRP[5].

Another MP-based heuristic is the set-partitioning approach that operates by enumerating the cost of a number of candidate vehicle routes, see [9, 87, 2]. Once again, the Set Partitioning Problem (SPP), which involves the partitioning of a set of characteristics (customers) into a number of activities (routes) such that each characteristic is included exactly once among the chosen activities, is a well known COP and exact solutions can be found for problems of large size, see [120]. The related formulation for the VRP is given in Section 2.3.1. Needless to say, if all feasible routes are initially included in this formulation then an optimal solution to the VRP would be found, however the key

[5]Fisher and Jaikumar's algorithm actually solved the VRPTW, not the VRP.

behind a successful heuristic of this type is the efficient selection of candidate solutions due to the excessive memory requirements that occur in the complete case.

Both these heuristic methods outperform the simple heuristics given above. In general, the assignment heuristic provides better quality solutions however the set-partitioning heuristic is more applicable since certain kinds of complex constraints can be added without too much difficulty, i.e. the more constrained a VRP becomes, the smaller the overall number of routes that need to be considered.

Generation 3A: Expert Systems and AI-based Heuristics

In recent years, due to a reticence for applying existing VRP solution methods in practice because of a lack of robustness in dealing with practical constraints and assumptions, an increasing number of studies describe the implementation of interactive methods and expert systems in the arena of vehicle routing. The former involves routing systems that allow the user to input customers into routes or to alter delivery sequences within a heuristic solution, see [171]. The latter involves the tuning and selection of particular heuristics for use in different types of problems, see [62, 130].

AI search techniques involve constructing a neighbourhood of solutions "around" an initial starting solution according to a number of specified rules. Ordinarily, the procedure operates in an iterative fashion with a new solution being created from particular neighbourhood solution by the combination of routes, e.g. by transferring a customer from one route to another or by alternating two customers in one route with another two customers in a different route. Each solution in a neighbourhood is assessed via a cost function which is usually a simplified version of the exact TSP-based cost since to find optimal solutions in a large number of cases would require excessive computational time.

An example of an AI-based heuristic is Simulated Annealing (SA), see [135, 176]. SA is a probabilistic hill-climbing technique that, according to some probability function, allows a "downhill move" (the selection of an inferior solution) during the neighbourhood search process. Typically, the probability that a non-improving solution is accepted reduces as the number of successive changes in the objective function value decreases and as computational time increases. SA algorithms stop when a fixed number of searches have failed to obtain a suitable new incumbent solution. Tabu search is another example of an AI-based heuristic, see [103, 104, 176]. The basis of this approach concerns the use of flexible attribute-based memory structures that are designed to exploit historical search information. Restrictions are placed in the algorithm to prevent cycling and tabu search algorithms stop after a fixed number of iterations. A number of successful applications of this approach to the VRP can be seen in the literature, see [168, 195, 96].

The main advantages of these heuristics over those defined previously is that a range of possible solutions is maintained as the search progresses and it is possible to select non-improving solutions under certain conditions. Additionally, the methods are simple to implement, conceptualise and enhance which is especially important for VRP practitioners.

Generation 3B: Exact algorithms

Following the survey by Laporte and Norbert [141], exact algorithms for the VRP can be classified into three broad areas: Integer Linear Programming (ILP), direct tree search methods and dynamic programming. The number of algorithms contained in each classification is huge and we only provide a selection of such methods in this review.

ILP Methods: The first two ILP methods described here involve approaches that were previously discussed with reference to MP-based heuristics. The set partitioning formulation (2.25)-(2.28) and the generalised assignment formulation (2.6)-(2.13) both represent a MP framework with which to find an optimal solution to the VRP. However, for the set partitioning approach, exact methods are hindered by two problems: (i) there usually exist millions of y_r variables which can cause problems relating to excessive computational storage requirements, and, (ii) huge numbers of optimal TSP solutions, relating to d_r where $Q_r \leq Q$ from (2.24), must be found which can cause problems relating to excessive computational time requirements. A similar difficulty to (ii) arises in the GAP-based approach. Clearly, it is for these reasons that heuristics were originally considered however corresponding optimal algorithms have been utilised. Agarwal [1] and Agarwal *et al* [2] employ a column generation technique that forms a reduced problem containing only a restricted subset of all possible variables (columns). In this case, optimal solutions were found for VRPs of between 15 and 25 customers. In addition, Desrosiers *et al* [64] and Desrochers *et al* [60] obtain solutions of VRPTWs of a slightly larger size using a similar method. The latter are easier to solve using such an approach since the number of feasible routes is lessened.

The first clear success using an ILP method came about following the introduction of the two-index vehicle flow formulation given in Section 2.3.1. Utilising this formulation and employing a general purpose integer programming algorithm that operates according to constraint relaxation, VRPs of medium size were solved to optimality. The algorithm successively solves subproblems containing (2.16), (2.17) and (2.18), introduces any subtour elimination constraints (2.19) if they are violated and introduces the integrality constraints (2.20) and (2.21) if the current solution is non-integer. Computationally, the algorithm works very well for VRPs that are not tightly constrained, due to the presence of fewer subtour elimination/capacity constraints, and has solved such problems for up to fifty vertices [150]. Similar methods have been utilised by Fleischmann [85, 86] and Laporte *et al* [149] to solve different types of VRPs of similar size.

The last ILP method to be considered here concerns a formulation for the TSP that can be obtained by combining (2.16), (2.18) and (2.21) with the following constraint:

$$\sum_{\substack{i,j \in S \\ i \leq j}} x_{ij} \leq |S| - 1. \tag{2.35}$$

This formulation can be strengthened by the addition of additional valid inequalities called comb inequalities [48, 111]. Following work by Laporte and Norbert to generalise these constraints for the VRP, Cornuejois and Harche [51] were able to solve the first tightly constrained, fifty customer vehicle routing problem.

Direct Tree Methods: Direct tree search methods consist of sequentially building vehicle routes by means of a branch and bound tree. An important factor in the efficiency of such an approach is in the quality of the bound at each branching stage. It is possible to branch on arcs or routes and exact VRP algorithms have been developed for both these cases. Further details of this type of approach are given in the early stages of Chapter 3 and so will be omitted here.

Early research work was concerned with branching on arcs. Christofides and Eilon [44] consider the formulation of a bound on the VRP given by the Shortest Spanning Tree Problem (SSTP) and solve very small size problems employing a tree search technique. Laporte *et al* [148] consider the use of a bound based on the Assignment Problem (AP) and use a sophisticated algorithm to solve medium size VRPs. Christofides [42] was the first to consider branching in terms of vehicle routes. In this construction, each tree has at most m levels and branching is completed by considering a list of possible feasible routes after the addition of one customer that has not yet been routed. Needless to say, such a method requires a set of criteria to eliminate potential routes from consideration.

To highlight the construction of a direct tree search method in more detail, the work of Christofides *et al* [46] to derive high quality bounds on the VRP based on shortest path calculations will be considered. The first of these bounds is based on the use of k-degree centre trees. These are trees in which the degree of vertex v_1 is k, i.e. $d(v_1) = k$. Consider a m-TSP defined on a graph G where the edge set E is partitioned into four subsets given as follows.

(i) E_1: edges forming a k-degree centre tree, i.e. a spanning tree over G where $d(v_1) = k$ (where $k = 2m - y$).

(ii) E_2: y edges incident to vertex v_1.

(iii) E_3: $(m - y)$ edges not incident to v_1.

Let c_l be the cost of edge $l \in E$, let E^i represent the set of edges incident to vertex v_i and let $(S, \overline{S})$ be the set of all edges with one vertex in S and the other in $\overline{S}$. Given that x_l^t $\forall$ $t = \{1, 2, 3\}, l \in E$ are binary variables that take the value 1 if and only if edge l is in the optimal solution, then the m-TSP relaxation of the VRP can be represented as follows.

$$\min_x \; z = \sum_{l \in E} c_l(x_l^1 + x_l^2 + x_l^3) \tag{2.36}$$

subject to

$$\sum_{l \in (S,\overline{S})} x_l^1 \geq 1 \qquad (S \subset V;\ |S| \geq 1), \tag{2.37}$$

$$\sum_{l \in E^1} x_l^1 = 2m - y, \tag{2.38}$$

$$\sum_{l \in E} x_l^1 = n - 1, \tag{2.39}$$

$$\sum_{l \in E^1} x_l^2 = y, \tag{2.40}$$

$$\sum_{l \in E \setminus E^1} x_l^3 = m - y, \tag{2.41}$$

$$\sum_{l \in E^i} (x_l^1 + x_l^2 + x_l^3) = 2 \qquad (i = 2, \ldots, n), \tag{2.42}$$

$$x_i^t \in \{0, 1\} \qquad (t = 1, 2, 3;\ l \in E), \tag{2.43}$$

$$y \geq 0 \text{ and integer.} \tag{2.44}$$

Now, for the three distinct problems that arise for x_l^1, x_l^2 and x_l^3, constraints (2.42) can be relaxed in Lagrangean fashion. The objective is therefore rewritten in the following manner where $\alpha(l)$ and $\beta(l)$ are the two terminal vertices of edge l and $\lambda_1 = 0$.

$$V(\lambda, y) = \sum_{l \in E} (c_l + \lambda_{\alpha(l)} + \lambda_{\beta(l)}) - 2\sum_{i=1}^{n} \lambda_i. \tag{2.45}$$

A lower bound on the VRP is then given by:

$$\max_{m_1 \leq y \leq m} \left\{ \max_{\lambda} \sum_t V^t(\lambda, y) \right\}, \tag{2.46}$$

where m_1 is a lower bound on the number of single customer routes in the optimal solution and can be chosen so that:

$$(m - m_1)D \geq \sum_{i=m_1}^{n} q_i. \tag{2.47}$$

The second bound employed by Christofides *et al* is based on q-routes, a route weighting concept first derived by Houck *et al* [121] for the TSP. Let ω be the weight of a route and let $q(l)$ be the value of the l^{th} element of ω. Let $\psi_l(i)$ be the value of the least cost "q-route" that starts and ends at the depot, passes through v_i, has no loop of the form $v_i - v_k - v_i$ and has a total weight of $q(l)$. A lower bound on the VRP is then given by:

$$\sum_{i=1}^{n} \min_{l=1,\ldots,|\omega|} \left(\frac{\psi_l(i) d_i}{q(l)} \right), \tag{2.48}$$

where the $\psi_l(i)$'s can be calculated recursively and the computational effort to find the lower bound is clearly related to $|\omega|$. During tests, the q-route bound was found to be superior to the k-degree tree bound in an arc-based branch and bound however the best results were actually obtained by using the latter in a route-based branch and bound.

This original work has recently been extended by Hadjiconstantinou *et al* [114] by employing lower bounds that are based on *q-paths* and *k-paths*. The use of iterative combinations of such paths results in lower bounds of high quality and a set of corresponding reduction tests reduces the size of the problem even further. Embedding the lower bounds into a tree search procedure, based on a new branching strategy, enables optimal solutions to be found for up to fifty customers. Fisher [81] derive new lower bounds based on k-trees and provide exact solutions of VRPs of similar size in short computational times.

Dynamic Programming: DP methods usually provide lower bounds for direct tree search methods; they are classified separately due to the different nature of the methods involved. The primary problem with such an approach is in the reduction of the huge number of different states that need to be completely enumerated. The basic formulation for a VRP-based DP has been given in Section 2.3.1 and, in this section, one DP method called state-space relaxation will be briefly reviewed. Other methods include the use of feasibility constraints or dominance criteria.

State space relaxation operates with the use of a mapping function and was introduced by Christofides *et al* [47]. First, the original state (k, T) that appears in (2.29) is relaxed to $[k, g(T)]$ where g is a mapping function from the space of all subsets T to a lower dimensional space. Let:

$$g(T) = \sum_{i \in T} q_i \equiv t \; \forall \, T \subseteq X', \text{ and,} \tag{2.49}$$

$$g(T^*) = \sum_{i \in T^*} q_i \equiv t^* \; \forall \, T^* \subseteq X'. \tag{2.50}$$

The relaxed problem can then be represented in the following way, where $\overline{v}(t^*)$ is the minimum cost of a circuit that starts and ends at the depot and has a total load of t^*.

$$f(k, t) = \min_{t^* < t} \left[f(k-1, t-t^*) + \overline{v}(t^*) \right] \tag{2.51}$$

subject to

$$t - (k-1)Q \leq t^* \leq Q, \tag{2.52}$$

$$\tfrac{1}{m-k}[q(X') - t] \leq t^* \leq \tfrac{1}{k} t, \tag{2.53}$$

$$q(X') - (m-k)Q \leq t \leq kQ. \tag{2.54}$$

A lower bound on $\overline{v}(t^*)$ can easily be obtained and successive substitutions of this kind can be incorporated into (2.51) to obtain recursive lower bounds on a VRP given by (2.51)-(2.54). Christofides [43] reported that VRPs of up to fifty customers could be solved employing this DP-approach.

2.4 Basic Stochastic Vehicle Routing Problems

Stochastic vehicle routing problems arise whenever elements of a VRP are random. As a result, there exist many possible problem alternatives otherwise known as SVRP variations. In general, solution concepts are more intricate and solution methodologies are more complex in the stochastic case, even though the latter predominantly originate from the same approaches used in the deterministic VRP. Before discussing such issues and completing a comprehensive review, five basic versions of the stochastic vehicle routing problem are formally defined. In each case however, the objective of the problem is omitted since the issues concerning the meaning of a stochastic optimum in a routing environment are discussed in the following section. At this stage, each objective can be taken to signify the minimisation of expected routing costs.

Stochastic routing problems that are not considered in this thesis, since they are not seen as direct stochastic extensions of the BVRP, include the stochastic inventory routing problem [205], the problem of finding the least expected travel time path between two customers [117], a location problem with stochastic demands [144] and a dynamic stochastic routing problem with random arrival times [26, 28, 27].

2.4.1 The VRP With Stochastic Demands

The vehicle routing problem with stochastic demands was the first stochastic interpretation of the VRP to be studied [201] and remains the most widely researched SVRP. Indeed, the VRPSD is often simply referred to as the SVRP. Specifically, the Basic Vehicle Routing Problem with Stochastic Demands (BVRPSD) retains all the characteristics of the BVRP except that the customer demands, originally corresponding to the deterministic q_i's, are represented by discrete, independent, non-negative random variables, $\xi_i \in \xi$, with finite means, μ_i, and finite variances, σ_i. Each customer $i > 1$ is defined as having δ_i possible demand values or demand *realisations*, denoted by $\varepsilon_i^l \in \xi_i \;\forall\; l \in \{1, \ldots \delta_i\}$. In addition, each set of demand realisations, $\xi_i = \{\varepsilon_i^1, \varepsilon_i^2, \ldots, \varepsilon_i^{\delta_i}\} \;\forall\; i > 1$, is assumed to be ordered in ascending size, i.e. $\varepsilon_i^l < \varepsilon_i^k$, $l < k$ and $\min_l(\varepsilon_i^l) = \varepsilon_i^1, \max_l(\varepsilon_i^l) = \varepsilon_i^{\delta_i} \;\forall\; i > 1$, and has a corresponding probability distribution set, $\varphi_i = \{p_i^1, p_i^2, \ldots, p_i^{\delta_i}\} \;\forall\; i > 1$ where the following sum holds,

$$\sum_{l=1}^{\delta_i} p_i^l = 1 \;\forall\; i > 1. \tag{2.55}$$

The term p_i^l can then be interpreted as the probability that the demand value ε_i^l arises from the set ξ_i. By definition, each customer requires a service and the maximum demand of any customer is always equal to or less than vehicle capacity, i.e. $\varepsilon_i^1 > 0, \varepsilon_i^{\delta_i} \leq Q \;\forall\; i > 1$.

2.4.2 The VRP With Stochastic Customers

The vehicle routing problem with stochastic customers, sometimes referred to as the Probabilistic Vehicle Routing Problem (PVRP), involves customers which have deterministic demands but are present with a given probability. The VRPSC has received almost as much attention in the literature as the VRPSD and is an extension of the TSP with Stochastic Customers (TSPSC), or Probabilistic Travelling Salesman Problem (PTSP), first studied by Jaillet [125]. Specifically, the Basic Vehicle Routing Problem with Stochastic Customers (BVRPSC) retains all the characteristics of the BVRP except that each customer has a probability of being present denoted by $p_i \; \forall \; i > 1$, where $0 < p_i \leq 1$.

2.4.3 The VRP With Stochastic Customers and Demands

The Vehicle Routing Problem with Stochastic Customers and Demands (VRPSCD) is an obvious offshoot of the VRPSC. Specifically, the basic version of this problem, i.e. the BVRPSCD, retains all the characteristics of the BVRPSC except that the customer demands, for those customers that are present, are represented by discrete, independent, non-negative random variables, ξ_i, with finite means, μ_i, and finite variances, σ_i.

2.4.4 The VRP With Stochastic Travel Times

The vehicle routing problem with stochastic travel times, sometimes referred to as the multiple Travelling Salesman Problem with Stochastic Travel Times (m-TSPST)[6], is a direct extension of the TSP with Stochastic Travel Times (TSPST), first studied by Kao [133]. Specifically, the Basic Vehicle Routing Problem with Stochastic Travel Times (BVRPST) retains all the characteristics of the BVRP except that in addition to the costs, c_{ij}, corresponding to the edges in set E, there also exist travel times which are represented by discrete, independent, non-negative random variables, π_{ij}, with finite means, μ_{ij}, and finite variances, σ_{ij}. Each edge is defined as having γ_{ij} possible time values denoted by $\tau_{ij}^{l} \; \forall \; l \in \{1, \ldots \gamma_{ij}\}$ in the set π_{ij}. This set of travel times is assumed to be ordered in ascending size, i.e. $\tau_{ij}^{l} < \tau_{ij}^{k}$, $l < k \; \forall \; i,j$, and has a corresponding probability distribution

[6]It is clearly possible to consider the VRPST with stochastic demands however this problem, which has never been solved, is largely ignored in lieu of the VRPST with stochastic service times, i.e. the VRPSST.

set, $\zeta_{ij} = \{\omega_{ij}^1, \omega_{ij}^2, \ldots, \omega_{ij}^{\gamma_{ij}}\}$ $\forall\, i,j = 1,\ldots,n$, where the following sum holds,

$$\sum_{l=1}^{\gamma_{ij}} \omega_{ij}^l = 1. \tag{2.56}$$

The term ω_{ij}^l can be taken to refer to the probability that the time realisation τ_{ij}^l arises from the set π_{ij} and the capacity Q should be interpreted as an overall time restriction.

2.4.5 The VRP with Stochastic Service Times

The vehicle routing problem with stochastic service times is an offshoot of the VRPSD in which there exist service time demands at each customer point. Specifically, the definition of the basic version of this problem, i.e. the BVRPSST, retains all the characteristics of the BVRPSD except that the customers have service times that are represented by discrete, independent, non-negative random variables, ξ_i, with finite means, μ_i, and finite variances, σ_i. In this case, of course, the capacity Q is a restriction on both travel and service time.

2.5 Characteristics of a Stochastic Vehicle Routing Problem

In a similar fashion to the generic SVRP, the VRPSD, VRPST, VRPSC, VRPSST and VRPSCD are all collections of sub-problems that extend from an original interpretation. The reason behind this extra intrinsic ambiguity lies in the inherent structural subjectivity involved with any stochastic problem. In this section therefore, all the important general issues surrounding stochastic vehicle routing are clarified using the VRPSD as an illustration. This complete dissemination of ideas will lead to a substantive understanding of the structure of the problem that will create an unambiguous basis from which to define a comprehensive series of practical problem extensions of the BVRPSD and to outline clear objectives of the other SVRPs given in the last section.

There are two key issues that effect the construction of a clear SVRP definition. The first concerns the appropriate form an associated solution should take and how the objective of the problem is to be perceived. The second concerns the effect that information disclosure has on the associated problem environment and how the timing of variable realisation relates to the optimisation process. The former is discussed once the role of problem scenarios and solution viability in a stochastic vehicle routing environment have been highlighted. Following this, the concept of information-disclosure and its influence on the practical uses of SVRP modelling in different planning systems will be considered.

2.5.1 Problem Scenarios and Solution Viability

In the BVRP, where all the demands are fixed in advance, provided that there are enough vehicles to meet customer demand, a solution can always be found such that vehicle capacities are not exceeded, each vehicle starts and ends at the depot and each customer is visited exactly once. The key to understanding stochastic vehicle routing lies in the acceptance that, unlike its deterministic equivalent, its solution is not practically *viable* (implementable) at all times. To clarify this point, a number of definitions are required.

Definition 2.5.1 (Problem Scenario) *Given that x_r, $r \in \{1,\ldots,\hat{r}\}$, corresponds to the discrete random variables involved in any stochastic vehicle routing problem, let a problem scenario, ϕ_s, $s \in \{1,\ldots,\hat{s}\}$, be a complete set of any single realisations of these random variables, i.e. $|\phi_s| = \hat{r}$.*

Definition 2.5.2 (Realisation Index) *Let $\iota(r,s)$ denote the realisation index of the discrete random variable, x_r, $r \in \{1,\ldots,\hat{r}\}$, in a given problem scenario, ϕ_s, $s \in \{1,\ldots,\hat{s}\}$.*

A full set of realisations of the random variables involved in the VRPSD would, therefore, correspond to $\phi_s = \{\varepsilon_2^{\iota(2,s)}, \varepsilon_3^{\iota(3,s)}, \ldots, \varepsilon_n^{\iota(n,s)}\}$, where $\iota(i,s)$ corresponds to the index of the realised demand value of customer i in the problem scenario indexed by s, i.e. the demand of customer i in scenario s is $\varepsilon_i^{\iota(i,s)}$. Of course, each problem scenario has an associated probability of occurrence, P_s. For any VRPSD scenario indexed by s, this would correspond to the following:

$$P_s = \prod_{i=2}^{n} (p_i^{\iota(i,s)}), \tag{2.57}$$

where:

$$\sum_{s=1}^{\hat{s}} P_s = 1. \tag{2.58}$$

Now, consider a simple VRP ($n = 8$, $m = 2$, $Q = 24$, $q_i = \{14, 3, 10, 10, 4, 2, 4\}, i = 2,\ldots,8$) where the optimal solution involves two routes, shown in Figure 2.2, given by $v_1 - v_5 - v_2 - v_1$ and $v_1 - v_3 - v_6 - v_7 - v_8 - v_4 - v_1$. The cost matrix can be ignored for the

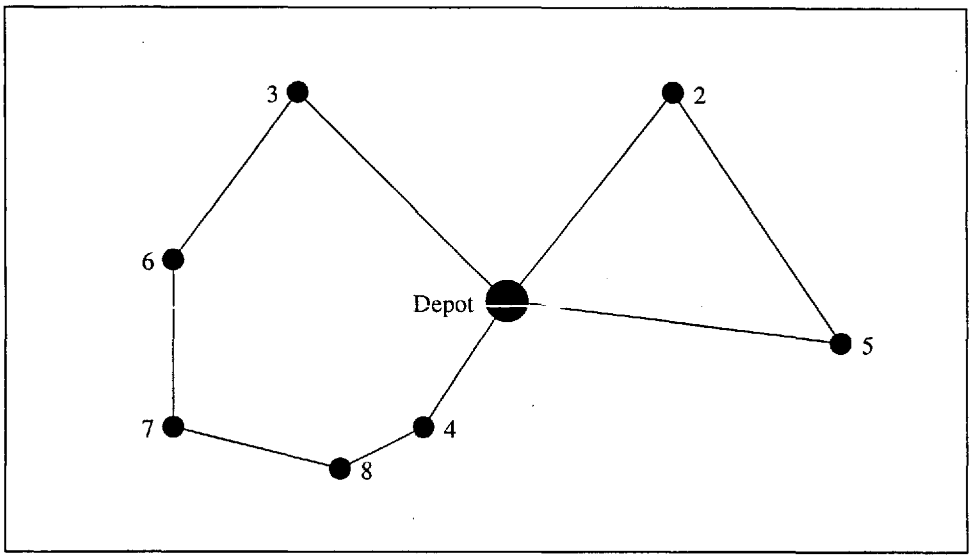

Figure 2.2: A VRP Solution

requirements of this discussion. If the VRP is transformed into an elementary VRPSD by converting the demand at customer $i = 4$ to a discrete uniform stochastic demand varying from 10 to 12, i.e. $\xi_4 = \{10, 11, 12\}$ with $\varphi_4 = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$, how is the existing optimal VRP solution effected? There are three different problem scenarios that occur, namely $\phi_1 = \{14, 3, 10, 10, 4, 2, 4\}$, $\phi_2 = \{14, 3, 11, 10, 4, 2, 4\}$ and $\phi_3 = \{14, 3, 12, 10, 4, 2, 4\}$ with $P_s = \frac{1}{3}\ \forall\ s \in \{1, 2, 3\}$. In the first two cases the original VRP solution remains practically viable, however if the latter arises then the demand of one customer is not satisfied (customer $i = 3$ or $i = 4$ depending on the direction of travel). In fact the whole problem becomes practically non-implementable since the total demand required (49) is greater than the maximum capacity of the two vehicles (48), i.e. $\left(\varepsilon_2^{\iota(2,3)} + \varepsilon_3^{\iota(3,3)} + \ldots + \varepsilon_8^{\iota(8,3)}\right) > 2Q$.

Clearly, a VRPSD solution cannot simply represent the solution to a corresponding VRP where the demands relate to the average demands of the VRPSD. Furthermore, it is evident that the only way to achieve viability over all problem scenarios is to assume each customer demand is at its maximum realisation, i.e. $\varepsilon_i^{\delta_i}\ \forall\ i > 1$, then solve the VRPSD as a deterministic VRP (this would require at least three vehicles in the example). Unfortunately however, due to issues of impracticality and expense, this trouble free, completely viable routing method would not seriously be considered except in the most extreme of cases when service deficiencies are never allowed to occur. Indeed, the impracticality of these types of solutions and the need to incorporate unviability into the evaluation of a route is the very reason why stochastic-type solutions are sought at all.

2.5.2 A Priori Optimisation and Planning

The motivation for studying a stochastic system is to find a solution which copes best with uncertainty, and subsequent solution unviability, without actually solving each problem scenario as it arises. In this way, SVRPs belong to a set of a priori optimisation problems defined as follows.

Definition 2.5.3 A Priori Optimisation *A priori optimisation is the act of optimising over all possible problem scenarios, prior to full information and the realisation of any single scenario.*

Following this definition, consider the role of a priori SVRP modelling both in a long term planning context, i.e. when a single solution is required over time, and a short term planning context, i.e. when many solutions are required over time. If there exists any uncertainty within either planning system then optimisation occurs a priori and a SVRP model should be used. (The only exception to this occurs when customer service is vital and no unviable solutions are allowed, e.g. medical services.) Moreover, depending on certain attributes of the system involved, even if the system is deterministic then SVRP modelling may also be of use. These practical attributes are discussed below.

- Long term planning: Given that relevant data is available and of adequate quality, a VRP model can be used. If not, the fixed system is in essence stochastic and a SVRP model should be utilised.
- Short term planning: Given that data is available and of suitable quality, the computational power continually required to evaluate and model such data is present, the associated labour assignments are of a flexible nature and a fixed system is not desired, then a VRP model would be used as this would always guarantee an optimal solution. If either of these five conditions does not apply then a SVRP model should be used. (The last two conditions are satisfied in the case of a priori optimisation since by definition a fixed, single solution is always obtained.)

2.5.3 Optimality, Route Failure and the Cost of Recourse

The use of SVRP, a priori optimisation lies in the role of planning so that, over a period of time, the most cost effective solution can be obtained. However, what conditions does the search for such a cost effective solution enforce on the structure of a SVRP objective? What are the consequences to such an objective on the inability to maintain practical viability? Indeed, what do we mean by an "optimal" solution to such a stochastic planning problem?

Consider the VRPSD once more. Its objective is highly subjective since its optimal solution exists somewhere between total serviceability/maximum transportation costs (i.e. a VRP-based solution where the customer demands are set at their maximum values) and zero serviceability/minimum transportation costs (i.e. a VRP-based solution where the customer demands are set at their minimum values). If total service is not to be guaranteed explicitly, there must be some element of trade-off between cutting cost and supplying an allowable level of customer service. A VRPSD solution addresses this aspect of balance with the use of two possible solution outlooks, both related to the concept of *route failure.* Forming the basis of all work concerning the SVRP, route failure can be interpreted as the action that occurs if the random variables involved in a stochastic routing problem exceed a certain level (previously referred to as the "unviability of a solution" or the "inability to serve a customer"). In the case of the VRPSD, a route failure occurs when a vehicle cannot complete all the deliveries in a designed route because its supply has been exhausted at some point along that route. Since each customer will not necessarily be serviced in practice by the designed routes alone, SVRP solution outlooks differ due to how route failure is incorporated into their objectives and how the trade off, mentioned earlier, is addressed. The first solution outlook regards serviceability. A VRPSD objective could be to fully service customers a certain percentage of the time, i.e. transportation costs can be reduced as much as possible given that a certain service level is maintained. In the example given in Section 2.5.1, an average demand (VRP-based) solution provides a complete service 66% of the time (route failure occurs 33% of the time) and so this would be an adequate solution if the required serviceability level was 66%. The second outlook involves the addition of a *recourse cost* (R), sometimes referred to as the penalty cost, to the overall cost of the system that is contemporaneous with a route failure. The importance of customer service differs for different problem situations and the cost of route failure differs depending on where, when and how the route fails. Therefore, incorporating these costs into a SVRP, by for example including the extra "transportation" costs necessary to re-serve the customer in question, the subjectivity of its objective can be reduced. Possible recourse cost factors include the location of the route failure, translating into the cost of

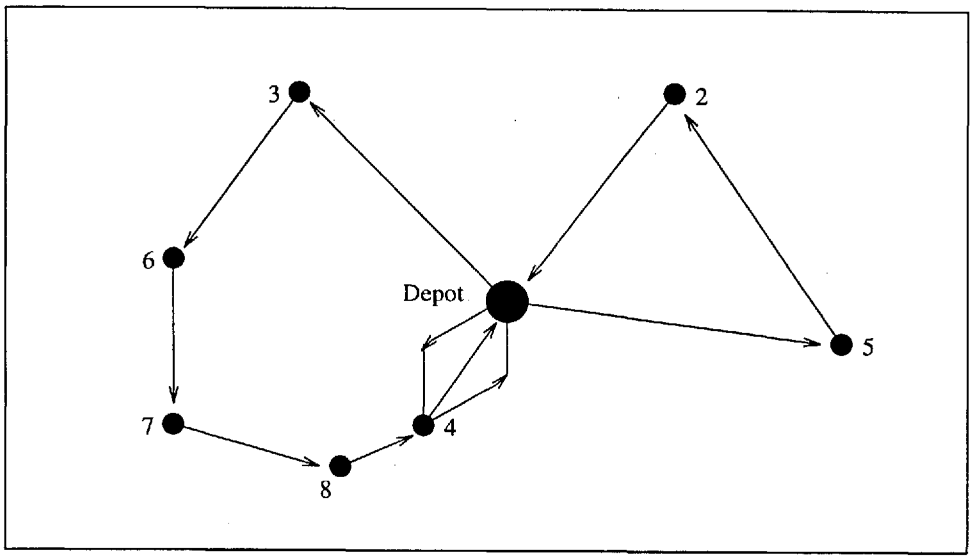

Figure 2.3: The Cost of Recourse in a VRPSD

a vehicle returning to the depot from that specific location and continuing on its journey, individual customer dissatisfaction costs (the supplier may regard particular customers as more important than others) and recourse costs relating to the extent of failure.

A transportation-based recourse situation is shown graphically in Figure 2.3 for the previous example where a route failure occurs at customer $i = 4$. The cost of recourse in this situation is given as follows:

$$R = 2c_{1i} \tag{2.59}$$

where i is the index of the customer at which there exists a route failure, i.e. $2c_{14}$.

2.5.4 Information Disclosure

The final subjective issue concerning stochastic routing involves the timing of *information disclosure* and how different temporal definitions can effect a SVRP solution. In the VRPSD, the timing of customer demand information disclosure, in relation to when the fixed/a priori routes are determined, is integral to the effectiveness of the solution itself. Random demands, which by definition arise out of given distributions, can be interpreted as withheld pieces of information that are disclosed at particular time slots in a system. For example, if the time slot occurs before routing, even though we initially had random demands, we have time to route the system as a VRP. At the other extreme, individual demands may not be known until exactly the moment a vehicle arrives at the given customer point, after routing has occurred. In between these two cases, there are a myriad of other possibilities. Indeed, on a wider scale, information disclosure essentially traverses the spectrum between the deterministic case, through the stochastic case, to the completely dynamic case. In the latter, information is consistently being disclosed and action is continuously being taken, the problem redefined and the solution redesigned.

One possible classification of information disclosure for the VRPSD was introduced by Laporte and Louveaux [141, 142] and involves the recognition of late and early information. In this discussion, these initial definitions are augmented with knowledge-based factors and are generalised for use over all SVRPs.

- Late information disclosure involves the realisation of discrete information after the vehicles have been routed and at the last possible moment once the routes are put into practice, e.g. for the VRPSD, demand values are only disclosed when a vehicle arrives at a customer point. Clearly, this disclosure specification is still fairly fallacious since routing could have originally occurred with the use of some form of system knowledge. For example, do we know the customer demand distributions, their mean and variance and, if so, can we estimate the maximum, minimum, or average demand possible for a typical customer before the routes are devised? Such system knowledge could lead to a form of decision making taking place whilst the vehicles are on their respective routes but it would not require any on-line information transfer since only existing knowledge is in use. Two separate cases of SVRP late information disclosure are defined: (i) *late information with no knowledge*, in which no knowledge is retained concerning the nature of the random variables involved, and (ii) *late information with existing knowledge*, in which some form of general knowledge about the system is put to use once the routes have been devised.

- Early information disclosure involves the realisation of discrete information at some point during the time frame between when the vehicles are routed and the moment before the variables are required to be realised, i.e. for the VRPSD, demand values are realised sometime before vehicles arrive at a relevant customer point. Early information covers a whole range of practical possibilities including an initial "total knowledge" state that involves the realisation of all the associated random variables before putting the previously determined routes into practice. Once again, two separate knowledge-based cases are defined: (i) *early information with dynamic*

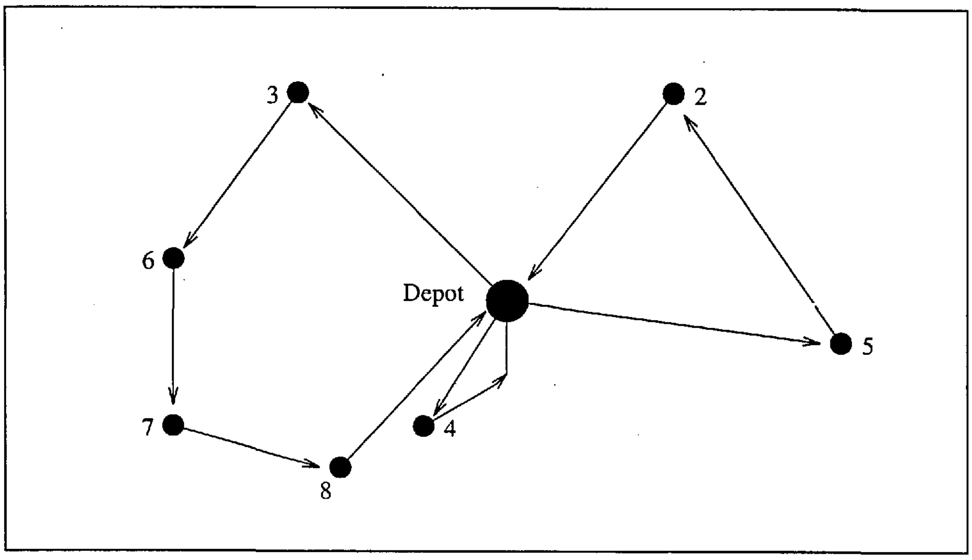

Figure 2.4: A Route Break in a VRPSD

knowledge, in which random variable information is realised while the vehicles are on their specified routes, and (ii) *early information with total knowledge*, in which random variable information is realised before vehicles set out on their specified routes. In practice, unlike an early information with dynamic knowledge system, the latter does not require an on-line procedural set-up since pre-emptive action can be taken at a secondary planning stage. In addition, note that early information with dynamic knowledge is a multi-stage definition in that it is not specified when information is made available while the vehicles are on their routes. Throughout this thesis, this definition is therefore taken to denote the decision stage before the decision stage involving the realisation of the random variables. In the case of the VRPSD, this translates into realising a customer demand one customer before arriving at the customer in question.

2.5.5 Decision Points and Route Breaks

The disclosure of information can allow a routing decision to be made while a driver is on a previously determined route. Customers can form a series of *decision points* where, by making some form of decision, an attempt can be made to reduce the effect of route failure. The latter is achieved by disconnecting a previously defined route and forming what is known as a *route break*. Intuitively, for the VRPSD, these breaks will always involve returning to the depot to refill before continuing on a route. In a system of late information with no knowledge, a vehicle can only return to the depot and refill from and to the point of route failure. This has been shown previously in Figure 2.3 for the previous example where a route failure occurs at customer $i = 4$. As shown in Figure 2.4, with the inclusion of a decision point (either by late information with existing knowledge or either of the early information systems), a more cost-effective situation can arise. A decision concerning the requirements of customer $i = 4$ can be made at customer $i = 8$ which allows a route break to be implemented and a saving, compared to the previous cost of recourse, which is given as follows,

$$2c_{14} - (c_{14} + c_{18} - c_{48}) = c_{14} + c_{48} - c_{18}. \tag{2.60}$$

This comparative saving, i.e. $c_{1j} + c_{ij} - c_{1i}$ where a failure occurs at j and j is preceded by i in the vehicle route, is always greater than zero due to the triangular inequality. In general, the cost of recourse in this situation is given as follows:

$$R = c_{1j} - c_{ij} + c_{1i}. \tag{2.61}$$

Needless to say, when information disclosure is considered and decision points are in place, the possibilities of enhanced recourse through predicting route failure and implementing route breaks are practically endless.

2.6 A General Framework of BVRPSD Extensions

A standard group of extensions, with clear and widely applicable objectives, need to be defined for any basic SVRP variation so that they can provide the unambiguous basis for associated research. As shown in Section 2.5, stochastic vehicle routing is linked to its role in practice in such a distinct way that this linkage cannot be ignored in related theoretical work. This section introduces a standard and definitive set of BVRPSD extensions. Similar classifications can easily be used for the analysis of other SVRPs given in Section 2.4.

2.6.1 Introduction

The fundamental structural characteristics of a SVRP that require distinct classification fall into two separate categories. The first, arising out of the discussion concerning serviceability and the cost of recourse given in Sections 2.5.1 to 2.5.3, involve the objectives of the problem and will usually depend on the role of management decision making and strategic planning. The second, arising out of the discussion concerning information disclosure, associated knowledge and route breaks given in Sections 2.5.4 to 2.5.5, can be directly linked to the system in which the routing operation is to function since it will largely depend on the existing procedural environment.

It is important to note that a SVRP solution does not arise out of tinkering with a corrective recourse action or with an alteration in the timing of information disclosure. The key is to extend any basic interpretation of the SVRP so that the subjectivity required in any particular practical example does not cloud the solution procedures and technical aspects of the general problem. Indeed, the invention of new and wonderful system requirements is like changing the problem to reach the desired answer; one does not win a sporting contest by changing the opponents.

2.6.2 BVRPSD Extensions

The standard extensions of the BVRPSD are shown in Table 2.2. In each case, the optimal solution involves finding a minimum cost set of routes, including any associated recourse costs, given that there exists a fixed system of information disclosure and fixed objective-related criteria. The first extension is the most straightforward. P1 is the problem of finding a set of routes which meet a limited service requirement regardless of the particular effects of any individual route failures. There exists a probability of a group of solution routes being unviable and this equates to a serviceability percentage. The optimal solution is a set of routes that provides this percentage of expected service at least cost.

Problems P2, P3 and P4 involve clearer definitions of the BVRPSD. If a route fails, a penalty cost is added to the overall routing cost. In P3 this cost is taken as a fixed number regardless of where or how the relevant route failure arises. In P4 this cost is enhanced by taking into account various cost elements involved in the theoretical re-serving of the customer. P3 would serve well in practice if knowledge of the cost of route failure, and the system in general, was very limited. P4 could be used when more about location and recourse possibilities are known. P2 corresponds to a combination of P1 and P4. In contrast to P1 that minimises cost while controlling the probability of route failure, P2 minimises *expected* cost while controlling the probability of route failure.

	BVRPSD Extension	Information Disclosure	Knowledge	Decision Points	Additional Costs per Route Failure
P1	Fixed serviceability	Late	None	None	None*
P2	Fixed Serviceability with recourse	Late	None	None	Variable*
P3	Fixed costs of recourse	Late	None	None	Fixed
P4	Variable costs of recourse	Late	None	None	Variable
P5	Existing knowledge	Late	Minimum demand of next customer	Yes	Variable
P6	Dynamic knowledge	Early	Exact demand of next customer	Yes	Variable
P7	Total knowledge	Early	Exact demand of all customers	Yes	Variable

(*In this case there exists a constraint on the percentage chance of route failure.)

Table 2.2: The Standard Extensions of the BVRPSD

The "VRPSD with Knowledge" problems, P5, P6 and P7, differ from the recourse cost problems by the inclusion of specifications relating to system design and the presence of information disclosure. Decision points arise in these problems and so their analysis should be applied towards slightly more dynamic situations where decisions can be made after the vehicles have left on their respective routes. In the first case, P5, there exists no on-line system as only existing knowledge is required; this is further specified as a vehicle returning to the depot if its load is below the minimum possible demand of the following customer. In this case, a route failure is predicted without any information being received whilst the vehicles are on their specific routes. P6 involves a similar situation to P5 but relies on a dynamic on-line system being in place. There must exist an early information system where decision points are present and the exact demand of the next customer on the route is known. Notice that P5 and P6 offer only two examples stemming from the wide class of possible information-disclosure interpretations. P7 involves the "final" information disclosure example and the most interesting case where demands are known after routing however before the drivers set out on their specified routes. This problem effectively develops the concept of variable recourse since drivers can "choose" where to fail along a route.

	BVRPSD Extension	Transportation Costs per Route Failure
P2	Fixed serviceability with recourse	c_{1j}
P4	Variable costs of recourse	c_{1j}
P5	Existing knowledge	c_{1j} or $c_{1j} - c_{ij} + c_{1i}$
P6	Dynamic knowledge	$c_{1j} - c_{ij} + c_{1i}$
P7	Total knowledge	$c_{1l} - c_{kl} + c_{1k}$

(Note: A route failure occurs at customer j, j is preceded by i on the given route and l is any customer before j on the route where l is preceded by k.)

Table 2.3: Costs per Route Failure of the BVRPSD Extensions

2.6.3 Route Failure Costs

In the above extensions, except P1 and P3, costs per route failure have been loosely defined as "variable". As stated in Section 2.5.3, possible route failure costs come in a variety of forms, e.g. costs per extent of failure and individual customer dissatisfaction costs. In the remainder of the thesis, these costs will always translate into the transportation costs related to the theoretical reserving of the customer involved. Provided as examples in Section 2.5, these costs are more formally defined in Table 2.3. The following two points should be noted, (i) two possible costs are present in the existing knowledge case since a vehicle may or may not pre-empt a route failure, and, (ii) transportation costs per route failure are complicated in the case of the total knowledge case since a route failure can be pre-empted at any point in the route.

2.6.4 Summary

Seven standard VRPSDs have been constructed that differ in their respective objective, handling of recourse or information-disclosure specifications. These extensions are considered in greater depth in Chapter 5. The adaptation of such a classification process for the other SVRPs defined earlier should be fairly intuitive. For the remainder of this thesis, particular extensions of alternative SVRPs will be constructed whenever necessary.

2.7 SVRP Formulations and Solution Methods

As in Section 2.3.2 for the VRP, in this section a variety of SVRP formulations are presented before introducing existing solution methods and completing a comprehensive literature review. The general methodologies used to formulate SVRPs fall into two separate categories: Stochastic Programming (SP) and Markov decision processes. Stochastic programming is by far the most important SVRP-related research field, especially with regard to this thesis, and remains the most widespread and well-researched methodology used to consider such problems. For this reason, SP will initially be briefly summarised below. For a more in depth review see [57, 131, 210, 132, 31].

2.7.1 Stochastic Programming

In their most basic form, mathematical programming problems are of the following type:

$$\left.\begin{array}{lll} \text{Minimise} & f(x) & \\ \text{subject to} & g_i(x) \leq 0 & \forall\, i = 1, \ldots, r \\ & x \in S \subseteq I\!R^s & \end{array}\right\} \qquad (2.62)$$

where the real functions f and g are assumed to be deterministic and known, i.e. they are given by the modelling process. Including random parameters in (2.62) leads to the following stochastic program:

$$\left.\begin{array}{lll} \text{`Minimise'} & f(x, \xi) & \\ \text{subject to} & g_i(x, \xi) \leq 0 & \forall\, i = 1, \ldots, r \\ & x \in S \subseteq I\!R^s & \end{array}\right\} \qquad (2.63)$$

where ξ is a vector of random variables varying over a set $\Xi \subset I\!R^k$. More specifically, consider a family of complete events F that are subsets of Ξ where the probability distribution P on F is given, i.e. there exists a distinct chance of a particular event occurring. This arrangement forms what is known as an induced probability space (Ω, F, P) where there exists some space Ω of outcomes, a collection of subsets F of these outcomes called events and a probability measure P assigning each $f \subset F$ with the probability with which it occurs. Given certain measurability conditions on the set of outcomes, this reduces to an induced probability space (Ξ, F, P) where ξ represents a random vector and ε corresponds to one possible realisation of ξ.

The practical consequence of this stochastic construction is that the objective function and constraints of (2.63) become random variables themselves. The requirement of choosing $x \in S$ is not clearly specified and this makes the definition of such a problem ambiguous in nature. Kall [131] mentions the idea of a *solution concept* which should be appropriate for each particular instance of the problem in question. Indeed, the majority of Section 2.5 involved the nature of problem subjectivity and the need to define this solution concept for the SVRP and, more specifically, for the VRPSD; an element of each standard BVRPSD extension is an appropriately defined solution concept. It has been shown that the practical ambiguity involved in modelling stochastic problems can be reduced by considering serviceability or by the introduction of a recourse cost. Similarly, in stochastic programming, we consider either *chance constrained (probabilistic) programming* or *stochastic programming with recourse.*

For purposes of simplicity, in the following discussion we will refer to the following linear program.

$$\left.\begin{array}{lll} \text{Minimise} & z = \sum_j c_j x_j & \\ \text{subject to} & \sum_j a_{ij} x_j \le b_i & \forall\, i = 1, \dots, m \\ & x_j \ge 0 & \forall\, j = 1, \dots, n \end{array}\right\} \qquad (2.64)$$

A stochastic program is formed if either of the objective function coefficients c_j, the constraint coefficients a_{ij} or the right-hand side values b_i in (2.64) are converted into random variables. [Note that if the only stochastic component corresponds to the c_j's then it is possible to consider the minimisation of the expected objective function value since the solution concept remains clearly defined. It is only when either the a_{ij}'s or the b_i's are randomised that the latter has no explicit meaning and one of the two stochastic programming variants must be employed.]

Chance-constrained programming

Chance-constrained stochastic programming, first considered by Charnes and Cooper [41] in 1959, involves replacing the constraints in (2.64) with a series of probabilistic constraints. Consider the probability that constraint i is satisfied, i.e.

$$P\left[\sum_j a_{ij} x_j \le b_i\right]. \qquad (2.65)$$

Now, if α_i denotes the maximum allowable probability that constraint i is violated, a chance constrained programming formulation of (2.64) can be formed by substituting the original constraints with the following chance constraints.

$$P\left[\sum_j a_{ij}x_j \leq b_i\right] \geq 1-\alpha_i \; \forall \, i=1,\ldots,m. \tag{2.66}$$

Given that the c_j's are known, this formulation minimises the objective function where the constraints are not allowed to exceed certain threshold/violation values, α_i. Needless to say, it is possible to consider a formulation where all constraints are bounded by the same probability, i.e.

$$P\left[\sum_j a_{ij}x_j \leq b_i, \; i=1,\ldots,m\right] \geq 1-\alpha. \tag{2.67}$$

Many authors, including Charnes and Cooper, show how the probabilistic constraints given above can be transformed into equivalent deterministic constraints for stochastic variables of a given type, see Section 2.7.2. These constraints may or may not be linear however existing mathematical programming techniques can still be applied and optimal solutions can, therefore, be obtained.

Stochastic programming with recourse

With the use of Lagrangean multipliers, it is possible to move the constraints (2.66) into the associated chance-constrained objective function. In this manner, a number of alternative formulations, known as stochastic programs with recourse, can be obtained.

Denoting λ_i as the fixed penalty cost incurred by bringing a constraint i into feasibility, a stochastic program with fixed recourse can be presented as follows.

$$\begin{array}{ll} \text{Minimise} & E[z] = \sum_j c_j x_j + \sum_i \lambda_i P\left[\sum_j a_{ij}x_j > b_i\right], \\ \text{subject to} & x_j \geq 0 \; \forall \, j=1,\ldots,n. \end{array} \tag{2.68}$$

Likewise, it is possible to consider a variable recourse interpretation of the above that incurs a penalty cost depending on how badly a particular constraint is violated. If λ_i now denotes the cost of bringing constraint i one unit closer to feasibility, a stochastic program with variable recourse can be given as follows.

$$\begin{array}{ll} \text{Minimise} & E[z] = \sum_j c_j x_j + \sum_i \lambda_i \sum_{l>0} l.P\left[\sum_j a_{ij}x_j - b_i = l\right], \\ \text{subject to} & x_j \geq 0 \; \forall \; j = 1, \ldots, n. \end{array} \tag{2.69}$$

A General Recourse Model

Stochastic programs with recourse belong to a set of *two-stage stochastic programs* that contain two component parts in their objective functions. In the first stage, before realising the random variables involved, a decision on x is taken. In the second stage, having realised the associated random variables, a second "decision" is taken to correct any infeasibilities that have arisen because of x. (A-priori optimisation is a form of two-stage stochastic programming.) Generalising this process further, it is possible to consider *multi-stage stochastic programs* that involve a successive number of decision points and a corresponding number of successive random variable realisations. It is important at this stage to distinguish between a regular multi-stage stochastic problem and a special case of the problem that can still be modelled as a two-stage stochastic program. Such special cases, known as having a *block-separable* property, refer to multi-stage stochastic problems in which corresponding decisions can be made at the outset and the future only involves specific reactions to these outcomes.

We now describe a general two-stage stochastic program with recourse that is based on the original stochastic program given in (2.63). Consider the i^{th} constraint of (2.63). This constraint is only violated if the function given by:

$$g_i^+(x,\varepsilon) = \begin{cases} 0 & \text{if } g_i(x,\varepsilon) \leq 0, \\ g_i(x,\varepsilon) & \text{otherwise.} \end{cases} \tag{2.70}$$

is greater than zero for a given decision x and a given realisation ε. To compensate for constraint violation a recourse or second stage activity $y_i(\varepsilon)$ can be chosen so that it satisfies:

$$g_i(x,\varepsilon) - y_i(\varepsilon) \leq 0 \tag{2.71}$$

If this extra effort is assumed to cause an extra cost or penalty of q_i per unit, total additional costs would amount to:

$$Q(x,\varepsilon) = \min_y \left\{ \sum_{i=1}^{r} q_i y_i \mid y_i(\varepsilon) \geq g_i^+(x,\varepsilon),\ i = 1,\ldots,r \right\} \tag{2.72}$$

So, if $g_0(x,\varepsilon)$ corresponds to a solution at the first stage, total cost becomes:

$$f_0(x,\varepsilon) = g_0(x,\varepsilon) + Q(x,\varepsilon) \tag{2.73}$$

The general stochastic program shown in (2.63) can then be reduced to the following two-stage stochastic program with recourse:

$$\min_{x\in X} E_\xi f_0(x,\xi) = \min_{x\in X} E_\xi(g_0(x,\xi) + Q(x,\xi)). \tag{2.74}$$

Clearly, given a deterministic first stage, this can be reduced to:

$$\min_x \ [g_0(x) + \mathcal{Q}(x)], \tag{2.75}$$

where $\mathcal{Q}(x) = E_\xi(Q(x,\xi))$ corresponds to second stage solution values, "$\min g_0(x)$" is the objective function of the first stage problem and "$\min E_\xi(Q(x,\xi))$" is the objective function of the second stage problem.

Summary

The majority of work on stochastic programming involves the solution of two-stage recourse problems that have an adequate and computationally tractable representation of their second stage. As we shall see in the following section, such a representation is not always possible to find. The remainder of research on stochastic programming involves either approximation techniques (that approximate the second stage recourse function), see [32], or Monte Carlo sampling methods (that use statistical estimates to obtain confidence intervals on associated results), e.g. the quasi-gradient [80] method and the decomposition method [119]. In this thesis, the latter statistical methods are not considered.

2.7.2 SVRP Formulations

We now consider three different types of formulations that can be used to model SVRP:

- Chance-constrained stochastic programs.
- Stochastic programs with recourse.
- Markov decision process models.

For purposes of clarity and without loss of generality at this stage, we use a selection of existing VRPSD formulations to provide an illustration of the more general SVRP formulaic issues. This treatment of the literature is reasonable because of a number of reasons. Firstly, formulations for particular SVRP interpretations can in general be interchanged with the minimum of difficulty. This is due primarily to the use of general SP-based formulations in virtually all types of SVRP modelling. Secondly, and most importantly, SVRP formulations are rarely used explicitly in any associated solution method and, therefore, from the outset they are often left in their most ambiguous and transferable form. This point is made clear in the following discussion. Thirdly, we present a representative number of each formulation type (two chance-constrained formulations, three recourse-based formulations and one Markov decision process model) and this range of formulaic structure is unavailable for any SVRP other than the VRPSD.

A Three-Index Chance-Constrained Formulation

The first formulation for the VRPSD, relating to the BVRPSD extension P1, was presented by Stewart [192] and is a loosely defined combination of the most basic three-index VRP formulation given by (2.2)-(2.4) and the chance-constrained stochastic programming approach described above. If, as before, S_m corresponds to the set of feasible solutions to the m-TSP and the random variables ξ_i correspond to the customer demands, the proposed VRPSD formulation is given as follows:

$$\min_x \sum_{i,j} \sum_k c_{ij} x_{ijk} \tag{2.76}$$

subject to

$$P\left[\sum_{ij} \xi_i x_{ijk} \leq Q\right] \geq 1-\alpha, \ (k=1,\ldots,m), \tag{2.77}$$

$$x = [x_{ijk}] \in S_m. \tag{2.78}$$

The constraints (2.77) imply that the maximum allowable probability that a vehicle on any one route cannot serve all customers to which it has been assigned is equal to α.

As stated earlier, these constraints can be transformed into deterministic equivalents in a straightforward manner. Given that the means and standard deviations of the ξ_i's are given by μ_i's and σ_i's respectively, a vehicle route k has a mean demand M_k and a standard deviation of demand D_k, where M_k and D_k are given as follows:

$$M_k = \sum_{ij} \mu_i x_{ijk}, \tag{2.79}$$

$$D_k = \sqrt{\sum_{ij} \sigma_i^2 x_{ijk}^2}. \tag{2.80}$$

Clearly, the constraints (2.77) can be replaced by the deterministic constraints:

$$M_k + D_k T \leq Q \;\; \forall \;\; k = 1, \ldots, m \tag{2.81}$$

given that there exists a constant T such that:

$$P\left[\left(\sum_{ij} \xi_i x_{ijk} - M_k\right) / D_k \leq T\right] = 1 - \alpha \;\; \forall \;\; k = 1, \ldots, m. \tag{2.82}$$

A Two-Index Chance-Constrained Formulation

Laporte *et al* [145] present a similar two-index chance-contrained formulation that incorporates greater simplicity but does not for allow hererogeneous vehicle capacities. Let $V_\alpha(S)$ correspond to the minimum number of vehicles required to serve the customers in a customer set S such that the probability of route failure in serving S is less than or equal to α, i.e. $V_\alpha(S)$ satisfies the following:

$$P\left[\sum_{i \in S} \xi_i > QV_\alpha(S)\right] \leq \alpha. \tag{2.83}$$

P1 can then be represented as follows:

$$\min_x \sum_{i,j} c_{ij} x_{ij} \tag{2.84}$$

subject to

$$\sum_{\substack{i \in S \\ j \notin S}} x_{ij} \geq V_\alpha(S) \qquad (S \subseteq \{2, \ldots, n\}, |S| \geq 2) \tag{2.85}$$

$$x = [x_{ij}] \in S_m. \tag{2.86}$$

Clearly, the constraints given by (2.85) constitute a simple extension to the deterministic connectivity constraints given by (2.22), see [150].

A Three-Index Fixed Recourse Formulation

The first penalty function-based formulation for the VRPSD, relating to BVRPSD extension P3, was presented by Stewart [192] and combines the basic VRP formulation given by (2.2)-(2.4) and the fixed recourse formulation given above. Given that λ_k corresponds to a fixed penalty incurred every time the vehicle route k fails, then the VRPSD can be given as follows.

$$\min_x \sum_{i,j} \sum_k c_{ij} x_{ijk} + \sum_k \lambda_k P \left[\sum_{ij} \xi_i x_{ijk} > Q \right], \tag{2.87}$$

$$\text{subject to} \quad x \in S_m. \tag{2.88}$$

A Three-Index Variable Recourse Formulation

Likewise, a simple stochastic program with variable recourse can be adapted for the VRPSD. Stewart [192] consider a version of the BVRPSD extension P4 where variable recourse is taken to correspond to the extent of failure, not the location of failure. Given that λ_k corresponds to a penalty per unit of demand in excess of Q on a vehicle route k and $E[l_k]$ is the expected number of units by which demand will exceed Q on a vehicle route k where $E[l_k]$ is given as follows:

$$E[l_k] = \sum_{l_k > 0} l_k . P \left[\sum_{ij} \xi_i x_{ijk} - Q = l_k \right], \; k = 1, \ldots, m, \tag{2.89}$$

then the VRPSD can be represented as follows.

$$\min_x \sum_{i,j} \sum_k c_{ij} x_{ijk} + \sum_k \lambda_k . E[l_k], \tag{2.90}$$

$$\text{subject to} \quad x \in S_m. \tag{2.91}$$

General Variable Recourse Formulations

Following from the general recourse model with a deterministic first-stage given in (2.75), if $Q(x, \phi_s)$ represents the recourse cost associated with x over a problem scenario denoted by ϕ_s, $s = 1, \ldots, \hat{s}$, a general recourse model of the SVRP can be written as follows:

$$\min_{x} \quad [cx + E_{\phi_s \in \xi}(Q(x, \phi_s))] \tag{2.92}$$

$$\text{subject to} \quad x \in S_m. \tag{2.93}$$

There are many possible ways to practically represent this recourse function, see Section 2.6.3. Indeed, since evaluating $x \in S_m$ is "obligatory" for the deterministic VRP, the primary modelling question in stochastic vehicle routing attenuates to dealing with the recourse function $E_{\phi_s \in \xi}(Q(x, \phi_s))$ which in itself is a minimisation.

Not only is it unreasonable to expect to adequately describe the exact sequence of decisions and events in a stochastic program, constraining the expected recourse function to be the result of a computation based on a system in steady state (i.e. a fixed first stage in a two-stage stochastic program), the recourse function itself is often so complex that it is not always available in a computationally tractable form, i.e. in the form of a simple equation. This complexity applies to the SVRP because of the nature of the problem and in the recognition that stochastic vehicle routing is, in fact, a multi-stage problem (see Section 2.7.1) that is modelled using two-stage stochastic programming.

All SVRPs are multi-stage in nature since random variables are recognised at successive stages during routing and recourse action is successively being taken. Nevertheless, for almost all SVRP interpretations (with late or early information and with existing knowledge), although no decision making is allowed whilst a vehicle is on a route, reactions to particular stochastic outcomes are pre-specified. Due to a corresponding property of block-separability, two-stage stochastic programming can, therefore, be applied. However, this is at the expense of incorporating a second stage recourse function that is far from easy to mathematically define. (For SVRPs that allow explicit second stage decision making, multi-stage stochastic programming needs to be applied.[7]) In stochastic programming methodology, there are two possibilities that exist to allow some treatment of such intractable recourse functions:

- Developing a closed form expression for the expected second stage value function $Q(x)$.

[7]The only BVRPSD extension that is actually multi-stage is the BVRPSD with Total Knowledge. This is because in the dynamic knowledge case, a limit was set that a driver would only return to the depot at the customer before a failure was due to occur, see Section 2.5.4.

- Developing an expression that allows the computation of the expected second stage value function $Q(x)$ given a first stage decision x.

In SVRP modelling, the exact closed form representation of $Q(x)$ for specific problem extensions is (rarely) obtainable and, even if available, such a formulation is incredibly cumbersome and of no obvious practical use. Although similarly complex, the expected second stage value function, which translates into the expected length of a series of routes, can be computed when the initial routes (x) are given. Since specific computational versions of the latter are presented throughout the thesis, we now only present a version of the former. Inspired by the commodity flow formulation for the VRP, first presented by Gavish and Graves [95], this formulation was introduced by Laporte and Louveaux [142].

Laporte and Louveaux present a VRPSD model that contains a form of existing knowledge. The formulation corresponds to a form of P5 where a vehicle driver pre-empts route failure if and only if a vehicle has zero load (as opposed to a general minimum demand level). To describe the recourse problem, let $z_{ij}^{\phi_s}$ = the commodity flow on arc (i,j) in ϕ_s for $i,j \in V, i \neq j, s \in \{1,\ldots,\hat{s}\}$, and let:

$$w_{ij}^{\phi_s} = \begin{cases} 1 & \text{if } \exists \text{ a route break on arc } (i,j) \text{ in } \phi_s, \\ 0 & \text{otherwise,} \end{cases} \qquad (i,j \in V\backslash v_0, i \neq j, s \in \{1,\ldots,\hat{s}\}),$$

$$w_{i}^{\phi_s} = \begin{cases} 1 & \text{if } \exists \text{ a depot return trip from } i \text{ in } \phi_s, \\ 0 & \text{otherwise.} \end{cases} \qquad (i,j \in V\backslash v_0, i \neq j, s \in \{1,\ldots,\hat{s}\}),$$

Given that x_{ij}, $i,j \in V, i \neq j$, are variables that take the value 1 iff arc (i,j) is in the optimal solution, then the recourse model can be represented by (2.92)-(2.93) where:

$$Q(x,\phi_s) = \min_{w_i^{\phi_s}, w_{ij}^{\phi_s}} \left[\sum_{i \in V\backslash v_1} 2c_{1i} w_i^{\phi_s} + \sum_{i \in V\backslash v_1} \sum_{j \in V\backslash v_1} (c_{1i} + c_{1j} - c_{ij}) w_{ij}^{\phi_s} \right] \tag{2.94}$$

subject to

$$\sum_{j \in V} z_{ij}^{\phi_s} - \sum_{j \in V\backslash v_1} z_{ji}^{\phi_s} \geq \varepsilon_i^{\iota(i,s)} - Q w_i^{\phi_s} - 2Q \sum_{j \in V\backslash v_1} w_{ij}^{\phi_s}, \tag{2.95}$$

$$\sum_{j \in V} z_{ij}^{\phi_s} - \sum_{j \in V} z_{ji}^{\phi_s} \leq \varepsilon_i^{\iota(i,s)} \left(1 - \sum_{j \in V\backslash v_1} w_{ij}^{\phi_s} \right), \tag{2.96}$$

$$\sum_{j \in V} z_{ij}^{\phi_s} \geq \varepsilon_i^{\iota(i,s)} \left(1 - \sum_{j \in V\backslash v_1} w_{ij}^{\phi_s} \right) - Q(1 - w_i^{\phi_s}), \tag{2.97}$$

$$\sum_{j\in V} z_{ij}^{\phi_s} \le \varepsilon_i^{\iota(i,s)}\left(1-\sum_{j\in V\backslash v_1} w_{ij}^{\phi_s}\right)+Q(1-w_i^{\phi_s})-Q\sum_{j\in V\backslash v_1} w_{ij}^{\phi_s}, \tag{2.98}$$

$$Q\left(w_i^{\phi_s}+\sum_{j\in V\backslash v_1} w_{ij}^{\phi_s}\right)\le \varepsilon_i^{\iota(i,s)}+\sum_{j\in V} z_{ji}^{\phi_s}, \tag{2.99}$$

$$\varepsilon_i^{\iota(i,s)}+\sum_{j\in V} z_{ji}^{\phi_s}\le Q(1+w_i^{\phi_s}), \tag{2.100}$$

$$w_i^{\phi_s}+\sum_{j\in V\backslash v_1} w_{ij}^{\phi_s}\le 1, \tag{2.101}$$

$$z_{ij}^{\phi_s}\le Qx_{ij}, \tag{2.102}$$

$$w_{ij}^{\phi_s}\le x_{ij}, \tag{2.103}$$

$$\sum_{j\in V\backslash v_1} z_{1j}^{\phi_s}=0, \tag{2.104}$$

$$z^{\phi_s}\ge 0, \tag{2.105}$$

$$0\le w^{\phi_s}\le 1 \text{ and integer}, \tag{2.106}$$

(2.95)-(2.101) for $i\in V\backslash v_1, i\neq j, s\in\{1,\ldots,\hat{s}\}$,

(2.102)-(2.103) for $i,j\in V\backslash v_1, i\neq j, s\in\{1,\ldots,\hat{s}\}$,

(2.104)-(2.106) for $s\in\{1,\ldots,\hat{s}\}$.

Notice that, for purposes of convenience, customer demand is interpreted as implying a series of collections as opposed to deliveries. Although fairly complicated, the recourse function can then be interpreted as follows: constraints (2.99) imply that $w_i^{\phi_s}$ and $w_{ij}^{\phi_s}$ both equal zero $\forall\, j\in V\backslash v_1$ if, after a collection at i, vehicle load is strictly smaller than Q. Constraints (2.95) and (2.96) imply that the flow leaving i is then equal to the flow entering i plus the amount collected at i and, in addition, the remaining constraints are all satisfied. Constraints (2.100) and (2.101) imply that $w_i^{\phi_s}=1$ and $w_{ij}^{\phi_s}=0$ respectively $\forall\, j\in V\backslash v_1$ if, after a "collection" at i, vehicle load is strictly greater than Q. Constraints (2.97) and (2.98) imply that the vehicle load is then equal to $\varepsilon_i^{\iota(i,s)}$ and, in addition, the remaining constraints are all satisfied. Constraints (2.99)-(2.101) imply that $w_i^{\phi_s}=0$ and $\sum_j w_{ij}^{\phi_s}=1$ if, after a "collection" at i, vehicle load is exactly equal to Q (this is the case when $C=(c_{ij})$ satisfies the triangular inequality and when v_1, v_i and v_j are not collinear).

A Markov Decision Process Model

As an alternative to stochastic programming, it is possible to model SVRP as a Markov decision process where optimal actions are taken at discrete points in time. Actions are influenced by random outcomes and states change over different stages. Here, we briefly summarise how to form a related state space for a SVRP Markov decision process. For more information see [70].

Consider a one vehicle version of P3. A vehicle that arrives at a customer site has only one possible decision action, i.e. replenish the customer (either fully or partially) and move to another location. The latter location is the depot when either all the customers have been serviced or vehicle load is zero. The initial state corresponds to a full vehicle with all customers to be served and the final state of the system corresponds to all the customers having been served and the vehicle, by definition, returning to the depot. An observation occurs each time a vehicle arrives at a location for the first time. Let τ_i, $i \in \{1, \ldots, n\}$, correspond to the total distance travelled up to the point of the i^{th} observation, i.e. $\tau_1 = 0$ and $\tau_i \leq \tau_{i+1}$. These transition distances are analogous to the orthodox transition times in Markov notation and correspond to the distance travelled up to the points at which decisions are taken. The state of the system at a given transition (distance) is described by $s = \{r, l, x_1, \ldots, x_n\}$ where r denotes the location of the vehicle, l denotes the commodity level of the vehicle and x_i represents the remaining demand of customer i (if i is not yet visited then $x_i = -1$). The state space is a subset of:

$$[\{1, \ldots, n\} \times [0, Q] \times (\{-1\} \cup [0, Q])^n]$$

which satisfies the above conditions.

A decision is taken at each transition (distance) so that a vehicle goes to a customer whose demand is undetermined and, on its route, replenishes, on a shortest path, a subset of customers whose demand is already known. Consequently, if the system is in a particular state when a decision is taken, the distance corresponding to the next transition is deterministic. The next state is then generated according to the probability distribution of the demand variables. Needless to say, the aim of the decision maker is to find a policy that minimises the expected distance of a decision sequence.

2.7.3 Computational Complexity

The SVRP is clearly at least as difficult as the VRP. The presence of nonlinearities in the constraints (or the objective function) makes it considerably more difficult in most cases. Chance-constrained programs are easier to resolve compared to their recourse-based counterparts since they can often be modelled as single-stage deterministic equivalents. As stated in the previous section, the primary issue of complexity for recourse representations of the SVRP, in contrast to the deterministic case, revolves around the expected second-stage recourse value function. Closed form representations of the latter are highly complex and almost impossible to formulate.

In general, a recourse-based stochastic programming SVRP model is a highly complex composite programs of two parts. Initially, a first stage integer program needs to be solved to find a feasible solution structure (one of a number of feasible sets of routes) that can be implemented into the next stage of the solution method. Then, a stochastic recourse formulation must be utilised to find the cost of the penalty function for the given first stage solution and so enabling the derivation of a solution value for the entire SVRP. Clearly, finding an optimal solution to such a complex problem in reasonable time and avoiding computer memory problems is a difficult task.

2.7.4 Solution Methods

Due to the intrinsic difficulty of SVRPs, the few existing studies in the literature have focused on heuristic methods. Indeed, apart from the work in this thesis, only one other exact algorithm has ever been developed for the SVRP and has been used to solve the VRPSD, VRPSC, VRPSCD and VRPST. We therefore simplify Fisher's review methodology for the VRP, see Section 2.3.2, by referring to two generations of research. The first corresponds to the use of simple heuristics, based exclusively on previous techniques for the deterministic VRP, that obtain computationally fast solutions of comparatively poor quality. The second is similar to the third VRP generation and refers to a combination of two attempts: to tune existing AI-based approximate methods for the SVRP and to develop existing MP and SP formulaic approaches to obtain exact stochastic routing solutions. Notice that the corresponding approach for the deterministic problem features prominently in each case.

Generation 1: Simple Heuristics

Simple SVRP heuristics can be categorised into the same three methods described for the deterministic case (see Section 2.3.2): route building, route improvement and two-phase.

The most straight-forward heuristic for the VRP is the Clark-Wright savings algorithm. Least cost insertions are successively incorporated into an existing feasible solution so that the capacity constraints (2.3) in the VRP formulation given by (2.2)-(2.4) are not violated. A simple extension of this method for a SVRP is to substitute the capacity check for a separate criteria. Tillman [201] introduces the dual cost aspect of a SVRP by considering the stochastic demand case and including the cost of hauling an amount of commodity which is not needed (excess) and the cost of not hauling enough to satisfy the demands of the customers on a particular route (route failure). More specifically, if $g_1(L)$ is the cost of hauling excess commodity and $g_2(L)$ is the cost of not hauling enough commodity, where $L = (\xi_1 + \ldots + \xi_k)$ corresponds to the sum of the random demand variables at the k stops on a given route and $h(L)$ is the probability density function of L then, for a given number of stops on a proposed route, the expected cost of L, $E(L)$, is given by:

$$E(L) = \sum_{L=0}^{Q} g_1^L h(L) + \sum_{L=Q+1}^{\infty} g_2^L h(L). \tag{2.107}$$

Tillman solves the first SVRP by using this cost function embedded in a savings algorithm to obtain a heuristic solution to a small (7 customer) multiple depot VRPSD. The extra cost criteria were given the following values: $g_1(L) = 30(L-Q)^2$ and $g_2(L) = 10(L-Q)^2$. Stewart [192] extends this and other work [108, 109] by introducing the concept of an artificial capacity $\overline{Q}$ that satisfies a VRP, i.e. the Clark Wright algorithm can be used in the normal way where the probabilistic chance-constraints given by (2.77) are replaced by an equivalent set of constraints that are linear in μ_i. In Section 2.7.2, it was shown how corresponding probabilistic constraints can be converted into deterministic equivalents. Under special conditions[8] the distribution of D_k in (2.82) can be given as follows:

$$D_k = \sqrt{\sum_{ij} \sigma_i^2 x_{ijk}^2} = \sqrt{\sum_{ij} \pi \mu_i x_{ijk}^2} = \sqrt{\pi}\sqrt{M_k}. \tag{2.108}$$

[8]Several important distributions, e.g. Poisson, binomial and gamma, satisfy the relevant conditions. Often, route demand distributions may be approximated by a normal distribution by appealing to the central limit theorem and the value T can be replaced by the appropriate standard normal deviate z.

By (2.81) we have:

$$M_k + T\sqrt{\pi M_k} \leq Q. \tag{2.109}$$

Therefore, the artificial capacity can be given by:

$$M_k \leq \frac{2Q + T^2\pi - \sqrt{T^4\pi^2 + 4QT^2\pi}}{2} = \overline{Q}. \tag{2.110}$$

Stewart also considers a savings approach for the three-index fixed and three-index variable recourse formulations given in Section 2.7.2. Both require a change in the existing savings function given by (2.33). For the fixed recourse case, the new saving that is obtained by joining i and j on the same route is given as follows:

$$s_{ij} = c_{1i} + c_{1j} - c_{ij} + \lambda P_i + \lambda P_j - \lambda P_{ij}, \tag{2.111}$$

where P_k is equal to the probability that the route customer k is on fails and P_{kl} is equal to the probability that the route connected by arc (k, j) fails.

For the variable recourse case, s_{ij} is given by:

$$s_{ij} = c_{1i} + c_{1j} - c_{ij} + \lambda L_i + \lambda L_j - \lambda L_{ij}, \tag{2.112}$$

where L_k is equal to the number of units "short" on the route that customer i is on and L_{kl} is equal to the number of units "short" on the route that is connected by the arc (k, j). More recent route building heuristic work has been completed to incorporate the location of route failure in the savings approach [72, 36].

A route improvement heuristic for the VRP [193] is also adapted for the SVRP by Stewart. A 3-opt exchange is used to find heuristic solutions to successive m-TSPs formed by moving the capacity constraints into the objective function. The application of this method to the chance-constrained case with an artificial capacity requires no explanation. For the recourse case, the Lagrange multipliers involved become the penalty multipliers given in (2.87) and (2.90). Stewart found that the savings heuristic was computationally faster but numerically outperformed by the 3-opt heuristic.

One of only two 2-phase heuristics proposed for the SVRP is given by Teodorovic *et al* [196]. They consider a route-first cluster-second approach where a giant TSP tour is constructed before vehicles are assigned to parts of the tour. This method was originally proposed for the VRP by Beasley [15]. The second 2-phase method is given by Teodorovic and Pavkovic [198] where a fuzzy logic approach is used to shrink a stochastic routing problem before applying a general sweep algorithm.

Generation 2A: AI-based Heuristics

Two AI-based heuristics have been used to solve SVRPs: Teodorovic and Pavkovic [197] apply simulated annealing to the VRPSD and Seguin [187] applies tabu-search to the VRPSCD. Both methods essentially follow their corresponding approaches for the deterministic case, see Section 2.3.2, however extensive testing has only been completed for the tabu search approach. We therefore omit the algorithmic details and concentrate on the performance of the latter approach.

The tabu search presented by Seguin is an adaptation of a method used by Gendreau *et al* [96] for the VRP. The main contribution of this work is in the construction of a suitable proxy objective function that approximates the total routing cost of a first stage solution. For a series of VRPSCDs, the average error between values obtained with the heuristic and associated known optimal solutions is reported to be 0.4%. These results appear to be highly encouraging, however it is important to note the following points:

(i) Approximately 50% of the tests completed were for problems where $n < 10$ (in fact over 90% of the tests completed were for problems where $n < 19$). In such cases, optimal solutions (see the next section) were found within the same computational time limit that was used for the tabu-search algorithm.

(ii) The test problems only involved VRPSCDs that had been solved exactly and the majority of these were of relative simplicity, i.e. the constraints are set so that the problem is in essence deterministic.

More extensive testing on a group of standard, computationally difficult test problems need to be completed before concrete conclusions can, therefore, be drawn about the performance of this algorithm. It should be noted that the same algorithm was also used for the VRPSD and the reported gap between the optimal solution and the heuristic solution averaged 9% and could be as high as 30% [100].

Generation 2B: Exact Methods

Prior to this research, only one exact solution method has ever been developed to solve the SVRP in any form. Laporte *et al* [145] and Laporte *et al* [146] use a simple branch and cut approach for a depot location version of the VRPSD and a VRPST respectively. Laporte *et al* [147] and Seguin [187] describe an enhancement of this algorithm that can be applied to the VRPSC and VRPSCD respectively. In all four reported cases, the branch and bound algorithm used is based on the 1973 Land-Powell code for integer linear programming [139]. For convenience, we summarise this algorithm before detailing the respective contributions of these studies.

Land-Powells B&B Algorithm

1. Let $z^* = \infty$. Define an initial subproblem and define upper bounds on the variables involved. Insert this subproblem in a list.

2. If the list is empty, stop.

3. Select the next subproblem to be solved from the list.

4. Solve the subproblem using the Simplex method. Let $\overline{z}$ be the solution value. If the problem is infeasible or $z^* < \overline{z}$ then goto 2. If the solution is integer then goto 6.

5. The solution is non-integer. Partition the current subproblem into a number of subproblems: this is done by branching on a fractional variable according to a criterion set by Land-Powell. For each problem, compute a lower bound $\underline{z}$ on the value of its objective. If $z^* < \underline{z}$ discard the subproblem. Insert all non-dominated subproblems in the list then goto 3.

6. The current solution is feasible and $z^* > \overline{z}$. Set $z^* = \overline{z}$ and goto 2.

Laporte et al (1989,1992): The importance of the branch and cut approach in this context is only realised in it forming the basis to the first exact results reported for any kind of SVRP. Laporte *et al* simply include a feasibility check before step 6 in the above algorithm that differs for particular problems. They consider two VRPSDs, both of which incorporate a variable specifying possible depot locations, and one VRPST. These three problems are further specified as follows:

(i) The two index chance-constrained VRPSD formulated in Section 2.7.2.

(ii) A version of the fixed recourse VRPSD formulated in Section 2.7.2 that is sometimes referred to as a bounded recourse VRPSD. In this case, the corresponding fixed cost arises when a bound on the size of the recourse cost is exceeded (this model uses the return trip back to the depot as the cost per route failure that contributes to the bounded recourse cost).

(iii) A fixed recourse version of the VRPST formulated in a similar manner to the bounded recourse model given above. In this case, the amount of travel time over a particular given amount (capacity) corresponds to the cost per route failure that in turn contributes to the bounded recourse cost.

The authors solve (i) by incorporating "connectivity constraints", given by (2.85), into the feasibility check. If all the constraints are satisfied then the algorithm proceeds to step 6, otherwise each violated constraint is included in the current subproblem and the algorithm returns to step 4. In (ii) and (iii) such a feasibility check involves ordinary subtour connectivity constraints and a recourse limit, i.e. a check occurs to see whether the current solution is connected to the depot and that it does not have too large a penalty.

The authors solve (i) for selected problems of up to $|N| = 30$, where $|N|$ represents the number of customers and possible depot locations, and they solve (ii) for selected problems of up to $|N| = 20$. In both cases, the parameters are set so that the problem is almost deterministic in nature. For example, in problem (i), the maximum probability of route failure was set to at 10%. This essentially means that the solution will be very similar to the solution obtained if the demands are set to their maximum value and the problem is solved as a deterministic VRP. Problem (iii) is solved for selected problems up to $n = 20$.

Laporte et al (1994): Laporte *et al* utilise the general recourse formulation given in Section 2.7.2 and consider an early information VRPSC[9]. In this problem, a negative cost arises when a customer is not present, i.e. a "failing" customer is bypassed and an associated saving is realised. Laporte *et al* extend Land-Powell's algorithm presented above by the implementation of an additional feasibility check known as an optimality cut. The procedure involved is known as the integer L-shaped method, first presented by

[9]By definition, the VRPSC assumes the presence of early information.

Laporte and Louveaux [143], and is an extension of the classical L-shaped solution method for stochastic programming problems introduced by Van Slyke and Wets [189] in 1969.

Consider a variable y that is a lower bound on the expected recourse function $Q(x) = E_{\phi_s \in \xi} Q(x, \phi_s)$ in (2.92). Initially equal to zero, it is gradually increased in the course of the algorithm through the introduction of lower bounding optimality cuts. This cut is obtained as follows for the one vehicle VRPSC. Let x^k (with edge set E^k) be an optimal solution to a subproblem. Then:

$$\sum_{(v_i, vj) \in E^k} x_{ij}^k = n, \tag{2.113}$$

and any other solution $x^{k'}$ (with edge set $E^{k'}$) is such that:

$$\sum_{(v_i, vj) \in E^k} x_{ij}^{k'} \leq n - 2, \tag{2.114}$$

Therefore, if $Q(x^k)$ is the value of the expected penalty, then y must satisfy $y \geq Q(x^k)$ for solution x^k, and $y \geq L$ for any solution $x^{k'}$, where L is a simple lower bound on the expected recourse function. This is expressed as the following optimality cut:

$$y \geq \frac{(Q(x^k) - L)}{2} \left(\sum_{(v_i, vj) \in E^k} x_{ij} - n \right) + Q(x^k). \tag{2.115}$$

The resulting algorithm is tested on a series of randomly generated one vehicle VRPSCs and selected problems were solved to optimality for $n = 50$.

Seguin (1994): Seguin simply extends the optimality cut given above for the general VRPSC and includes stochastic customer demands in associated test problems. In the latter, as well as negative recourse costs that arise due to bypassed customers, there exists the original positive recourse costs that arise when vehicle capacities are exceeded, i.e. a route break to the depot occurs before "failing" customers.

For the VRPSCD, since y must satisfy $y \geq Q(x^k)$ for a solution x^k, and $y \geq 0$ for any solution $x^{k'}$, a simpler optimality cut is given as follows:

$$y \geq Q(x^k) \left(\sum_{(v_i, vj) \in E^k} x_{ij} - n + m + 2 \right). \tag{2.116}$$

Seguin considers a series of randomly generated problems. The only reported complete results for all problems, tightly constrained or not, are found for $n = 10$. Some selected problems are solved for $n = 70$. In these problems however, the solution equates to the solution of the m-TSP. Seguin also solves similar size VRPSDs.

2.8 Summary of Main Contributions to SVRP Research

In this section, all known work on stochastic vehicle routing is gathered together and each SVRP is classified in turn. A tabular summary of all the contributions to SVRP research is given in Table 2.4 and the important contributions are collected together in Table 2.5.

A comprehensive summary of dynamic vehicle routing and vehicle routing with stochastic customers can be found in Powell *et al* [172] and Bertsimas and Schmi-Levi [29]. The only other survey for the SVRP is given by Gendreau *et al* [98].

2.8.1 VRPSD

Tillman [201] was the first to consider a stochastic vehicle routing problem of any kind by developing an adapted Clark-Wright Savings algorithm to account for stochastic demands. A very small size problem of seven customers with Poisson demands was used in the testing process. Golden and Stewart [108] introduce artificial capacities and are the first to apply stochastic programming to the VRPSD. They consider one large scale VRPSD of 75 customers with Poisson demands. Golden and Yee [109] extend this work and consider correlated demands and alternative demand distributions. Yee and Golden [214] are the first to consider the location of route failure and present a dynamic programming recursion that can be used to compute full recourse values.

A major contribution to VRPSD research is Stewarts PhD thesis [192]. Full stochastic programming formulations for the chance constrained and penalty function cases are presented and possible heuristic approaches are compared. Stewart and Golden [194] summarise the work found in Stewart's thesis. Dror and Trudeau [72] consider route failures in a VRPSD and investigate an extension of the Clark-Wright algorithm to include the location of route failure. Apart from two exceptions, a savings approach for the VRPSD with split deliveries [36] and a two-phase approach for the VRPSD [196], this paper marks the end of the use of simple heuristics for the VRPSD.

VRPSD	VRPSC
Tillman (1969)	Jaillet (1985)
Golden and Stewart (1978)	Jezequel (1985)
Golden and Yee (1979)	Jaillet (1988)
Yee and Golden (1980)	Jaillet and Odoni (1988)
Stewart (1981)	Berstimas (1988)
Stewart and Golden (1983)	Waters (1989)
Dror and Trudeau (1986)	Jaillet (1989)
Larson (1988)	Bertsimas et al (1990)
Laporte et al (1989)	Bertsimas (1992)
Dror and Trudeau (1989)	Jaillet (1993)
Laporte and Louveaux (1990)	Benton and Rossetti (1993)
Bastian and Kan (1992)	Bertsimas and Howell (1993)
Teodorovic and Pavkovic (1992)	Laporte et al (1994)
Bouzaiene-Ayari et al (1993)	Bertsimas et al (1995)
Dror (1993)	VRPST
Dror et al (1993)	Kao (1978)
Popovic (1995)	Cook and Russell (1978)
Gendreau et al (1995)	Sniedovich (1981)
Teodorovic et al (1995)	Carraway et al (1989)
Gendreau et al (1996)	Laporte et al (1992)
Teodorovic and Pavkovic (1996)	Lambert et al (1993)
VRPSCD	SVRP
Seguin (1994)	Powell et al (1995)
Gendreau et al (1995)	Gendreau et al (1996)
Gendreau et al (1996)	Bertsimas et al (1997)

Table 2.4: Contributions to SVRP Research

Paper	Description
Tillman (1969)	Savings heuristic - First SVRP paper
Golden and Stewart (1978)	Savings heuristic, CC approach
Stewart (1981)	Heuristics, SP formulations, application
Jaillet (1985)	Solution properties
Jezequel (1985)	Simple heuristics
Dror and Trudeau (1986)	Savings heuristic, properties of route failure
Bertsimas (1988)	Heuristics and solution properties
Laporte et al (1989)	Branch and cut (first exact SVRP solutions)
Teodorovic and Pavkovic (1992)	Simulated annealing (first SVRP AI-heuristic)
Laporte et al (1992)	Branch and cut (selected VRPSTs up to n=20)
Laporte et al (1994)	L-shaped method (selected TSPSCs up to n=50)
Seguin (1994)	L-shaped method (complete VRPSCDs up to n=10)
Seguin (1994)	Tabu-search (complete VRPSCD heuristic results)

Table 2.5: The Important Contributions to SVRP Research

The first of many articles proposing a VRPSD formulation but without any associated solution method was given by Dror and Trudeau [70]. In this paper the authors present a number of SP models and a Markov decision process model, together with some properties relating to the solution of a VRP that do not apply to a solution of a VRPSD since a stochastic routing solution may intersect itself. Further formulations have been developed by a number of authors, these include:

(i) an early information model by Laporte and Louveaux [142],

(ii) a late information model by Laporte and Louveaux [142],

(iii) a full service model by Bastian and Kan [13],

(iv) a full service model by Dror [68],

(v) a model with a limit on the number of route failures by Dror *et al* [69], and,

(vi) a Bayesian model by Popovic [170].

These have yielded no exact solutions apart from a special location-routing model presented by Laporte *et al* [145] and a special case of the PVRP with stochastic demands and deterministic customer presence by Seguin [187] (this work is summarised in Gendreau *et al* [97]). In the former paper, problems are solved to optimality for $N = |30|$, where N represents both the number of customers and possible depot sites. In the latter paper, exact solutions are given for problems of up to 70 customers however the parameters set such that the problem is in essence deterministic.

Teodorovic and Pavkovic [197] present a simulated annealing AI-heuristic and a fuzzy logic approach is employed by Teodorovic and Pavkovic [198]. In addition, there are some isolated cases where the problem has been investigated and documented for real-world applications. Stewart [192] considers bus-routing where the amount of people to pick-up at stops are given as a random variable and Larson [152] investigates the problem of sludge disposal where accumulation in any plant is a random process.

2.8.2 VRPSST

The VRPSST has never before been formulated or solved in the literature, however the problem has very close links to two other stochastic routing problems: the Vehicle Routing

Problem with Stochastic Travel Times and the Vehicle Routing Problem with Stochastic Demands. The only study to deal with the VRPST in its complete form, including demands that may or may not be stochastic, is given by Cook and Russell [50] who consider a simulation approach.

2.8.3 VRPST

The objective of the VRPST - and its one vehicle counterpart, the Travelling Salesman Problem with Stochastic Travel Times (TSPST) - usually involves finding an a priori solution such that the probability of completing a tour within a given deadline is maximised. As such, the VRPST is interchangeable with the multiple vehicle TSPST (m-TSPST).

Kao [133] proposes two heuristics for the TSPST, the first based on dynamic programming (DP) and the second based on implicit enumeration. Sniedovich [190] shows that obtaining optimal solutions using the former DP approach is reliant on the property of monotonicity and Carraway *et al* [38] presents a generalised DP method that overcomes this problem. Lambert *et al* [138] derive a heuristic solution algorithm for the m-TSPST, based on the well-known Clark-Wright savings procedure, and find cost effective cash-collection routes through bank branches where the amount of cash collected is limited by an insurance company and late arrival incurs a penalty relating to lost interest. Laporte *et al* [146] are the first to consider an alternative objective for the VRPST. They present a three-index simple recourse model and a two-index recourse model for a VRPST based on finding a minimum cost a priori solution where the penalty for late arrival is proportional to the length of the delay. Using the branch and cut method previously described they present exact results for selected VRPSTs of up to twenty customers.

2.8.4 VRPSC

The VRPSC is a direct multiple vehicle extension of the TSPSC, see [125]. Here we review both types of problems. Several simple heuristics have been proposed for the TSPSC including a savings approach and a 2-opt method [129]. These two studies are summarised by Jaillet [126] and Jaillet and Odoni [128]. The main contribution to the study of VRPSC however, is the PhD thesis of Bertsimas [21]. The array of bounds and properties presented are outside the scope of this thesis, however for reference the work is summarised by Bertsimas *et al* [25] and Bertsimas [22]. Using the L-shaped method, Laporte *et al* [147] solve medium size TSPSCs and Seguin [187] (summarised in Gendreau *et al* [97]) solve small size VRPSCs. Other theoretical studies have been completed by Jaillet [127] and Bertsimas and Howell [24].

Waters [207] discusses an application to cost cutting for a wholesale distributor and provides the only case where customers are treated with non-binary demands. In addition, Waters empirically compares strategies including following the planned route, skipping absent customers and reoptimising the remaining route whenever the customer is revealed. A similar simulation approach is used by Benton and Rossetti [18]. Bertsimas *et al* [23] considers an empirical study of the reoptimisation strategy and compares it to the original a priori solution, see Chapter 7.

2.8.5 VRPSCD

The only authors to treat this problem are Gendreau *et al* [97, 99]. In the former, a tabu-search heuristic has been applied and comprehensively tested. In the latter, the integer L-shaped method has been used to obtain complete optimal results for small size problems.

2.9 Summary

The purpose of this chapter has been to introduce deterministic vehicle routing as a research field in operations research before detailing its more applicable stochastic equivalent. Existing VRP solution methods were considered in detail at this stage since the adaptations of such techniques form the basis to practically all existing research in stochastic vehicle routing.

Stochastic vehicle routing was introduced and a series of basic interpretations were defined, together with appropriate notation. The characteristics of a SVRP solution were discussed and this highlighted a number of practical difficulties that arise when considering stochasticity in a routing context. During this description, a number of definitions, e.g. problem scenarios, route failure and a priori optimisation, were presented and subjective issues, e.g. information disclosure and the cost of recourse, were clarified. Such a discussion eventually led to the development of a series of extensions that would adequately describe the appropriate practical forms of a stochastic vehicle routing problem. The particular SVRP variation that was used to illustrate this process was the VRPSD.

In Chapter 5 such problems will be explored further.

The final part of this chapter has considered the solution of SVRPs. Stochastic programming was introduced before detailing possible SVRP formulations and solution methods and completing a comprehensive literature review. Clearly, finding optimal solutions to SVRPs is a difficult task. Without doubt, the limited amount of associated research in the literature and the predominance within such studies for heuristic methods attests to such a fact. Nevertheless, several indicators, over and above the more practical reasons given in Chapter 1, point to exploring a new exact method for the problem. Firstly, designing a good heuristic is no easy task in a stochastic context. Gendreau *et al* [99] state that major difficulties arise when designing efficient heuristics for the SVRP because the "shape of optimal solutions is quite often counter intuitive from a geometric standpoint." Since both simple and AI-heuristic approaches rely heavily on geometric propositions to construct and alter routing configurations, highly sub-optimal solutions can result from approximate methods. In addition, heuristic methods are not well equipped to include stochasticity in their algorithmic construction and in many ways it may be more beneficial to solve an "average-value" deterministic equivalent. On the other hand, the only exact method produced has also largely ignored the presence of associated stochastic distributions. More specifically, no exact solution method has ever included any criterion based around the presence of stochastic demand or, apart from two exceptions, has ever actually found solutions to stochastic routing problems that are computationally difficult (in the stochastic sense): Seguin's very small size VRPSCDs and Laporte's medium size one vehicle VRPSCs. Both these contributions only accounted for the stochasticity in customer presence. Moreover, the exact (and heuristic) methodologies produced thus far have been based predominantly on similar VRP approaches and have therefore failed to create genuinely new, useful and applicable techniques for a particularly new, useful and applicable stochastic combinatorial optimisation problem.

In summary, in addition to the reasons for developing an exact approach as opposed to a heuristic approach (see Chapter 1), it was considered that these considerable algorithmic omissions in the literature provided added research impetus towards developing a new exact method for SVRP. In addition, during the construction of the algorithm, emphasis was placed on the applicability and generic nature of the model for the reasons already disclosed in Chapter 1.

Chapter 3

The Paired Tree Search Algorithm

3.1 Introduction

This chapter describes a new exact algorithm, known as the Paired Tree Search Algorithm (PTSA), which forms the basis of the theoretical and practical work in this thesis. Throughout the chapter, the component parts of the algorithm are demonstrated using the simplest BVRPSD extension that incorporates variable recourse, namely P4. In Section 3.2, a complete formulation for this problem is developed and the motivation behind the creation of the PTSA approach is described. In Sections 3.3 and 3.4 respectively, the foundations of the methods employed in the algorithm are presented and algorithmic details are described. Following the introduction of a set of test problems in Section 3.5, computational results in Section 3.7 show the effectiveness of the PTSA and of the specific procedures used by the algorithm.

3.2 A Two-Stage VRPSD Formulation (Extension P4)

From Chapter 2, a general two-stage stochastic program with recourse can be described as follows:

$$\min_{x \in X} E_\xi f_0(x, \xi) = \min_{x \in X} E_\xi (g_0(x, \xi) + Q(x, \xi)). \tag{3.1}$$

Given a deterministic first stage, this can be reduced to:

$$\min_{x} \ [g_0(x) + Q(x)], \tag{3.2}$$

where $Q(x) = E_{\varepsilon \in \xi}(Q(x, \varepsilon))$ corresponds to second stage recourse value function, $\min g_0(x)$ is the objective function of the first stage problem and "min $E_{\varepsilon \in \xi}(Q(x, \varepsilon))$" is the objective function of the second stage problem.

The general recourse model for the SVRP is then given as follows.

$$\min_{x} \quad [cx + E_{\phi_s \in \xi}(Q(x, \phi_s))], \tag{3.3}$$

$$\text{subject to} \quad x \in S_m. \tag{3.4}$$

where S_m corresponds to the set of feasible solutions to the m-TSP and $Q(x, \phi_s)$ represents the recourse cost associated with a set of routes x over a problem scenario denoted by ϕ_s, $s = 1, \ldots, \hat{s}$. It has been shown that there are many possible ways to represent this recourse function, both mathematically (e.g. closed form representations and computational form representations) and practically (e.g. fixed and variable costs of recourse). The former is, to a certain extent, governed by the latter which in turn depends on the particular SVRP extension being modelled. Here we consider the BVRPSD with Variable Costs of Recourse, P4.

The two practical stages of P4 can be stated as follows. In a first stage, a set of m vehicle routes of minimal cost are determined so that (i) each route starts and ends at the depot and (ii) each customer is visited exactly once by one vehicle. In a second stage, the first stage routes are followed as planned but whenever capacity is exceeded along a route vehicles return to the depot to refill and then continue along their pre-defined route. Given that second stage recourse costs are represented by the values of such return trips to the depot (see Table 2.6.3), the objective is to design a minimum expected cost-set of routes such that demand is met, (i) and (ii) are satisfied and exactly m vehicles are used. In formulaic terms, these two stages are presented independently below.

3.2.1 The First Stage Problem

The first stage solution corresponds directly to the solution of a m-TSP where a complete feasible set of routes are required to minimise cx. A simple two-index formulation for this problem, similar to Laporte *et al*'s two-index vehicle flow VRP formulation given in Section 2.3.1, is presented here.

With c_{ij} and x_{ij} interpreted as c_{ji} and x_{ji} whenever $i > j$, we define integer decision variables x_{ij} as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is used in the first stage solution and } 1 \leq i < j \leq n, \\ 2 & \text{if } (i,j) \text{ is used as a return trip and } i = 1, j > 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

A feasible set of routes is then obtained by solving the following:

$$\min_{x} \; g_0(x) = cx \tag{3.6}$$

subject to

$$\sum_{j=2}^{n} x_{1j} = 2m, \tag{3.7}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad (v_k \in V\backslash\{v_1\}), \tag{3.8}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \qquad (S \subset V\backslash\{v_1\}\,,\; 3 \leq |S| \leq n-2), \tag{3.9}$$

$$0 \leq x_{1j} \leq 2 \qquad (v_j \in V\backslash\{v_1\}), \tag{3.10}$$

$$0 \leq x_{ij} \leq 1 \qquad x_{ij} \text{ integer}, \; (1 \leq i < j \leq n). \tag{3.11}$$

These first stage deterministic constraints specify that m vehicles enter and leave the depot, (3.7), that every customer receives a visit exactly once, i.e. the vertex degree constraints (3.8), and that individual routes disconnected from the depot are prohibited, i.e. the classical connectivity constraints (3.9).

3.2.2 The Second Stage Problem

The second stage solution, relating to $E_{\phi_s \in \xi}(Q(x, \phi_s))$ in (3.3), is less well defined. In Section 2.7.1, it was shown that it is very difficult to formulate a function corresponding to the recourse value in closed form. Indeed, no such closed form formulation has ever been presented for P4 in the literature. It is generally considered to be more useful to consider the value of the recourse cost in computational form given a fixed first-stage solution.

We present two possible formulations:

- A new closed form representation of the recourse value function for P4, and,
- A new recursive-based formulation that can be used to compute the value of the recourse cost function for P4 given a fixed first-stage set of routes.

A Closed Form Representation of Q(x)

The following formulation is adapted from the commodity flow based formulation (2.94)-(2.106) that was originally proposed by Laporte and Louveaux [142] for a different VRPSD extension. Notice that, for the sake of convenience, customer demand is interpreted as implying a series of collections as opposed to deliveries.

To describe the recourse problem, let $z_{ij}^{\phi_s}$ = the commodity flow on arc (i,j) in ϕ_s for $i,j \in V, i \neq j, s \in \{1,\ldots,\hat{s}\}$, and let:

$$w_i^{\phi_s} = \begin{cases} 1 & \text{if } \exists \text{ a depot return trip from } i \text{ in } \phi_s, \\ 0 & \text{otherwise.} \end{cases} \qquad (i \in V\backslash v_0, s \in \{1,\ldots,\hat{s}\}).$$

The recourse value function can then be given as follows:

$$Q(x,\phi_s) = \min_{w_i^{\phi_s}} \sum_{i \in V\backslash v_1} 2c_{1i}w_i^{\phi_s} \tag{3.12}$$

subject to

$$\sum_{j\in V} z_{ij}^{\phi_s} - \sum_{j \in V\backslash v_1} z_{ji}^{\phi_s} \geq \varepsilon_i^{\iota(i,s)} - Qw_i^{\phi_s}, \tag{3.13}$$

$$\sum_{j\in V} z_{ij}^{\phi_s} - \sum_{j\in V} z_{ji}^{\phi_s} \leq \varepsilon_i^{\iota(i,s)}, \tag{3.14}$$

$$\sum_{j\in V} z_{ij}^{\phi_s} \geq \varepsilon_i^{\iota(i,s)} - Q(1-w_i^{\phi_s}), \tag{3.15}$$

$$\sum_{j\in V} z_{ij}^{\phi_s} \leq \varepsilon_i^{\iota(i,s)} + Q(1-w_i^{\phi_s}), \tag{3.16}$$

$$Qw_i^{\phi_s} \leq \varepsilon_i^{\iota(i,s)} + \sum_{j\in V} z_{ji}^{\phi_s}, \tag{3.17}$$

$$\varepsilon_i^{\iota(i,s)} + \sum_{j\in V} z_{ji}^{\phi_s} \leq Q(1+w_i^{\phi_s}), \tag{3.18}$$

$$w_i^{\phi_s} + \sum_{j\in V\backslash v_1} w_{ij}^{\phi_s} \leq 1, \tag{3.19}$$

$$z_{ij}^{\phi_s} \leq Qx_{ij}, \tag{3.20}$$

$$w_i^{\phi_s} \leq x_{ij}, \tag{3.21}$$

$$z^{\phi_s} \geq 0, \tag{3.22}$$

$$0 \leq w^{\phi_s} \leq 1 \text{ and integer}, \tag{3.23}$$

(3.13)-(3.21) for $i \in V\backslash v_1, i \neq j, s \in \{1,\ldots,\hat{s}\}$,

(3.22)-(3.23) for $s \in \{1,\ldots,\hat{s}\}$.

By considering the following three occurrences at customer i, the second stage constraints (3.13)-(3.23) can be interpreted in the following way:

- If, after collection at customer i, the vehicle load is strictly smaller than the vehicle capacity, i.e. (amount to be collected + amount before entering) $< Q$, by (3.17) $w_i^{\phi_s} = 0 \ \forall \ i \in V \backslash v_1$. Then, by (3.13) and (3.14), the vehicle load when leaving customer i is equal to the amount collected plus the amount of vehicle load when entering the vertex, i.e. the vehicle collects and continues as required. The remaining constraints (3.15), (3.16), (3.19)-(3.23) are satisfied.

- If it is not possible to load customer i's supply, i.e. (amount to be collected + amount before entering) $> Q$, assuming that:

$$\sum_{j \in V} z_{ji}^{\phi_s} \leq 2Q, \tag{3.24}$$

  by (3.19) $w_i^{\phi_s} = 1 \ \forall \ i \in V \backslash v_1$. Then, by (3.15) and (3.16), the vehicle load when leaving customer i is equal to the amount collected, i.e. the vehicle has returned to the depot to deposit its load. The remaining constraints, (3.13), (3.14), (3.19)-(3.23) are satisfied.

- If, after collection at customer i, the vehicle load is exactly equal to the vehicle capacity, i.e. (amount to be collected + amount before entering) $= Q$, constraints (3.17) and (3.18) are redundant and $w_i^{\phi_s} = 0$ or $1 \ \forall \ i \in V \backslash v_1$. Therefore, since we are dealing with a minimisation and $w_i^{\phi_s}$ is present in the objective function, no failure will occur at i. Clearly, however, there will always be a failure at the following customer.

A Recursive Formulation For Q(x)

The following dynamic programming-based recursive formulation is adapted from a similar formulation given by Bertsimas [22] for the VRPSCD. It should be noted that the formulation corresponds to a computational form of the recourse function in that it can only be used to evaluate the value of $Q(x)$ for a fixed first-stage solution x.

Consider a first-stage feasible solution characterised by the vector $x^v = [x_{ij}^v]$. Given that the problem scenario ϕ_s, $s \in \{1, \ldots, \hat{s}\}$, is realised from the random demand variable

ξ, let $T(x^v)$ denote the expected second stage costs where:

$$T(x^v) = E_{\phi_s \in \xi}(T(x^v, \phi_s)). \tag{3.25}$$

Let $T^k(x^v, \phi_s)$ denote the cost of route k in x^v and let the expected cost of any route k be computed in the following way:

$$T^k(x^v) = E_{\phi_s \in \xi}(T^k(x^v, \phi_s)). \tag{3.26}$$

The expected cost of m vehicle routes with a fixed x^v is then given as follows:

$$T(x^v) = \sum_{k=1}^{m} T^k(x^v). \tag{3.27}$$

From the definition of the VRPSD given in Section 2.4.1, let p_i^l represent the probability of occurrence of the l^{th} ordered demand level ε_i^l originating from the demand set ξ_i of customer i. In addition relabel the vertices of the k^{th} route of x^v so that the route becomes $(v_1, v_2, \ldots, v_{t_k}, v_{t_k+1} = v_1)$. Now, if d is the load of the vehicle upon arrival at customer i, the expected cost of a route k is as follows:

$$T^k(x^v) = E_{\phi_s \in \xi}(T^k(x^v, \phi_s)) = \alpha_2^k(Q) \tag{3.28}$$

where

$$\alpha_{t_k}^k(d) = 2c_{1i} \sum_{l|\varepsilon_{t_k}^l > d} p_{t_k}^l \qquad (0 < d \leq Q), \text{ and,} \tag{3.29}$$

$$\alpha_i^k(d) = p_i^{l^*} \alpha_{i+1}^k(0) +$$

$$\sum_{l|\varepsilon_i^l > d} p_i^l(\alpha_{i+1}^k(Q - \varepsilon_i^l) + 2c_{1i}) +$$

$$\sum_{l|\varepsilon_i^l < d} p_i^l \alpha_{i+1}^k(d - \varepsilon_i^l) \qquad (i = 2, \ldots, t_{k-1}; 0 < d \leq Q). \tag{3.30}$$

and

$$p_i^{l^*} = \begin{cases} p_i^l & \text{if } \varepsilon_i^l = d, \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

The proof of (3.28)-(3.30), which represents the expected total of the extra costs incurred (when a vehicle exceeds capacity) on an original a priori routing sequence, follows directly from the definition of $\alpha_i^k(d)$ which represents the expected penalty cost from vertex v_i of route k given that the load of the vehicle before entering vertex v_i is d.

Consider a customer i where $i < t_k$. For a vehicle entering i with load d, the expected recourse cost is made up of three component parts:

(i) The cost, $\alpha_{i+1}^k(0)$, that is incurred when the load at the following customer will be exactly zero, i.e. vehicle load has been exhausted (to zero) at i but no failure occurs at this customer because of the presence of late information.

(ii) The cost, $\alpha_{i+1}^k(Q - \varepsilon_i^l) + 2c_{1i}$, that is incurred when the load at the following customer will be $Q - \varepsilon_i^l$, i.e. a route failure has occurred at i and a trip back to the depot is necessary.

(iii) The cost, $\alpha_{i+1}^k(d - \varepsilon_i^l)$, that is incurred when the load at the following customer will be $d - \varepsilon_i^l$, i.e. no route failure has occurred at i and the vehicle continues along its route.

From these observations, (3.30) follows since the probabilities attached to each respective cost are given as follows:

(i) The probability that vehicle load at customer i is exactly equal to the demand at customer i, i.e. $p_i^{l^*}$.

(ii) The probability that vehicle load at customer i is strictly less than the demand at customer i, i.e. $\sum_{l|\varepsilon_i^l > d} p_i^l$.

(iii) The probability that vehicle load at customer i is strictly greater than the demand at customer i, i.e. $\sum_{l|\varepsilon_i^l < d} p_i^l$.

The corresponding function for a vehicle with load d entering the last customer ($i = t_k$) on a route is simply made up of the costs incurred when a route failure has occurred at i. This is expressed by (3.29). Hence, it follows that the recourse cost can be computed recursively from (3.28) using (3.29) and (3.30).

3.2.3 Summary

The resulting stochastic programming model for P4 is a highly complex composite program of two parts. Initially, a first stage integer program, (3.6)-(3.11), needs to be solved to find the cost of a feasible solution structure (one of a number of feasible sets of routes) that can be implemented into the next stage of the solution method. Then, either an integer program (3.13)-(3.23) or a recursive dynamic program (3.27)-(3.30) must be solved to optimality to obtain the recourse cost of the first-stage solution.

For the reasons discussed in general in Section 1.3 and more specifically in Section 2.9, optimisation appears to be the more obvious research tool for SVRP over approximation: to the best of our knowledge, only one exact solution method exists in the literature, rapidly decreasing costs of computation make optimisation more attractive (indeed reducing computational time is not usually seen as being a priority for SVRPs) and heuristics have struggled to find high quality solutions to SVRPs and stochastic problems in general. Clearly, there remains a lot to understand about stochastic routing as a research field and optimisation will play a very important role in the learning process.

It was shown in Section 2.9 that a new solution method, optimal or not, should somehow allow for the presence of stochasticity so that for the first time computationally difficult SVRPs can be solved. Since the SVRP is at least as difficult as the VRP (and therefore the complexities that the deterministic problem encounters will be at least as restricting in the stochastic case) the size of attempted problems does not necessarily need to be particularly large. VRPs have, after all, only been solved exactly up to fifty customers. However, it was considered that a new solution algorithm could be devised to incorporate and therefore counter stochasticity in a way that was not previously possible. In addition, it was hoped that such an approach could also be generic enough to cope with a range of previously defined stochastic extensions, problem variations and applications.

To summarise, research was directed towards developing a new, generic, exact method for SVRP (initially P4). From the outset, the algorithm was to be in the form of a direct tree search, to provide a suitable structure for the two stage problem, however a new stochastic emphasis was to be included. More specifically, while operating in reasonable time and retaining a suitable limit on computer memory requirements, the algorithm was to locate optimal solutions by providing an adequate structure for (i) the mathematical implementation of the first stage problem and (ii) the formation of a method to evaluate and limit the computation of the second stage problem by somehow incorporating stochasticity in a dynamic setting.

3.3 Algorithmic Foundations

The PTSA is based on a direct tree search method, see Section 2.3.2, that is adapted for stochastic problems. The new method is also referred to as the stochastic branch and bound method.[1] The algorithm has its foundations in a stochastic decision tree approach and therefore, before detailing the PTSA, general dynamic solution structures will be summarised with respect to their role in modelling stochastic problems.

3.3.1 Dynamic Solution Structures for Deterministic Problems

The dynamic environment that is used to solve most combinatorial optimisation problems is the general tree search method or branch and bound approach. This method is based on the idea of intelligently enumerating all the feasible solutions of a COP. For all but the most elementary examples it is a hopeless task to investigate all the points in a feasible set and the efficiency of the method arises in the effective manner in which the possible solutions are enumerated. Effective enumeration depends on two factors, the quality of the *bound* and the proficiency of the *branching strategy*. At each node of a search tree the calculation of the bound on the value of the optimal solution to the sub-problem corresponding to that node forms the basis to any associated algorithm. As well as limiting the tree search, calculating bounds helps to guide the partitioning of the feasible solution subsets and subsequently identifies the optimal solution. The branching strategy corresponds to the decision, at each node in the search tree, to choose or reject an arc which would branch from a node of the search tree and extend a partially completed solution. During the expedient traversal of a tree, the search method can be limited through the process of fathoming. A node is fathomed (during minimisation) when the lower bound at a certain stage is greater than or equal to the current upper bound (the best available solution so far). Backtracking can then occur in a similar manner to when a feasible solution has been found at a leaf of the tree.

[1]The PTSA or stochastic branch and bound is not to be confused with probabilistic tree search methods that incorporate probability-driven approximate methods to achieve heuristic solutions.

For the SVRP, each node can refer to the depot or an individual customer and the branching process can correspond to the choice of whether or not a partial route is allowed between one location and another. Notwithstanding the simplicity of this structure, difficulties arise concerning how the cost of each partial route, or arc, is to be explicitly evaluated and indeed implemented in such a dynamic procedure. Each arc coincides with the construction of the two separate sub-problems given by (3.6)-(3.11) and (3.28)-(3.30), if the recursive formulation for $Q(x)$ is used. If the ascendent node corresponds to customer i and the descendent node corresponds to customer j, the former involves the simple addition of the primary travel cost, c_{ij}, to the composite objective function (3.3) given none of the first stage constraints are violated. The contribution of the latter recourse function to the composite objective function is not so easily calculated dynamically however. In fact, it is neither easy enough to be evaluated explicitly nor simplified enough (dynamically speaking) to be used as the basis for any necessary bounding procedures. Indeed, it is this sub-problem complexity that means the general tree search approach, if taken in isolation, is obsolete apart from tiny SVRP problems. For example, to obtain the complete arc cost from a vertex representing customer i to a descendent vertex representing customer j in a VRPSD tree search, the following expression is required.

$$c_{ij} + RF_j.2c_{1j} \tag{3.32}$$

where

$$\begin{aligned} RF_j = & \sum_{k=1}^{j-2} P[q_{k+1} = kQ].P[q_{k+2} > kQ \mid q_{k+1} = kQ] + \\ & \sum_{k=0}^{j-2} P[kQ < q_{k+1} < (k+1)Q].P[q_{k+2} > (k+1)Q \mid kQ < q_{k+1} < (k+1)Q]. \end{aligned} \tag{3.33}$$

RF_j denotes the conditional probability of route failure "between" i and j and can be proved in the following way. Given that q_l denotes the total demand on a particular fixed routing sequence up to and including customer l, consider q_i. There are three possibilities relating to comparison between q_i and Q:

(A) $q_i > Q \Longrightarrow q_i > kQ$ for $k \in \{1, \ldots, i-1\}$.

(B) $q_i < Q$.

(C) $q_i = kQ$ for $k \in \{1, \ldots, i-1\}$.

Condition C imposes the only definite failure at j. If the capacity of the vehicle is met exactly at i, the vehicle will travel to j and fail since the demand at j must be greater than zero. Condition B corresponds to the only case where the number of previous route failures is definitely zero and imposes a failure at j if $q_j > Q$. Condition A implies that a route failure has definitely occurred previously and, like condition B, may or may not imply a route failure at j. This is expressed by (3.33).

3.3.2 Dynamic Solution Structures for Stochastic Problems

Stochastic solution structures have been developed to aid the solution search process for problems of indeterminacy. In the case of the PTSA, they also provide an appropriate background for efficient tree search sub-problem construction. Three possible stochastic solution structures are briefly introduced below.

Stochastic Decision Trees: Stochastic Decision Trees (SDT) are direct stochastic extensions of their deterministic counterparts. A SDT requires a tree that branches for each possible decision and each possible realisation of the stochastic variables involved. The tree is then built up of a series of decision nodes and chance nodes where the outcome of a problem scenario is obtained at each leaf of the tree.

Stochastic Dynamic Programming: Dynamic programming requires a decision to be made for each possible state in each stage. These decisions are used to produce a series of optimal solutions, one for each stage. In the stochastic case, decisions depend on what has happened previously. A continuous state space can be developed however, creating a flexibility in terms of the random variables involved, but the process is clearly limited to a relatively small number of states, see [132].

Scenario Aggregation: Scenario aggregation is a dynamic process that is event-based. Clearly, if there exists a finite number of problem scenarios (see Definition 2.5.1), each with a deterministic optimal solution, it is possible to find the most common optimal solution. This corresponds to an approximation of the "stochastic" optimal solution since not all feasible solutions are considered and even if other feasible sets are considered then not all solutions may be implementable at all times. It is possible to successively aggregate a number of similar scenarios and eventually locate an overall optimal solution though, due to a large number of associated deterministic problems, such a method is almost always employed as a heuristic using a simplified objective function.

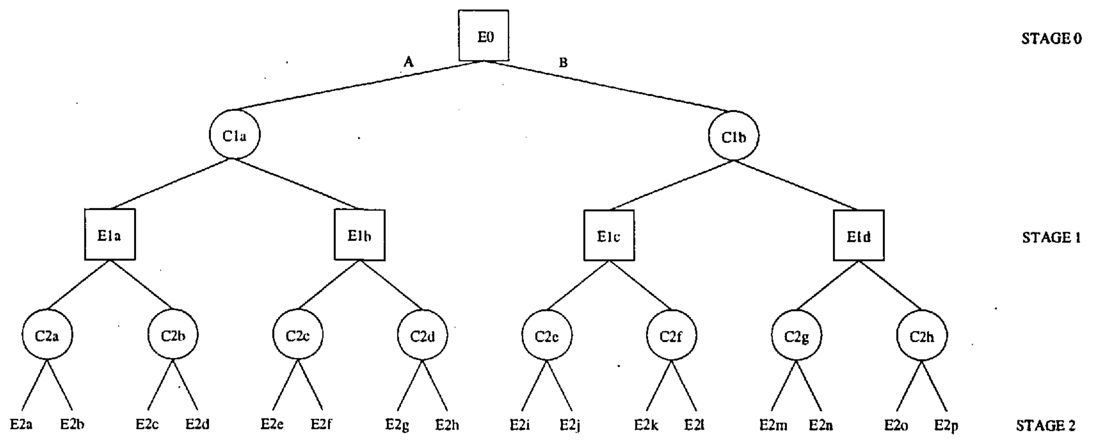

Figure 3.1: A Stochastic Decision Tree

3.3.3 A SDT Approach For The VRPSD

Due to the similarity between the SDT method and the branch and bound approach in general, stochastic decision trees were selected to form the basis of the PTSA. However, an alternative design structure and a parallel binary tree search significantly modify the original method. The reasoning behind such an adaptation is discussed in pedagogical stages below.

Stochastic Decision Trees (SDT)

The stochastic decision tree approach is an extension of the deterministic decision tree procedure. Figure 3.1 displays a SDT for a two stage problem. The square nodes represent decision nodes and the circular nodes represent chance nodes. Starting at the first decision node, representing event $E0$, two decisions are possible, A and B. Following decision A, two events can occur depending on the outcome of chance node $C1a$. Each of these events, $E1a$ and $E1b$, have a corresponding conditional probability of occurrence, $P(E1a \mid E0)$ and $P(E1b \mid E0)$ respectively. Similarly, every decision node in a SDT has an associated probability of occurrence that is conditional on the previous 'decisions' and 'chances', e.g. the probability of $E2j$ occurring is $P(E2j \mid E1c \cap E0)$. Importantly, as the objective of a decision tree is to maximise a function over a series of decisions, all the leaves corresponding to a set of decisions need to be grouped together to form an accurate solution structure and to evaluate the correct solution value. The role of the tree is then realised when all the leaves have been evaluated, the decision sets compared and the best group of decisions chosen.

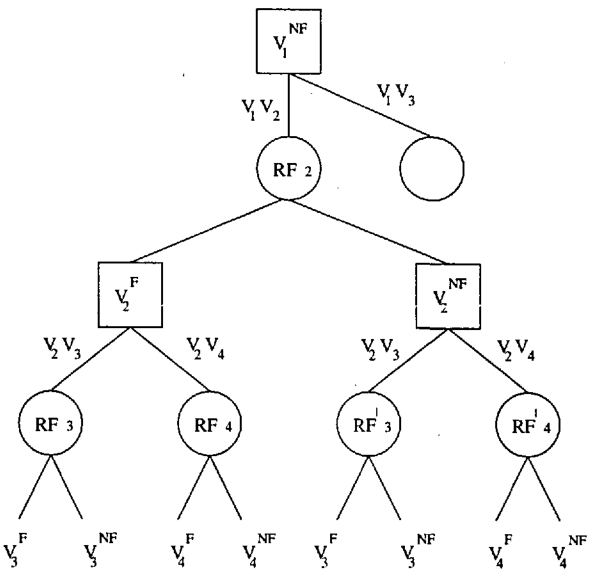

Figure 3.2: A SDT Representation Of The VRPSD

A SDT-VRPSD Approach

For the VRPSD, SDT decision nodes can represent the choice of an arc on a graph contributing to a vehicle route, as in the general tree search method, and SDT chance nodes can correspond to independent events generated after the customer demands have been realised. Clearly however, the latter approach for the VRPSD soon leads to dimensionality problems unless the events themselves are properly limited. One such "reduced" event is the occurrence or non-occurrence of route failure, i.e. the "probability" present at a given chance node will be based on RF_j. The first few branches of such a SDT can be seen in Figure 3.2. Following the decision to branch from v_1 to v_2 for example, the corresponding chance node uses RF_2 to create either a "failure at v_2" (v_2^F) or a "no failure at v_2' (v_2^{NF}). In addition, the first node corresponds to v_1^{NF} and, by definition, $RF_3 \neq RF_3'$ and $RF_4 \neq RF_4'$ since the preceding decision nodes are dissimilar. As in all SDTs, each leaf of the tree corresponds to a feasible solution potentiality.

Due to the nature of such an event-based tree and the lack of an obvious structure to evaluate feasible solutions, the SDT method in its current form still remains computationally difficult to implement. Further modifications are two fold: (i) by constructing the SDT around a general tree search, the "solution grouping" complication can be overcome and the first stage problem can be properly represented, (ii) by introducing a load-based emphasis to SDT branching, second stage recourse representation can be simplified.

SDT Modification I: A Parallel Binary Tree Search

Due to the possibility of recourse, a VRPSD solution refers to a set of planned routes that may not be completed in practice. However, analogous to all tree search methods is the principle that each node should refer to a partial or complete feasible solution, i.e. it is not possible to form any kind of alternative primary tree structure for a VRPSD that could not apply to a VRP. To represent the first-stage problem therefore, a search tree is linked to a SDT, i.e. a group of SDT nodes index a single node on a separate tree.

SDT Modification II: A Load-leaving Approach

Unless events are properly limited, the branching from SDT chance nodes can lead to dimensionality problems. Previously, the SDT was considered in terms of branching according to RF_j however such reduced events are not simplistic enough to envisage a clarified second stage problem construction. In the PTSA therefore, SDT branching occurs according to the possible load a vehicle can have after satisfying the demand of the customer in question, i.e. events equate to alternative *load-leaving* levels. In this way, chance node branching is limited by the capacity of the vehicles involved irrespective of the stochastic demand requirements in the problem and therefore the search is additionally restricted by the distinct number of load-states possible in practice. [Note, it is still not possible to find the best descendent in such a load-based SDT system and use it as a feasible solution construct since the branching process does not operate according to divisions of potential feasible solution subsets. There still remains the need, therefore, for a binary tree search to act as an index to the SDT-type tree that results out of the discrete load-leaving possibilities.]

The PTSA: A Précis

In an effort to find a generic and efficient method for solving the VRPSD exactly, several conditions were classified as being the most influential in designing the algorithmic procedure. Most importantly, at this stage, the structure of the PTSA needed to be such that bounds to the first stage deterministic problem as well as the stochastic recourse second stage problem could be efficiently implemented. Also, however, it was important to find a method that simplified the solution procedure in the absence of a non-complex second stage problem. This has been achieved with the twin methodology highlighted above.

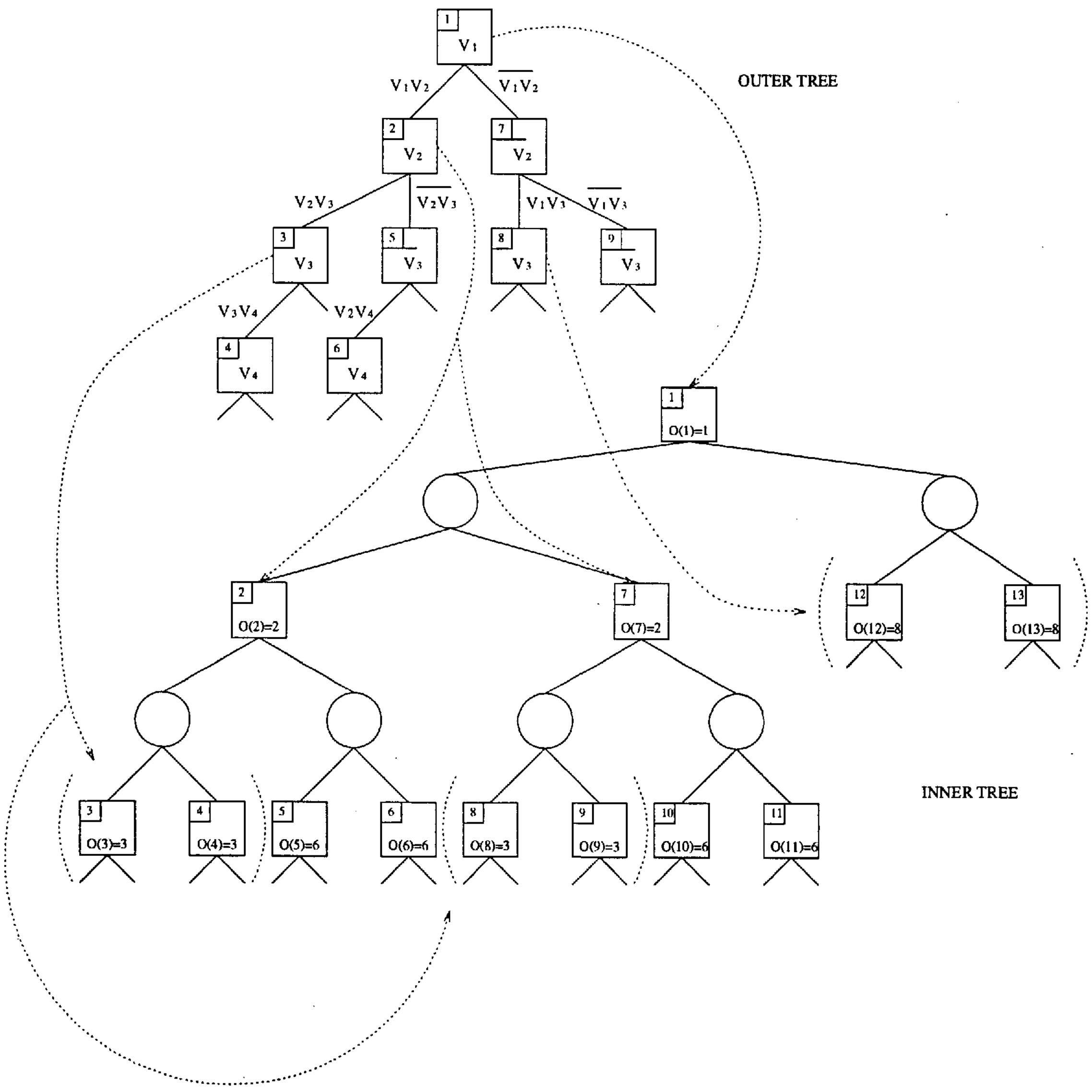

Figure 3.3: Linked Trees In The PTSA

3.4 Algorithmic Details

Although many different data structures are utilised to overcome computational difficulties in the PTSA and additional parameters are included to represent the VRPSD accurately, the basic SDT methodology remains unchanged. The chance nodes and decision nodes present in a stochastic decision tree are established along with the ordinary tree search necessary for representing the first-stage problem with the use of two linked trees.

3.4.1 The Linked Trees

Figure 3.3 displays the pair of search trees associated with the PTSA. One, a binary search decision tree (OUTER tree), relates to the first-stage deterministic problem and the other, a SDT-based tree (INNER tree), relates to the second stage stochastic recourse problem. The OUTER tree conforms exactly to the simple branch and bound method described in Section 3.3.1. Every branch, corresponding to a possible routing segment in the VRPSD, divides the feasible solution subset into two independent sets. On the OUTER tree, each decision node ρ is identified by a customer index $c(\rho)$ that refers to a vertex v_i in the VRPSD, i.e. $c(\rho) = v_i$ or $c(\rho) = \overline{v_i}$ where the latter refers to a not-node. For example, no route constructed below the node $\rho = 6$ can include the arc v_2v_3 however it must include the arcs v_1v_2 and v_2v_4. In addition, each node in the OUTER tree indexes at least one event node on the INNER tree. Such pointers are shown as dotted lines in the diagram.

3.4.2 The INNER Tree

The INNER tree is a modified SDT where chance nodes branch according to the load-leaving level of the next customer following from the demand of the given customer and the load-leaving level of the preceding customer.

Decision Nodes

Each INNER tree decision node λ is denoted by a series of indices:

(i) the load-leaving level, $l(\lambda) \in \{0, \ldots, Q\}$,

(ii) the OUTER tree index, $O(\lambda)$,

(iii) the INNER tree index, $I(\lambda)$,

(iv) the probability $p(\lambda)$ of leaving customer $c(O(\lambda))$ with load $l(\lambda)$ (depending on the characteristics of the parent chance node), and

(v) the recourse cost, $r(\lambda)$.

All these indices are self-explanatory except $I(\lambda)$ which corresponds to the unique index specifying the current node within the set of all possible nodes on the INNER tree that correspond to OUTER tree node $O(\lambda)$. By the definition of the VRPSD, note that $l(\lambda) = Q \ \forall \ \lambda$ where $c(O(\lambda)) = v_1$, i.e. the amount with which a vehicle leaves the depot always corresponds to the vehicle capacity Q, and, $0 \leq l(\lambda) \leq Q - 1 \ \forall \ \lambda$ where $c(I(\lambda)) \neq v_1$, i.e. the load-leaving level of all other INNER tree event nodes must be less than the vehicle capacity since at each customer in a VRPSD a delivery of some kind must be made. [The total number of decision nodes on the INNER tree representing one vertex on the OUTER tree varies depending on the number of customers on the partial route of which $c(O(\lambda))$ is a part and the respective levels of stochasticity of their demand values, see Section 3.4.6.]

Chance Nodes

Each INNER tree chance node has an associated series of branches relating only to the stochastic demand at the customer represented by the descendent vertex and the discrete event load-leaving state at the customer represented by the ascendent vertex. Hence, creating descendent nodes that result from the alternative values of corresponding load-leaving states is a relatively explicit task. Indeed, on the INNER tree, each decision node has a independent probability of occurrence, in comparison with other nodes with the same parent chance node, namely $p(\lambda)$ for a decision node λ with load $l(\lambda)$.

Probabilities of Contribution

It has been shown that feasible solutions of a SDT are formed by combining groups of decision nodes on the INNER tree, hence the need for an OUTER tree index. Clearly, during an INNER tree branching process, it is important to consider the contribution of each decision node to the corresponding node on the OUTER tree. In parallel to each $p(\lambda)$ therefore, a *probability of contribution* is introduced to each decision node on the INNER tree. These probabilities correspond to the total contribution of each INNER tree node λ to a node of the OUTER tree $O(\lambda)$ and are denoted by the index $cp(\lambda)$. For

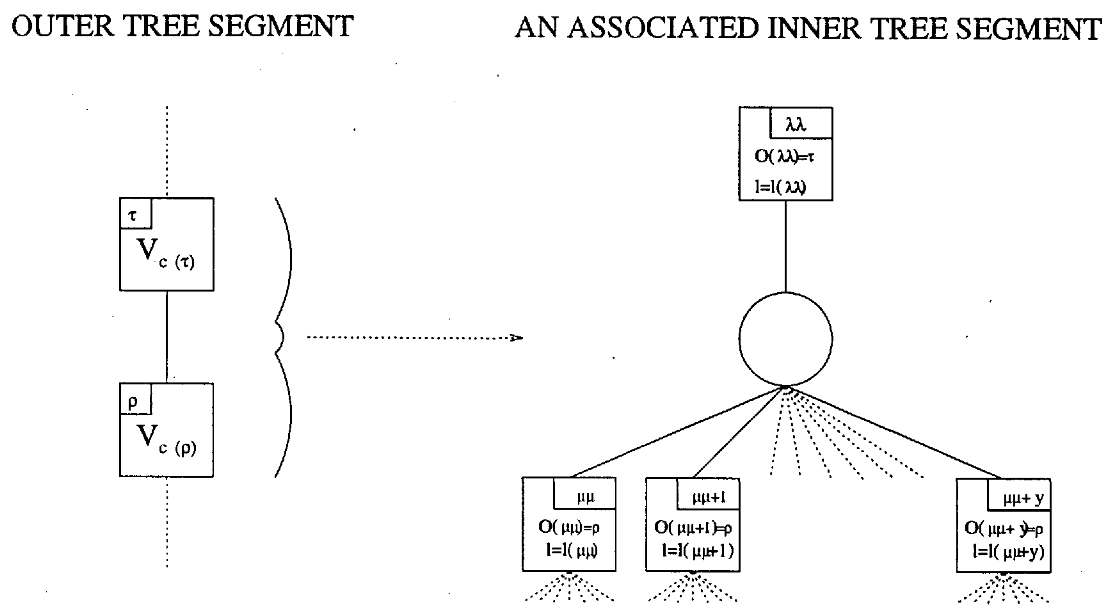

Figure 3.4: The Crossover Between Linked Trees In The PTSA

example, if in Figure 3.3 the probability of an event occurring from each chance node is distributed equally then the probability of contribution of the nodes $\lambda = 2$ and $\lambda = 7$ to the OUTER tree node $\rho = 2$ is 1 in total and 0.5 individually, i.e. $cp(1) = 1$, $cp(2) = 0.5$ and $cp(7) = 0.5$. Similarly, the event nodes $\lambda = 3, 4, 8, 9$ have associated probabilities of contribution of 0.25 each to the decision set (routing segment) that they complete, namely $v_1v_2v_3$, i.e. $p(3) = p(4) = p(8) = p(9) = 0.5$ and $cp(3) = cp(4) = cp(8) = cp(9) = 0.25$.

3.4.3 Creating INNER Tree Nodes

The creation of INNER tree event nodes can be generalised as follows. If there exists a branch from vertex τ to ρ on the OUTER tree, corresponding to customers $c(\tau)$ and $c(\rho)$ in the VRPSD, then this will point to any number of branches on the INNER tree that connect an associated node $\lambda\lambda$, where $O(\lambda\lambda) = \tau$, to $(y+1)$ other nodes (starting with $\mu\mu$) such that $O(\mu\mu + \mu) = \rho \; \forall \; \mu = 0, \ldots, y$. One such branch is shown in Figure 3.4.

Now, if $P(l', l'', d, Q)$ represents the probability of a load-leaving level, l'', being realised after entering a node with a vehicle of capacity Q and current load-level l' and encountering a demand of d (see Figure 3.5), then:

$$P(l', l'', d, Q) = \begin{cases} \text{Prob}[l' - d = l''] & \text{if } l' > d, \\ \text{Prob}[Q + l' - d = l''] & \text{if } l' \leq d. \end{cases} \tag{3.34}$$

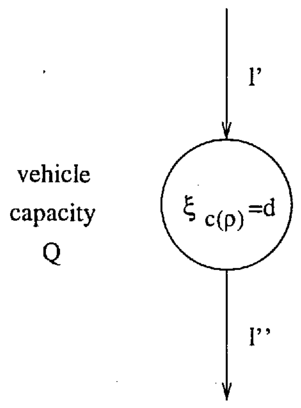

Figure 3.5: The Factors Affecting The Load-Leaving Level of an INNER Tree Node

In the generalisation therefore, each chance node will split the new decision nodes, $\mu\mu + \mu \; \forall \; \mu = 0, \ldots, y$ according to the following probabilities:

$$p(\mu\mu + \mu) = P(l(\lambda\lambda), ll, \xi_{c(\rho)}, Q) \tag{3.35}$$

for

$$\varepsilon_{c(\rho)} \in \xi_{c(\rho)} \tag{3.36}$$

$$ll = 0, \ldots, Q-1 \tag{3.37}$$

where

$$P(l(\lambda\lambda), ll, \xi_{c(\rho)}, Q) = \begin{cases} \text{Prob}[l(\lambda\lambda) - \xi_{c(\rho)} = ll] & \text{if } l(\lambda\lambda) > \xi_{c(\rho)}, \\ \text{Prob}[Q + l(\lambda\lambda) - \xi_{c(\rho)} = ll] & \text{if } l(\lambda\lambda) \leq \xi_{c(\rho)}. \end{cases} \tag{3.38}$$

Clearly, the following sum holds:

$$\sum_{ll=0}^{Q-1} P(l(\lambda\lambda), ll, \xi_{c(\rho)}, Q) = 1. \tag{3.39}$$

There will be $y+1 \leq Q$ discrete events that result from the branching from node $\lambda\lambda$, each corresponding to an alternative load-leaving level of customer $c(\rho)$, such that:

$$y = \sum_{k=0}^{Q-1} f(k), \tag{3.40}$$

where

$$f(k) = \begin{cases} 0 & \text{if } P(l(\lambda\lambda), k, \xi_{c(\rho)}, Q) = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{3.41}$$

Hence, the probability of contribution at each decision node $\mu\mu + \mu \; \forall \; \mu = 0, \ldots, y$ is given by $cp(\mu\mu + \mu) = cp(\lambda\lambda)p(\mu\mu + \mu)$ where $cp(\lambda\lambda) \geq cp(\mu\mu + \mu) \; \forall \; \mu = 0, \ldots, y$ since:

$$cp(\lambda\lambda) = \sum_{\mu=0}^{y} cp(\mu\mu + \mu). \tag{3.42}$$

Clearly, each branch on the INNER tree has a finite cost that is dependent on the load-leaving states relating to the two corresponding adjacent nodes. These recourse costs $r(\mu\mu + \mu)\ \forall\ \mu = 0, \ldots, y$ are given by:

$$r(\mu\mu + \mu) = \begin{cases} 0 & \text{if } l(\mu\mu + \mu) \leq l(\lambda\lambda), \\ 2.c_{1c(\rho)} & \text{otherwise.} \end{cases} \tag{3.43}$$

Each node then contributes a cost of $cp(\mu\mu+\mu)r(\mu\mu+\mu)$ to OUTER tree node $O(\mu\mu+\mu)$.

3.4.4 Linkage and Crossover

The dynamic linkage between a node on the OUTER tree and a series of nodes on the INNER tree is computationally non-trivial. A fairly complicated search is required to locate and branch from ascendent nodes on the INNER tree. For example, OUTER tree node $\rho = 3$ in Figure 3.3 corresponds to four nodes $\lambda = 3, 4, 8, 9$ on the INNER tree with respective load-leaving levels given by $l(\lambda)$. A search would require first the location of all those INNER tree nodes that correspond to node $\rho = 2$ on the OUTER tree and second, a series of branches to form the new load-leaving nodes.

3.4.5 The PTSA Computational Procedure

The algorithm adopts a nested branching scheme and can be summarised as follows. At each OUTER tree decision-node ρ, which has a customer index $c(\rho)$ and a parent node τ, an arc $(v_{c(\tau)}, v_{c(\rho)})$ is assigned to a current partial route k. If a feasible first stage set of routes containing k cannot be found then backtracking occurs; otherwise the current solution value z^ρ is updated, i.e. $z^\rho \leftarrow z^\rho + c_{c(\tau)c(\rho)}$, and the search continues by transferring to the INNER tree. The set of INNER tree nodes, Λ^τ, that has previously been developed and used to index the parent of ρ, are located. Full branching occurs on the INNER tree from all nodes in Λ^τ to generate a series of new nodes linked to ρ, namely Λ^ρ, with alternative non-unique load-leaving levels. The current solution value z^ρ is updated accordingly. If z^ρ is feasible and greater than the best incumbent VRPSD solution value z^*, the node ρ is fathomed; otherwise the search is transferred to the OUTER tree and

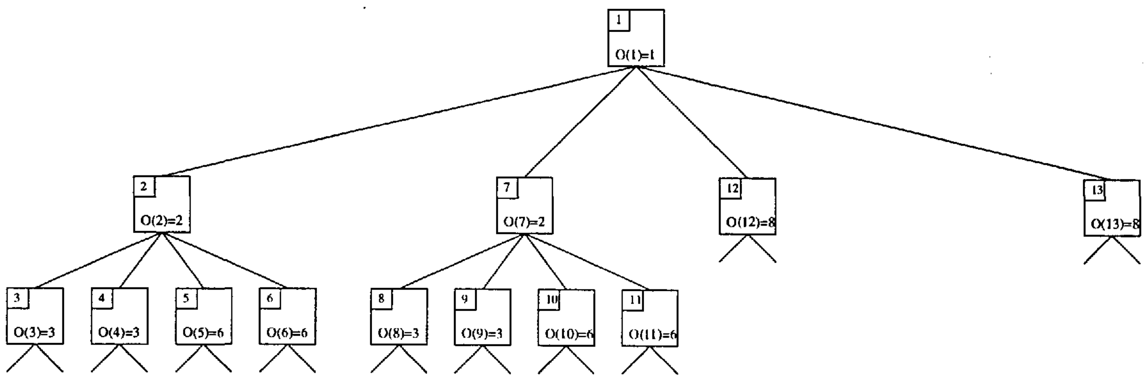

Figure 3.6: A Reduced INNER Tree

the iterative branching process continues.

The OUTER tree is based on the depth-first branching scheme and requires a branching strategy to aid the search process (see later). The INNER tree is based on a depth-first branching scheme but, to conserve computational storage space, no chance nodes are explicitly generated. Instead, the branching from decision nodes in the INNER tree to alternative descendent nodes is considered without the explicit extension of a series of chance nodes. All the relevant mechanics of the INNER tree procedure remain in place since the properties of each chance node can be incorporated into their ascendent event nodes. For example, the INNER tree segment shown in Figure 3.3 reduces to the segment shown in Figure 3.6 without a concurrent loss of information.

To highlight the full procedure in detail, the algorithm is given in two parts, relating to the OUTER tree (Algorithm 3.1) and the INNER tree (Algorithm 3.2). The latter is connected to the former following the fathoming test (2a) in Algorithm 3.1.

3.4.6 Computational Complexity and Rebranching

In its current form, the PTSA is limited in those problems it can solve by two factors. Firstly, bounds are required to speed up the search process and secondly, the number of INNER tree nodes that must be generated to find a feasible solution to the VRPSD limits the performance of the PTSA due to high computational storage requirements. If each customer in the VRPSD has δ_i discrete demand values and $m = 1$, the number of INNER tree nodes required to find a feasible solution, $|I^z|$, is as follows:

$$|I^z| = 1 + \sum_{i=2}^{n-1} \left(\prod_{j=2}^{i} \delta_j \right) \tag{3.44}$$

Algorithm 3.1 The PTSA: OUTER tree

```
1 Initialisation
  initialise the cost matrix $C = (c_{ij})$;
  set the OUTER tree node from which branching will commence to $\rho = 1$, $c(\rho) = 1$;
  set the level of the tree to $L_O = 1$;
  set $z^\rho = 0$ and $z^* = \infty$;
  set the vertices available to branch to as $V' = \{v_2, \ldots, v_n\}$;
  initialise the first INNER tree node $\lambda = 1$, $O(\lambda) = 1$;

2 Branching
  while ($V' \neq \emptyset$) and ($z^\rho \leq z^*$) do
    choose a customer $v_i \in V'$ to branch to;
    branch and number the new node accordingly, i.e. $\rho = \rho + 1$, $c(\rho) = i$;
    set $\tau$ as the parent of $\rho$
    set $L_O = L_O + 1$;
    calculate $z^\rho$ (first stage), i.e. $z^\rho \leftarrow z^\rho + c_{c(\tau)c(\rho)}$;
    2a Fathoming Test
    if ($z^\rho \leq z^*$) then
      complete INNER Tree Branching (Algorithm 3.2);
      calculate $z^\rho$ (second stage);
      2b Feasibility and Optimality Test
      if ($z^\rho$ is feasible) and ($z^\rho < z^*$) then
        set $z^* = z^\rho$;
        record routes;
      end if
    end if
    update $C$ and $V'$ for node $\rho$;
  end while

3 Backtracking
  repeat
    set $L_O = L_O - 1$;
    if $L_O = 0$ then
      return $z^*$;
      return routes;
      stop;
    end if
  until (alternative not node has not been examined)
  backtrack to new node;
  update $C$ and $V'$;
  goto 2;
```

Algorithm 3.2 The PTSA: INNER tree

```
1 Initialisation
  set the INNER tree node from which tree search will commence to $\lambda = 1$;
  set the level of the tree to $L_I = 1$;
  set $\lambda_0$ to be the number of nodes generated on the INNER tree;

2 Initial Search
  while ($O(\lambda) \neq \tau$) do
    if ($\lambda$ corresponds to the last node on level $L_I$) then
      set $L_I = L_I + 1$;
    end if
    goto next INNER tree node on level $L_I$;
    set current node as $\lambda$;
  end while

3 Branching
  set $Y$ to be the set of load-leaving possibilities from node $\lambda$;
  while ($Y \neq \emptyset$) do
    choose a load-leaving possibility $l$ to branch to;
    branch and number the new node accordingly, i.e. $\lambda = \lambda_0 + 1$;
    set $L_I = L_I + 1$;
    set $O(\lambda) = \rho$;
    set $l(\lambda) = l$;
    calculate $p(\lambda)$, $cp(\lambda)$ and $r(\lambda)$;
    calculate $z^\rho$ (second stage), i.e. $z^\rho \leftarrow z^\rho + cp(\lambda)r(\lambda)$;
  end while
  return to parent node;
  set $L_I = L_I - 1$;

4 Lateral Search
  if (current node corresponds to the last node on level $L_I$) then
    stop;
  else
    repeat
      goto next INNER tree node on level $L_I$;
      set current node as $\lambda$;
      if ($O(\lambda) = \tau$) then
        goto 3;
      end if
    until (all INNER tree nodes on level $L_I$ have been examined)
  end if
  stop;
```

If $m > 1$, the maximum value of $|I^z|$ for all feasible solutions involves one route with $n - m - 1$ customers, another route of two customers and, if $m > 2$, another $m - 2$ single customer routes. More specifically:

$$|I^z| = 1 + \sum_{i=2}^{n-m-1} \left(\prod_{j=2}^{i} \delta_j \right) + \sum_{i=n-m-1}^{n-1} \left(\prod_{j=2}^{i} \delta_j \right) = 1 + \sum_{i=2}^{n-1} \left(\prod_{j=2}^{i} \delta_j \right) + \prod_{j=2}^{n-m-1} \delta_j \quad (3.45)$$

Given that several indices are stored at each INNER tree node, storage requirements can become enormous, e.g. if $n = 11$, $m = 2$ and $\delta_i = 5 \; \forall \; i$, $|I^z| = 12{,}207{,}031$ and if $n > 12$, $|I^z| > 1$ billion. In its present form, therefore, limits would need to be set on δ_i, m or n.

Resulting from a need to reduce PTSA storage requirements even further, a branching extension is added to the INNER tree. An aggregation process is established where the load-leaving level of each INNER tree node contributes to new nodes formed in a *re-branching* procedure. Once full INNER tree branching has occurred, all recently constructed nodes (indexing ρ on the OUTER tree) are replaced by a series of new nodes, each corresponding to a unique load-leaving level. The explicit procedure requires one level of the INNER tree to be ascended before all the nodes that have just been formed, now below the active level, are deleted. Following this, the first node on the current level is located and a new branching scheme is incorporated where each new node, a son of the current node, corresponds to a load-leaving level. Each of the "re-branched" nodes then corresponds to a unique load-leaving level, thereby limiting the number of nodes retained on each level of the INNER tree to Q. $|I^z|$ is now given by $|I^z| = 1 + Q(\max_i[\delta_i] + n - 1)$. In the earlier example, with any value of m, $n = 11$, $\delta_i = 5$ and $Q = 100, |I^z| = 1501$. A new version of the PTSA (Algorithm 3.2) that incorporates rebranching is given in Algorithm 3.3. The original branching process now involves forming a series of temporary nodes which are later deleted. The contribution of these nodes, in terms of probability and cost, is stored so that a series of permanent nodes can be created in their place.

3.5 Test Problems

The following set of test problems were formed to assess the performance of the PTSA, together with the impact of a number of parameters. They derive from similar tests used by Gendreau *et al* [97] for the VRPSCD.

Algorithm 3.3 The PTSA: INNER tree (with rebranching)

```
1 Initialisation
  as Algorithm 3.2

2 Initial Search
  as Algorithm 3.2

3 Branching
  set $Y$ to be the set of load-leaving possibilities from node $\lambda$;
  while ($Y \neq \emptyset$) do
    choose a load-leaving possibility $l$ to branch to;
    branch and number the new node accordingly, i.e. $\lambda = \lambda_0 + 1$;
    set $L_I = L_I + 1$;
    set $O(\lambda) = \rho$;
    set $l(\lambda) = l$;
    calculate $p(\lambda)$,
    calculate and record $cp(\lambda)$ and $r(\lambda)$ for the relevant $l(\lambda)$;
    calculate $z^\rho$ (second stage), i.e. $z^\rho \leftarrow z^\rho + cp(\lambda)r(\lambda)$;
  end while
  return to parent node;
  set $L_I = L_I - 1$;

4 Lateral Search
  if (current node corresponds to the last node on level $L_I$) then
    goto 5;
  else
    repeat
      goto next INNER tree node on level $L_I$;
      set current node as $\lambda$;
      if ($O(\lambda) = \tau$) then
        goto 3;
      end if
    until (all INNER tree nodes on level $L_I$ have been examined)
  end if

5 Rebranching
  backtrack to earliest node on $L_I$ where $O(\lambda) = \rho$;
  delete all other nodes on $L_I$ where $O(\lambda) = \rho$;
  rebranch from $\lambda$ for all load-leaving levels using recorded values;
  stop;
```

3.5.1 Problem Generation

Ten test problems were created as follows: n vertices v_i are generated in the $[1, 99]^2$ square according to a continuous uniform distribution and each c_{ij} is then computed as the Euclidean distance between v_i and v_j. Customers are arbitrarily assigned to one of three groups with discrete demands of mean $\mu_i = 5, 10, 15$ respectively. A uniform distribution is utilised and δ_i values of demand are randomly generated in the intervals $[1, 9]$, $[5, 15]$ and $[10, 20]$ where the probability of each customer demand value ε_i^l arising is identical, i.e. $p_i^l = 1/\delta_i \ \forall \ i$ and l. For all tests, the ten problems were solved for each setting of the associated parameters.

3.5.2 Vehicle Utilisation

During the testing process, problem tightness is controlled by selecting the vehicle capacity Q (integer) in order to achieve a pre-specified expected 'vehicle utilisation coefficient' U, where:

$$U = \frac{\sum_{i=2}^{n} \sum_{l=1}^{\delta_i} p_i^l \varepsilon_i^l}{mQ}. \tag{3.46}$$

3.6 An Example of the PTSA

To further illustrate the PTSA, a worked example is now presented. The example involves a simple seven customer, one vehicle problem. The input data describing the example is shown in Table 3.1. The depot is located at $(17, 9)$, vehicle capacity Q is set at 47 and $\delta_i = 7 \ \forall \ i$, i.e. $p_i^l = 1/7 \ \forall \ i$ and l.

3.6.1 Branching on the INNER Tree

Following an initial branch on the OUTER tree from the depot to customer v_8, i.e. $\rho = 1$ to $\rho = 2$ where $c(1) = 1$ and $c(2) = 8$, initial branching takes place on the INNER tree according to the demand set ξ_8 as shown in Table 3.1. Figure 3.7 displays the resulting tree and the characteristics of the relevant nodes are shown in Table 3.2. Following the rebranching procedure, the (temporary) nodes are deleted and new permanent nodes are formed. In this case, since branching previously took place from a single node ($\lambda = 1$), INNER tree structure remains identical.

i	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	ϵ_i^4	ϵ_i^5	ϵ_i^6	ϵ_i^7
2	24	5	5	6	7	8	12	13	15
3	83	69	5	6	7	9	10	13	15
4	51	20	1	2	3	4	5	6	9
5	94	37	1	2	3	5	7	8	9
6	42	41	11	12	14	16	17	19	20
7	26	75	11	12	14	16	17	18	20
8	60	93	6	8	9	11	12	13	14

Table 3.1: Input Data

λ	$O(\lambda)$	$l(\lambda)$	$p(\lambda)$	$cp(\lambda)$	$r(\lambda)$
1	1	47	-	1	0
2	2	33	0.1428570	0.1428570	0
3	2	34	0.1428570	0.1428570	0
4	2	35	0.1428570	0.1428570	0
5	2	36	0.1428570	0.1428570	0
6	2	38	0.1428570	0.1428570	0
7	2	39	0.1428570	0.1428570	0
8	2	41	0.1428570	0.1428570	0

Table 3.2: INNER Tree Data I

λ	$O(\lambda)$	$l(\lambda)$	$p(\lambda)$	$cp(\lambda)$	$r(\lambda)$
9	3	13	0.1428570	0.0204081	0
10	3	15	0.1428570	0.0204081	0
11	3	16	0.1428570	0.0204081	0
12	3	17	0.1428570	0.0204081	0
13	3	19	0.1428570	0.0204081	0
14	3	21	0.1428570	0.0204081	0
15	3	22	0.1428570	0.0204081	0
16	3	14	0.1428570	0.0204081	0
17	3	16	0.1428570	0.0204081	0
18	3	17	0.1428570	0.0204081	0
19	3	18	0.1428570	0.0204081	0
20	3	20	0.1428570	0.0204081	0
21	3	22	0.1428570	0.0204081	0
22	3	23	0.1428570	0.0204081	0
23	3	15	0.1428570	0.0204081	0
24	3	17	0.1428570	0.0204081	0
25	3	18	0.1428570	0.0204081	0
26	3	19	0.1428570	0.0204081	0
27	3	21	0.1428570	0.0204081	0
28	3	23	0.1428570	0.0204081	0
29	3	24	0.1428570	0.0204081	0
30	3	16	0.1428570	0.0204081	0
31	3	18	0.1428570	0.0204081	0
32	3	19	0.1428570	0.0204081	0
33	3	20	0.1428570	0.0204081	0
34	3	22	0.1428570	0.0204081	0
35	3	24	0.1428570	0.0204081	0
36	3	25	0.1428570	0.0204081	0
37	3	18	0.1428570	0.0204081	0
38	3	20	0.1428570	0.0204081	0
39	3	21	0.1428570	0.0204081	0
40	3	22	0.1428570	0.0204081	0
41	3	24	0.1428570	0.0204081	0
42	3	26	0.1428570	0.0204081	0
43	3	27	0.1428570	0.0204081	0
44	3	19	0.1428570	0.0204081	0
45	3	21	0.1428570	0.0204081	0
46	3	22	0.1428570	0.0204081	0
47	3	23	0.1428570	0.0204081	0
48	3	25	0.1428570	0.0204081	0
49	3	27	0.1428570	0.0204081	0
50	3	28	0.1428570	0.0204081	0
51	3	21	0.1428570	0.0204081	0
52	3	23	0.1428570	0.0204081	0
53	3	24	0.1428570	0.0204081	0
54	3	25	0.1428570	0.0204081	0
55	3	27	0.1428570	0.0204081	0
56	3	29	0.1428570	0.0204081	0
57	3	30	0.1428570	0.0204081	0

Table 3.3: INNER Tree Data II

λ	$O(\lambda)$	$l(\lambda)$	$cp(\lambda)$	$r(\lambda)$
9	3	13	0.0204081	0
10	3	14	0.0204081	0
11	3	15	0.0408162	0
12	3	16	0.0612244	0
13	3	17	0.0612244	0
14	3	18	0.0816325	0
15	3	19	0.0816325	0
16	3	20	0.0612244	0
17	3	21	0.1020406	0
18	3	22	0.1020406	0
19	3	23	0.0816325	0
20	3	24	0.0816325	0
21	3	25	0.0612244	0
22	3	26	0.0204081	0
23	3	27	0.0612244	0
24	3	28	0.0204081	0
25	3	29	0.0204081	0
26	3	30	0.0204081	0

Table 3.4: INNER Tree Data III

λ	$O(\lambda)$	$l(\lambda)$	$p(\lambda)$	$cp(\lambda)$	$r(\lambda)$
27	4	0	0.1428570	0.0029154	0
28	4	3	0.1428570	0.0029154	0
29	4	4	0.1428570	0.0029154	0
30	4	6	0.1428570	0.0029154	0
31	4	7	0.1428570	0.0029154	0
32	4	8	0.1428570	0.0029154	0
33	4	45	0.1428570	0.0029154	178.4

Table 3.5: INNER Tree Data IV

λ	$O(\lambda)$	$l(\lambda)$	$cp(\lambda)$	$r(\lambda)$
27	4	0	0.0087463	0
28	4	1	0.0116618	0
29	4	2	0.0145772	0
30	4	3	0.0233235	0
31	4	4	0.0262390	0
32	4	5	0.0291544	0
33	4	6	0.0437316	0
34	4	7	0.0466471	0
35	4	8	0.0583089	0
36	4	9	0.0670552	0
37	4	10	0.0641397	0
38	4	11	0.0670552	0
39	4	12	0.0787170	0
40	4	13	0.0641397	0
41	4	14	0.0699706	0
42	4	15	0.0641397	0
43	4	16	0.0553944	0
44	4	17	0.0524780	0
45	4	18	0.0437316	0
46	4	19	0.0291544	0
47	4	20	0.0262390	0
48	4	21	0.0174927	0
49	4	22	0.0145772	0
50	4	23	0.0087463	0
51	4	24	0.0058309	0
52	4	25	0.0029154	0
53	4	45	0.0029154	178.4
54	4	46	0.0029154	178.4

Table 3.6: INNER Tree Data V

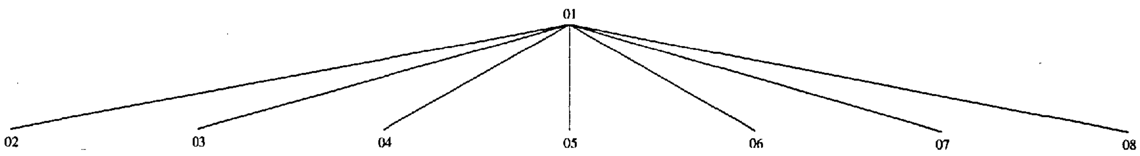

Figure 3.7: Temporary INNER Tree Nodes Corresponding to OUTER Tree Arc (1, 8)

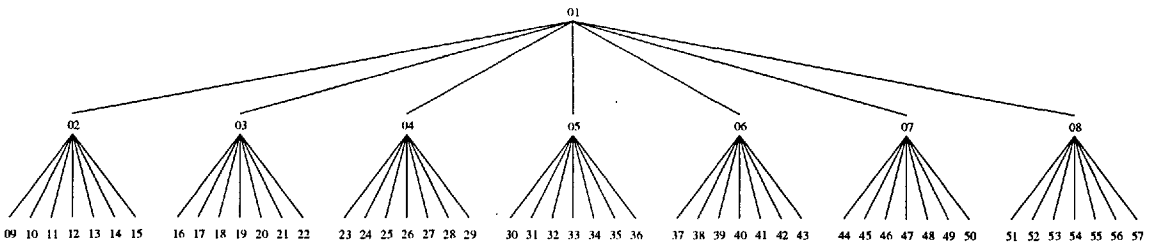

Figure 3.8: Temporary INNER Tree Nodes Corresponding to OUTER Tree Arc (1, 8, 7)

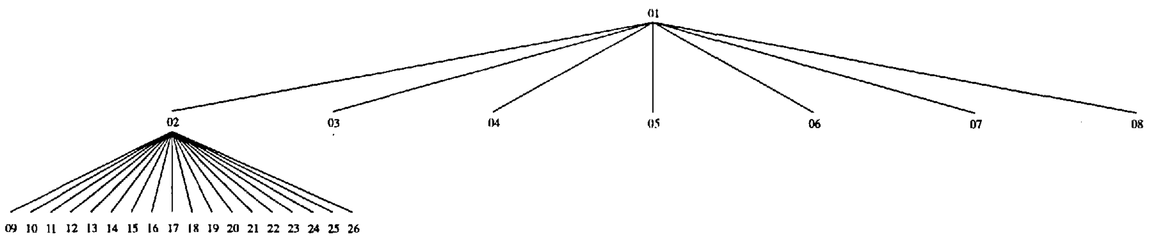

Figure 3.9: Rebranched INNER Tree Nodes Corresponding to OUTER Tree Arc (1, 8, 7)

Now, consider another branch on the OUTER tree, from node $\rho = 2$ to node $\rho = 3$ where $c(3) = 7$. Temporary branching takes place on the INNER tree as shown in Figure 3.8. The characteristics of the newly formed nodes are shown in Table 3.3. Following the next rebranching procedure, the new INNER tree is as shown in Figure 3.9 and the indices of the new nodes are shown in Table 3.4. Now, consider another branch on the OUTER tree, from node $\rho = 3$ to node $\rho = 4$ where $c(4) = 3$. The newly formed temporary nodes total 126. Consider only the branch from node $\lambda = 9$. Seven nodes are formed and their characteristics, according to the demand set ξ_3 given in Table 3.1, are shown in Table 3.5. Notice that a recourse cost from the node $\lambda = 33$ contributes to the relevant OUTER tree node and vehicle load increases following a "trip back to the depot" to $l(33) = 45$. Following the next rebranching procedure, the indices of the new nodes on level $l_I = 4$ of the INNER tree are shown in Table 3.6.

3.6.2 Branching on the OUTER Tree

Figure 3.10 displays the initial construction of the OUTER tree for the example given above. The first few nodes correspond to the $1-8-7-3$ branch utilised in Section 3.6.1 to describe the INNER tree.

Each INNER tree node λ contributes a recourse cost $r(\lambda)$, to the OUTER tree node $O(\lambda)$. The total recourse cost attributed to each OUTER tree node is therefore given by the following.

$$\sum_{\lambda \in \Lambda^{\rho}} r(\lambda) cp(\lambda) \tag{3.47}$$

where Λ^{ρ} corresponds to the set of all INNER tree nodes for which $O(\lambda) = \rho$.

Figure 3.10 corresponds to the formation of the first three leaves of the OUTER tree and Table 3.7 displays all the relevant OUTER tree node information. This includes fixed arc costs, recourse costs, obtained by (3.47), and overall routing costs given by z^{ρ}. Each of the final OUTER tree nodes, $\rho = 8, 11, 15$, include an additional cost referring to the final trip back to the depot so that the feasible solution cost in each case is actually $425.4, 466.7$ and 427.5 respectively. The best feasible solution obtained after the branching process shown in the figure therefore corresponds to node $\rho = 8$. If the PTSA is run to optimality, using approximately 20 seconds of computational time and generating $12,219$ and $1,580,267$ OUTER tree and INNER tree nodes respectively, $z^{*} = 368.5$ and the optimal route is $1-4-5-3-8-7-6-2-1$.

3.7 Computational Results

The paired tree search algorithm has been coded in FORTRAN and run on a Silicon Graphics Workstation Indigo R4000 (100MHz). This CPU is roughly 25% faster than a 100MHz Pentium, see Dongarra [67].

The results for the basic version of the PTSA for P4 are split into two sections. Firstly, we consider the overall results. The effect that a number of parameters, e.g. n, m, U and δ_i, have on the performance of the algorithm (in terms of computational time and INNER/OUTER tree node generation) is discussed, together with an evaluation of particular PTSA components such as fathoming and rebranching. Secondly, we consider possible branching strategies and how they can be used to enhance the search process.

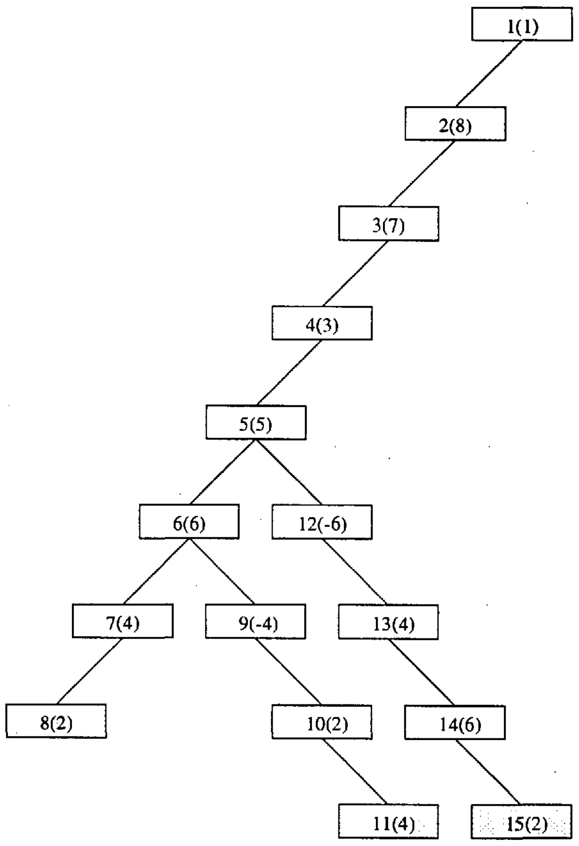

Figure 3.10: Example OUTER Tree

Node ρ	Customer $c(\rho)$	Fixed Cost $\equiv c_{ij}$	Recourse Cost $\sum cp(\lambda)r(\lambda)$	Total $\equiv c_{ij} + \sum cp(\lambda)r(\lambda)$	Overall Cost z^ρ
1	1	0	0	0	0
2	8	94.4	0	94.4	94.4
3	7	38.5	0	38.5	132.9
4	3	57.3	1.0	58.3	191.2
5	5	33.8	18.1	51.9	243.1
6	6	52.2	62.3	114.5	357.6
7	4	22.8	5.4	28.2	385.8
8	2	30.9	0.6	31.5	417.3
10	2	40.2	1.8	42.0	399.6
11	4	30.9	0.5	31.4	431.0
13	4	46.2	14.6	60.8	303.9
14	6	22.8	51.9	74.7	378.6
15	2	37.6	3.2	40.8	419.4

Table 3.7: OUTER Tree Data

3.7.1 Results I: Parameters and PTSA Components

Numerical experiments were completed to assess the effect of four problem characteristics: problem size - n, vehicle utilisation - U, vehicle number - m and the number of discrete demand values per customer - δ_i. In addition, two specific algorithmic components, fathoming (F) and re-branching (R) were tested by examining four cases of the PTSA.[2] These are given as follows ($\overline{X}$ refers to the omission of procedure X):

- complete enumeration and no-rebranching - $(\overline{F}, \overline{R})$,
- complete enumeration and rebranching - $(\overline{F}, R)$,
- fathoming and no-rebranching - $(F, \overline{R})$,
- fathoming and re-branching - (F, R).

The ten test problems, described in Section 3.5, were run for a number of problem instances where $n = 5, 6, 7$, $m = 1, 2, 3$, $\delta_i = 5, 6, 7$ and $U = 0.5, \ldots, 1$, i.e. $(10 \times 3 \times 3 \times 3 \times 6) = 1620$ separate stochastic problems were solved. Tables 3.8, 3.9 and 3.10 display the average values of these tests and compare the number of OUTER tree nodes used ($|O|$), the number of INNER tree nodes used ($|I|$) and the computation time in seconds over the four cases of PTSA. The values in each table are averages of 540 problem instances.

Complete Enumeration

Testing the PTSA against complete enumeration provided not only an indication of the performance of the algorithm but also of the computational complexity of the problem in general. Clearly for $(\overline{F}, \overline{R})$, the number of OUTER tree nodes is independent of the customer demand characteristics (δ_i and U), and only alters with the number of vehicles (m) and the number of vertices (n). [The reason for the slight variation in the value of $|O|$ in the table is due to the requirement of choosing an integer Q which can slightly vary for different demand characteristics.] For complete enumeration, $|O|$, $|I|$ and cpu increase most sharply over n. As expected, this implies that problem size is the biggest hindrance to the performance of the algorithm at this stage. Notice also that in cases where $m > 1$

[2]The PTSA with no fathoming both on the INNER and OUTER trees (complete enumeration) was tested to highlight the size of stochastic problems of this kind and to highlight the complexity involved in evaluating all problem scenarios.

		$(\overline{F},\overline{R})$			$(\overline{F},R)$			$(F,\overline{R})$			(F,R)		
		$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*
n	5	141.7	65815.3	0.95	141.7	8776.0	0.23	135.6	48252.8	0.78	135.6	7157.7	0.22
	6	1054.3	3150685.6	40.93	1054.3	76974.0	1.24	919.3	1656476.3	23.38	919.3	53202.0	0.95
	7	8340.2	161491238.3	2116.74	8340.2	706522.3	10.75	6281.6	49509291.1	733.39	6281.8	377391.7	6.28

Table 3.8: Results for Variable n ($m = 1, 2, 3$, $\delta_i = 5, 6, 7$, $U = 0.5, \ldots, 1$)

		$(\overline{F},\overline{R})$			$(\overline{F},R)$			$(F,\overline{R})$			(F,R)		
		$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*
m	1	1069.7	17238804.8	221.87	1069.7	170012.7	2.44	933.1	6438958.4	96.70	933.2	103281.9	1.64
	2	4092.1	70007636.9	926.88	4092.1	357283.5	5.41	3117.0	19854910.3	296.15	3117.1	181360.9	3.04
	3	8430.0	154455651.0	2017.18	8430.0	534301.3	8.63	6289.0	48835855.8	716.24	6288.9	298614.7	5.27

Table 3.9: Results for Variable m ($n = 5, 6, 7$, $\delta_i = 5, 6, 7$, $U = 0.5, \ldots, 1$)

		$(\overline{F},\overline{R})$			$(\overline{F},R)$			$(F,\overline{R})$			(F,R)		
		$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*
δ_i	5	3825.9	18480390.9	231.61	3825.9	254911.0	3.97	2921.9	6002570.2	88.10	2921.9	139000.4	2.42
	6	3873.9	54037498.2	715.23	3873.9	322450.9	4.98	2968.8	16985562.9	246.48	2968.8	177219.2	2.98
	7	3847.1	131623427.3	1728.78	3847.1	386467.2	5.86	2959.6	40349297.2	602.47	2959.7	213225.1	3.55

Table 3.10: Results for Variable δ_i ($n = 5, 6, 7$, $m = 1, 2, 3$, $U = 0.5, \ldots, 1$)

problem complexity increases significantly. This is due to the huge increase in the number of routes that need to evaluated in the VRPSD where, unlike the VRP, the direction of travel also must be taken into consideration.

Rebranching

Rebranching effects the amount of INNER tree nodes that are required to find an optimal solution. In the worst case, i.e. where $\delta_i = 7$, the reduction between complete enumeration without rebranching and complete enumeration with rebranching can be as much as over 130 million INNER tree nodes. If fathoming is in place, such a reduction totals over 40 million nodes. The effect on computational time is no less significant. For example, comparing *cpu* in the cases where algorithm construction was given by $(\overline{F}, \overline{R})$ to $(\overline{F}, R)$ and $(F, \overline{R})$ to (F, R) respectively, the reduction was by as much as 99.6% and 99.4%.

Problem Parameters

In general, using the complete algorithm (F, R), problem size is the most important parameter. The value of n effects $|O|$, $|I|$ and *cpu* almost in equal measure. The parameter of secondary importance is m. In the main, this parameter effects $|O|$ and *cpu*. The number of discrete demand values (δ_i) predominantly effects $|I|$ but also slightly effects computational time.

An interesting part of the PTSA can be highlighted in the consideration of U. Table 3.11 displays the changing values of $|O|$, $|I|$ and *cpu*, using (F, R), as U increases for problems where $n = 6$. As U increases and the problem becomes increasingly harder to solve in the conventional sense, computational time decreases. This unusual result is due to the manner in which the PTSA constructs INNER tree nodes. This is highlighted by considering the case where $U = 0.9$ and $U = 1.0$. At the point where $U = 1.0$, the amount of INNER tree nodes generated is less than the case where $U = 0.9$, even though $|O|$ has increased. In this case, the problem is harder to solve due to increased recourse costs, i.e. OUTER tree nodes increase, however the problem is easier to solve on the SDT due to a reduced value of Q, i.e. INNER tree nodes decrease.

U	$\|O\|$	$\|I\|$	cpu
0.5	1013.8	60068.3	1.10
0.6	998.1	55835.1	0.99
0.7	985.2	53349.8	0.97
0.8	867.1	50658.8	0.90
0.9	769.8	48288.8	0.85
1.0	775.0	46778.2	0.84

Table 3.11: $|O|$, $|I|$ and Time (secs.) for the (F, R) - ($n = 6$, $m = 1, 2, 3$, $\delta_i = 5, 6, 7$)

Strategy	$\|O\|$	$\|I\|$	cpu
CLOSEST	15889	1425619	23.3
LOWEST	16214	1474704	24.3
HIGHEST	16221	1475164	24.6
FARTHEST	16757	1558723	26.1

Table 3.12: PTSA Components For Different Branching Strategies

3.7.2 Results II: Branching Strategies

The branching strategy - deciding which customer to examine next in the OUTER tree - is based on the choice of OUTER tree arcs, i.e. if an arc v_iv_j is chosen for branching at a node of the search tree in order to extend a partially completed route $(v_1, v_k, \dots, v_i)$, an alternative branching is to reject arc v_iv_j as a possible extension of the partially completed route. This process was highlighted in 3.3.1 and utilised in the backtracking section of the PTSA. The branching strategy is of significance in the level of efficiency of the algorithm and has two separate applications, in the choice of which customer to branch to on the first route (i.e. from the depot) and in the choice of which customer to branch to from another customer. Depending on the algorithm, these may or may not be the same strategy.

Four elementary strategies were considered: CLOSEST customer, FARTHEST customer, LOWEST average demand and HIGHEST average demand. Each strategy was partially incorporated into the complete PTSA (F, R), i.e. the choice of where to branch to from the depot was random, and run for 240 problem instances where $n = 7, 8$, $m = 1, 2$, $\delta_i = 7$, $U = 0.5, \dots, 1$. The results are shown in Table 3.12. Broad generalisations are speculative since the above alterations are so problem specific. In addition, any improvements will be small due to the fact that PTSA tree node construction and computational time is largely reliant on the parameters discussed earlier. Nevertheless, it is possible to

Strategy	$\|O\|$	$\|I\|$	*cpu*
CLOSEST	16006	1418209	24.1
HIGHEST	16180	1479907	24.3
LOWEST	16275	1458606	24.6
FARTHEST	16708	1540799	25.0

Table 3.13: PTSA Components For Different Initial Branching Strategies

see that the most successful strategy is branching according to the CLOSEST customer. On average, the number of nodes generated is reduced by at least 3.3% compared with any of the other strategies. Tests were also completed using only the initial branching strategy as a parameter, i.e. randomising the general strategy. These results can be seen in Table 3.13. Once again, the CLOSEST customer to the depot proved to be the most effective strategy. Needless to say, this strategy is now fully incorporated into the PTSA.

3.8 Summary

The algorithm presented in this chapter has been constructed as an optimal solution method for a range of stochastic (routing) combinatorial optimisation problems. The primary computational difficulties to overcome in tackling such problems is to adequately represent the second stage problem and obtain an optimal solution in reasonable time while still retaining a suitable limit on computer memory requirements. The PTSA, based around a stochastic branch and bound methodology, has overcome these complications by dynamically accounting for stochasticity (INNER tree) and simultaneously maintaining the basic structure of a normal branch and bound (OUTER tree).

The problem used to illustrate the particular components of the PTSA and to highlight initial results has been the BVRPSD extension P4. Small size problems have been solved exactly (for the first time) without the use of any bounding procedures and the approach has maintained its generic outlook throughout the development phase.

Chapter 4

Lower Bounds For The Paired Tree Search Algorithm

4.1 Introduction

The PTSA is based around a stochastic branch and bound methodology and its primary operation is to dynamically account for stochasticity while retaining the basic structure of an orthodox branch and bound. The problem used to illustrate the particular components of the PTSA and to highlight initial results has been the BVRPSD extension P4. In this chapter, we continue to consider P4 by deriving and embedding new bounds for both the first stage and the second stage of the problem into the PTSA before solving computationally difficult, medium size VRPSDs for the first time. The strength of the algorithmic procedure, especially in the dynamic incorporation of the second stage bound within the INNER tree, is made apparent during the description.

In Sections 4.2 and 4.3 respectively, lower bounds on the first stage and second stage problems of P4 are derived. In each case, the computational implementation of the relevant bounds on the PTSA is fully detailed. In Section 4.4, the PTSA is updated and a simple upper bound used to obtain an initial feasible solution at the root node of the PTSA is detailed. In the first part of the testing phase, a large number of medium size, randomly generated problems and large size, (VRP) benchmark problems are solved to optimality. Finally, in the second part of the testing phase, we provide heuristic solutions of known quality to VRPSDs of larger size.

4.2 A Lower Bound on the First Stage of P4

The PTSA has so far been used without directly considering the sub-problem of P4, defined on a graph $G^s = (V^s, E^s)$, which is present at each OUTER tree node. In the same manner as the overall problem, such a sub-problem has a first stage and second stage. The only difference arises because certain branches of the OUTER tree, corresponding to edges in the edge-set of a graph describing P4, are fixed.

As described in Section 3.2.1, the first stage of P4 corresponds directly to a m-TSP. In the following section, we present a straight forward lower bound for the m-TSP, denoted by L1, that is transformed during the computational implementation phase to become an effective tree search limiter. The bound has significantly helped to solve larger size VRPSDs as the computational results will testify.

4.2.1 Lower Bound L1

Before describing the lower bound, the two-index formulation for the m-TSP is restated. Let:

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is used in the first stage solution and } 1 \le i < j \le n, \\ 2 & \text{if } (i,j) \text{ is used as a return trip and } i = 1, j > 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

In what follows, c_{ij} and x_{ij} are interpreted as c_{ji} and x_{ji} whenever $i > j$, i.e. we have a symmetric graph. The first stage problem can then be given as follows.

$$\min_x \; g_0(x) = cx \tag{4.2}$$

subject to

$$\sum_{j=2}^{n} x_{1j} = 2m, \tag{4.3}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad (v_k \in V\backslash\{v_1\}), \tag{4.4}$$

$$\sum_{i,j \in S} x_{ij} \le |S| - 1 \qquad (S \subset V\backslash\{v_1\}\,,\ 3 \le |S| \le n-2), \tag{4.5}$$

$$0 \le x_{1j} \le 2 \qquad (v_j \in V\backslash\{v_1\}), \tag{4.6}$$

$$0 \le x_{ij} \le 1 \qquad x_{ij} \text{ integer}, \ (1 \le i < j \le n). \tag{4.7}$$

As stated earlier, these first stage deterministic constraints specify that m vehicles enter and leave the depot (4.3), that every customer receives a visit exactly once (4.4) and that individual routes disconnected from the depot are prohibited (4.5).

Consider the relaxation of the vehicle number constraints (4.3) and the subtour elimination constraints (4.5) from this problem. The following problem is obtained.

$$\min_{x} \; cx \tag{4.8}$$

subject to

$$\sum_{j=2}^{n} x_{1j} = 2, \tag{4.9}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad (v_k \in V \backslash \{v_1\}), \tag{4.10}$$

$$0 \leq x_{1j} \leq 2 \qquad (v_j \in V \backslash \{v_1\}), \tag{4.11}$$

$$0 \leq x_{ij} \leq 1 \qquad x_{ij} \text{ integer}, \ (1 \leq i < j \leq n). \tag{4.12}$$

The problem described by (4.8)-(4.12) corresponds to what is known as a minimum cost 2-perfect matching problem. This problem belongs to a well-known set of graph theoretic problems which are all extensions of the 1-matching problem. The latter problem involves choosing a subset of edges from a graph which "pair" (match) the vertices of the graph so that no two edges are adjacent. The problem is of considerable interest because of the existence of efficient polynomial algorithms that can be used to obtain its optimal solution.

We now describe the 1-matching problem and its b-matching extension as a prerequisite for detailing the computational implementation of the 2-perfect matching bound. We also outline the advantages and disadvantages of employing such a bound in the PTSA. For more information on general matching problems and their solution see [110, 200].

1-Matching Problems

Given a simple non-directed graph $G^m = (V^m, E^m)$ where V^m corresponds to the set of vertices and E^m corresponds to the set of edges, a 1-matching of G^m is a subset of edges $K \subset E^m$ such that no two edges of K are adjacent, i.e. each vertex is incident with exactly one arc. An example 1-matching of a graph G^m, shown in Figure 4.1(a), is displayed in Figure 4.1(b). Clearly, an objective-driven adaptation of this problem corresponds to finding a minimum/maximum cost 1-matching K^* of G^m.

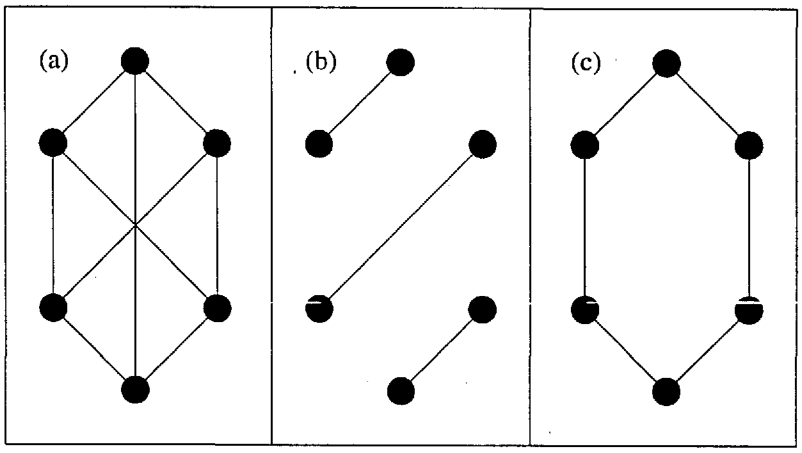

Figure 4.1: Example Matchings

b-Matching Problems

Let G^m denote a non-directed multi graph (V^m, E^m) where V^m corresponds to the set of vertices, E^m corresponds to the set of edges, and the degree of vertex $i \in V^m$ is given by $d_{G^m}(i)$. Let there exist an associated subset of edges $K \subset E^m$ such that in a partial multi-graph of G^m, denoted by $G^{m'} = (V^{m'}, K^{m'})$, the expression $0 \leq b_i \leq d_{G^m}(i)$ is satisfied where b_i are integer values associated with each vertex $i \in V^m$. A b-matching of G^m is then given by a set of edges K such that $0 \leq d_{G^{m'}}(i) \leq b_i$. A b-matching is known as a *perfect* b-matching if $d_{G^{m'}}(i) = b_i \; \forall \; i \in V$. Needless to say, the 1-matching problem is a b-matching where $b_i = 1$ and the 1-perfect matching problem is a b-matching where $d_{G^{m'}}(i) = b_i = 1$. An example 1-perfect, 2-perfect and 3-perfect matching of graph G^m, shown in Figure 4.1(a), are displayed in Figures 4.1(b), 4.1(c) and 4.1(a) respectively.

Solution Methods For Perfect Matching Problems

The ease with which a perfect matching problem can be solved depends on the nature of the underlying graph involved. Indeed, finding a 1-perfect matching of a bipartite graph - $G = (V^a \cup V^b, E)$ where $V^a \cap V^b = \emptyset$ and $E = \{(i,j) | \; i \in V^a, j \in V^b\}$ - is equivalent to solving the well-known Assignment Problem (AP). Therefore, the first exact algorithms for 1-perfect matching problems were AP-based. The Hungarian method was devised by Kuhn [137] as early as 1955. Primal methods that use the concept of polyhedral combinatorics have been devised by Barr [12] and methods based on a shortest augmenting path approach have been presented by Tomizawa [203]. For a 1-perfect matching of a complete, non-bipartite graph (describing the type of graph involved in the m-TSP), each of these methods have been extended by Edmonds [77, 76] (blossom algorithm), Cunningham and Marsh [54] and Derigs [58, 59] respectively.

The b-perfect matching problem on complete, non-bipartite graphs can be solved by either of the following two methods.

- *Generalisation:* Generalising one of the three methods given above.
- *Conversion:* Converting the underlying graph into a separate graph for which a corresponding 1-perfect matching is equivalent to a b-perfect matching of the original problem.

2-Perfect Matching as a Lower Bound

The relaxation of the m-TSP given in (4.8)-(4.12) describes a 2-perfect matching of any sub-problem defined on a graph G^s. This means that if the problem is solved to optimality, every vertex in V^s will have exactly two adjacent edges. The problem is a relaxation of the original problem since subtours are allowed, i.e. customers can be disconnected from the depot, and the requirement that there are m vehicle routes that start and end at the depot is omitted. Nevertheless, the optimal solution to (4.8)-(4.12) is a lower bound to the m-TSP since the additional constraints will always increase, never decrease, the value of the associated objective function. [Notice that a tighter bound can easily be obtained by adding $(m-1)$ artificial depots to the graph describing the sub-problem on the OUTER tree. If this is the case and each depot has an inter-connecting arc of infinite cost, then m vehicle routes will by definition be established. In essence, this adaptation simply converts the constraint given by (4.9) to the original constraint given by (4.3).]

The matching problem has been used to heuristically solve vehicle routing problems, see [20], however to our knowledge the 2-perfect matching problem has never been used as the lower bound to a VRP-based sub-problem. Indeed, it has been suggested that to obtain optimal solutions to such a problem (using Edmond's blossom algorithm) and incorporate them into a branch and bound approach of any kind is computationally expensive, see [16, 8]. Procedures to obtain fast lower bounds of the 2-perfect matching problem itself have for this reason been presented in recent years, see [166]. Nevertheless, after due consideration, to obtain a minimum cost 2-perfect matching lower bound for a first-stage sub-problem in the PTSA, the "conversion" approach was used along side Derigs' shortest augmenting path method for the 1-perfect matching problem. The reasons for taking this approach were as follows.

- Only medium size VRPSDs are to be solved, thereby restricting corresponding 2-perfect matching sub-problems to a manageable size. In addition, transformation methods can be applied to reduce associated 1-perfect matching problems (see later).
- Irrespective of the fact that computational time is not of top priority for SVRP in general, the method proposed by Derigs is considerably faster than the blossom algorithm.

4.2.2 Computational Implementation of L1

It is straightforward to incorporate a lower bound (value) of the first stage problem into the OUTER tree of the PTSA (see Section 4.4). Therefore, the computational steps that remain are five fold.

(i) *Graph Expansion:* Artificial depots, corresponding to m vehicle routes, are added to the original graph which currently only represents a TSP sub-problem. The new graph is denoted by G^a.

(ii) *Graph Reduction:* A simple reduction process is applied to G^a to shrink fixed arcs into one "new" substitute arc. The new graph is denoted by G^b.

(iii) *Graph Conversion:* Graph G^b, representing a m-TSP sub-problem, is converted into an incremental graph, G^c, from which only a 1-perfect matching is required.

(iv) *Graph Transformation:* The new graph is transformed into a smaller, more manageable graph. The new graph is denoted by G^d.

(v) *Matching Solution:* The transformed graph G^d is solved by Derigs' shortest augmenting path method.

Each stage is described independently below for an original sub-problem described by the simple three vertex graph G^s shown in Figure 4.2.

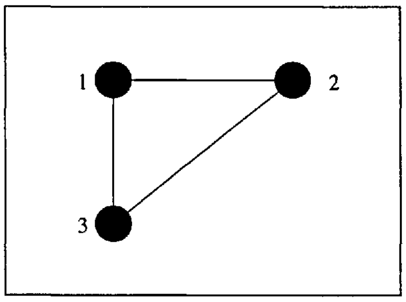

Figure 4.2: Original Graph G^s

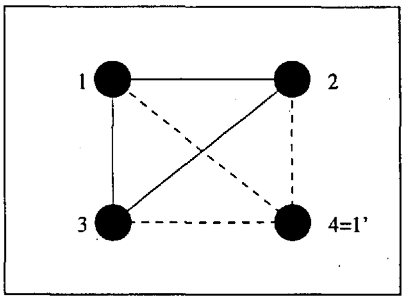

Figure 4.3: Graph Expansion

Graph Expansion

This is a very simple stage where $(m-1)$ vertices are added to G^s to obtain a new graph G^a where $G^a = (V^a, E^a)$. The number of vertices in G^a is given by $n^a = n + m - 1$ and the new vertex and edge sets correspond to $V^a = \{v_a, \ldots, v_{n^a}\}$ and $E^a = E \cup \{(i,j) | \, i,j \in V^a, i \neq j, i \text{ or } j \in V^a \backslash V\}$ respectively. The extended cost matrix $C^a = (c^a_{ij})$ associated with E^a is given as follows:

$$c^a_{ij} = \begin{cases} c_{ij} & \text{if } i,j \in V, \\ c_{i1} & \text{if } i \in V\backslash\{1\}, ; j \in V^a\backslash\{V\}, \\ c_{1j} & \text{if } i \in V^a\backslash V, ; j \in V\backslash\{1\}, \\ \infty & \text{if } i,j \in (V^a\backslash V) \cup \{1\}. \end{cases} \tag{4.13}$$

The graph G^a for the original graph G where $m = 2$ is shown in Figure 4.3.

Graph Reduction

Graph reduction corresponds to shrinking any fixed arcs in G^a into one arc (with a corresponding summation of arc costs). Such a reduction is possible if the number of fixed arcs is greater than one. This simple arrangement is shown diagrammatically in Figure 4.4 for

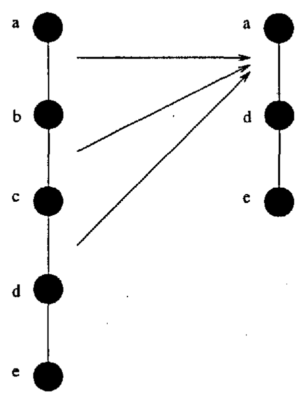

Figure 4.4: Graph Reduction

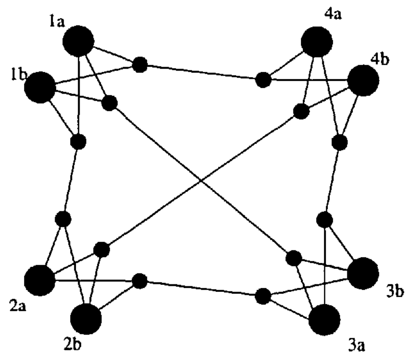

Figure 4.5: Graph Conversion

a set of arcs $\{(a,b),(b,c),(c,d),(d,e)\}$ where $\{(a,b),(b,c),(c,d)\}$ are fixed. Note that the new cost of arc (a,d) is given by $(c_{ab}+c_{bc}+c_{cd})$.

Graph Conversion

It is possible to obtain a 2-perfect matching solution of G^b by generating an incremental graph G^c. The importance of G^c stems from the fact that solving a 1-perfect matching over the incremental graph corresponds to the solution of a b-matching of G^b, see Berge [19].

The graph G^c is defined as follows. With each vertex $i \in V$ of G^b, associate two disjoint sets of vertices of G^c:

A_i: A vertex set of cardinality $d_{G^b}(i)$ where each element α_i^u of A_i corresponds to an edge u of G^b incident with i.

B_i: A vertex set of cardinality $d_i' = d_{G^b}(i) - b_i$ where each element of B_i is denoted by β_i^r $\forall r = 1,\ldots,d_i'$. (For 2-perfect matching, a reduced version of the incremental graph, denoted by $G^{c'}$, can be obtained by setting $d_i' = 2$.)

The set of vertices of G^c is then given by $A_i \cup B_i \; \forall \; i \in V$. The edges of G^c are obtained for each $i \in V$, by joining every vertex of A_i with every vertex of B_i and for each $u = (i, j) \in E$ by joining the vertices α_i^u with α_j^u. Berge proved that every matching of G^c which saturates all vertices of B_i $(i = 1, \ldots, n)$ induces in G^b a b-matching and conversely. An incremental graph $G^{c'}$, for the original 4-vertex graph given in Figure 4.3, is shown in Figure 4.5. The cost matrix of $G^{c'}$ is altered in the following way: each α to α arc has a cost of zero, each α to β arc has a cost equalling half of the arc cost on G^b that corresponds to vertex α_i^u.

Graph Transformation

To provide an additional computational advantage, the expanded, reduced and incremented graph is finally transformed before a 1-perfect matching solution is found. The transformation process, first presented by Thornton [200] and shown here in revised form, operates by exploiting the symmetry of an incremental graph (for every distinct 2-matching on G^b there are 2^4 equivalent 1-matchings on $G^{c'}$).

The transformation works by forming a string of assumptions and subsequent reductions. For instance, start by assuming that if vertex 1 is connected to vertex 2 in G^b then either vertex $1a$ or vertex $1b$ is connected to vertex $2a$ in $G^{c'}$. The incremental graph shown in Figure 4.5 can then be reduced to that shown in Figure 4.6. Similarly, by assuming that if vertex 2 is connected to vertex 3 in G^b then either vertex $2a$ or vertex $2b$ is connected to vertex $3a$ in $G^{c'}$, the incremental graph can be transformed even further to that shown in Figure 4.7. Repeating this process for $3a$ or $3b$ to $4a$ (Figure 4.8) and $4a$ or $4b$ to $1a$, results in a significantly reduced graph, G^d, that is shown in Figure 4.9.

Matching Solution

The exact minimum cost 1-perfect matching solution method used in the PTSA is the same as that first presented by Derigs [58, 59]. The method involves solving successive shortest paths, using Dijkstra's algorithm [66], and the specific details are omitted here.

4.2.3 Summary

The above 2-perfect matching solution method has been implemented at each OUTER tree node of the PTSA so that a lower bound of the first stage of a sub-problem of P4 can be found. The excessive computational requirements of using such an incremental method

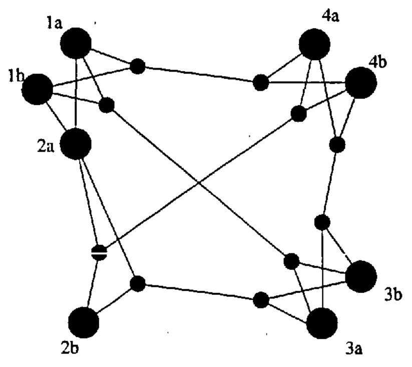

Figure 4.6: Transformation I

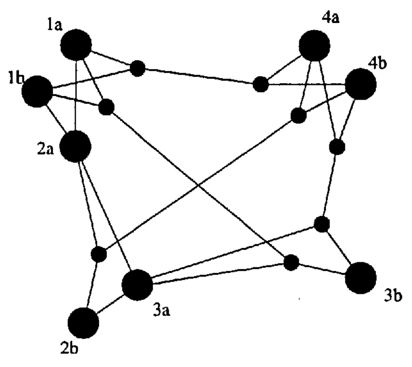

Figure 4.7: Transformation II

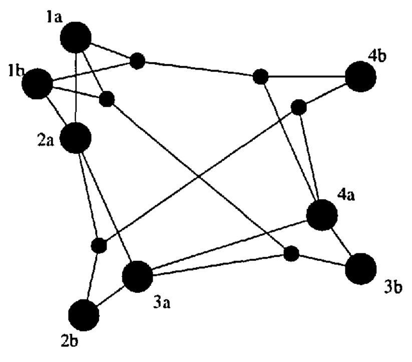

Figure 4.8: Transformation III

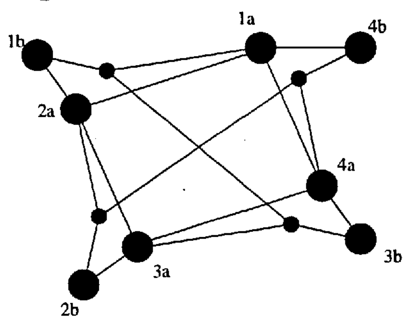

Figure 4.9: Transformation IV

Graph	$\|V\|$	$\|E\|$
$G^a = G^b$	n	$n/2(n-1)$
G^c	$n(2n-4)$	$n^2/2(n-3)(n-1)^2$
$G^{c'}$	$n(n+1)$	$n/2(5n-5)$
G^d	$n(n-1)$	$n/2(5n-11)$

Table 4.1: The Computational Improvements of the Matching Transformations

are to some extent overcome by a series of reductions and transformations. As opposed to the original incremental graph G^c, in the final transformed graph the number of arcs $|E|$ are reduced by a factor of five and the number of vertices $|V|$ are reduced by a factor of two. The specific details are shown in Table 4.1.

4.3 Lower Bounds on the Second Stage of P4

Following the derivation and computational implementation of a first stage bound that can be embedded on the OUTER tree, attention is given to second stage bounds that can be applied to both the OUTER tree (L2) and the INNER tree (L3). L2 and L3 are both formed by considering the minimum total demand to be satisfied through recourse at any customer in a sub-problem defined on a graph G^s.

4.3.1 Lower Bound L2

A second stage bound can be obtained by examining the sub-problem at each OUTER tree node ρ in which a set of customers S has previously been served. Consider the minimum total demand to be satisfied through recourse at ρ. Such a quantity depends on the remaining customer demand and the combined capacity of the remaining vehicles. If the set of customers still needing a service is $S' = V\backslash(S \cap \{v_1\})$, the number of vehicles available is m^* and each demand set $\xi_i = \{\varepsilon_i^1, \ldots, \varepsilon_i^{\delta_i}\}$ is ordered in ascending size, then the minimum remaining demand $d(\rho)$ to be satisfied through recourse is given by:

$$d(\rho) = \begin{cases} \sum_{v_k \in S'} (\varepsilon_k^1) - Q(m^* + 1) + 1 & \text{if } c(\rho) \neq 1, \\ \sum_{v_k \in S'} (\varepsilon_k^1) - Q(m^* + 1) & \text{if } c(\rho) = 1. \end{cases} \tag{4.14}$$

The proof of 4.14 is straight-forward and can be given as follows. We examine two cases: when a vehicle is situated at a customer and when a vehicle is situated at a depot.

- $c(\rho) = 1$: If the current vehicle is situated at the depot then its vehicle load is given by its capacity, Q. In addition, there are m^* remaining vehicles, all of capacity Q. Therefore the total "capacity" of the remaining vehicles available to serve the remaining customers is given by $Q + m^*Q$. The minimum remaining demand to be satisfied through recourse is then:

$$\sum_{v_k \in S'} (\varepsilon_k^1) - Q(1 + m^*). \tag{4.15}$$

- $c(\rho) \neq 1$: If the current vehicle is situated at a customer then its its vehicle load is less than its capacity, Q, i.e. its maximum is $Q - 1$. In addition, there are m^* remaining vehicles, all of capacity Q. Therefore the total "capacity" of the remaining vehicles available to serve the remaining customers is given by $Q(1 + m^*) - 1$. The minimum remaining demand to be satisfied through recourse is then:

$$\sum_{v_k \in S'} (\varepsilon_k^1) - Q(1 + m^*) + 1. \tag{4.16}$$

Now, if $d(\rho) > 0$ then a route failure will definitely occur regardless how the remaining customers are routed. Indeed, the number of route failures, $f(\rho)$, that must occur while serving the remaining customers is given by:

$$f(\rho) = \lceil d(\rho)/Q \rceil \tag{4.17}$$

where $\lceil * \rceil$ represents the smallest integer not less than $*$. Given that c_0 represents the remaining least-cost single-trip to the depot, i.e.

$$c_0 = \min_{v_k} [c_{1k} \mid v_k \in S'], \tag{4.18}$$

the lower bound L2 is then given as follows:

$$L2(\rho) = \begin{cases} f(\rho).2c_0 + z_2^T & \text{if } d(\rho) > 0, \\ z_2^T & \text{otherwise.} \end{cases} \tag{4.19}$$

where z_2^τ is the contribution of the recourse cost to the overall cost at the parent of ρ denoted by τ.

4.3.2 Lower Bound L3

The lower bound L3 operates in a similar manner to L2 however the construction of f is now a function of an INNER tree node. As the INNER tree nodes are scanned, L2 is successively applied to each individual node based sub-problem and L3, occurring at different stages in the scanning process, is incremented accordingly. The most important computational improvement from such an arrangement, as well as time efficiency, is in the construction of INNER tree nodes. The bound is described below in conjunction with the derivation of L2 given above. (Note the bound offers no new information in comparison to L2 when $c(O(\lambda)) = 1$.)

To obtain L3, consider the sub-problem at an INNER tree node λ in which a set of customers S has previously been served and the load of the vehicle is $l(\lambda)$. The minimum total demand to be satisfied through recourse at λ, $d(\lambda)$, is given by:

$$d(\lambda) = \begin{cases} \sum_{v_k \in S'} (\varepsilon_k^1) - Q(m^*) + l(\lambda) & \text{if } c(O(\lambda)) \neq 1, \\ \sum_{v_k \in S'} (\varepsilon_k^1) - Q(m^* + 1) & \text{if } c(O(\lambda)) = 1. \end{cases} \tag{4.20}$$

The proof of 4.20 is similar to that given for L2 in Section 4.3.

Now, given that $f(\lambda) = \lceil d(\lambda)/Q \rceil$ where $\lceil * \rceil$ represents the smallest integer not less than $*$, Λ^ρ corresponds to the set of all INNER tree nodes where $O(\lambda) = \rho$ and $cp(\lambda)$ corresponds to the probability of contribution of λ to the OUTER tree node $O(\lambda)$, then a new lower bound L3, defined on an OUTER tree node ρ, can be given as follows:

$$L3(\rho) = z_2^\tau + \sum_{\lambda \in \Lambda^\rho} \eta(\lambda) cp(\lambda) \tag{4.21}$$

where

$$\eta(\lambda) = \begin{cases} f(\lambda).2c_0 & \text{if } d(\lambda) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.22}$$

where z_2^τ is the contribution of the recourse cost to the overall cost at the parent of ρ denoted by τ.

4.4 Computational Implementation

The implementation of each of the lower bounds, L1 and L2 on the OUTER tree and L3 on the INNER tree, requires little explanation. In this section, a simple upper bound to be placed at the root node of the OUTER tree is briefly highlighted before detailing an updated version of the PTSA that includes all the relevant bounds.

The upper bound used at the beginning of the PTSA is based on the 2-perfect matching solution obtained at the root note. The upper bound is not a bound in the conventional sense since all that is provided at the root node is a feasible set of routes (based on the matching solution) that is then used as a first feasible solution. Any illegal subtours are connected to the depot in a heuristic fashion.

The new updated version of the algorithm, given in detail in Algorithms 4.1 and 4.2, can be summarised as follows. At each OUTER tree decision-node ρ, which has a customer index $c(\rho)$ and a parent node τ, an arc $(v_{c(\tau)}, v_{c(\rho)})$ is assigned to a current partial route k. If a feasible first stage set of routes containing k cannot be found then backtracking occurs; otherwise the search continues and lower bounds on the first stage problem, $L1$, and the second stage problem, $L2$, are computed on the OUTER tree. If $(L1 + L2)$ is greater than the best incumbent VRPSD solution value, z^*, then node ρ is fathomed; otherwise the current solution value z^ρ is updated and the search continues by transferring to the INNER tree. The set of INNER tree nodes, Λ^τ, that has previously been developed and used to index τ, are located and an improved lower bound, $L3$, of the recourse problem can then be obtained. Given that $(L1 + L3) < z^*$, full branching occurs on the INNER tree from all nodes in Λ^τ to generate a series of new nodes linked to ρ, Λ^ρ, with alternative non-unique load-leaving levels; otherwise node ρ is fathomed. The current solution value z^ρ is updated accordingly at each new branch and, once branching is completed, the search is transferred to the OUTER tree.

4.5 An Example of the PTSA With Lower Bounds

In this section we briefly update the example given in Section 3.6 and include the contribution of the bounds presented in this chapter. Needless to say, the only changes that occur to the example arise after the first feasible solution is found (assuming that the first feasible route obtained at the root node of the OUTER tree is the same as that given

Algorithm 4.1 The PTSA: OUTER tree

```
1 Initialisation
  initialise the cost matrix $C = (c_{ij})$;
  set the OUTER tree node from which branching will commence to $\rho = 1$, $c(\rho) = 1$;
  set the level of the tree to $L_O = 1$;
  set $z^\rho = 0$ and $z^* = \infty$;
  set the vertices available to branch to as $V' = \{v_2, \ldots, v_n\}$;
  initialise the first INNER tree node $\lambda = 1$, $O(\lambda) = 1$;
  calculate L1 and calculate L2;

2 Branching
  while $(V' \neq \emptyset)$ and $(L1 + L2 \leq z^*)$ do
    choose a customer $v_i \in V'$ to branch to;
    branch and number the new node accordingly, i.e. $\rho = \rho + 1$, $c(\rho) = i$;
    set $\tau$ as the parent of $\rho$
    set $L_O = L_O + 1$;
    calculate L1, i.e. $z^\rho = L1$;
    2a Fathoming Test
    calculate L2, i.e. $z^\rho = L1 + L2$;
    if $(L1 + L2 < z^*)$ then
      complete INNER Tree Branching (Algorithm 4.2);
      if $(L1 + L3 > z^*)$ then
        goto 3;
      end if
      calculate $z^\rho$ (second stage), i.e. $z^\rho = max(L1 + L2, L1 + L3)$;
      2b Feasibility and Optimality Test
      if ($z^\rho$ is feasible) and $(z^\rho < z^*)$ then
        set $z^* = z^\rho$;
        record routes;
      end if
    end if
    update $C$ and $V'$ for node $\rho$;
  end while

3 Backtracking
  repeat
    set $L_O = L_O - 1$;
    if $L_O = 0$ then
      return $z^*$;
      return routes;
      stop;
    end if
  until (alternative not node has not been examined)
  backtrack to new node;
  update $C$ and $V'$;
  goto 2;
```

Algorithm 4.2 The PTSA: INNER tree (with rebranching)

```
1 Initialisation
  set the INNER tree node from which tree search will commence to $\lambda = 1$, set $L_I = 1$;
  set $\lambda_0$ to be the number of nodes generated on the INNER tree;
2 Initial Search
  while $(O(\lambda) \neq \tau)$ do
    if ($\lambda$ corresponds to the last node on level $L_I$) then
      set $L_I = L_I + 1$;
    end if
    goto next INNER tree node on level $L_I$ and set current node as $\lambda$;
  end while
3 Bounding (L3)
  calculate L3;
  while $(L1 + L3 \leq z^*)$ do
    goto next INNER tree node on level $L_I$ and set current node to $\lambda$;
    if $(O(\lambda) = \tau)$ then
      calculate L3;
    end if
  end while
  if $(L1 + L3 > z^*)$ then
    stop;
  end if
4 Branching
  set $Y$ to be the set of load-leaving possibilities from node $\lambda$;
  while $(Y \neq \emptyset)$ do
    choose a load-leaving possibility $l$ to branch to;
    branch and number the new node accordingly, i.e. $\lambda = \lambda_0 + 1$;
    set $L_I = L_I + 1$, $O(\lambda) = \rho$, $l(\lambda) = l$ and calculate $p(\lambda)$,
    calculate and record $cp(\lambda)$ and $r(\lambda)$ for the relevant $l(\lambda)$;
    calculate recourse cost (second stage), i.e. $cp(\lambda)r(\lambda)$;
  end while
  return to parent node and set $L_I = L_I - 1$;
5 Lateral Search
  if (current node corresponds to the last node on level $L_I$) then
    goto 6;
  else
    repeat
      goto next INNER tree node on level $L_I$ and set current node to $\lambda$;
      if $(O(\lambda) = \tau)$ then
        goto 4;
      end if
    until (all INNER tree nodes on level $L_I$ have been examined)
  end if
6 Rebranching
  set $L_I = L_I + 1$ and backtrack to earliest node on $L_I$ where $O(\lambda) = \rho$;
  delete all other nodes on $L_I$ where $O(\lambda) = \rho$;
  rebranch from $\lambda$ where $O(\lambda) = \tau$ for all load-leaving levels using recorded values, stop;
```

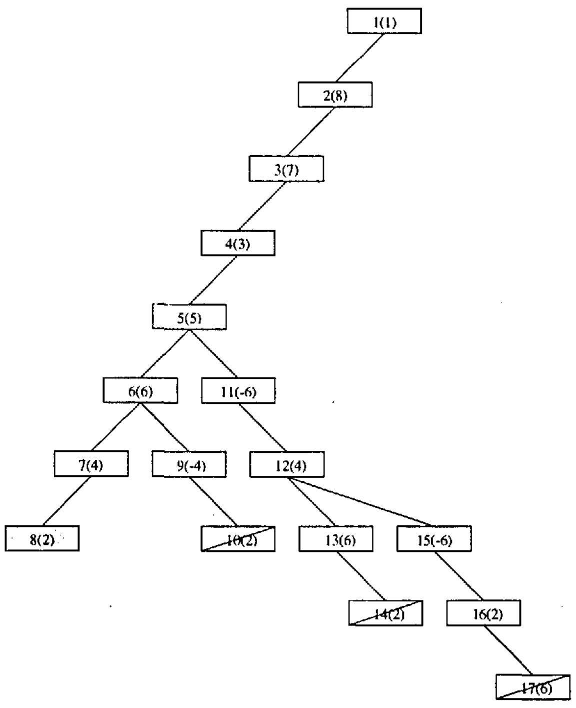

Figure 4.10: Updated OUTER Tree Example

earlier). We therefore first consider the OUTER tree node $\rho = 10$ and refer to the new version of the OUTER tree for this example given in Figure 4.10.

Before the INNER tree nodes corresponding to OUTER tree node $\rho = 10$ are formed, the two bounds $L1$ and $L2$ at $\rho = 10$ are calculated. Note at this point that the cost up to the parent of $\rho = 10$ is 357.6 and the best feasible solution so far obtained is $z^* = 425.4$ (see Section 3.6). Now, the solution to the two perfect matching problem at node $\rho = 10$ is $L1 = 383.0$. The first second stage lower bound is $L2 = 81.4$. Therefore, the initial total lower bound is $z^{10} = L1 + L2 = 464.4 > 425.4$. The node is fathomed before calculating $L3$ and before branching on the INNER tree. The search then proceeds to node $\rho = 12$ on the OUTER tree. At this node, $L1 = 341.3$ and $L2 = 19.1$. Consequently, $z^{12} = L1 + L2 = 360.4 < 425.4$ and the INNER tree algorithm is activated to first obtain $L3$. The INNER tree nodes which have OUTER tree indices $O(\lambda)$ that correspond to the parent of node $\rho = 12$, i.e. node $\rho = 5$, are scanned and $L3 = 31.1$. The new total lower bound $z^{12} = L1 + L3 = 372.4$ remains less than z^* and temporary branching continues on the INNER tree. Following the construction of the next OUTER tree node, $\rho = 13$, $L1 = 341.3$, $L2 = 33.6$, and $L3 = 59.7$. Therefore, neither $L1 + L2 = 374.9$ nor $L1 + L3 = 401.0$ are significant enough to fathom the node. Next, OUTER tree node

$\rho = 14$ is constructed and $L1 = 341.3$, $L2 = 85.6$. $L1 + L2 = 426.9 > 425.4$ and the node is fathomed. Consider one more leaf of the tree. Nodes remain unfathomed until $\rho = 17$. Here, $L1 = 381.9$, $L2 = 41.4$ and $L3 = 49.6$, therefore $L1 + L2 = 423.3.4 < 425.4$ but $L1 + L3 = 431.4 > 425.4$ and the second recourse based bound contributes to fathoming the node.

If the PTSA is run to optimality, the exact solution is found after approximately 6 seconds of computational time and after the construction of 2, 199 OUTER tree nodes and 80, 164 INNER tree nodes. Needless to say, this compares favourably with the case for the PTSA with no lower bounds used in Section 3.6.

4.6 Computational Results

The lower bounds presented in this chapter, all coded in FORTRAN, were added to the paired tree search algorithm. The testing process took place in five separate stages and, as in Chapter 3, all tests were run on a Silicon Graphics Workstation Indigo R4000 (100MHz). The tests can be summarised as follows:

- T1: The tests used in Chapter 3 are repeated for the full PTSA ($n \leq 7$).
- T2: The tests used in Chapter 3 are extended and the PTSA is run to optimality for medium size problems ($n \leq 20$). The tests created in T2 form a series of original VRPSD benchmark test problems known as *Test Data I.*
- T3: A new set of randomly generated problems of medium size ($n \leq 20$), where customer demands are described by Poisson distributions and large δ_i values, are run to optimality using the full PTSA. The tests created in T3 form a series of original VRPSD benchmark test problems known as *Test Data II.*
- T4: A set of larger size VRP-based benchmark problems ($n \leq 25$), with stochastic demands added, are run to optimality using the full PTSA.
- T5: A set of large randomly generated problems ($n \leq 40$) are solved heuristically using the full PTSA under a given time limit.

Each set of tests and their corresponding results, none of which have been obtainable previously, are described respectively below.

		$(\overline{F},\overline{R},\overline{B})$			$(F,R,\overline{B})$			(F,R,B)		
		$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*	$\|O\|$	$\|I\|$	*cpu*
n	5	141	65815	1.0	136	7158	0.2	36	718	0.2
	6	1054	3150686	40.9	919	53202	1.0	103	1703	0.4
	7	8340	161491238	2116.7	6282	377392	6.3	355	4879	1.3
m	1	1070	17238805	221.9	933	103282	1.6	93	2588	0.4
	2	4092	70007637	926.9	3117	181361	3.0	235	3215	0.8
	3	8430	154455651	2017.2	6289	298615	5.3	301	2476	1.2
δ_i	5	3826	18480391	231.6	2922	139000	2.4	186	2006	0.7
	6	3874	54037498	715.2	2969	177219	3.0	193	2770	0.7
	7	3847	131623427	1728.8	2960	213225	3.6	196	3590	0.8

Table 4.2: Results for Tests T1 (n =5 to 7, m =1 to 3, δ_i =5 to 7, U =0.5 to 1)

4.6.1 Tests T1

Numerical experiments based on those given earlier were re-run to assess the effect of the bounds. One specific algorithmic component, bounding (B), was added to the PTSA and the following three cases of the PTSA were examined ($\overline{X}$ refers to the omission of procedure X):

- no fathoming, no re-branching and no bounding - $(\overline{F},\overline{R},\overline{B})$, i.e. $(\overline{F},\overline{R})$ in Chapter 3,
- fathoming and re-branching and no bounding - $(F,R,\overline{B})$, i.e. (F,R) in Chapter 3,
- fathoming and re-branching and bounding - (F,R,B).

The tests described in Section 3.5 were run for 1620 problem instances where $n = 5,6,7$, $m = 1,2,3$, $\delta_i = 5,6,7$ and $U = 0.5,\ldots,1$. Table 4.2 displays the average values of these tests and compares the number of OUTER tree nodes used ($|O|$), the number of INNER tree nodes used ($|I|$) and the computation time in seconds over the three cases of PTSA. The computational savings that are realised by the construction of the PTSA - $(\overline{F},\overline{R},\overline{B}) \rightarrow (F,R,\overline{B})$ - and the use of the bounds - $(F,R,\overline{B}) \rightarrow (F,R,B)$ - are evident. The contribution of L2 and L3 are observed in the reduction of $|O|$ and $|I|$ respectively.

$n\backslash U$	0.7	0.8	0.9	1.0
6	0.003	0.004	0.004	0.005
7	0.005	0.005	0.007	0.009
8	0.008	0.008	0.011	0.016
9	0.020	0.019	0.024	0.044
10	0.030	0.032	0.043	0.111
11	0.081	0.081	0.093	0.316
12	0.212	0.211	0.239	0.729
13	0.393	0.419	0.452	1.288
14	0.481	0.505	0.609	2.269
15	1.174	1.184	1.336	8.885
16	2.058	2.082	2.621	16.998
17	16.933	6.765	17.805	72.416
18	35.142	34.371	36.715	166.598
19	81.609	61.432	64.470	325.347
20	50.141	50.597	51.881	409.063

Table 4.3: Tests T2 ($m = 1$, $\delta_i = 3$, Uniform): Average Computation Times (mins.)

4.6.2 Tests T2

The tests in T1 were extended to form a new series of tests with larger values of n where δ_i is fixed at 3, i.e. $p_i^l = 1/\delta_i = 1/3 \ \forall \ i$ and l. The ten underlying problems used in Tests T2 have been documented as *Test Data I* and form a series of original benchmark problems that can be found in Appendix A.1 (selected results can be found in Appendix C). Tables 4.3 and 4.4 display average computation times in minutes over the ten problem instances of T2 using (F, R, B) where the parameters m, n and U are set as follows: $m = 1$, $n = 6, \ldots, 20$, $U = 0.7, \ldots, 1$ and $m = 2$, $n = 6, \ldots, 16$, $U = 0.3, \ldots, 1$. Table 4.5 shows the average time when eventual optimum solutions were located. Primarily, these results show that the PTSA is an effective procedure to solve medium size VRPSDs to optimality. On average, one vehicle VRPSDs where $n = 20$ and $U \leq 0.9$ can be solved within one hour and fifteen customer VRPSDs where $m = 2$ and $U \leq 0.9$ can be solved within six hours. Moreover, in the latter problems, the average time required to locate the eventual optimal solution is less than one hour (the maximum time was 5.86 hours).

As well as indicating the effectiveness of the PTSA, Tests T2 have also been used to investigate the relative difficulty of obtaining solutions to particular VRPSDs. Table 4.6 displays the average optimal solution value for all ten problem instances of T2 depending on n and U where $m = 2$. Figure 4.11 displays the percentage increase in the average

$n\backslash U$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
6	0.004	0.004	0.004	0.005	0.006	0.007	0.007	0.009
7	0.006	0.006	0.007	0.010	0.016	0.019	0.024	0.027
8	0.012	0.013	0.014	0.023	0.041	0.076	0.074	0.122
9	0.028	0.028	0.035	0.086	0.207	0.236	0.274	0.387
10	0.049	0.049	0.080	0.311	0.679	0.917	1.142	1.586
11	0.128	0.129	0.180	0.756	1.498	2.023	3.085	5.503
12	0.376	0.373	0.570	1.758	3.342	5.731	8.755	20.140
13	0.935	0.911	1.074	3.392	7.895	10.106	15.080	33.309
14	1.301	1.227	1.569	6.277	18.389	27.569	38.878	134.034
15	2.891	2.890	3.214	17.818	69.769	70.288	184.744	473.826
16	4.534	4.330	5.970	41.659	176.137	187.798	353.427	1385.142

Table 4.4: Tests T2 ($m = 2$, $\delta_i = 3$, Uniform): Average Computation Times (mins.)

$n\backslash U$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
6	0.003	0.003	0.003	0.003	0.004	0.003	0.004	0.004
7	0.004	0.004	0.004	0.005	0.006	0.006	0.006	0.005
8	0.006	0.006	0.007	0.007	0.011	0.015	0.008	0.030
9	0.011	0.011	0.010	0.025	0.062	0.024	0.028	0.048
10	0.012	0.013	0.034	0.113	0.180	0.211	0.070	0.123
11	0.037	0.037	0.066	0.276	0.130	0.253	0.064	0.057
12	0.172	0.170	0.314	0.609	1.069	1.177	0.200	5.298
13	0.172	0.198	0.245	0.728	3.059	0.153	0.293	1.286
14	0.246	0.198	0.356	1.914	4.633	0.850	7.571	26.498
15	0.550	0.550	0.517	1.308	16.730	13.851	41.651	119.645
16	0.430	0.373	0.916	9.056	42.556	35.150	41.719	245.123

Table 4.5: Tests T2 ($m = 2$, $\delta_i = 3$, Uni.): Avg. Time to Locate Optimal Sols. (mins.)

$n\backslash U$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
6	249.9	249.9	252.1	260.3	275.1	292.1	312.4	334.3
7	267.4	267.4	269.1	280.7	295.7	307.5	329.6	349.8
8	275.0	275.0	277.0	292.6	310.8	327.3	344.2	364.0
9	288.8	288.8	291.1	310.4	329.5	339.8	351.3	374.8
10	295.4	295.4	298.2	318.5	335.7	344.5	360.2	383.6
11	311.0	311.0	314.5	337.4	353.5	360.7	374.8	401.4
12	321.3	321.3	323.7	344.3	357.6	370.5	383.9	409.5
13	343.5	343.5	346.5	370.5	381.5	389.0	401.9	427.0
14	348.4	348.4	351.5	376.6	389.3	396.1	410.1	437.3
15	354.7	354.7	358.1	382.9	394.9	401.4	417.4	444.8
16	358.2	358.2	362.4	387.3	401.6	408.8	422.3	451.5

Table 4.6: Tests T2 ($m = 2$, $\delta_i = 3$, Uniform): Average Optimal Solution Values

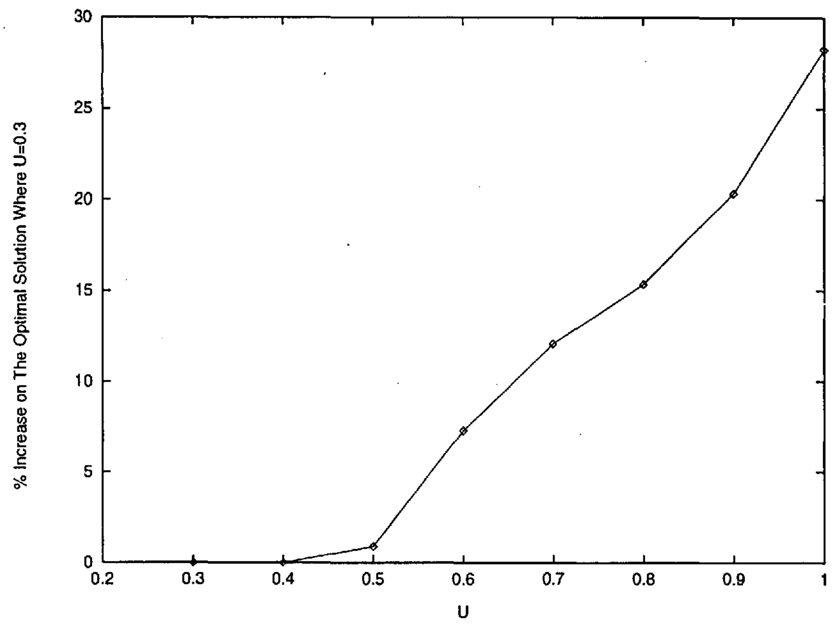

Figure 4.11: % Increase in the Value of the Optimal Solution ($m = 2$, $n = 6$ to 16)

value of the optimum over U compared to the average optimum value at $U = 0.3$. It is clear that to solve VRPSDs of certain characteristics, e.g. extension P4 where $U < 0.5$ and $m = 2$, may have no practical significance. Although stochasticity is present in these problems, vehicle capacity is set at such a high level that failures do not occur. Such problems have a solution equating to their corresponding m-TSP and can, therefore, be classed as computationally easy stochastic routing problems.[1] Indeed, in many cases where $U = 0.5$ and $m = 2$, the relevant optimal set of routes still do not differ from the case where $U < 0.5$, i.e. although the optimal solution value changes, the decisions required to obtain that optimum do not. For one vehicle VRPSDs, computationally difficult problems arise where $U \geq 0.9$. Note, apart from this chapter, no computationally "easy" problems are solved in the remainder of the thesis.

4.6.3 Tests T3

Tests T3 correspond to a new set of randomly generated test problems. Tests were constructed in a similar manner to T1 and T2, i.e. n vertices v_i were generated in the $[1, 99]^2$ square according to a continuous uniform distribution, each c_{ij} was then computed as the Euclidean distance between v_i and v_j and customers were arbitrarily assigned to one of

[1]A similar situation arises in the VRP where a large value of Q covers the demands involved and the problem transforms into a m-TSP.

three groups with discrete demands of mean $\mu_i = 5, 10, 15$ respectively. In T3, however, Poisson demands were utilised and $\delta_i = 11, 16, 19$ for each customer group since p_i^l was set to be not less than 0.01. The ten underlying problems used in Tests T3 have been documented as *Test Data II* and form a series of original benchmark problems that can be found in Appendix A.2 (selected results can be found in Appendix C). Tables 4.7 and 4.8 display average computation times in minutes over the ten problem instances of T3 using (F, R, B) where $m = 1$, $n = 7, \ldots, 20$, $U = 0.7, \ldots, 1$ and $m = 2$, $n = 7, \ldots, 16$, $U = 0.3, \ldots, 1$. Although computational times are comparable, it is clear that problems in Tests T3 are harder than those in Tests T2 due to the extra generation of INNER tree nodes. Table 4.9 shows the number of INNER tree nodes created (in millions) for T3 problems where $m = 2$. It is clear that an exponential-type relationship exists between U and $|I|$ and a symmetry exists between the complexity of the problem as U and n increase respectively. For example, the increase in I as U increases where $n = 16$ is almost identical to the increase in I as n increases where $U = 1.0$, i.e. to solve a problem where $n = 13$ and $U = 1.0$ is equal in difficulty to solving a problem where $n = 16$ and $U = 0.7$. This behaviour is shown clearly in Figures 4.12 and 4.13 which show the increase in average computation times for different n (for specific values of U) and different U (for specific values of n) respectively.

4.6.4 Tests T4

T4 involves a different set of problems that have been solved to optimality using the PTSA. The data for these tests are given by subsets of the two well known 50 and 75 customer VRP test problems given by Eilon *et al* [78]. Demands take the form of discrete uniform distributions generated in the same manner as in Tests T2. The data corresponding to these two tests can be found in Appendix B. Tables 4.10 and 4.11 display average computation times in minutes over the two problem instances of T4 using (F, R, B) where $m = 1$, $n = 15, \ldots, 25$, $U = 0.7, \ldots, 1$ and $m = 2$, $n = 15, \ldots, 25$, $U = 0.3, \ldots, 1$.

Randomly generated vehicle routing problems are known to be significantly harder to solve to optimality than ordinary benchmark problems. This is highlighted in the comparison between the average computational times in Tests T4 and those in Tests T2/T3. For instance, consider two "similar" VRPSD problems ($m = 2$, $n = 16$, $U = 1.0$): one from T4, one from T2. Their solutions are given in Figures 4.14 and 4.15 as VRPSD

$n\backslash U$	0.7	0.8	0.9	1.0
7	0.006	0.008	0.009	0.020
8	0.009	0.012	0.013	0.031
9	0.016	0.021	0.024	0.075
10	0.029	0.036	0.047	0.147
11	0.072	0.086	0.114	0.392
12	0.101	0.118	0.173	0.632
13	0.246	0.270	0.371	1.366
14	0.389	0.431	0.571	2.438
15	0.748	0.769	1.203	6.580
16	1.780	1.840	2.324	14.708
17	2.240	2.333	3.057	34.333
18	6.051	5.991	8.821	130.396
19	11.844	12.302	16.641	175.130
20	44.180	44.691	62.514	526.366

Table 4.7: Tests T3 ($m = 1$, $\delta_i \in \{11, 16, 19\}$, Pois.): Avg. Computation Times (mins.)

$n\backslash U$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
7	0.007	0.008	0.010	0.026	0.035	0.040	0.055	0.073
8	0.011	0.013	0.020	0.060	0.112	0.129	0.179	0.283
9	0.019	0.021	0.043	0.184	0.312	0.497	0.765	1.190
10	0.047	0.047	0.096	0.748	1.352	1.515	2.413	5.522
11	0.129	0.117	0.244	1.531	4.482	6.221	9.828	18.390
12	0.220	0.230	0.473	4.167	9.198	14.551	23.996	54.527
13	0.343	0.319	0.662	5.261	15.542	24.703	51.316	138.843
14	0.494	0.445	0.921	9.764	35.444	69.744	127.840	353.345
15	1.220	1.098	2.117	24.420	88.768	162.536	318.390	817.663
16	2.010	1.859	3.449	74.250	190.079	410.981	922.778	2086.770

Table 4.8: Tests T3 ($m = 1$, $\delta_i \in \{11, 16, 19\}$, Pois.): Avg. Computation Times (mins.)

$n\backslash U$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
7	0.00	0.01	0.01	0.04	0.05	0.05	0.07	0.09
8	0.01	0.01	0.02	0.07	0.14	0.13	0.18	0.29
9	0.01	0.01	0.03	0.20	0.31	0.44	0.64	1.03
10	0.03	0.03	0.08	0.84	1.29	1.14	1.72	3.59
11	0.06	0.05	0.16	1.21	3.62	4.00	5.65	11.07
12	0.10	0.09	0.31	2.95	5.27	7.07	10.69	26.02
13	0.14	0.11	0.38	2.31	7.29	9.65	18.32	50.51
14	0.18	0.11	0.38	3.71	14.24	22.70	36.70	98.99
15	0.31	0.24	0.66	8.26	29.19	45.13	68.95	175.37
16	0.41	0.37	0.65	20.21	49.30	86.40	176.42	407.80

Table 4.9: Tests T3 ($m = 1$, $\delta_i \in \{11, 16, 19\}$, Poisson): Average Value of $|I|$ (millions)

$n\backslash U$	0.7	0.8	0.9	1.0
15	0.251	0.255	0.334	1.565
16	0.393	0.394	0.498	3.611
17	0.317	0.316	0.527	5.253
18	0.920	0.922	1.174	21.144
19	1.659	1.661	2.011	46.005
20	4.532	4.541	5.094	125.133
21	10.070	10.092	11.522	212.312
22	3.155	3.139	4.525	100.691
23	9.565	9.575	12.099	332.861
24	15.233	14.879	18.598	484.283
25	26.939	27.042	32.987	1507.607

Table 4.10: Tests T4 ($m = 1$, $\delta_i = 3$, Uniform): Average Computation Times (mins.)

$n\backslash U$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
15	0.348	0.348	1.075	1.589	3.035	3.819	6.684	49.224
16	0.886	0.887	1.764	3.687	4.944	4.427	12.359	85.286
17	1.564	0.915	2.241	4.370	9.826	11.062	17.612	395.820
18	1.797	1.799	3.329	8.620	8.096	17.997	25.850	396.784
19	1.868	1.932	4.480	26.318	21.722	20.233	33.899	771.785
20	7.138	7.141	11.992	36.392	30.961	30.834	64.024	3153.863
21	19.994	18.557	55.475	107.139	149.816	146.429	531.678	-
22	14.142	14.237	26.835	93.840	89.175	111.693	342.151	-
23	21.751	22.481	48.939	195.613	169.536	162.727	488.556	-
24	38.762	38.766	83.099	440.172	468.912	471.492	1267.995	-
25	34.765	34.797	77.547	375.351	321.985	601.418	2022.673	-

Table 4.11: Tests T4 ($m = 2$, $\delta_i = 3$, Uniform): Average Computation Times (mins.)

Problem	Fixed Cost	Recourse Cost (% of Overall Cost)	Overall Cost
VRPSD I (T4)	238.6	18.8(7%)	257.4
VRPSD II (T2)	393.6	42.5(10%)	436.1

Table 4.12: A Comparison of Randomly Generated and VRP-based Problems

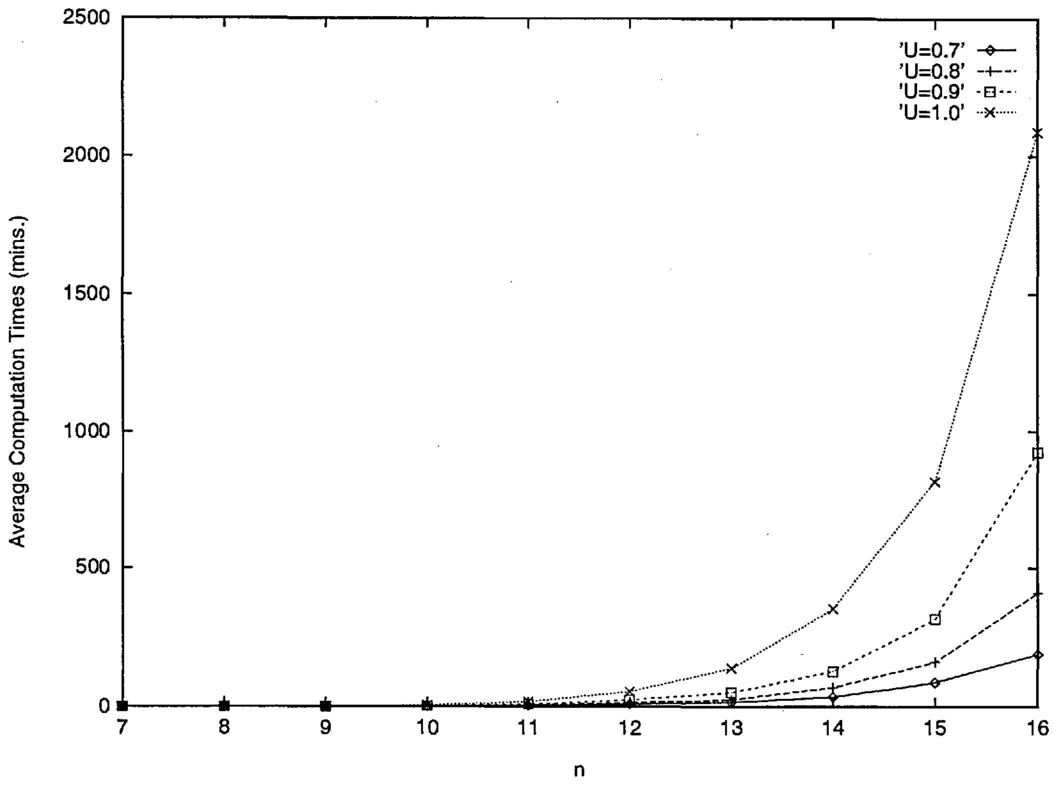

Figure 4.12: Tests T3 (m=2): The Effect of n on Computational Time

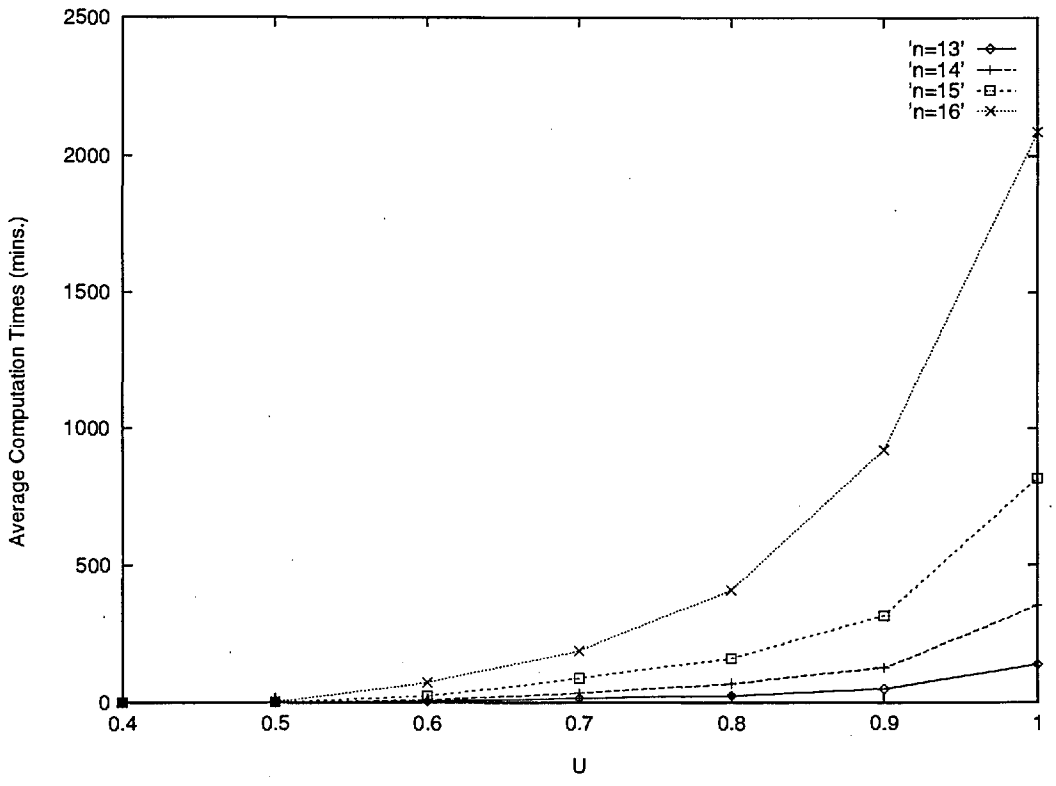

Figure 4.13: Tests T3 (m=2): The Effect of U on Computational Times

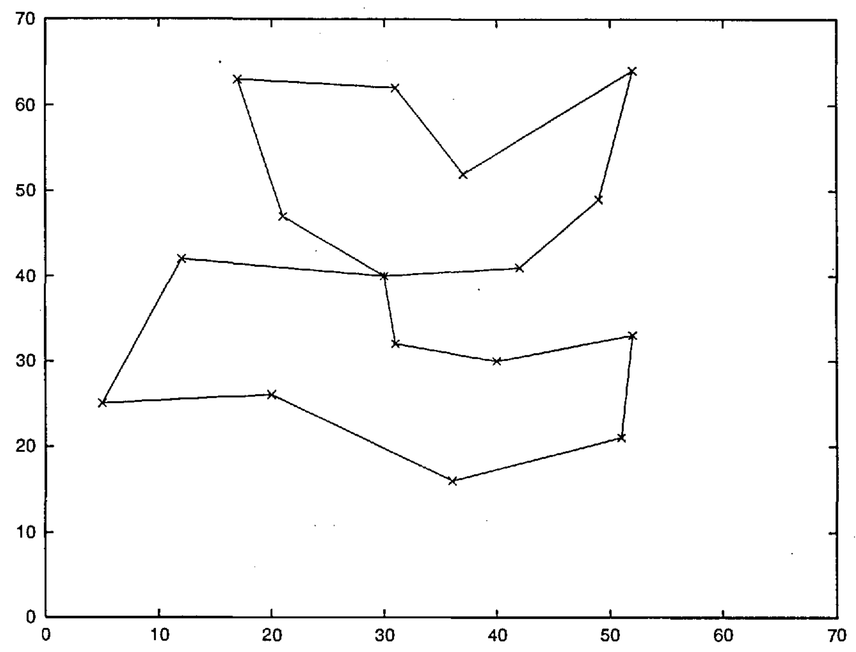

Figure 4.14: VRPSD Solution I

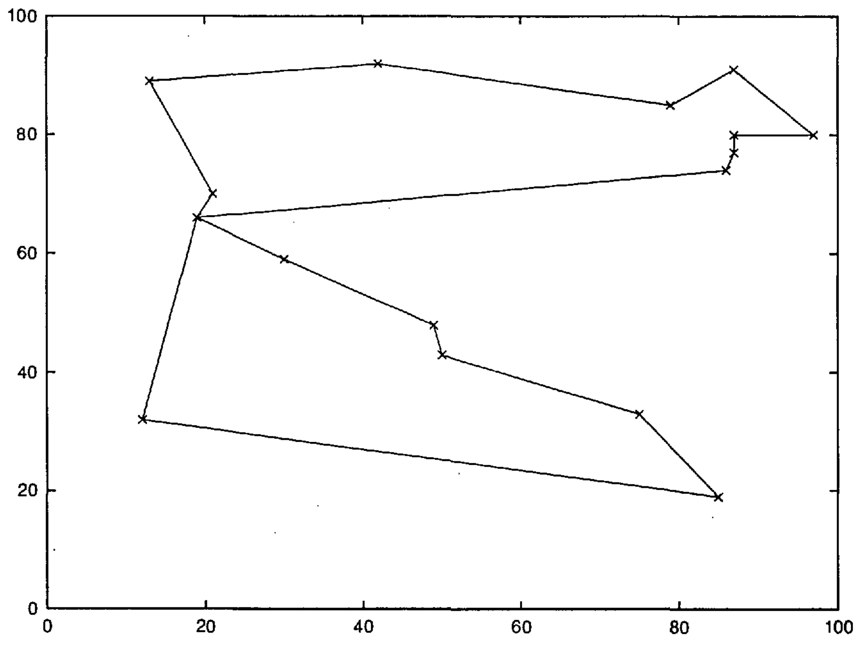

Figure 4.15: VRPSD Solution II

$n\backslash m$	1	2
20	4.9	11.8
25	4.0	11.1
30	10.6	17.2
35	14.0	19.7
40	12.8	17.8

Table 4.13: The Quality (G) of the PTSA Heuristic

Solution I and VRPSD Solution II respectively. Unlike the deterministic VRP, the even spread of customers around the depot in VRPSD Solution I indicates an easier problem since, in such cases, the original matching based upper bound usually coincides with the m-TSP and the best routes in terms of recourse are easy to find. This difference is highlighted in respective running times of 52.87 minutes and 3278.09 minutes. [The randomly generated problem concerned was, in fact, the hardest problem of its kind in Tests T2.] In harder problems, the comparative contribution of the recourse cost to the overall routing cost is also in general higher, see Table 4.12.

4.6.5 Tests T5

T5 involves the use of the same set of problems that were used in T2, i.e. Test Data I. In this set of tests, however, larger size problems have been solved heuristically by setting a computational time limit of 3 hours on the PTSA. In addition, the lower bounds given in this chapter were used to obtain a measure of quality on the heuristic solution obtained.

We define G^{lb} as the percentage gap between the best lower bound (lb) and the best feasible solution obtained (z), i.e. $G^{lb} = 100(z - lb)/z$. Table 4.13 displays average values of G^{lb} over the ten problem instances of T5 where $m = 1$, $n = 20, 25, 30, 35, 40$, $U = 0.9$ and $m = 2$, $n = 20, 25, 30, 35, 40$, $U = 0.7$. In most cases the gap between the best obtained solution and the lower bound is fairly large; for $n < 30$ the gap is approximately 4.4% where $m = 1$ and 12.5% where $m = 2$.

4.7 Summary

In this chapter large size, computationally difficult VRPSDs have been solved for the first time. This has been made possible due to the structure of the PTSA and the use of three lower bounds on both the first stage and second stage of P4. A large number of problems have been solved to test the use of the bounds and the computational performance of the PTSA including randomly generated problems, VRP-based benchmark problems and problems incorporating a large amount of stochasticity. The only comparable work in the literature, see Seguin [187], has solved similar computationally difficult VRPSDs where $n \leq 10$.

Chapter 5

Applying the PTSA to Extensions of the BVRPSD

5.1 Introduction

A standard group of problem extensions, with clear and widely applicable objectives, have been defined for the VRPSD in Section 2.6.2. In this chapter, the adaptability of the PTSA is recognised as an entire class of these extensions are solved to optimality for the first time.

Each BVRPSD extension involves finding a minimum cost set of routes, including any associated recourse costs, given that there exists a fixed system of information disclosure and fixed objective-related criteria. In the following discussion, all the extensions specified in Table 2.2 are analysed apart from P1 (a simplified version of P2) and P3 (a simplified version of P4). For each extension we first describe how the PTSA was adapted before presenting associated computational results.

5.2 BVRPSD Extension P2

P1 corresponds to a VRPSD in which a set of routes must be found that meet a particular service requirement, regardless of the specific effects of any individual route failures. There exists a probability of a group of solution routes being unviable, i.e. failing, and this equates to a serviceability percentage. The optimal solution corresponds to a set of routes that provides this percentage of expected service at least cost. P2 involves a clearer definition of the BVRPSD since routing costs take into account the theoretical re-serving of failed customers. More specifically, P2 represents a VRPSD that minimises expected costs while controlling the probability of route failure.

For serviceability related problems such as P1 and P2, the PTSA must be adapted to somehow incorporate a measure for the probability of route failure. This measure then needs to be constrained to obtain appropriate feasible solutions. Before specifically describing how the PTSA can be adapted for BVRPSD extension P2, the probability of route failure and its incorporation into the PTSA is discussed below.

5.2.1 The PTSA and the Probability of Route Failure

In the paired tree search algorithm, INNER tree nodes are created according to a branching process that is dependent on the load leaving level of the particular customer involved. A range of indices specify this and other characteristics of each node, e.g. recourse costs and various probabilities. It is comparatively simple to include another set of indices to represent different practical occurrences and one such index can correspond to the possible occurrence of route failure.

Refer to the notation introduced in Section 3.4.3 and consider an INNER tree node λ branching to another INNER tree node μ. Let a new index for μ be defined as follows.

$$rf(\mu) = \begin{cases} 1 & \text{if } l(\lambda) < \xi_{c(O(\mu))}, \\ 0 & \text{if } l(\lambda) \geq \xi_{c(O(\mu))}. \end{cases} \tag{5.1}$$

The index $rf(\mu)$ corresponds to a binary variable that takes the value 1 if and only if the INNER tree node μ represents a route failure, i.e. the probability of μ representing a route failure (given a particular load-leaving level of λ) is $p(\mu)rf(\mu)$.

Following this, it is possible to introduce another new index $crf(\lambda)$ that corresponds to the contribution of the INNER tree node λ to the OUTER tree node $O(\lambda)$ with regard to the probability of route failure. It is easy to see that $crf(\mu)$ is given as follows.

$$crf(\mu) = \begin{cases} crf(\lambda).p(\mu) & \text{if } (rf(\mu) = 1 \text{ and } rf(\lambda) = 0) \text{ or } (rf(\lambda) = 1), \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Note that from (5.2), in addition to the case where $rf(\mu) = 1$, an INNER tree node μ

will contribute its probability of route failure to the OUTER tree node $O(\mu)$ if $rf(\lambda) = 1$, i.e. if the previous node corresponds to a route failure then the new node will by definition be recognised as a "failing" node. This construction means that we are interested in the probability of any kind of failure along a route; a route that fails six times will be equivalent to a route that fails once.

Now, given that Λ^ρ corresponds to the set of all INNER tree nodes where $O(\lambda) = \rho$ and $crf(\lambda)$ corresponds to the route failure probability of contribution of λ to the OUTER tree node $O(\lambda) = \rho$, then the probability of route failure at node ρ is given by:

$$RF(\rho) = \sum_{\lambda \in \Lambda^\rho} crf(\lambda). \tag{5.3}$$

Therefore, without loss of generality, the probability of route failure corresponding to a customer $c(\rho)$ on a particular route is given by $RF(c(\rho)) = RF(\rho)$. (Note this is different to the meaning of RF_j given in (3.33) that corresponds to the probability of failure at customer j in a VRPSD. In this context, the meaning is extended to define a failure up to and including a particular customer in a VRPSD.) Moreover, for any feasible set of routes z, the maximum value of this summation for any customer in z can be given as follows,

$$RF^z = \max_k [RF(k)] \tag{5.4}$$

where k is a customer in a route of z. The probability of route failure in an optimal set of routes z^* is therefore given by RF^{z^*}.

Now, before using a constraint on RF^z to model P2, it is important to consider the values RF^{z^*} takes for different unconstrained problems. This can be illustrated using P4.

5.2.2 P4 and the Probability of Route Failure

We have solved a whole range of problems relating to extension P4 in Chapter 4. For each problem, the maximum probability of route failure of any route in each optimal set of routes was obtained using the method described in the previous section. In the following, we present the corresponding results to Tests T2 and T3.

For one vehicle VRPSDs the probability of route failure can be determined before an optimal set of routes is found. Nevertheless, RF^{z^*} can be used to provide some measure of the difficulty of a problem. The average value of RF^{z^*} for Tests T2 and T3 where $m = 1$,

U	T2	T3
0.7	0.00	0.01
0.8	0.06	0.70
0.9	4.62	10.44
1.0	46.02	47.08

Table 5.1: Tests T2 and T3 ($m = 1$): Average Value of RF^{z^*} (%)

$n \backslash U$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
6	0.00	0.00	2.59	4.44	16.17	20.00	42.22	85.19
7	0.00	0.00	4.07	12.14	16.50	16.05	55.93	81.73
8	0.00	0.00	5.95	20.10	16.54	27.41	36.05	81.60
9	0.00	0.00	6.60	15.41	25.44	7.08	38.27	77.37
10	0.00	0.00	7.38	17.90	24.70	23.15	34.29	76.60
11	0.00	0.00	8.91	13.41	4.99	10.68	19.05	70.21
12	0.00	0.00	5.90	25.17	21.19	12.13	29.69	87.49
13	0.00	0.00	6.79	20.39	15.17	19.94	28.65	76.43
14	0.00	0.00	7.36	20.68	10.57	10.57	38.49	76.31
15	0.00	0.00	7.96	8.69	1.06	11.25	32.13	83.81
16	0.00	0.00	7.89	11.60	5.98	13.81	19.03	79.53
Average	0.00	0.00	6.49	15.45	14.39	15.64	33.98	79.66

Table 5.2: Tests T2 ($m = 2$): Average Value Of RF^{z^*} (%)

$n \backslash U$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
7	0.00	0.14	13.34	31.81	24.25	27.88	56.79	77.72
8	0.00	0.11	12.47	37.55	22.39	23.82	52.50	80.60
9	0.00	0.11	13.90	35.70	19.37	38.85	48.51	75.89
10	0.00	0.05	16.02	35.33	32.21	11.86	46.31	83.95
11	0.00	0.05	15.38	26.45	24.83	16.91	49.51	85.15
12	0.00	0.03	16.23	47.58	11.45	12.21	41.13	80.12
13	0.00	0.02	12.00	38.61	22.00	15.47	33.05	77.50
14	0.00	0.01	12.41	38.52	16.66	16.21	33.05	77.23
15	0.00	0.00	12.93	42.97	6.44	15.17	35.75	74.70
16	0.00	0.00	14.76	34.89	19.58	10.02	26.98	80.80
Average	0.00	0.05	13.94	36.94	19.92	18.84	42.36	79.37

Table 5.3: Tests T3 ($m = 2$): Average Value Of RF^{z^*} (%)

$U = 0.7, \ldots, 1$ and $n = 6, \ldots, 16$ are shown in Table 5.1.

Tables 5.2 and 5.3 display the average value of RF^{z^*} for Tests T2 and T3 respectively where $m = 2$, $U = 0.3, \ldots, 1$ and $n = 6, \ldots, 16$. As n increases, the average value of RF^{z^*} remains largely unchanged. However, not unexpectedly, this is not the case as U increases. Figure 5.1 displays the average value of RF^{z^*} for both T2 and T3. It is clear that for both cases the overall probability of route failure increases, then decreases and eventually increases sharply up to its value at $U = 1.0$. This may seem to be a counter intuitive result until one considers the trade-off between service and transportation costs first mentioned in Chapter 2. Clearly, as vehicle capacity decreases, route failure increases until it costs more to continuously fail (and return to the depot) than it does to "cover" the customers (and not return to the depot). Whenever this occurs, the percentage chance of route failure will decrease slightly before increasing once again as U becomes increasingly large.

Additionally, the reason for the difference in RF^{z^*} over Tests T2 and T3 provides another explanation as to why problems in T3 are harder to solve than those in T2, see Section 4.6.3. The percentage chance of route failure for each n is always larger in T3 due to a greater spread of customer demands (a Poisson distribution is used with many more discrete demand possibilities). The service/cost trade-off mentioned above is much more pronounced in T3 and, hence, the "decline" is greater. Indeed, there is less percentage chance of route failure on average if $U = 0.7$ and $U = 0.8$ than if $U = 0.6$. Furthermore, unlike problems in T2, T3 problems where $U = 0.4$ have optimal solutions that actually involve a small amount of route failure. Also, in no problems where $U = 0.3$ (and $U = 0.3, 0.4$ for T2) is there any percentage chance of route failure within the optimal set of routes, i.e. the m-TSP solution is being used and no demands are left unsatisfied. This corresponds to the case where recourse costs are zero and the VRPSD involved can be referred to as computationally easy, see Section 4.6.2.

Figure 5.2 displays the changes in RF^{z^*} for different values of n in Tests T2. A series of extra problems have been added to allow for the case where $U = 1.1$. There appears to be no connection between the change in RF^{z^*} for different n but, needless to say, $U > 1.0 \Longrightarrow RF^{z^*} \rightarrow 100\%$.

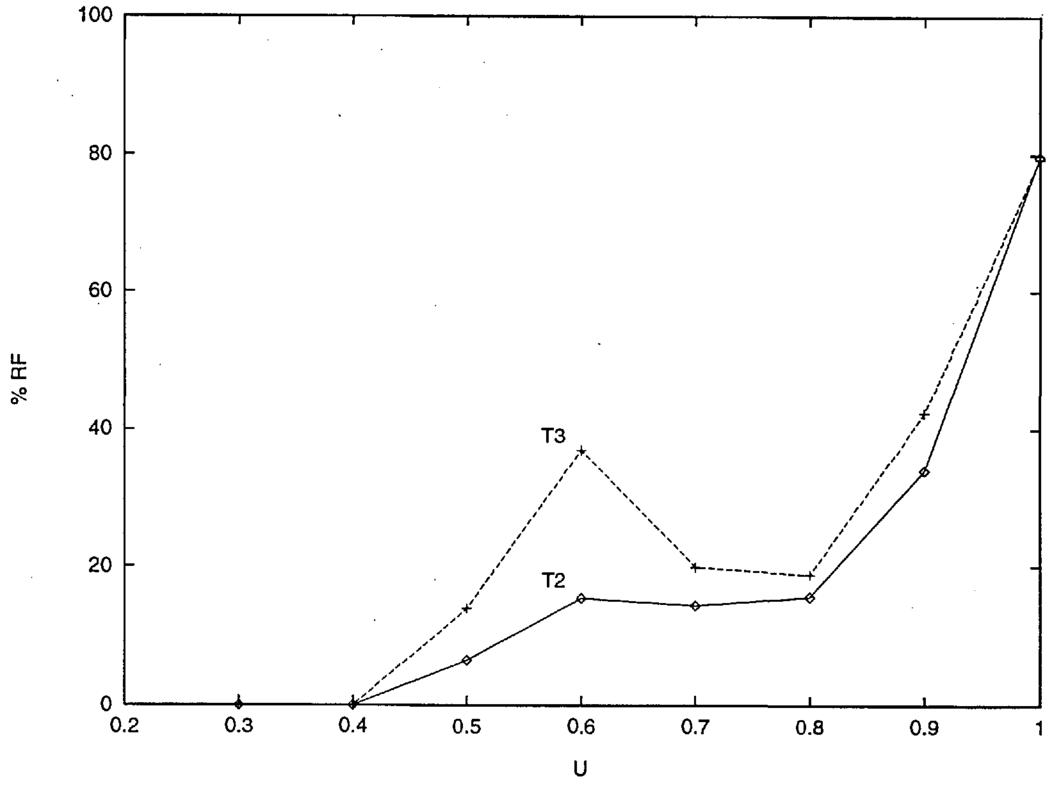

Figure 5.1: Average % Chance of Route Failure For Tests T2 and T3 ($m = 2$)

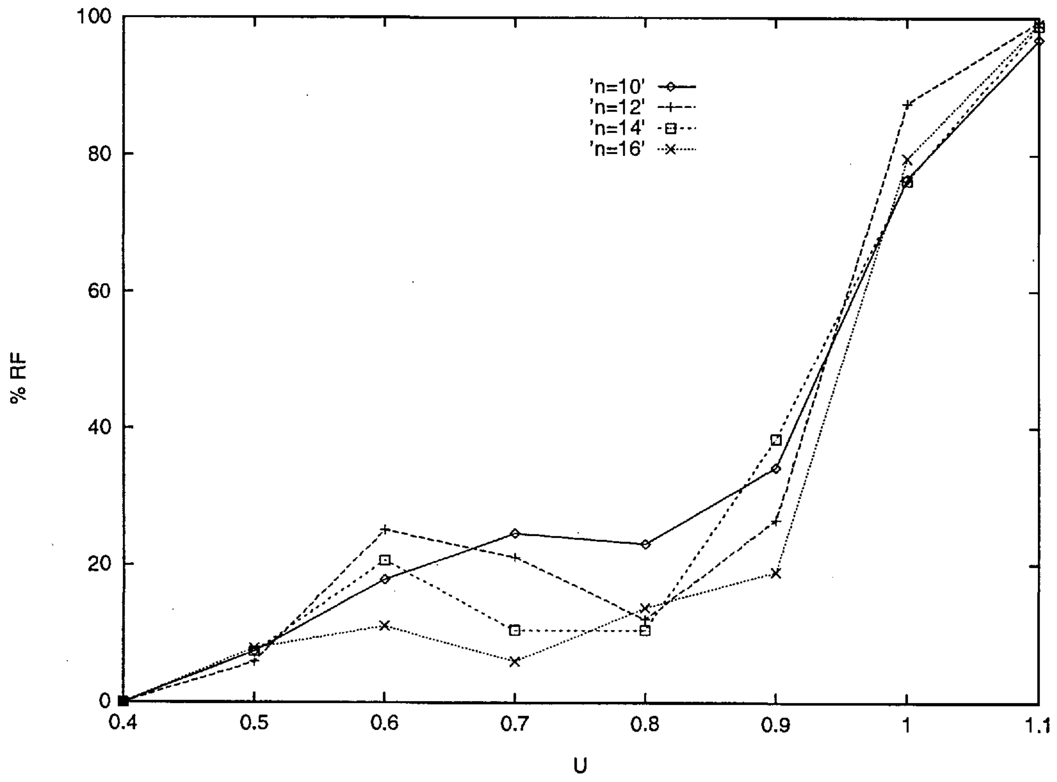

Figure 5.2: Effect of U on Average % Chance of Route Failure Per n (Tests T2, $m = 2$)

5.2.3 P2 and the Probability of Route Failure

P2 involves a constraint limiting the amount of route failure in a feasible solution. It was shown in Section 5.2.1 how the probability of route failure for a set of routes can be obtained. It is, therefore, straight-forward to introduce a constraint on the OUTER tree that prohibits the value of $RF^z = \max_k[RF(k)]$ to be above a certain value $\overline{RF^z}$ for any customer k in a feasible set of routes, i.e.

$$RF^z \leq \overline{RF^z}. \tag{5.5}$$

Since this implies that $RF^{z^*} \leq \overline{RF^z}$, (5.5) equates to a constraint on the serviceability of an optimal set of routes and can therefore be used to model BVRPSD extension P2.

5.2.4 Computational Results

By incorporating the constraint (5.5) into the OUTER tree of the PTSA and implementing a simple demand-based heuristic at the root node (to obtain a first feasible solution), it is possible to obtain exact solutions to chance constrained VRPSDs such as P2 for the first time. The heuristic obtains an initial feasible set of routes by assigning customers to particular routes so that the maximum total demand along all routes is minimised. Two sets of eleven customer VRPSDs from Tests T2 and T3 respectively have been used to illustrate the problem.

Firstly, it is important to note from the discussion in Section 5.2.2 that depending on the value of U for a particular problem, constraint (5.5) may be unnecessary, i.e. the optimal set of routes may have a value of RF^{z^*} which is already below $\overline{RF^z}$. Similarly, setting too low a value for $\overline{RF^z}$ may render a problem infeasible. The choice of $\overline{RF^z}$ is therefore particularly problem specific and in the following tests we are most interested in how a change in the value of $\overline{RF^z}$ affects the overall routing cost for a given instance of P2. [Note, if $\overline{RF^z} = 0\%$ then P2 is equivalent to a VRP, with associated demands corresponding to $\varepsilon_i^{\delta_i}$ $\forall\, i$, and if $\overline{RF^z} = 100\%$ then P2 is equivalent to BVRPSD extension P4.] Given that for a particular test problem t, $RF^{z^*}(t)$ corresponds to the route failure of the (unconstrained) optimum solution of t and $\underline{RF^z}(t)$ corresponds to the minimum possible route failure for any set of routes in t, then $\overline{RF^z}(t)$ is clearly constrained in the following way.

$$\underline{RF^z}(t) \leq \overline{RF^z}(t) \leq RF^{z^*}(t) \tag{5.6}$$

From Table 5.2, for example, over the ten problem instances of T2 where $m = 2$, $n = 12$ and $U = 0.7$ (denoted by $t_1, \ldots, t_{10}$), the minimum, average and maximum values for RF^{z^*} are 0.00%, 21.19% and 88.84% respectively, i.e. $\min_i \underline{RF^z}(t_i) = 0.00\%$ and $\max_i RF^{z^*}(t_i) = 88.84\% \; \forall \; i = 1, \ldots, 10$. In addition, from Table 5.3, over the ten problem instances of T3 where $m = 2$, $n = 12$ and $U = 0.6$ (denoted by $t_{11}, \ldots, t_{20}$), the minimum, average and maximum values for RF^{z^*} are 0.18%, 47.58% and 87.74% respectively, i.e. $\min_i \underline{RF^z}(t_i) = 0.18\%$ and $\max_i RF^{z^*}(t_i) = 87.74\% \, \forall \, i = 11, \ldots, 20$.

To analyse P2, each test $t_i \in \{t_1, \ldots, t_{20}\}$ was solved for different upper bounds on the percentage chance of route failure given by (5.6) where $\overline{RF^z}(\%) \in \{0, 10, 20, \ldots, 100\}$. The results are shown in Tables 5.4 and 5.5. In all cases, $\underline{RF^z} = 0.0$. From Table 5.4, $RF^{z^*} \leq 10\%$ for four out of ten problems in T2 whereas from Table 5.5, despite a lower value of U, $RF^{z^*} \leq 50\%$ for only four problems in T3. Once again, this is because a Poisson distribution is being used instead of a uniform distribution. Needless to say, the optimal solution of any particular problem always decreases as $\overline{RF^z}$ increases. This is shown clearly in Figure 5.3 that displays the optimal solution value for three tests over differing values of $\overline{RF^z}$. As the constraint on serviceability is weakened, the cost of the optimal set of routes decreases.

Clearly, the interesting measure in these types of chance constrained problems is the relative "loss" in terms of the additional routing costs incurred when extra serviceability is required. For a given upper bound on route failure R, such a measure, denoted by $g(R)$, can be established by considering the gap between the optimal solution when $\overline{RF^z} = R$, i.e. $z(R)$, and the unconstrained optimal solution z^*. We denote $g(R)$ as follows.

$$g(R) = \frac{z(R) - z^*}{z^*} \tag{5.7}$$

For the test problems presented here, the average value for this measure in the most severe case, i.e. total serviceability, is $g(0.0) = 3.33\%$ for T2 and for $g(0.0) = 7.85\%$ for T3. For a given problem, $g(0.0)$ can be as much as 20%. Table 5.6(a) and 5.6(b) show the values of g for all values of R for Tests T2 and T3. In addition, the number of problems that have reached the optimal value of their test, N(Optimal), is also shown.

$t_i \backslash \overline{RF^z}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	405.5	397.7	-	-	-	-	-	-	-	-
2	329.9	-	-	-	-	-	-	-	-	-
3	331.5	327.1	322.1	-	-	-	-	-	-	-
4	370.2	347.1	345.5	-	-	-	-	-	-	-
5	395.8	361.8	-	-	-	-	-	-	-	-
6	358.9	358.9	348.8	-	-	-	-	-	-	-
7	379.6	379.6	379.6	378.5	-	-	-	-	-	-
8	434.5	427.2	-	-	-	-	-	-	-	-
9	400.4	396.6	392.6	392.6	379.4	-	-	-	-	-
10	288.4	288.4	288.4	288.4	288.4	286.5	286.5	286.5	286.5	285.3

Table 5.4: Tests T2 ($m = 2$, $n = 12$, $U = 0.7$): Optimal Solutions for Different $\overline{RF^z}$

$t_i \backslash \overline{RF^z}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
11	430.9	382.6	382.0	382.0	378.4	-	-	-	-	-
12	449.6	429.0	-	-	-	-	-	-	-	-
13	413.1	413.1	413.1	413.1	413.1	413.1	405.3	-	-	-
14	341.9	341.9	338.4	338.4	338.4	338.4	338.4	338.4	338.4	334.4
15	410.6	394.9	394.9	394.9	392.3	392.3	392.3	392.3	388.4	-
16	451.5	403.9	403.9	403.9	403.9	403.9	378.1	378.1	373.3	-
17	369.1	339.1	-	-	-	-	-	-	-	-
18	461.4	414.5	414.5	414.5	414.5	414.5	405.7	-	-	-
19	348.6	333.0	-	-	-	-	-	-	-	-
20	413.0	387.6	387.6	387.6	385.0	385.0	385.0	385.0	385.0	382.6

Table 5.5: Tests T3 ($m = 2$, $n = 12$, $U = 0.6$): Optimal Solutions for Different $\overline{RF^z}$

R	$g(RF^z)$	N(Optimal)
0.0	3.33	1
0.1	1.06	4
0.2	0.50	7
0.3	0.48	8
0.4	0.08	9
0.5	0.01	9
0.6	0.01	9
0.7	0.01	9
0.8	0.01	9
0.9	0.00	10

(a) Tests T2

R	$g(RF^z)$	N(Optimal)
0.0	7.85	0
0.1	1.85	3
0.2	1.75	3
0.3	1.75	3
0.4	1.52	4
0.5	1.52	4
0.6	0.40	6
0.7	0.40	6
0.8	0.19	8
0.9	0.00	10

(b) Tests T3

Table 5.6: Average Values of $g(R)$ For Tests in T2 and T3 (%)

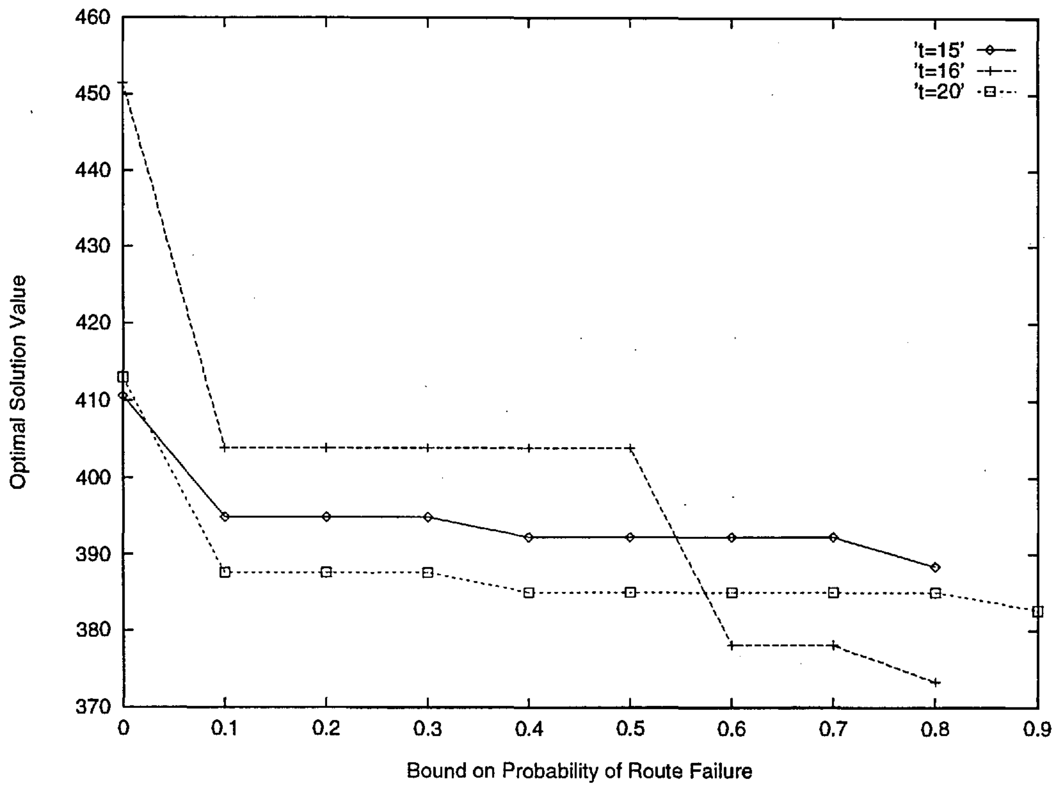

Figure 5.3: Test problems $t = 15, 16, 20$: Optimal Solution Values for Different $\overline{RF^z}$

	$\overline{RF^z} = 0.0$	$\overline{RF^z} = 0.9$
$\lvert O\rvert$ (thousands)	117.5	44.4
$\lvert I\rvert$ (millions)	19.7	5.1
cpu (seconds)	3214.4	1161.2

Table 5.7: Average Computational Results For Tests $\{t_{11}, \ldots, t_{20}\}$

Finally, it should be noted that corresponding computational times and node construction in the use of the PTSA for extension P2 is comparable with that for extension P4. This is illustrated by Table 5.7 which displays the average number of OUTER tree nodes used ($|O|$), the number of INNER tree nodes used ($|I|$) and the computation time in seconds over $\{t_{11}, \ldots, t_{20}\}$ where $\overline{RF^z} = 0.0$ and $\overline{RF^z} = 0.9$ respectively. Moreover, two larger size chance constrained VRPSD problems have been solved using the PTSA. Two problem instances of the VRP-based Tests T4, see Chapter 4, have been solved to optimality where $m = 2$, $n = 20$, $U = 0.9$ for $\overline{RF^z} = \underline{RF^z} \approx 0$. The value of $g(\underline{RF^z})$ in these two problems is 2.53% and 3.38% respectively.

5.3 BVRPSD Extensions P5/P6/P7

The "VRPSD with Knowledge" problems - P5, P6 and P7 - differ from P4 by the inclusion of specifications relating to system design and the presence of information disclosure. In P5 there exists no on-line system as only existing knowledge is required; this is further specified as a vehicle returning to the depot if its load is below the minimum possible demand of the following customer. P6 involves a similar situation to P5 but relies on a dynamic on-line system being in place. There must exist an early information system where decision points are present and the exact demand of the next customer on the route is known. P7 involves the case where demands are known after routing however before the drivers set out on their specified routes. This problem effectively develops the concept of variable recourse since drivers can "choose" where to fail along a route.

Dror and Trudeau [72] were one of the first to consider a route failure in terms of its location. They stated that previous VRPSD models that did not account for particular return trips to the depot "required modification". This, however, is mistaken since it is clear that each extension P5, P6 and P7 (and P4) are essentially different models and each has its own practical use. Primarily, the important issue is to model all the problems individually and for the first time provide a measure corresponding to how "improved" (in terms of cost) one system is compared to another. This has a practical benefit since a decision-maker may, for instance, be interested in i) training drivers to make certain decisions on a route or ii) incorporating an on-line system into the routing operation. For either of these cases, the comparative savings in total routing costs are of importance.

Clearly, if an optimal solution to P4 is represented by z^4, an optimal solution to P5 is represented by z^5 and so on, then the following conjecture holds:

$$z^4 \geq z^5 \geq z^6 \geq z^7. \tag{5.8}$$

The proof of (5.8) involves the straight-forward use of the triangular inequality and is similar to that shown in Section 2.5.5 for an example recourse cost.

Obtaining solutions to extensions P5 and P6 involve relatively simple extensions of the PTSA. In each case, the recourse cost function associated with each INNER tree node, given by (3.43) for P4, must be altered according to the particular definitions given in Table 2.3 from Section 2.6.3. P7 is more complex and only heuristic solutions can be obtained. In the following, each extension is considered in turn. A subset of Tests T2 are used to illustrate each problem and primary interest is given to optimal solution "performance" relative to that for P4, i.e. z^4.

5.3.1 BVRPSD Extension P5

For P5, (3.43) must be altered to allow for the case when a route break occurs if the load in the vehicle falls below the minimum demand of the following customer. Consider an INNER tree node λ branching to another node μ, the new recourse cost index can then be given as follows.

$$r(\mu) = \begin{cases} 0 & \text{if } l(\lambda) > \xi_{c(O(\mu))}, \\ 2c_{1c(O(\mu))} & \text{if } l(\lambda) < \xi_{c(O(\mu))} \neq \varepsilon^1_{c(O(\mu))}, \\ c_{1c(O(\mu))} + c_{1c(O(\lambda))} - c_{c(O(\lambda))c(O(\mu))} & \text{if } l(\lambda) < \xi_{c(O(\mu))} = \varepsilon^1_{c(O(\mu))}. \end{cases} \tag{5.9}$$

To test P5, in relation to P4, optimal solutions were first obtained using the original PTSA (solving P4) over the ten problem instances of T2 where $m = 1$, $n = 10, 12, 14, 16$, $U = 1.0, 1.1, 1.2$ and $m = 2$, $n = 10, 12, 14, 16$, $U = 0.9, 1.0, 1.1$. Following this, the above recourse cost index was incorporated in to the INNER tree and the same set of problems were solved to optimality using the new PTSA. The results are shown in Tables 5.8 and 5.9. The percentage cost improvement in using P5 in comparison with P4 is shown on average in Table 5.10. From this table, $z^5 \approx (0.996)z^4$.

n	U	P4	P5	P6	P7
10	1.0	291.1	290.4	287.1	286.3
	1.1	313.0	311.0	302.0	300.2
	1.2	327.5	324.7	315.3	312.5
12	1.0	317.1	316.7	313.7	313.0
	1.1	339.6	337.8	330.3	329.4
	1.2	353.4	351.8	344.7	340.5
14	1.0	345.1	344.6	341.0	340.5
	1.1	368.9	366.8	358.8	357.5
	1.2	383.2	381.7	375.2	371.3
16	1.0	356.2	355.7	352.3	351.3
	1.1	382.4	380.0	373.1	370.4
	1.2	398.7	396.3	387.8	383.8

Table 5.8: Tests T2 ($m = 1$): Average Value of the Optimal Solution

n	U	P4	P5	P6
10	0.9	360.2	359.4	354.5
	1.0	383.6	381.1	370.2
	1.1	403.6	401.7	386.3
12	0.9	383.9	383.7	381.0
	1.0	409.5	407.3	399.0
	1.1	430.3	428.1	413.8
14	0.9	410.1	409.9	407.9
	1.0	437.3	436.3	430.0
	1.1	460.4	458.5	446.4
16	0.9	422.3	422.1	419.9
	1.0	451.5	449.6	441.6
	1.1	471.4	469.3	459.5

Table 5.9: Tests T2 ($m = 2$): Average Value of the Optimal Solution

Extension	$m = 1$	$m = 2$
P4	348.0 (0.00%)	418.7 (0.00%)
P5	346.5 (0.43%)	417.2 (0.36%)
P6	340.1 (2.27%)	409.2 (2.27%)
P7	338.1 (2.84%)	-

Table 5.10: Average Cost Improvements In Comparison To P4 (%)

5.3.2 BVRPSD Extension P6

For P6, (3.43) must be altered so that whenever a route failure occurs a route break is actioned. Consider an INNER tree node λ branching to another node μ, the new recourse cost index can then be given as follows.

$$r(\mu) = \begin{cases} 0 & \text{if } l(\lambda) \geq \xi_{c(O(\mu))}, \\ c_{1c(O(\mu))} + c_{1c(O(\lambda))} - c_{ic(O(\mu))} & \text{if } l(\lambda) < \xi_{c(O(\mu))}. \end{cases} \tag{5.10}$$

To test P6, in relation to P4 and P5, we incorporated the above recourse cost index into the PTSA and ran the new algorithm over the ten problem instances of T2 where $m = 1$, $n = 10, 12, 14, 16$, $U = 1.0, 1.1, 1.2$ and $m = 2$, $n = 10, 12, 14, 16$, $U = 0.9, 1.0, 1.1$. The results are shown in Tables 5.8 and 5.9. The percentage cost improvement in using P6 in comparison with P4 is shown on average in Table 5.10. From this table, $z^6 \approx (0.98)z^4$.

5.3.3 BVRPSD Extension P7

The final extension operates in a similar manner to P6 however a vehicle can break its route at the most cost-effective point. Clearly, such a problem is very difficult to model since an element of memory must be retained so that once a route failure occurs a route can be "traversed backwards" to find the least cost route break. Moreover, there is a constraint on how far back along a route a break can be actioned since by definition a vehicle must retain enough load to service the remaining customers that follow the route failure. This convoluted situation can be illustrated with the use of a simple example.

Possible route breaks for a portion of a route are shown in Figures 5.4, 5.5 and 5.6. In all these examples, a route failure occurs at the final node. Figure 5.4 corresponds to the most simple case (P4) where a return trip back to the depot occurs. Figure 5.5 corresponds to the dynamic knowledge case (P6) where one arc of the original route is skipped. Figure 5.6 corresponds to the total knowledge case, where one arc of the original route is skipped but at the most advantageous point possible (assuming Euclidean distances between vertices). The complex issue, in terms of using the PTSA to model P7, arises because it is possible that breaking at such a point (and therefore servicing two extra customers before the failed customer) may result in not being able to service a set of subsequent customers. Since some form of dynamic memory is required in cases of more than one route to achieve an accurate

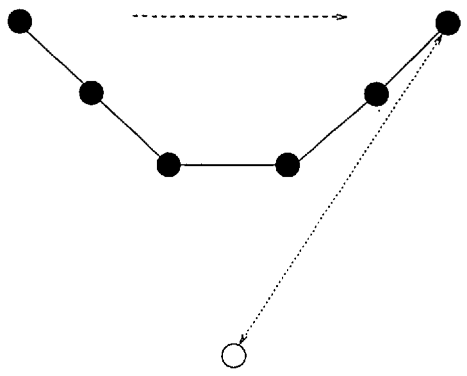

Figure 5.4: Route Break I: No Knowledge

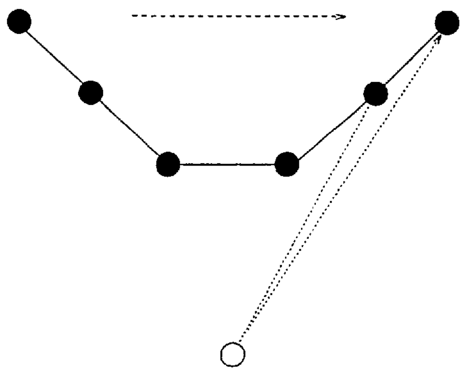

Figure 5.5: Route Break II: Dynamic Knowledge

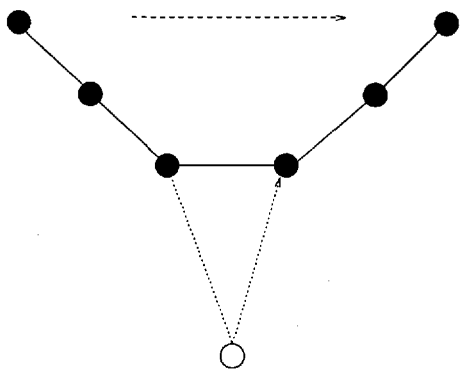

Figure 5.6: Route Break III: Total Knowledge

routing cost, it is therefore only possible at this stage to use the PTSA to model P7 where $m = 1$. In addition, the PTSA can only approximate the value of the optimal solution since an earlier route break is permitted if and only if the subsequent customers never fail. Nevertheless, a solution of this problem has never been produced before and so the corresponding heuristic will still provide an invaluable method to obtain a measure for the improvement using total knowledge over other extensions of the BVRPSD. Importantly, total knowledge provides the most cost-effective form of stochastic vehicle routing apart from VRP-based reoptimisation.

For P7, (3.43) must be altered so that whenever a route failure occurs the least cost route break is actioned such that the maximum demands of the following customers are less than total vehicle capacity. Consider an INNER tree node λ branching to another node μ and assume that an OUTER tree node ρ corresponds to a routing segment in which a set of customers S^ρ has previously been served according to the edge set E^ρ. If the set of customers still needing a service at an OUTER tree node ρ is $S^{\rho'} = V \backslash (S^\rho \cap \{v_1\})$ and each demand set $\xi_i = \{\varepsilon_i^1, \ldots, \varepsilon_i^{\delta_i}\}$ is ordered in ascending size, then the new recourse cost index would be given as follows.

$$r(\mu) = \begin{cases} 0 & \text{if } l(\lambda) \geq \xi_{c(O(\mu))}, \\ c^t & \text{if } l(\lambda) < \xi_{c(O(\mu))}. \end{cases} \tag{5.11}$$

where c^t is obtained from the following minimisation.

Let (k^*, l^*) be the value of (k, l) that minimises the expression:

$$c^t = \min_{k,l}(c_{1l} + c_{1k} - c_{kl}) \tag{5.12}$$

where

$$(k, l) \in E^{O(\mu)} \tag{5.13}$$

$$\sum_{v_i \in S^{\rho'} | c(\rho') = l^*} (\varepsilon_i^{\delta_i}) \leq Q. \tag{5.14}$$

To test P7, in relation to P4, P5 and P6, we incorporated (5.11) into the PTSA and solved the previous set of test problems, i.e. the ten problem instances of T2 where $m = 1$, $n = 10, 12, 14, 16$ and $U = 1.0, 1.1, 1.2$. The results are shown in Table 5.8. The percentage cost improvement in using P7 in comparison with P4 is shown on average in Table 5.10.

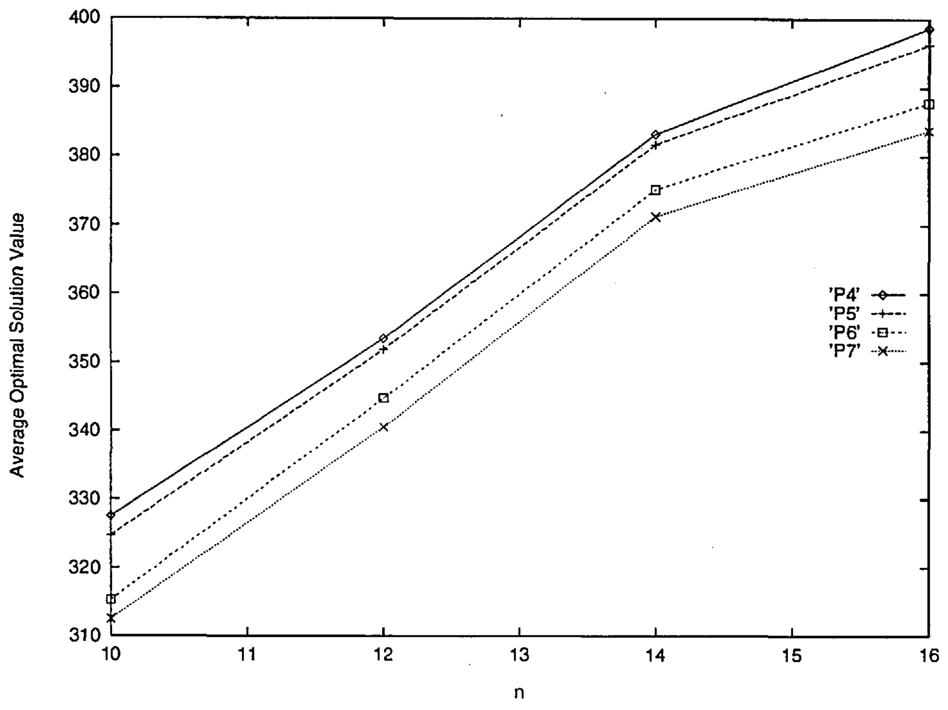

Figure 5.7: Tests T2 ($m = 1$): Average Optimal Solutions Per Extension Over n

From this table, $z^7 \approx (0.97)z^4$. Figure 5.7 displays the difference in average optimal solution cost between the four extensions over n for $m = 1$.

5.4 Summary

In this chapter, a number of extensions of the VRPSD that were first defined in Chapter 2 have been solved to optimality using adaptations of the PTSA. A number of issues have been discussed including different levels of demand-related information disclosure and route-related reliability. Clearly, the extensions are defined to cross the spectrum between serviceability and cost and, for the first time, it has been possible to quantify the relative costs of the particular trade-offs involved.

Chapter 6

A VRPSST Application in the Scheduling of Field Service Engineers

6.1 Introduction

The PTSA has been shown to be an effective procedure to find optimal solutions to the vehicle routing problem with stochastic demands. The VRPSD with Variable Costs of Recourse has been used to illustrate its capabilities and a group of alternative extensions have been analysed and solved to optimality for the first time. In this chapter, we discuss the adaptation of the PTSA to solve other SVRPs highlighted in Section 2.4. Algorithms are proposed for the VRPST and the VRPSST. Furthermore, the PTSA is employed within a VRPSST-based model applied to the scheduling of field service maintenance engineers.

6.2 The VRP With Stochastic Service Times

Let $G = (V, E)$ be a graph where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices and $E = \{(v_i, v_j) : v_i, v_j \in V\}$ is a set of edges. The vertices have known and fixed locations and every edge (v_i, v_j) has an associated non-negative cost c_{ij} and non-negative travel time t_{ij}. It is assumed that the graph is symmetrical and the matrices (c_{ij}) and (t_{ij}) satisfy the triangular inequality, i.e. (v_i, v_j) is only defined for $i < j$ and $(c_{ik} + c_{kj} \geq c_{ij}, t_{ik} + t_{kj} \geq t_{ij} \; \forall \; i, j, k)$. Vertex v_1 represents a depot at which a homogeneous fleet of m vehicles, each with an overall working (service and travel) time restriction of T, is based. The remaining vertices correspond to a set of customers where each customer v_i has associated service time requirements given by discrete, independent, non-negative random variables ξ_i with finite means μ_i and variances σ_i^2. In a first stage, a set of m vehicle routes of minimal cost are determined so that (i) each route starts and ends at the depot and (ii) each customer is visited exactly once by one vehicle. In a second stage, the first stage routes are followed as planned but whenever T is exceeded along a route, as a consequence of the deterministic travel times t_{ij} and the stochastic service times ξ_i, the vehicle returns to the depot and then continues along its pre-defined route with a replenished time allowance of T. Given that second stage recourse costs are represented by the values of such return trips to the depot, the objective is to design a minimum expected cost-set of routes such that all service time requirements are met, (i) and (ii) are satisfied and exactly m vehicles are used.

The similarities between the VRPSST as it is described here[1] and the VRPSD are clear. The only real complication arises in the incorporation of deterministic travel times in the constraint on total working time. Indeed, the VRPSST can be represented by a similar first stage and second stage as that given for the VRPSD in Section 3.2. The only alteration stems from the second stage which can be represented as follows. Consider a first-stage feasible solution characterised by the vector $x^v = [x_{ij}^v]$. Given that the problem scenario ϕ_s, $s \in \{1, \ldots, \hat{s}\}$, is realised from the random service time variable ξ, let $W(x^v) = E_{\phi_s \in \xi}(W(x^v, \phi_s))$ denote the expected second stage routing costs and let $W^k(x^v, \phi_s)$ denote the expected recourse cost of route k in x^v given ϕ_s The expected cost of K vehicle routes given a current feasible solution x^v is then simply:

$$W(x^v) = \sum_{k=1}^{K} W^k(x^v) \tag{6.1}$$

where the expected cost of any route k can be computed separately, i.e. $W^k(x^v) = E_{\phi_s \in \xi}(W^k(x^v, \phi_s))$.

Let p_i^l represent the probability that the l^{th} service time ε_i^l originates from the set of realisations $\{\varepsilon_i^1, \ldots, \varepsilon_i^l, \ldots, \varepsilon_i^{\delta_i}\}$ of customer v_i, such that $\varepsilon_i^l < \varepsilon_i^k \; \forall \; l < k$. In addition,

[1]Alternative interpretations of the recourse cost are possible; these include a fixed cost of route failure or a penalty per unit of time in excess of T along a route.

relabel the vertices of the k^{th} route of x^v so that the route becomes $(v_1, v_2, \ldots, v_{t_k}, v_{t_{k+1}} = v_1)$. By denoting g to be the remaining working time available for a vehicle upon arrival at a customer v_i, the expected cost of a route k is then as follows (the proof is similar to that given in Section 3.2.2 for P4).

$$W^k(x^v) = E_{\phi_s \in \xi}(W^k(x^v, \phi_s)) = \alpha_2^k(T) \tag{6.2}$$

where:

$$\alpha_{t_k}^k(g) = 2c_{1i} \sum_{l|\varepsilon_{t_k}^l > g} p_{t_k}^l \qquad (0 < g \leq T), \text{ and} \tag{6.3}$$

$$\alpha_i^k(g) = p_i^{l^*} \alpha_{i+1}^k(0) + \sum_{l|\varepsilon_i^l > g} p_i^l(\alpha_{i+1}^k(T - \varepsilon_i^l) + 2c_{1i}) + \ldots \tag{6.4}$$

$$+ \sum_{l|\varepsilon_i^l < g} p_i^l \alpha_{i+1}^k(g - \varepsilon_i^l) \qquad (i = 2, \ldots, t_{k-1}; 0 < g \leq T).$$

and

$$p_i^{l^*} = \begin{cases} p_i^l & \text{if } \varepsilon_i^l = g, \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

In an appropriate PTSA therefore, SDT branching occurs according to the residual working time a vehicle can have after satisfying the service time requirements of the customer in question, i.e. events equate to alternative *service time leaving* levels. The bound on the first stage remains identical to that for the VRPSD. The second stage bounds are similar but require modification. Only L2 is described here since the adaptation of L2 into the INNER tree based bound of L3 is straight-forward.

A second stage lower bound on the VRPSST can be obtained by considering the recourse problem at each OUTER tree node ρ in which a set of customers S has previously been served. Let $c(\rho)$ denote the customer associated with node ρ. Consider the minimum total service time to be satisfied via return trips to the depot at ρ. Such a quantity of time depends on the service time distributions of the remaining customers, the combined total time restriction of the remaining vehicles and the minimum travel time required to cover the remaining customer locations. If the set of customers still needing a service at node ρ is $S' = V \backslash (S \cap \{v_1\})$, the number of vehicles available is m', the minimum travel time required to visit the customers in S' is P' (a lower bound of which can be obtained using a 2-perfect matching approach) and each service time set, $\{\varepsilon_i^1, \ldots, \varepsilon_i^{\delta_i}\}$, is ordered in ascending size, then the minimum remaining working time $g(\rho)$ to be satisfied through recourse is given by:

$$g(\rho) = \begin{cases} P' + \sum_{v_k \in S'} (\varepsilon_k^1) - T(m'+1) + 1 & \text{if } c(\rho) \neq 1, \\ P' + \sum_{v_k \in S'} (\varepsilon_k^1) - T(m'+1) & \text{if } c(\rho) = 1. \end{cases} \tag{6.6}$$

If $g(\rho)$ is greater than zero then a route failure will definitely occur irrespective of how the remaining customers are routed. Indeed, the number of route failures, $f(\rho)$, that must occur while serving the remaining customers is given by $f(\rho) = \lceil g(\rho)/T \rceil$ where $\lceil * \rceil$ represents the smallest integer not less than $*$. Now, given that c_0 represents the remaining least-cost single-trip to the depot, i.e. $c_0 = \min_{v_k}[c_{1k} \mid v_k \in S']$, the lower bound L^2 is given by:

$$L^2(\rho) = \begin{cases} f(\rho).2c_0 & \text{if } g(\rho) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.7}$$

The version of the paired tree search algorithm that incorporates the above alterations and can therefore solve the VRPSST has been coded in FORTRAN and run on a Silicon Graphics Workstation Indigo R4000 (100MHz) for a range of real-life problems. The computational results can be found in the description of the applied model. Significantly, computational times and PTSA tree search structure is similar to that for the VRPSD version of the PTSA.

6.3 The VRP With Stochastic Travel Times

Let $G = (V, E)$ be a graph where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices and $E = \{(v_i, v_j) : v_i, v_j \in V\}$ is a set of edges. The vertices have known and fixed locations and every edge (v_i, v_j) has an associated non-negative cost c_{ij} and travel times given by discrete, independent, non-negative random variables ξ_{ij} with finite means μ_{ij} and variances σ_{ij}^2. It is assumed that the graph is symmetrical and the matrices $C = (c_{ij})$ and $\Xi = (\xi_{ij})$ (for average values) satisfy the triangular inequality, i.e. (v_i, v_j) is only defined for $i < j$ and $(c_{ik} + c_{kj} \geq c_{ij}, \mu_{ik} + \mu_{kj} \geq \mu_{ij} \ \forall \ i, j, k)$. Vertex v_1 represents a depot at which a homogeneous fleet of m vehicles, each with an overall working (service and travel) time restriction of T, is based. The remaining vertices correspond to a set of customers where each customer v_i has associated non-negative, service time s_i. In a first stage, a set of m vehicle routes of minimal cost are determined so that (i) each route starts and ends at the depot and (ii) each customer is visited exactly once by one vehicle. In a second stage, the first stage routes are followed as planned but whenever T is exceeded along a route, as a consequence of the deterministic service times s_i and the stochastic travel times ξ_{ij}, the vehicle returns to the depot and then continues along its pre-defined route with a replenished time allowance of T. Given that second stage recourse costs are represented by the values of such return trips to the depot, the objective is to design a minimum expected cost-set of routes such that all service time requirements are met, (i) and (ii) are satisfied and exactly m vehicles are used.

The similarities between this interpretation of the VRPST and the VRPSST given in Section 6.2 are evident. The difference arises since the stochastic and deterministic interpretation of the service and travel times have been alternated. This results in more random variables in the case of the VRPST and a minor modification in both the second stage formulation and the second stage lower bound L2. The latter is straight forward in that the interpretation of (6.2)-(6.4) is changed so that the stochastic service times, i.e. $\varepsilon_i \in \phi_s$, now include the deterministic service times and the stochastic travel times, i.e. $(\varepsilon_i \in \phi_s) + s_i$ where ε_i corresponds to the travel time up to customer i in the route k. The lower bound on the second stage would correspond to that given in Section 6.2 except that the summation of service times in (6.6) would equate to a summation of the deterministic service times and P' would correspond to a lower bound on the minimum possible travel times over the remaining customers.

The version of the paired tree search algorithm that incorporates the above alterations and can therefore solve the VRPST has not been implemented since the corresponding model is very similar in terms of complexity to the model described in Section 6.2 for the VRPSST.

6.4 The Applied Maintenance Scheduling Problem

The PTSA has been used to solve a real-life operational problem at a utility company, which has been modelled as a VRPSST. The company has a large number of major assets, including depots, work sites, buildings and machinery, and employs Field Service Engineers

(FSEs) to maintain all of these assets. FSEs are home-based and work independently in a set geographical region. An average day for a FSE involves eight hours and fifteen minutes work and overtime is paid for work completed over this allotted time. Typically, a number of jobs (usually less than ten) are completed per day at a number of alternative sites (usually less than five). Accordingly, a FSE may complete up to thirty jobs per week at up to twenty different locations.

6.4.1 The Maintenance Scheduling System in Practice

Jobs are assigned to a FSE in a variety of ways, however each job has a basic form of prioritisation and, for all but the most reactive jobs, requires some form of localised scheduling and routing. Currently, before deciding which jobs to complete each day, a FSE considers a variety of factors including:

- *priority* which is a known upper limit of time before which a job must be completed,
- *site location* which is known and fixed,
- *travel time* which is estimated based on FSE knowledge of the geographical area and local traffic systems etc., and,
- *service time* which is the length of time taken to complete a job and is estimated according to 'incomplete' knowledge and FSE experience.

In this study, FSE jobs are defined according to their associated priority and belong to one of the following three categories: (i) reactive (R) - emergency call-outs with a priority given in terms of hours, (ii) pre-planned (P) - regular jobs with a priority given in terms of months, and, (iii) unplanned (U) - irregular jobs that require some form of local prioritisation usually given in terms of days and/or weeks. The characteristics of these *job-types*, obtained from a database storing information for eight months of FSE work, are shown in Table 6.1. Specifically, column 2 shows the proportion of jobs that have been classified under a particular job type during this time period (Total Number), column 3 displays the proportion of total time spent completing jobs of a particular job type (Total Time), column 4 highlights the average duration of time taken to complete individual jobs of a particular job type (Average Service Time) and column 5 displays approximations of the upper limits of priority per job type that accord with FSE efficiency targets.

Job Type	Total Number	Total Time	Average Service Time	Priority
Pre-planned-P	59%	23%	1 hour	3 months
Unplanned-U	11%	19%	5 hours	2 weeks
Reactive-R	30%	58%	6 hours	1 hour

Table 6.1: Job Characteristics for an Engineer at the Utility Company

6.4.2 The Stochastic Problem Setting

In any stochastic environment, there exists a specific 'information' state that refers to the amount of information available to the decision makers at the time of decision making as opposed to the time when full information becomes available. In this study, the decision-makers are the engineers, decision-making refers to local scheduling/routing and full information occurs 'with hindsight' after a job is completed. The presence of a stochastic 'information state' is highlighted by the fact that: (i) service times are deemed stochastic as opposed to fixed, and, (ii) there exist inconsistencies in the way jobs are reported, e.g. a qualitative study with FSEs revealed an estimated ratio of 20:70:10 in the 'Total Number' of P, U and R jobs in contrast to the actual ratio of 59:11:30 (see Table 6.1). In practice, there also exist a variety of reoptimisation methods, i.e. operational systems, that can be utilised in such a problem environment.

Table 6.1 shows that over 50% of a FSE's work time is spent doing reactive jobs. Such jobs, however, total only 30% of all the total number of jobs completed. Indeed, as P and U jobs are large in number and have shorter duration's which are stochastic in nature they are seen as 'schedulable'. Conversely, reactive jobs are seen as uncontrollable and reducible only by improved engineering techniques and preventative maintenance, i.e. an increased number of planned 'maintenance' jobs should decrease the overall number of emergency cases.

To summarise, two uncertainties are present in the FSE scheduling/routing system. Firstly, certain maintenance jobs completed by FSEs can arise in a probabilistic manner and, secondly, the time required to complete individual jobs is unknown, i.e. the occurrence of FSE jobs can be stochastic and the completion times of FSE jobs is stochastic. Consequently, the problem of determining optimal schedules is very complex. To simplify the approach, consider a finite period of time within which a series of P, U and R jobs, have to be completed by a FSE (note that the geographical boundary of such jobs will be specified by the site locations themselves.) Specifically, one such list would start with a set of U jobs (by definition arising in the previous week) and end with a long list of P jobs. R jobs would not be included as they arise instantaneously. The basic routing/scheduling problem can then be described as follows: *if a FSE has a list of U and P jobs to complete within a finite planning horizon (e.g. a five day working week), how should those jobs be scheduled to minimise overall cost taking into account both reactive call-outs (R jobs) and the uncertain nature of job service times?*

6.5 Modelling the Applied Problem as a VRPSST

The main objective of the study was to examine the possible restructuring and refinement of the existing FSE scheduling and routing system with a view to reducing costs and/or improving productivity and the level of service associated with maintenance operations at the utility company. These issues have been addressed by developing a VRPSST-based optimisation model of the basic FSE scheduling/routing problem given above and validating the model using historical information. More specifically, the optimisation model can be used to identify optimal schedules of P/U jobs for a given FSE and, therefore, can be used to recommend the most efficient daily routes taking into account stochastic service times (and reactive call-outs). The relative performance of the model can then be evaluated by analysing existing FSE schedules, i.e. model output can be used to predict schedules based on historical information and a comparison can then be made between results obtained manually and results that could have been obtained with the use of the model. Finally, it is possible to investigate the impact of using the model at two different stages of implementation. Such analysis provides a measure of the comparative efficiency of the current 'manual' system against that of the model at two stages of practical use primarily concerned with reactive call-out recognition.

To model the FSE problem as a VRPSST, each "vehicle" corresponds to a "day" in a given planning horizon. The time restriction, T, then equates to the normal hours of each working day. In addition, VRPSST "customers" correspond to "jobs" that require a service and, as before, each job has an assigned geographical location, each route starts and ends at a fixed point (the depot) and each job is serviced on one day only, i.e. each customer is visited exactly once by one "vehicle". The time matrix represents the travel times between customer sites and the cost matrix may represent either travel time, travel cost or travel distance between customer sites. The objective of the VRPSST described in this context is then to design a minimum expected "cost-set" of routes given that recourse costs (represented by return trips back to the depot) are interpreted as the cost incurred to return to a location to complete a particular job on a day outside the planning horizon.

6.6 Model Input

The implementation of the optimisation model requires the availability of input data in the format required by the model. The primary operations involved in scheduling a FSE include the prioritisation of jobs to be scheduled, the classification of individual job times from a host of job characteristics and the inclusion of geographical site locations. Such processes are essentially independent of the model which, because of its generic nature, can be utilised in a variety of operational systems. Nevertheless, the accuracy of two of the inputs required - job locations (hence travel times/distances) and job durations (service times) - becomes the sole determinant in the exactness of the results and such data issues are, therefore, further discussed below.

6.6.1 Job-Locations and the Road Network

The processing power and storage ability of computers enable the storage on a computer of the road network for a specific geographical area. This network can be processed and used to calculate accurately the distance between any two points. Typically such computer based road maps store a large number of road junctions (identified by grid references) and details of the links (roads) between such junctions, see Beulens [30] and Beasley [14].

The results presented in this case study are based on real distances calculated between any two site locations using a real Road Network System (RNS). The scope of this system is to cover the pilot study region of the utility company and, for this reason, ten figure OS references were obtained for all pilot study-based sites and FSE home locations. The road network contains over 4000 road segments (arcs) and about 1600 intersections of road segments (nodes identified by grid references). Basically, for each pair of site locations to be considered in the routing problem, the RNS first identifies these locations in the road network and then provides accurate corresponding time, distance and path information.

The RNS stores a considerable amount of information about the links between road junctions, not only the length of the link but also the type of road that makes up the link, e.g. a motorway, A road or B road. With such information, it becomes possible to adjust for differing vehicle speeds on different types of road and, hence, to calculate accurate vehicle travel times between any two locations. Route planning, therefore, is based on actual map distances and actual average speeds used on a motorway (60mph), A road (40mph) and B road (20mph). Note that the route that gives the minimum vehicle travel time between any two sites may well be different from the minimum mileage route between the same two sites. In this study, vehicle routes are planned on the basis of a minimum vehicle travel 'cost' between sites which represents an equally weighted combination of both factors. Shortest routes between any two site locations on the road network and associated path information are determined using a shortest path algorithm, see Dijkstra [66].

6.6.2 Service Times

A standard mathematical distribution is required to describe the service time of a particular job and act as input into the VRPSST maintenance model. For our purposes, a FSE job is not defined by precise engineering detail but by what a FSE predicts a particular job to entail since, for all but the most trivial of jobs, the precise specifics of a job are unknown until the problem is diagnosed on site. Indeed, the definition of a job needs to be rich enough to take into account what is estimated by an experienced FSE whilst not defining the job explicitly. Initially, we examined the service time distributions representing the P, U and R job-types described earlier (together with more detailed job specifiers such as asset type and asset fault), using data across the whole of the utility company. The latter consisted of data for every FSE over the whole of the company for an eight month period. Totalling $65,923$ jobs and $201,841$ man-hours, this information was deemed to be broad enough to give accurate service time distributions. Nevertheless, it was found that the lack of a human factor (and indeed a regional factor) influenced the distributions with more significance than the type of jobs themselves and, accordingly, no matching mathematical distribution could be found. Figure 6.1, for example, displays the service time distribution for U jobs company wide. This hypothesis proved correct following an analysis of the service time distributions for the same job-types using data for individual FSEs. Log-normal distributions with differing means and standard deviations were found to fit with adequate statistical confidence (using the student's t-distribution test). Figure 6.2 (using information for one FSE only) shows this graphically for one particular example.

Presently therefore, when a service time distribution is required, the FSE's mean service time, and standard deviation, for a given job-type contribute to a discretised log-normal distribution that can be entered into the model. An example of such an input distribution, where there exists twelve discrete service time possibilities, is shown in Figure 6.3. Notice that for modelling purposes, a limit of 6 hours is maintained, i.e. the probability that a time above 6 hours is realised contributes to a summed discrete probability of occurrence corresponding to exactly 6 hours.

6.7 Model Output: Computational Considerations

The model was coded in FORTRAN and run on a Silicon Graphics Workstation Indigo R4000 (100MHz), see Section 6.2. The evaluation of the computational performance of the model is based on twelve scenarios (each scenario corresponding to one week's data) from one months historical data for three FSEs based in the pilot study region.

6.7.1 A Framework for the Analysis of Results

The actual input for each scenario, obtained from historical information, displays when and where each job was completed, what its priority was and how long each job took to be completed (in hours) for a given FSEs working week. The following information can be obtained from such a weekly input: (i) a list of all P and U jobs to be scheduled on Monday morning, (ii) the actual service time per day (with/without reactive call outs), (iii) the actual travel time per day (with/without reactive call outs), (iv) the actual distance travelled per day (with/without reactive call outs) and (v) the actual overtime per day (with/without reactive call outs), i.e. the time beyond 8 hours and 15 minutes.

The manual sequence of jobs completed ('scheduled') in practice by a given FSE for each day of the week will be referred to as the *manual schedule.* The corresponding optimal sequence of P/U jobs, obtained by the optimisation model at the beginning of the planning week, will be referred to as the *optimal schedule.* Note: days in the optimal schedule are not ordered over the planning period and, for that reason, a secondary ordering process is

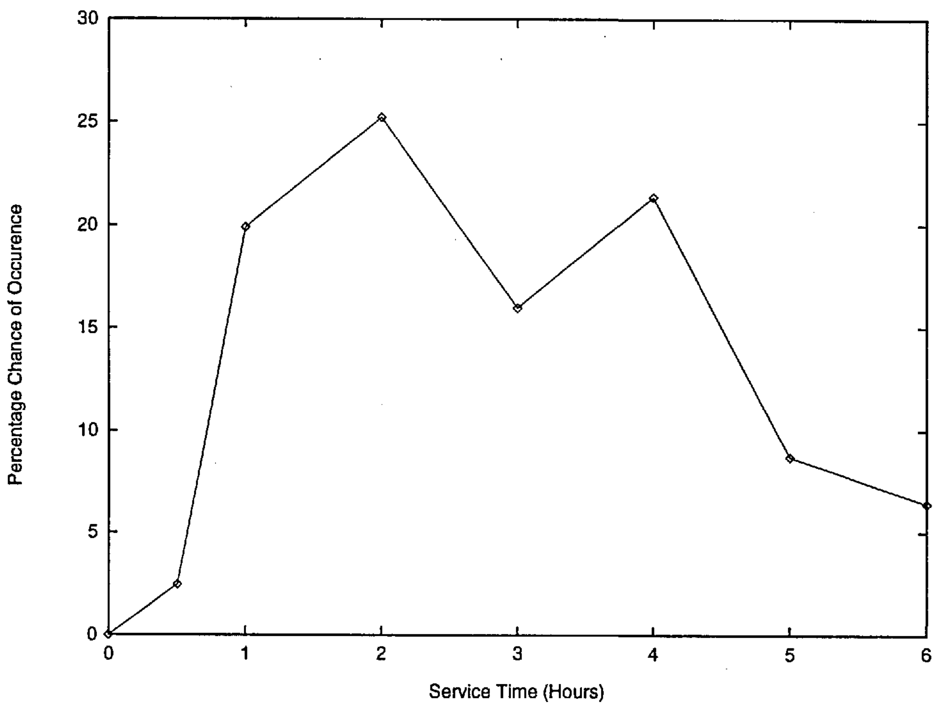

Figure 6.1: Service Time Distribution For *U* Jobs Over The Entire Company

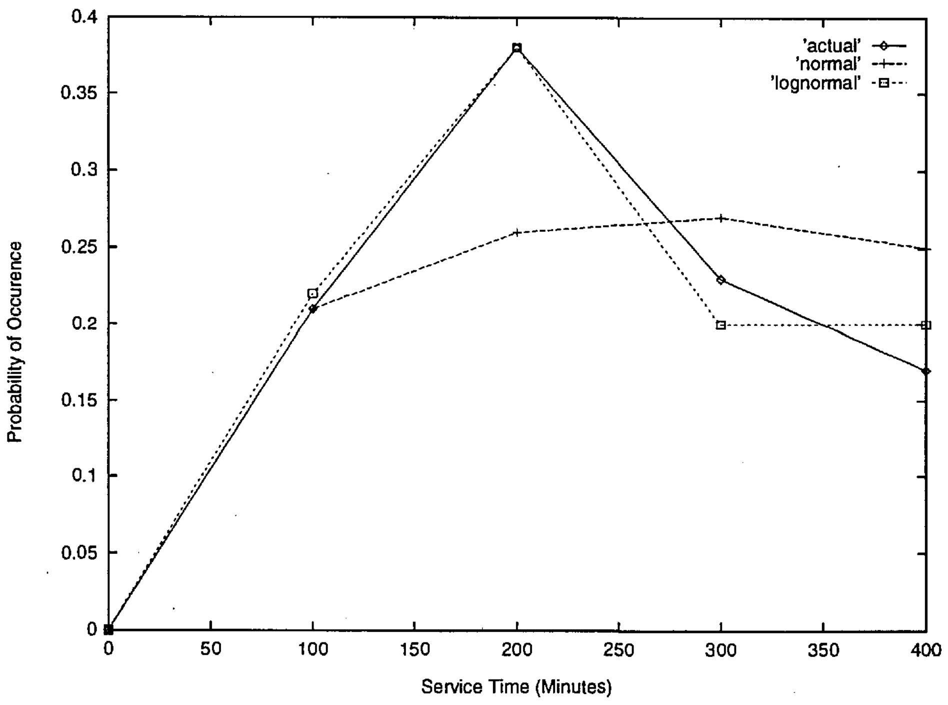

Figure 6.2: Service Time Distribution For *U* Jobs For One FSE

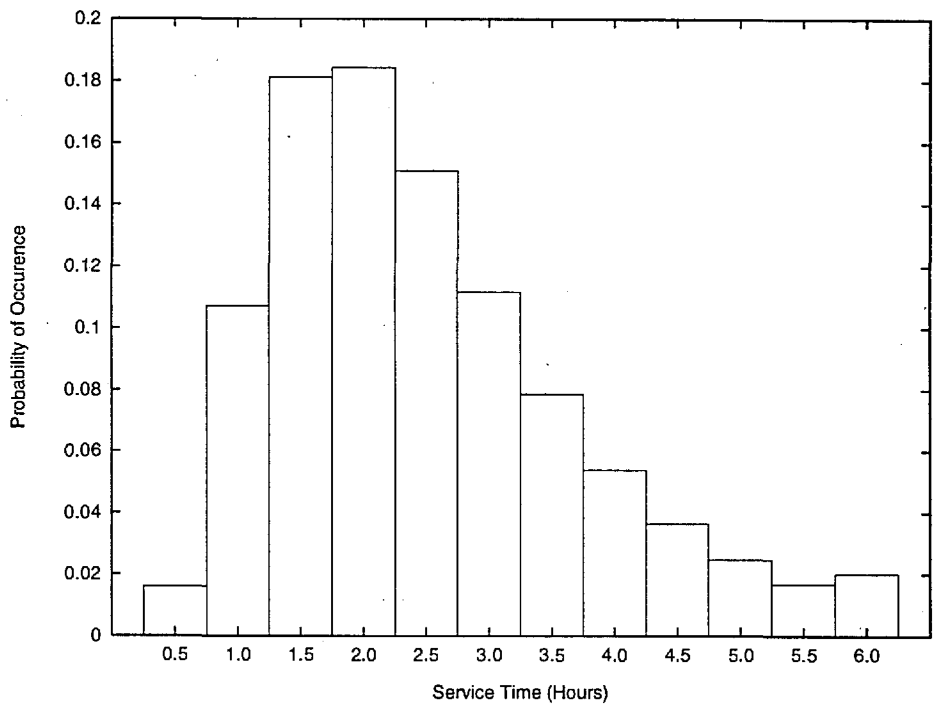

Figure 6.3: Example FSE Service Time Distribution Input

established based on the number of jobs per day and the expected service time per day.

When the actual job completion times obtained from the manual data for a particular scenario are entered into the optimal schedule, then the latter becomes the *actual-optimal schedule* which can be compared to the existing *manual schedule.* The following assumptions have been used in developing actual-optimal schedules based on the optimal model output.

1. All P/U jobs completed in the manual schedule are available for scheduling at the beginning of that week.

2. There exists a fixed limit of overtime allowed per day within the actual-optimal schedule that corresponds to the average amount of overtime used in the historical data, i.e. a job in an actual-optimal schedule will only create overtime when the total time used in that day, plus the expected time of the new job, is less than the working day plus the fixed amount of overtime allowed on average per day in the manual schedule.

3. If a FSE does not have time to complete a P/U job, even allowing for overtime, then the job will be completed at the end of any permitted subsequent daily schedule.

Conversely, if a FSE has some spare time at the end of a day then the last scheduled job in the week will be completed.

4. If a reactive call-out occurs in the historical data, then the reactive job is completed in the actual-optimal schedule and the current P/U job is abandoned to be completed later on in the week (see Section 6.7.2).

By finding the optimal schedule at the beginning of each scenario and using the above framework to compare the manual schedule with the actual-optimal schedule, it is possible to find estimates for the reduction in total travel time, distance travelled and overtime, that can be obtained by using the optimisation model instead of the manual method. In addition, by investigating the impact of implementing the model in two stages, described below, it is possible to consider the issue of reoptimisation and the 'cost' of going on-line.

6.7.2 Reoptimisation

The model initially schedules P/U jobs allowing for stochastic service times. Reactive call-outs are not included as they do not 'exist' at the time of scheduling. Indeed, one possible manifestation of the model is to exclude the contribution of reactive call-outs entirely and to only consider jobs that can be scheduled; this would correspond to a FSE system in which reactive call-outs are never encountered and would therefore partially invalidate any associated results. Two different levels of inclusion of reactive call-outs, which correspond to two different stages of model implementation, are therefore included:

- *Simple Inclusion* - Jobs occur as scheduled however when reactive call-outs arise they are implemented just as they occur in reality and when they end the original schedule is continued. No secondary optimisation is completed once the original schedule has been interrupted and so the method is similar to the manual system of dealing with reactive call-outs. [This system corresponds to one run of the optimisation model at the beginning of the planning horizon.]

- *Reoptimisation* - Jobs occur as scheduled however whenever reactive call-outs arise, and have been completed, reoptimisation occurs. [This system corresponds to the running of the model within an on-line system by, for example, continually re-running the model following the completion of a day which has included a reactive call out.]

6.8 An Example Scenario

Table 6.2 displays data for a week of FSE work in the pilot study region. Five reactive jobs occur during the week: one on Monday morning, one in the middle of the day on Monday, one on Tuesday, one on Thursday morning and one on Friday afternoon. Table 6.3 displays the corresponding inputs to the VRPSST model. Reactive jobs are omitted since they do not exist at the time of scheduling. The index column is simply used to identify inputted jobs and allows for a combination of P jobs when total expected service time is small[2]. For example, indices 7a and 7b are used to identify two small jobs that are combined to generate a single service time distribution which should be inputted into the model and represented as job 7. The expected service times (μ_i), the standard deviations (σ_i) and the number of discrete service time values (δ_i) describing each log-normal distribution of a given job are also shown.

When no reactive call-outs are considered, the output of the optimisation model is the optimal schedule shown in Table 6.4. This list would have been used to schedule the FSE in an implemented system. Notice that there is a 5.1% difference between the expected service time and the actual service time (ST) over the whole week. Using the list of assumptions given in Section 6.7.1, this optimal schedule can now be used to obtain the actual-optimal schedule shown in Table 6.5. The corresponding manual schedule is also shown in Table 6.5. Notice that the actual-optimal schedule differs from the optimal schedule due to the fact that, since extra time was available on Thursday, jobs 5 and 11 could be added to Thursday's schedule. Contrasting the actual-optimal schedule with the manual schedule, the following points can be noted: (i) the five day original schedule becomes a four day schedule in the optimised case, (ii) the spread of jobs is more even in the optimised schedule and, hence, overtime (OT) is cut from 3.17 hours to a total of 0.08 hours (a reduction of 97.5%), (iii) travel time (TT) decreases dramatically in the optimised schedule from 7.17 hours to 4.83 hours (a reduction of 32.6%), and (iv) travel distance (TD) decreases in the optimised schedule from 460 to 320 kilometres (a reduction of 30.4%).

Employing simple inclusion in the example scenario results in a 5.9% reduction in travel time, an 11% reduction in overtime and a 5% reduction in distance travelled; profiles

[2]Small jobs are defined as having a mean time of less than 30 minutes.

Day	Job-type	Service Time (hrs.)	Index*
Monday	R	1.50	-
	U	2.25	3
	R	2.50	-
	U	2.00	4
Tuesday	R	3.25	-
	U	1.50	5
Wednesday	U	5.00	2
	U	3.25	6
Thursday	R	4.00	-
	U	1.50	8
	P	0.25	7a
	P	0.25	7b
	U	2.25	12
Friday	P	0.50	11a
	P	0.25	9a
	U	2.25	10
	P	0.25	11b
	P	0.25	11c
	P	0.25	9b
	P	0.50	9c
	U	2.00	13
	U	1.00	14
	R	1.00	-

(*Index 1 represents the depot.)

Table 6.2: Actual Data For The Example Scenario

Job-type	Index (*combined jobs)	Probability Distribution		
		$\mu_i(hrs.)$	σ_i	δ_i
U	2	3.04	1.80	34
U	3	3.04	1.80	34
U	4	3.04	1.80	34
U	5	3.04	1.80	34
U	6	3.04	1.80	34
P	7*	0.61	0.46	17
U	8	3.04	1.80	34
P	9*	0.92	0.69	24
U	10	3.04	1.80	34
P	11*	0.92	0.69	24
U	12	3.04	1.80	34
U	13	3.04	1.80	34
U	14	3.04	1.80	34

Table 6.3: Input Data For The Example Scenario

	Sequence of Jobs	Expected Service Time (Hours)
M	H-7-9-10-8-H	6.40
T	H-4-14-6-H	7.30
W	H-2-13-H	4.87
T	H-12-3–H	4.87
F	H-11-5-H	3.35
		26.79

Table 6.4: The Optimal Schedule For The Example Scenario

			ST (hrs)	TT (hrs)	OT (hrs)	TD (km)
Manual schedule	M	H-3-4-H	4.25	1.33	0.00	80
	T	H-5-H	1.50	0.67	0.00	40
	W	H-2-6-H	8.25	1.00	1.00	80
	T	H-8-7a-7b-12-H	4.25	1.00	0.00	60
	F	H-11a-9a-10-11bc-9bc-13-14-H	7.25	3.17	2.17	200
			25.50	7.17	3.17	460
Actual-optimal schedule	M	H-7-9-10-8-H	5.25	1.33	0.00	110
	T	H-4-14-6-H	6.25	1.33	0.00	80
	W	H-2-13-H	7.00	0.83	0.00	60
	T	H-12-3-11-5-H	7.00	1.33	0.08	70
			25.50	4.83	0.08	320

Table 6.5: Schedules For The Example Scenario With No Reactive Call-outs

			ST (hrs)	TT (hrs)	OT (hrs)	TD (km)
Manual schedule	M	H-R-3-R-4-H	8.25	1.83	1.83	120
	T	H-R-5-H	4.75	1.00	0.00	70
	W	H-2-6-H	8.25	1.00	1.00	80
	T	H-R-8-7a-7b-12-H	8.25	1.33	1.33	70
	F	H-11a-9a-10-11bc-9bc-13-14-R-H	8.25	3.33	3.33	200
			37.75	8.50	7.50	540
Actual-optimal schedule	M	H-R-7-9-R-10-H	7.75	2.67	2.17	190
	T	H-R-4-14-6-H	9.50	1.67	2.92	100
	W	H-3-5-11-13-H	6.75	1.33	0.00	80
	T	H-R-2-H	9.00	1.00	1.75	70
	F	H-8-12-R-H	4.71	1.00	0.00	70
			37.75	7.67	6.83	510

Table 6.6: Schedules For The Example Scenario With Reoptimisation

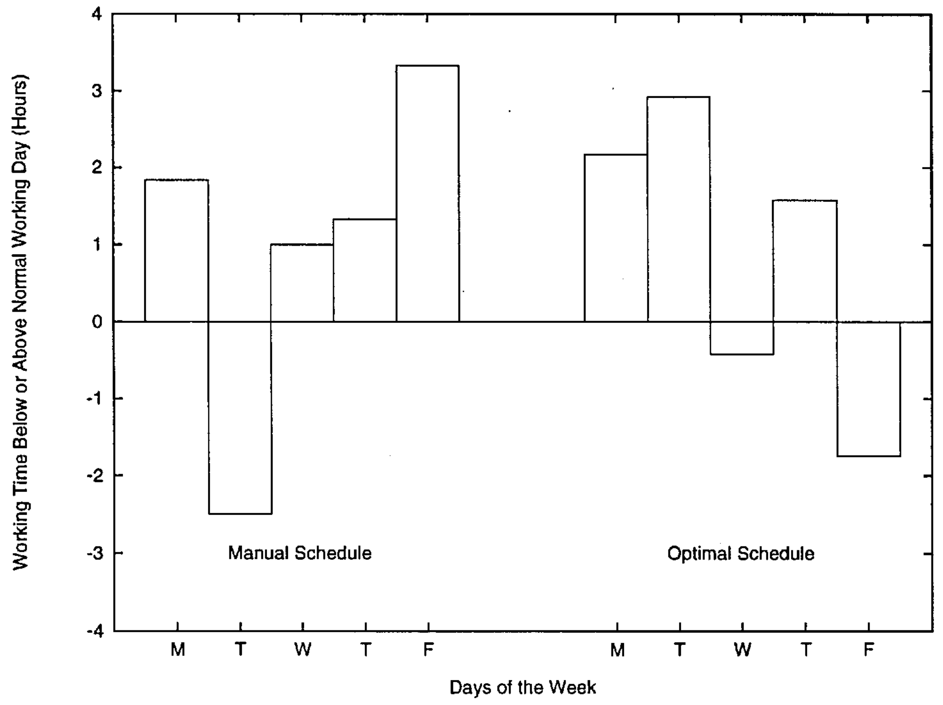

Figure 6.4: A Time Profile of Manual and Actual-Optimal Schedules

of the use of FSE time in this case are shown in Figure 6.4. Table 6.6 displays both schedules if reoptimisation is employed. The three re-runs of the VRPSST model, that occur due to reactive call-outs on Monday, Tuesday and Thursday, alter the nature of the original optimal schedule and result in a 9.8% reduction in travel time, a 8.9% reduction in overtime and a 5.6% reduction in distance travelled. Notice that, in this particular scenario, although travel time decreases under reoptimisation, the percentage reduction in overtime is slightly less than in the case of simple inclusion.

6.9 Overall Computational Results

The results for all scenarios are shown in Tables 6.7-6.9. These tables display the percentage reduction in travel time, overtime and distance travelled achieved when the optimisation model is used for each scenario in the case of no reactive call-outs, reactive call-outs with simple inclusion and reactive call-outs with reoptimisation. Computation times required to obtain the optimal schedule for each scenario, together with their associated VRPSST problem sizes, are shown in Table 6.10.

If reactive call-outs are ignored, see Table 6.7, the average results over all scenarios indicate a substantial reduction in travel time (14%) and distance travelled (18%), together

Scenario	TT (% Reduction)	OT (% Reduction)	DT (% Reduction)
1	21.4	9.8	20.3
2	0.0	26.4	5.2
3	15.3	25.0	18.0
4	6.5	17.9	14.5
5	7.1	0.0	19.2
6	6.5	81.0	9.7
7	20.6	38.6	20.0
8	12.5	0.0	18.8
9	10.7	80.0	12.5
10	25.0	0.0	28.3
11	0.0	0.0	23.1
12	32.6	97.4	30.4
Average	13.6	32.5	18.2

Table 6.7: Results: Reactive Call-outs Ignored

Scenario	TT (% Reduction)	OT (% Reduction)	DT (% Reduction)
2	0.0	0.0	3.4
3	10.9	22.9	14.9
5	2.9	14.8	6.7
6	16.7	26.2	11.1
7	21.4	30.7	21.1
8	14.7	0.0	18.8
9	5.9	44.0	7.7
10	8.3	38.8	15.7
11	0.0	2.8	8.3
12	5.9	11.1	5.6
Average	8.3	20.1	11.0

Table 6.8: Results: Reactive Call-outs - Simple Inclusion

Scenario	TT (% Reduction)	OT (% Reduction)	DT (% Reduction)
2	0.0	0.0	3.4
3	10.9	22.9	14.9
5	8.8	7.4	6.7
6	22.2	23.1	22.2
7	21.4	30.7	23.7
8	14.7	0.0	18.8
9	5.9	30.8	7.7
10	8.3	38.8	15.7
11	0.0	41.7	4.2
12	9.8	8.9	5.6
Average	9.6	19.6	11.7

Table 6.9: Results: Reactive Call-outs - Reoptimisation

Scenario	Jobs $(n-1)$	Days (m)	Computational Times (hours)
1	13	5	8.41
2	14	6	11.16
3	10	5	8.87
4	16	6	13.58
5	11	5	11.64
6	10	5	8.46
7	10	5	8.73
8	6	4	0.00
9	9	4	4.99
10	11	4	3.70
11	13	5	7.58
12	13	5	8.38

Table 6.10: Problem Data and Computation Times For Each Scenario

Reactive Call-outs	TT (% Reduction)	OT (% Reduction)	DT (% Reduction)
Ignored	10.8	19.8	14.5
Simple Inclusion	5.5	11.5	9.2
Reoptimisation	5.5	11.5	9.2

Table 6.11: Average Results: FSE 1

Reactive Call-outs	TT (% Reduction)	OT (% Reduction)	DT (% Reduction)
Ignored	11.7	29.9	16.9
Simple Inclusion	13.9	17.9	14.4
Reoptimisation	16.8	15.3	17.9

Table 6.12: Average Results: FSE 2

Reactive Call-outs	TT (% Reduction)	OT (% Reduction)	DT (% Reduction)
Ignored	17.1	44.4	23.6
Simple Inclusion	5.0	24.2	9.3
Reoptimisation	6.0	30.0	8.3

Table 6.13: Average Results: FSE 3

with a dramatic reduction in overtime (33%). Table 6.8 displays the results when reactive call-outs are implemented using simple inclusion. No results are available for scenarios 1 and 4 as no reactive call-outs occurred during these two weeks. Notice that, although the optimal schedule clearly outperforms the manual schedule, the improvements over current practice are slightly less than in the previous case (8%, 11%, and 20% respectively). This decrease is due to the fact that reactive call-outs occur at random points of the week and no secondary optimisation is completed once the original schedule has been interrupted.

Once reoptimisation is implemented only minor increases in the reductions beyond the simple inclusion case are achieved. Clearly, these results indicate that reoptimisation may be unnecessary in certain practical cases when large amounts of un-scheduled reactive jobs disrupt the optimal routing system. Nevertheless, results were only completed up to the end of the planning horizon, not on a more practical rolling weekly reoptimisation basis. Hence, depending on costs and efficiency targets, reoptimisation may still be comparatively important to management.

Finally, Tables 6.11 to 6.13 display the average improvement per FSE for all types of reactive call-out inclusion (scenario 1-4 corresponds to FSE 1, scenario 5-8 corresponds to FSE 2 and scenario 9-12 corresponds to FSE 3). It is clear that the three FSEs used in the case study incur similar results.

6.10 Summary

The modelling approach presented in this applied chapter involved the construction of a VRPSST-based model and the application of a new solution method to identify optimal schedules of jobs for a FSE and to recommend the most efficient routes for the FSE taking into account stochastic service times and reactive call-outs. The performance of the model was evaluated by analysing existing FSE schedules and investigating the impact on FSE performance of using the model at two different stages of implementation, one of which is a simulated on-line system. The results of the optimisation model show that improvements of approximately 8% and 11% in total travel time and distance travelled respectively can be achieved when stochastic service times and reactive call-outs are included in a FSEs weekly schedule. In addition, the average reduction in overtime in such cases is approximately 20%.

Chapter 7

Measuring the Performance of an A Priori Optimum Using A Posteriori Optimisation

7.1 Introduction

It was shown in Chapter 2 that the SVRP, unlike its deterministic equivalent, belongs to a class of a priori optimisation problems for which it is impractical to consider an a posteriori approach that computes an optimal solution whenever the random variables are realised, see Bertsimas *et al* [25]. Instead, an a priori approach attempts to obtain the best solution, over all possible problem scenarios, prior to the realisation of any single scenario. In this chapter, we use the PTSA to obtain optimal results for VRPSD test problems that we can use to investigate the computational performance of such an a priori approach. The quality of the a priori solutions obtained are compared with sample averages of solutions obtained using an a posteriori strategy of reoptimisation in which each problem scenario is solved as a separate VRP.

In Section 7.2, a literature review of the few similar methods employed in the area of stochastic vehicle routing is presented. In Section 7.3, the method of sampling individual VRP solutions to obtain an approximation of the a posteriori solution is outlined, together with the method with which the quality of an a priori solution is tested. Finally, comprehensive results and conclusions are given in Sections 7.4 and 7.5 respectively.

The former includes an example of the complete computational solution procedure and an investigation into the effect of different demand distributions on a priori solution quality.

7.2 Literature Review

The only other relevant work in the literature considers the VRPSC. Bertsimas *et al* [23] show, for a selection of one vehicle VRPSCs, that the gap between an a priori heuristic solution and an a posteriori approximation, based on an average of heuristic solutions for a number of associated VRP scenarios, falls between −3% and 5% (negative values were obtainable in practice due to the absence of optimality). Since an a priori solution method involves a single routing strategy, designed in advance of specific demand information though still incorporating knowledge of the demand distributions, in this study emphasis is given to the optimality of the vehicle routes obtained. The work presented here will provide an accurate evaluation of the gap between an a priori exact solution to the VRPSD and an a posteriori approximation based on an average of exact solutions for a number of VRP scenarios. This is made possible through the availability of a fast exact VRP solution method and, now, an optimal SVRP solution method. For the first time, therefore, accurate results can be obtained that attest to the performance of stochastic routing problems as relevant problems to be studied in certain contexts. The SVRP chosen to illustrate this is the VRPSD.

7.3 The A Posteriori Approximation

The purpose of this section is to describe the method used to test the a priori strategy, using results for the VRPSD obtained from the PTSA, against the a posteriori strategy, using results obtained from scenario-based VRP solutions. We first describe the deterministic VRP solution method used to solve an individual problem scenario, before describing the sampling procedure and how the quality of an a priori solution is assessed.

7.3.1 A VRP Solution Method

The exact algorithm used in this study, developed by Hadjiconstantinou *et al* [114], involves the use of lower bounds that are based on the computation of q-paths and k-paths. The use of iterative combinations of such paths results in lower bounds of high quality and a set of corresponding reduction tests reduces the size of the problem and improves the quality of the bounds even further. Embedding the lower bounds into a tree search procedure, based on a new branching strategy, enables optimal solutions to be found for problems of up to fifty customers. Further information on direct tree search VRP algorithms is given in Section 2.3.2.

Hadjiconstantinou *et al*'s algorithm was used to solve individual problems on a Silicon Graphics Workstation Indigo R4000 (100MHz).

7.3.2 The A Posteriori Solution

An exact evaluation of an a posteriori solution to a VRPSD is impractical due to prohibitive computational problems that arise following the evaluation of a huge number of VRP-based scenarios. A sampling method is therefore required to simulate the reoptimisation based, a posteriori solution. This involves creating a number of independent problem scenarios from the set of possible customer demand realisations and computing the associated probability of occurrence for each scenario.

For a given test problem, if $s \in \{1, \ldots, \bar{s} \leq \hat{s}\}$ denotes a problem scenario ϕ_s from a sampled set of size $\bar{s}$, P_s denotes the probability that a scenario s arises, $\iota(i,s)$ denotes the discrete demand level realised per customer i in scenario s, $p_i^{\iota(i,s)}$ represents the probability of the demand level indexed by $\iota(i,s)$ arising for customer i and the optimal VRP solution of s is C_s, the a posteriori approximation, A, can be given as follows:

$$A = \frac{\sum_{s=1}^{\bar{s}} P_s C_s}{\sum_{s=1}^{\bar{s}} P_s} \tag{7.1}$$

where

$$P_s = \prod_{i \geq 2} p_i^{\iota(i,s)}. \tag{7.2}$$

Note, since each scenario arises with an equal chance for test problems utilising the uniform distribution, for such problems (7.1)-(7.2) reduces to the following.

$$A = \frac{\sum_{s=1}^{\bar{s}} C_s}{\bar{s}}. \tag{7.3}$$

Needless to say, the important question in such a sampling procedure is what value the parameter $\overline{s}$ should take? In addition, note that for any one VRPSD, the value of m will vary depending on the set of problem scenarios involved.

7.3.3 The Quality of an A Priori Solution

It was shown in Chapter 5 that different knowledge-based VRPSD extensions can be more "cost-effective" than the simplest recourse extension P4. Clearly, if we denote an a posteriori solution by z^a, then we can extend the conjecture given by (5.8) from Section 5.3, referring to the optimal solutions of particular extensions to a given VRPSD, in the following way.

$$z^4 \geq z^5 \geq z^6 \geq z^7 \geq z^a. \tag{7.4}$$

In this study, both extensions P4 and P6 are used to evaluate the performance of a VRPSD solution. To obtain the a priori solution, P, the PTSA is used to solve a number of VRPSD (P4 or P6) corresponding to the values of m (number of vehicles) used in the associated sampling procedure. The value of P is then given by $\min_m z^4$ and $\min_m z^6$ for P4 and P6 respectively.

The quality of the a priori solution is then measured by the gap G between the a priori solution and the a posteriori solution. This is denoted by $G4$ for extension P4 and $G6$ for extension P6 and can be given as follows.

$$G = \frac{P - A}{A}. \tag{7.5}$$

7.4 The Computational Study

In the following section, the type of test problems used to find G are described, a fully worked example is given as an illustration of the computational procedure involved and full computational results are presented.

7.4.1 The Test Problems

The data for the tests used in this study are subsets of the well-known 50 and 75 customer VRP test problems given in Eilon et al [14]. The same tests were also used as the basis to Tests T4 in Chapter 4. For each test set, summarised in Table 7.1 and detailed

Test Set	$n-1$	Q	Customers
Test 1a	15	55	$v_2, \ldots, v_{16}$
Test 1b	20	58	$v_{12}, \ldots, v_{31}$
Test 1c	25	48	$v_{17}, \ldots, v_{41}$
Test 2a	15	56	$v_2, \ldots, v_{16}$
Test 2b	20	71	$v_{12}, \ldots, v_{31}$
Test 2c	30	68	$v_2, \ldots, v_{31}$

Table 7.1: Test Problems Taken From The Literature

in Appendix B (test 1 data is taken from Eilon *et al*'s 50-customer problem and test 2 data is taken from their 75-customer problem), a number of alternative problems have been created according to different combinations of the vehicle capacity Q and the customer demand distribution. In addition, the number of vehicles used in each problem is quantified.

Demand distributions: Customer demand distributions took the form of either the discrete uniform, the discretised normal or the Poisson. The mean demand μ_i per customer in each test set remained identical across all distribution types. In the uniform case, δ_i demand values were generated according to a spread of one third around the mean, i.e. a range of demand values from $\lfloor 2\mu_i/3 \rfloor$ to $\lfloor 4\mu_i/3 \rfloor + 1$ was created where each demand has a probability of occurrence given by $p_i^l = 1/\delta_i \; \forall \; i \geq 2, \; l \in \{1, \ldots, \delta_i\}$, and $\lfloor x \rfloor$ represents the highest integer not greater than x. As before, for the Poisson and discretised normal case, demand values per customer were generated so that $p_i^l \geq 0.01 \; \forall \; i \geq 2, \; l \in \{1, \ldots, \delta_i\}$.

Vehicle Capacity: For certain problems, vehicle capacity was altered so that three possible values for each test set varied around the original values given in Table 7.1. Varying Q in this manner effects problem tightness and alters the expected vehicle utilisation.

Vehicle Number: In the a posteriori strategy of reoptimisation, m varied over the range of problem scenarios generated in each sampling procedure. The value of m in a VRP-based solution to a single scenario was given by the minimum number of vehicles required to satisfy the customer demand requirements of that scenario. Conversely, for a single solution to a VRPSD, the number of vehicles available is fixed. To obtain the best a priori solution for a given test problem therefore, m was varied within the range of possible values used in the associated a posteriori approach, provided such a value had a frequency of occurrence within the sampled VRP scenarios which was greater than 10%.

i	μ_i	δ_i	ε_i^1	$\varepsilon_i^{\delta_i}$
2	7	7	4	10
3	30	22	20	41
4	16	13	10	22
5	9	8	6	13
6	21	16	14	29
7	15	12	10	21
8	19	15	12	26
9	23	17	15	31
10	11	9	7	15
11	5	5	3	7
12	19	15	12	26
13	29	21	19	39
14	23	17	15	31
15	21	16	14	29
16	10	9	6	14

Table 7.2: Example Demand Characteristics For A Test Set 1a Problem

The best value of the VRPSD optimal solution for a given m was then selected as P.

7.4.2 An Example of the Computational Procedure

The basic problem arising from Test 1a is the 15 customer problem with $Q = 55$ and a discrete uniform demand distribution. The mean demands μ_i, the number of demand values δ_i, the minimum demand ε_i^1 and the maximum demand $\varepsilon_i^{\delta_i}$ for each customer in this example are shown in Table 7.2. To obtain the a posteriori solution, 211 scenarios were sampled from a possible total of 2.721×10^{16} and each scenario was then solved independently as a VRP. A graph of the solutions to these scenarios is shown in Figure 7.1. Scenarios with high values of U usually incurred higher costs and required six vehicles whereas "easier" scenarios incurred lower costs and required five vehicles. This increase in cost can clearly be seen in the figure where scenarios numbering over 150 correspond to problems requiring six vehicles. Figure 7.2 displays the moving (weighted) average of VRP solutions for the example, together with the final a posteriori approximation given by $A = 333.1$. It was found that a choice of $N \approx 200$ provided a good approximation of the a posteriori solution. This observation was based on the fact that the moving average of the VRP cost for all test problems converged to a gap of approximately $\pm 0.4\%$ away from the overall average once $\overline{s}$ was greater than 100.

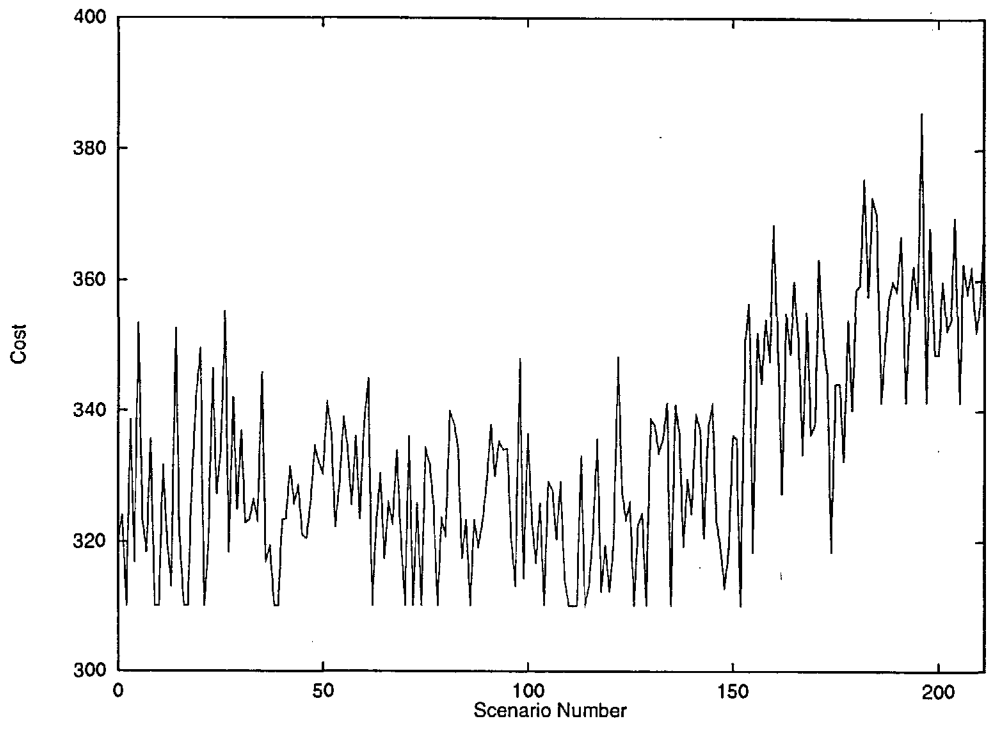

Figure 7.1: Individual VRP Solutions for the Example

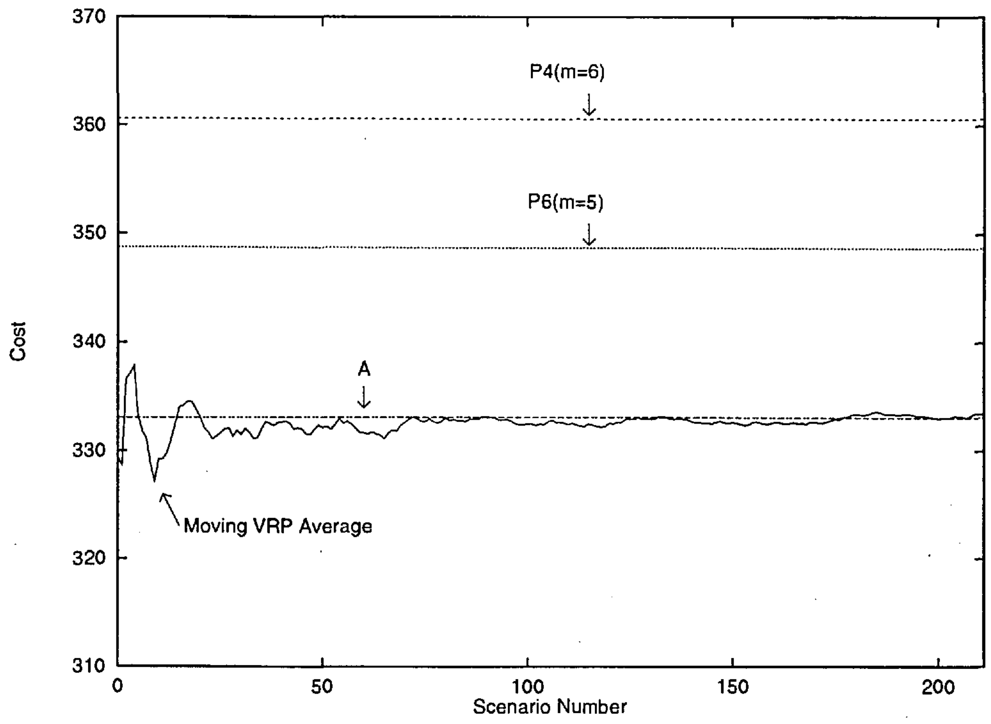

Figure 7.2: The Moving Average For The Example

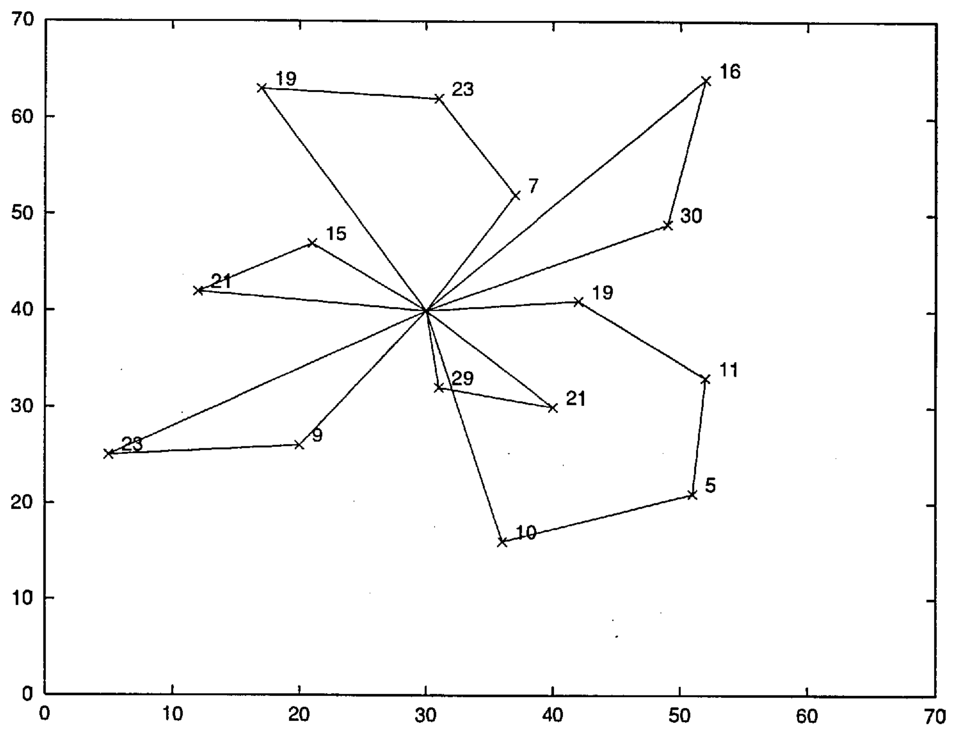

Figure 7.3: The A Priori Solution with Simple Recourse ($P4 = 360.6$, $G4 = 8.26\%$)

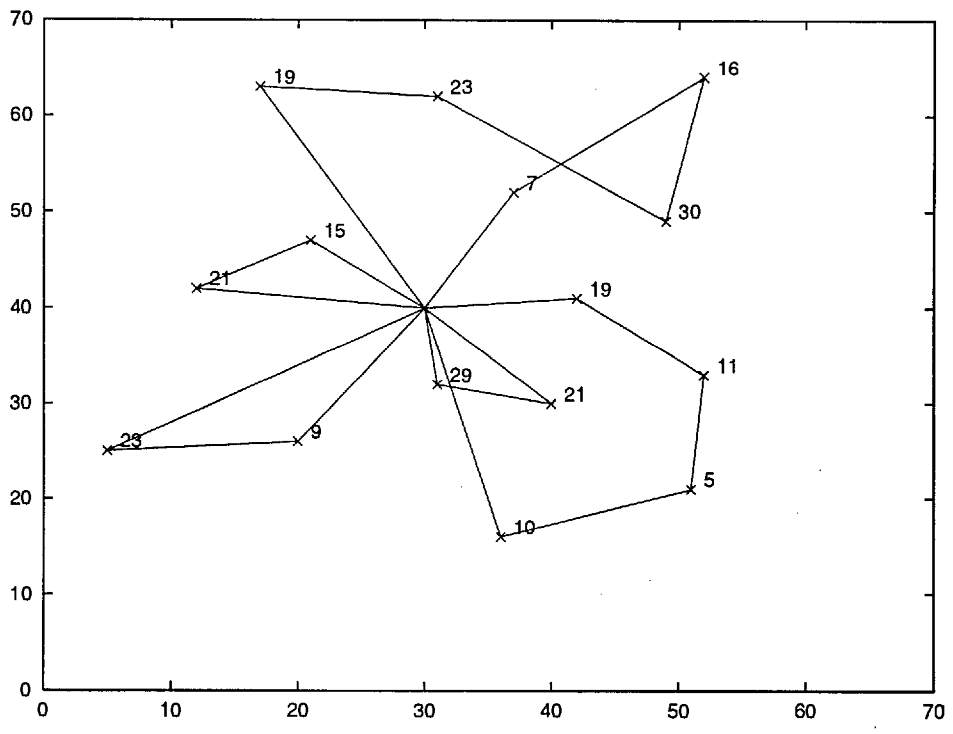

Figure 7.4: The A Priori Solution with Dynamic Recourse ($P6 = 348.7$, $G6 = 4.68\%$)

Test Set	$n-1$	Q	m	$\%U$	$\%\overline{s}$	A	P4	G4
1a	15	55	5	94	75	326.7	361.3	
			6	78	25	352.8	360.6	
			Total			333.1	360.6	8.26
2a	15	56	5		3	337.0		
			6		86	360.6	398.7	
			7		11	395.6	401.3	
			Total			364.0	398.7	9.53
1b	20	58	5	113	1	389.2		
			6	95	81	436.5	469.4	
			7	81	18	465.8	478.4	
			Total			441.6	469.4	6.30
2b	20	71	5		1	382.6		
			6		78	413.9	470.7	
			7		21	467.5	477.4	
			Total			425.0	470.7	10.75
1c	25	48	7	90	1	582.6	-	
			8	79	72	628.5	722.6	
			9	70	27	665.1	733.2	
			Total			637.8	722.6	13.30
2c	30	68	8	108	2	575.8	-	
			9	96	64	601.3	693.7	
			10	87	34	624.4	718.9	
			Total			608.8	693.7	13.95

Table 7.3: Results of the Basic Test Problems

To obtain the two a priori solutions, for P4 and P6, the PTSA was run twice for $m = 5$, with an optimal cost of $z^4 = 361.3$ ($z^6 = 348.7$), and $m = 6$, with an optimal cost of $z^4 = 360.6$ ($z^6 = 357.4$). The latter was then chosen to be P4 and the former to be P6. These solutions are also shown in Figure 7.2 and their associated routes are displayed in Figures 7.3 and 7.4. Using (7.5), $G4$ was found to be equal to 8.26% and $G6$ was found to be equal to 4.68%. This means that implementing the solution to P6 would result in an overall cost that was less than 5% away from the theoretical best that can be achieved when full information is available.

7.4.3 Results

A summary of the results to all the uniform demand problems for extension P4 with the original capacities is shown in Table 7.3. In each case $\overline{s} \geq 200$. Note that for problems involving more than twenty customers, the PTSA requires a high computational time to find an optimal solution. In such cases, the a priori solution is given by the best feasible solution obtained within an imposed time limit. Since the PTSA is an exact solution method, it is important to observe that any feasible solution obtained in such a manner is always of known quality.

Overall for these problems, the quality of the a priori solution, represented by G, varies between 6% and 14%. In the majority of cases, the smallest allowable value of m corresponds to the chosen a priori solution. In examples such as these, additional computational work should be completed to assess the serviceability of such routing solutions since a small increase in cost may result in a considerable reduction in the percentage chance of route failure along any one route. As expected, the a posteriori solution increased in parallel with an increase in m.

To test the effect of differing values of U on the quality of results, Q was varied for tests 1a and 1b. The results shown in Table 7.4 highlight that changing Q makes very little difference to the quality of results. The average gap in these problems, where $n = 15, 20$, is less than 6%. In addition, results corresponding to different demand distributions over tests 1a, 1b, 2a and 2b can be seen in Table 7.5. As expected, the VRPSDs requiring the highest cost occur when normal distributions are present as these are more "stochastic". Poisson and uniform distributions appear to incur similar costs. Overall, the average gap increases from 6.7% to 7.3% to 8.8% as the distributions change from Poisson, to uniform, to normal respectively.

Finally, we present comparative results for extensions P4 and P6. All the 15 customer tests were run for P6, including those for different demand characteristics. The comparative percentage gaps are shown in Table 7.6. When P6 is employed, the average reduction in the gap between the a priori solution and the a posteriori solution is approximately 3%.

7.5 Summary

The work in this chapter has shown that stochastic vehicle routing problems, in this case where demand is uncertain, can have a very useful role in a large number of practical situations when an organisation does not possess real time parameter information, does not possess enough flexibility in its labour assignments and/or cannot analyse data in an on-line manner. Following a simple a priori strategy involves a single VRPSD solution

Test Set	$n-1$	Q	A	P4(m)	G4
1a	15	50	355.5	378.1(5)	6.36
		55	333.1	360.6(6)	8.26
		60	319.9	333.4(5)	3.26
1b	20	52	472.8	497.6(6)	5.25
		58	441.6	469.4(6)	6.30
		64	421.3	444.4(5)	5.48

Table 7.4: The Effect of Q on the Gap

Test Set	Dist.	$n-1$	Q	A	P4(m)	G4
1a	U	15	55	333.1	360.6(6)	8.26
	N			342.1	368.1(6)	7.60
	P			325.2	354.0(5)	8.86
2a	U	15	56	364.0	398.7(6)	9.53
	N			363.4	393.5(5)	8.28
	P			367.2	390.3(7)	6.29
1b	U	20	58	441.6	469.4(6)	6.30
	N			445.3	489.8(7)	9.99
	P			447.5	467.8(6)	4.54
2b	U	20	71	425.0	470.7(6)	10.75
	N			419.3	446.6(6)	6.51
	P			411.4	440.4(6)	7.05

Table 7.5: The Effect of the Distribution Types on the Gap

Test Set	Dist.	$n-1$	Q	A	P4(m)	G4	P6(m)	G6
1a	U	15	55	333.1	360.6(6)	8.26	348.7(5)	4.68
	N			342.1	368.1(6)	7.60	355.4(5)	3.89
	P			325.2	354.0(5)	8.86	347.1(5)	6.73
2a	U	15	56	364.0	398.7(6)	9.53	377.9(6)	3.82
	N			363.4	393.5(5)	8.28	367.6(5)	1.16
	P			367.2	390.3(7)	6.29	373.4(7)	1.69

Table 7.6: Results of the Tests For P4 and P6

which is comparatively easy to implement and can result in only a small increase in cost in comparison to an a posteriori, reoptimisation-based, strategy. For problems of medium size, an average gap of 8% (no more than 14%) occurs between an optimal a priori solution and an a posteriori approximation. Furthermore, these results only account for the most simple VRPSD that incorporates a fixed cost of route failure with no route breaks. In general, incorporating a VRPSD with dynamic knowledge reduces the quality gap by approximately 3%.

Chapter 8

Conclusions

In this final chapter, we will briefly summarise the entire thesis and highlight the main contributions derived from our research. We will then consider some issues related to this research and discuss some current limitations of our proposed methods. Finally, some potential avenues for future research are suggested.

8.1 Summary of the thesis

This thesis began by describing a class of difficult Combinatorial Optimisation Problems (COPs) known as Vehicle Routing Problems (VRPs). Such problems have enjoyed an intensive period of research over the past two or three decades in an effort to provide practically useful solutions. The outcome has been a variety of exact and heuristic solution techniques. Exact methods aim to provide optimal solutions, but they are often limited to solving problems of small sizes or of particular structures. Heuristic methods, on the other hand, seek to provide approximate solutions for large-sized problems under limited time constraints, but the quality of the solutions they obtain can sometimes be very poor.

In the classical definition of the VRP, it is assumed that the associated parameters such as cost, customer demands and vehicle travel times, are deterministic. This conjecture is often too simplistic in today's dynamic environment where there exists increasing requirements on levels of productivity and service and a corresponding commitment to enlarged and more elaborate transportation systems. In parallel with the need to manage such a growing number of indeterminate systems there exists an increased amount of data augmentation and volatility. Hence, due to insufficient capital, a lack of computing facilities and/or fixed labour assignments, deterministic models cannot always be implemented and stochastic models need to be considered.

The Stochastic Vehicle Routing Problem (SVRP) differs from the VRP by the introduction of some element of variability within the system in question. Unlike its deterministic equivalent, the SVRP belongs to a class of a priori optimisation problems for which it is impractical to consider an a posteriori approach that computes an optimal solution whenever the random variables are realised. Instead, an a priori approach attempts to obtain the best solution, over all possible problem scenarios, prior to the realisation of any single scenario. Needless to say, the SVRP is particularly complex and few related solution methods, reviews or surveys exist in the literature. Indeed, the broad mission of this thesis was to fill this gap and to investigate ways to overcome the inherent difficulties found in examining stochastic problems of this type.

Following a necessarily comprehensive review of the VRP, individual SVRP variations were classified. These included the Vehicle Routing Problem with Stochastic Demands (VRPSD), the Vehicle Routing Problem with Stochastic Customers (VRPSC) and the Vehicle Routing Problem with Stochastic Service Times (VRPSST). Using the VRPSD as the main illustration, many concepts unique to stochastic vehicle routing, such as route failure, route breaks, recourse costs and information disclosure, were introduced in an attempt to classify a stochastic routing environment. Seven practical extensions of the VRPSD were defined (P1-P7). These extensions were then used throughout the thesis as an applicable and relevant group of VRPSD models that could provide the basis to a similar study for any other SVRP. To finish the classification and review process, a survey of all stochastic routing research was completed. Only one exact SVRP method could be found in the literature and, for the VRPSD, complete solutions had been found for up to nine customers for a broad class of computationally difficult problems.

The main objective of the research was to develop a new SVRP solution algorithm. For the following reasons, the algorithm was designed to provide exact solutions at the expense of computational time (and an alternative heuristic approach). Firstly, SVRP is a young problem and few optimal methods, which could enable a learning process around the problem and generate a series of original benchmark tests, had previously been devised. Secondly, since a priori routing problems involve a fixed routing strategy in a problem environment that is usually stochastic over time, solving stochastic problems in real time is rarely necessary. Thirdly, optimisation is increasingly being considered as a more practical approach for real problems due to decreased costs of computation. Finally, heuristic attempts to solve stochastic problems are often highly sub-optimal due to the shape of optimal solutions often being counter intuitive from a geometric standpoint.

Although the algorithm was originally developed to solve one extension of the VRPSD (P4), the process was designed to be as generic as possible to allow for solving alternative VRPSD extensions, other SVRPs and an applied case study.

The VRPSD was formulated using a two-stage stochastic programming approach. The primary difficulty involved with such an approach is to adequately represent the second stage problem and this was achieved using a new recursive-based formulation that could compute the value of the recourse cost function given a fixed first-stage set of routes. A new algorithm known as the Paired Tree Search Algorithm (PTSA) was developed to obtain optimal solutions to extension P4 for the first time. The primary operation of the algorithm, based around a stochastic branch and bound methodology, was to dynamically account for stochasticity (INNER tree) and simultaneously maintain the basic structure of a normal branch and bound (OUTER tree). Rebranching was introduced as a method to limit both computational time and the number of INNER tree nodes. After the first development phase, small size VRPSDs were solved without the use of any bounding procedures.

Following the initial construction of the algorithm, three lower bounds were introduced for use within the PTSA. Firstly, a 2-perfect matching based lower bound on the first stage problem (m-TSP) was implemented on the OUTER tree. This bound, which was most useful in reducing computational time, involved the generation of a transformed incremental graph before solving an associated 1-perfect matching problem. A new lower bound on the second stage recourse problem was also introduced on the OUTER tree. This lower bound was then applied successively on the INNER tree to limit computational time and the concurrent production of INNER tree nodes. Due to the success of the PTSA in providing an adequate solution structure and the ability of the bounds to speed up the search process, computationally difficult medium size VRPSDs ($n \leq 20$) were solved to optimality for the first time. A large number of problems were used to test the quality of the bounds and the computational performance of the PTSA including randomly generated problems, VRP-based benchmark problems and problems incorporating a large amount of stochasticity. In addition, the PTSA is perfectly capable of providing good approximate solutions to large VRPSDs that involve too great a computational effort to be solved exactly.

The remainder of the thesis involved adapting and applying the PTSA for a range of problems. We had previously been applying the PTSA to solve a single extension of the VRPSD, therefore we began by considering alternative extensions that incorporated different levels of demand-related information disclosure and route-related reliability. By solving these extensions (P1-P7) a complete picture of one particular SVRP could be presented for the first time. Empirical observations compared each extension and showed how optimal routing costs varied across the spectrum between maximum and minimum serviceability in a stochastic routing environment.

Following the introduction of a number of other SVRPs, the adaptation of the PTSA for the Vehicle Routing Problem With Stochastic Travel Times (VRPST) and the VRPSST was described. Only minor changes were required to adapt the algorithm and the original computational times observed for the VRPSD remained largely unchanged. Indeed, the PTSA was used to solve an operational problem, modelled using the VRPSST, at a utility company. Optimal schedules for maintenance engineers working in a stochastic environment were produced and, by using the PTSA, large savings of approximately 10% in travel time and travel distance were realised. Lastly, we considered the issue of reoptimisation. Using the PTSA, an empirical study was completed that assessed the usefulness of an a priori approach compared to an a posteriori approach (simulated by solving a sampled number of deterministic VRPs). It was found that the SVRP approach provided an excellent alternative to the reoptimisation strategy. For example, employing a VRPSD with dynamic knowledge instead of reoptimisation reduces the quality of the a posteriori solution by only 3%.

The main contributions of the work contained in this thesis can, therefore, be summarised as follows:

- *A new Paired Tree Search Algorithm (PTSA)* that can provide optimal solutions to solve a range of stochastic vehicle routing problems.

- *New lower bounds*, developed in conjunction with the PTSA, that enable larger problems to be solved with the use of less computational time.

- *A framework for studying SVRPs* that involved a full review of existing methods and the formation and solution of a range of problem extensions.
- *A SVRP application* that showed that through the use of a thorough and original modelling process, the study of complex stochastic problems can be particularly beneficial in an applied context.
- *A comparison between a priori and a posteriori optimisation* that showed the performance of a stochastic solution against that of a "best-case" reoptimisation strategy.

8.2 Suggestions for further research

Suggestions for further research fall into two broad areas. One, based around computational improvements, concurs with a number of current PTSA-based algorithmic limitations, while the other involves a more general furtherance of study that only limited research time prevented us completing ourselves.

8.2.1 Further Computational Study

The PTSA was developed over time in parallel with a considerable learning process. Not only is the algorithm original in its coding but also in its structure and design and, to our knowledge, such a stochastic-handling technique has never before been utilised. As a result of such learning-based algorithmic development, a number of computational improvements could, in future, be made to the coding of the algorithm that would improve both computational efficiency and, most importantly, computational time. Two obvious examples of this correspond to the construction and implementation of the INNER tree. The current structure is fairly dynamic and it may be more efficient to use a pointer system that references INNER tree nodes without the need for an explicit search. Moreover, following the introduction of rebranching on the INNER tree, it may be possible to transfer from the OUTER tree to the rebranched INNER tree in a more direct fashion.

In addition to improvements to the INNER tree, the more orthodox OUTER tree could be improved through the enhanced computation of the first stage lower bound L1. Currently, the matching-based bound is calculated in a particularly time-consuming manner. It may be possible to improve this process through alternative computational techniques.

8.2.2 Further Research Directions

There are two main areas of further work which would be beneficial to the research community. The first concerns the generalisation and theoretical documentation of the work contained in this thesis and the second involves more empirical work and the adaptation of the PTSA for alternative SVRPs and, moreover, different stochastic optimisation problems. We consider these areas separately below.

Theoretical Study

The majority of this thesis has been concerned with the construction of an algorithm and a series of empirical investigations to test its applicability and its ability to teach us more about stochastic routing environments. Further theoretical work on the behaviour of the algorithm could be completed and systematic research could be undertaken in order to broaden understanding of the behaviour of such a stochastic solution method in a more general environment. We propose that the PTSA could be a new stand alone, general stochastic solution method. In this study, the problem used to highlight such the method was predominantly the VRPSD, however, through follow-up theoretical work, it is hoped that the construct of the algorithm is such that the basic approach could be used to solve many other stochastic optimisation problems.

Empirical Study

Following on from the work in this thesis, and perhaps using a more theoretical and generalised foundation of the PTSA, the amount of empirical studies and applications that could be developed using the same methodology are practically endless. Not only could all the SVRP variations be solved in a similar manner, many practical applications involving stochasticity could be modelled, e.g. finance.

Appendix A

The VRPSD Benchmark Test Problems

A.1 Test Data I

Test Data I is made up of 10 sets of 40 randomly generated vertices, each with a set of randomly generated demands. Each vertex is generated in the $[1, 99]^2$ square according to a continuous uniform distribution and customers are arbitrarily assigned mean demands of $\mu_i \in \{5, 10, 15\}$. A uniform distribution is utilised and three values of demand are randomly generated in the intervals $[1, 9]$, $[5, 15]$ and $[10, 20]$ where the probability of each customer demand value arising is identical, i.e. $p_i^l = 1/\delta_i = 1/3 \ \forall \ i$ and l. Tables A.1 and A.2 display the (x, y) coordinates and the possible demand values for the vertices in each of the ten tests.

A.2 Test Data II

Test Data II is made up of 10 sets of 40 randomly generated vertices, each with a set of randomly generated demands. Each vertex is generated in the $[1, 99]^2$ square according to a continuous uniform distribution and customers are arbitrarily assigned mean demands of $\mu_i \in \{5, 10, 15\}$. Poisson demands are utilised and $\delta_i = 11, 16, 19$ for each customer group since p_i^l was set to be not less than 0.01. The demand sets for different values of μ_i are given in Tables A.3, A.4 and A.5 respectively. Table A.6 displays the (x, y) coordinates and the corresponding mean demand value for the vertices in each of the ten tests.

	Test 1					Test 2					Test 3					Test 4					Test 5				
i	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3
1	29	54	8	9	12	44	63	5	6	10	81	68	6	7	10	36	60	9	11	15	76	49	9	10	12
2	36	77	14	15	17	7	30	12	13	15	57	37	11	16	18	90	13	11	12	17	52	61	11	13	15
3	45	91	2	4	8	20	54	1	7	8	88	49	5	6	9	11	17	2	3	9	38	13	2	3	5
4	37	16	8	11	15	23	58	10	11	14	93	15	6	7	14	11	81	7	10	13	87	69	8	10	13
5	8	33	11	12	13	95	57	15	17	18	77	10	11	12	18	97	44	15	18	19	51	17	13	18	20
6	45	38	2	3	5	87	34	6	7	8	98	21	3	7	10	49	85	1	2	5	98	24	1	6	9
7	98	90	5	13	14	96	24	6	10	13	89	64	6	10	13	37	86	6	7	13	85	40	11	13	14
8	83	25	11	12	17	75	74	15	16	18	69	11	14	15	17	74	9	13	14	15	10	39	18	19	20
9	4	84	1	5	8	7	56	2	8	10	91	3	1	2	6	39	74	2	4	6	27	86	5	8	9
10	54	69	6	11	12	75	18	5	7	15	71	61	6	12	15	57	73	8	9	12	25	47	10	12	13
11	26	98	12	14	17	28	93	14	17	18	44	46	15	16	17	28	36	13	17	18	39	2	12	14	17
12	89	53	4	9	10	61	70	1	2	8	11	37	1	5	10	88	28	1	2	4	98	81	1	5	7
13	98	12	7	10	11	61	4	5	6	13	26	74	6	12	14	5	47	5	9	10	65	95	5	11	14
14	11	78	12	14	20	77	32	15	16	17	55	72	13	17	19	83	15	16	18	20	59	37	17	18	20
15	63	98	1	2	5	15	42	5	7	9	7	39	2	4	5	73	5	1	3	7	47	71	1	2	4
16	20	72	6	8	11	22	20	8	11	13	28	27	5	11	15	17	92	6	10	12	44	45	5	12	15
17	83	45	16	17	19	95	85	16	17	20	15	54	12	14	17	67	54	15	16	20	2	74	14	15	18
18	47	90	2	5	9	48	12	4	8	9	33	99	2	3	8	22	34	4	5	9	49	51	5	7	8
19	84	4	5	8	11	91	42	5	10	15	59	37	11	13	15	46	38	6	7	11	95	7	10	14	15
20	65	8	12	14	15	28	48	11	16	19	79	93	12	16	17	31	86	14	16	17	22	71	12	19	20
21	60	64	2	4	5	65	16	3	4	5	9	37	2	7	10	29	39	1	4	6	92	51	3	6	9
22	69	70	9	10	11	45	8	5	10	14	12	26	5	6	7	68	1	7	8	10	78	88	5	9	15
23	62	13	15	16	18	50	33	13	16	20	89	40	13	15	18	55	36	11	13	15	49	79	13	18	20
24	81	23	4	7	9	93	15	1	4	5	34	68	1	8	9	58	27	6	7	8	93	47	2	6	9
25	79	41	6	7	14	74	20	8	10	13	18	14	5	6	9	89	35	7	8	11	83	41	7	10	13
26	1	76	12	17	20	74	53	13	16	19	99	1	11	13	16	29	30	11	13	16	65	39	14	15	18
27	17	62	1	3	9	72	43	3	5	9	79	90	4	6	9	20	93	1	4	5	10	74	1	9	10
28	68	78	9	14	15	69	38	5	6	10	3	67	5	9	13	9	40	7	9	11	11	83	5	12	15
29	53	18	11	17	20	43	65	11	13	20	80	35	13	15	16	37	12	17	19	20	34	22	11	12	16
30	45	28	4	6	7	67	45	7	8	9	91	34	3	5	8	78	89	6	8	9	23	45	5	7	8
31	9	48	5	7	11	29	54	8	9	12	44	63	5	6	10	81	68	6	7	10	36	60	9	11	15
32	75	70	13	16	17	36	77	14	15	17	7	30	12	13	15	57	37	11	16	18	90	13	11	12	17
33	9	67	2	8	9	45	91	2	4	8	20	54	1	7	8	88	49	5	6	9	11	17	2	3	9
34	41	11	7	10	14	37	16	8	11	15	23	58	10	11	14	93	15	6	7	14	11	81	7	10	13
35	9	63	11	18	20	8	33	11	12	13	95	57	15	17	18	77	10	11	12	18	97	44	15	18	19
36	26	74	5	6	8	45	38	2	3	5	87	34	6	7	8	98	21	3	7	10	49	85	1	2	5
37	50	58	7	8	9	98	90	5	13	14	96	24	6	10	13	89	64	6	10	13	37	86	6	7	13
38	9	52	11	15	16	83	25	11	12	17	75	74	15	16	18	69	11	14	15	17	74	9	13	14	15
39	70	79	6	8	9	4	84	1	5	8	7	56	2	8	10	91	3	1	2	6	39	74	2	4	6
40	15	43	11	12	13	54	69	6	11	12	75	18	5	7	15	71	61	6	12	15	57	73	8	9	12

Table A.1: Test Data I: Tests 1-5

	Test 6					Test 7					Test 8					Test 9					Test 10				
i	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3	x	y	ϵ_i^1	ϵ_i^2	ϵ_i^3
1	19	66	8	12	14	89	48	5	8	15	2	83	8	11	13	98	28	6	8	9	9	48	5	7	11
2	85	19	13	15	20	94	85	13	18	20	21	61	11	12	18	26	75	11	15	16	75	70	13	16	17
3	13	89	2	5	6	92	86	2	6	9	27	25	1	4	8	82	30	2	3	4	9	67	2	8	9
4	87	77	9	10	15	32	50	9	11	13	93	60	5	7	9	35	12	11	13	14	41	11	7	10	14
5	79	85	12	14	16	99	58	14	15	16	1	2	12	18	19	50	46	11	15	18	9	63	11	18	20
6	86	74	7	9	10	33	87	3	9	10	81	20	1	3	9	9	13	3	6	7	26	74	5	6	8
7	97	80	6	12	14	69	3	6	8	13	50	83	7	11	13	61	43	8	9	15	50	58	7	8	9
8	50	43	15	18	19	83	40	11	15	16	16	83	12	13	18	28	98	16	17	20	9	52	11	15	16
9	49	48	2	7	8	19	34	2	7	8	24	39	2	4	6	45	58	2	5	7	70	79	6	8	9
10	87	91	8	12	13	46	9	6	8	14	75	28	10	11	14	93	48	9	12	13	15	43	11	12	13
11	42	92	13	15	16	2	15	14	19	20	87	72	13	14	16	32	97	11	16	20	19	94	11	19	20
12	30	59	5	8	9	73	92	2	9	10	71	40	2	3	10	85	30	5	6	7	9	43	6	7	8
13	12	32	7	9	12	4	90	7	9	11	30	99	10	11	13	57	13	7	10	12	57	34	6	7	14
14	75	33	18	19	20	4	37	11	16	20	83	7	15	16	19	41	72	13	19	20	27	42	13	14	16
15	87	80	1	4	8	16	76	2	8	10	41	58	3	6	7	23	13	5	8	10	72	42	3	4	8
16	21	70	6	12	13	32	67	7	10	12	39	25	9	14	15	48	7	6	8	10	52	63	5	13	15
17	89	23	11	17	19	67	79	12	15	16	21	56	11	17	20	60	91	13	14	18	56	26	14	16	19
18	37	16	1	4	8	17	22	3	4	10	25	36	3	6	10	22	3	1	2	6	97	13	2	4	8
19	73	29	7	10	15	80	60	8	12	15	74	87	5	7	13	24	80	12	14	15	42	20	9	13	15
20	82	44	11	12	17	39	69	12	17	19	91	54	13	17	19	30	16	12	17	18	42	89	12	16	19
21	49	65	3	4	10	36	90	1	4	6	29	57	1	3	7	67	5	4	5	6	52	40	3	4	7
22	61	79	10	13	15	76	15	6	12	15	30	19	5	10	13	87	94	5	9	11	11	63	7	9	12
23	91	68	11	13	18	83	63	11	13	14	17	69	13	14	17	44	28	13	15	17	16	78	12	14	19
24	32	80	3	7	8	56	70	5	6	9	14	52	6	9	10	10	99	2	8	10	7	91	3	4	8
25	16	41	6	9	13	17	91	10	12	15	66	26	9	13	15	88	32	8	10	13	23	60	5	12	14
26	17	11	12	17	18	36	61	13	15	19	81	51	11	16	20	39	9	13	14	15	79	70	11	13	18
27	14	5	6	8	9	5	25	4	5	6	94	22	7	9	10	23	16	1	2	3	39	28	1	3	5
28	40	76	10	12	15	38	73	5	8	13	76	55	11	13	15	99	18	6	9	12	82	90	8	10	12
29	94	8	12	14	19	20	96	15	16	17	63	73	11	15	18	21	12	11	17	19	94	72	11	16	17
30	45	45	1	4	6	39	28	4	7	8	92	34	2	4	9	50	32	3	6	9	74	38	1	2	6
31	76	49	9	10	12	19	66	8	12	14	89	48	5	8	15	2	83	8	11	13	98	28	6	8	9
32	52	61	11	13	15	85	19	13	15	20	94	85	13	18	20	21	61	11	12	18	26	75	11	15	16
33	38	13	2	3	5	13	89	2	5	6	92	86	2	6	9	27	25	1	4	8	82	30	2	3	4
34	87	69	8	10	13	87	77	9	10	15	32	50	9	11	13	93	60	5	7	9	35	12	11	13	14
35	51	17	13	18	20	79	85	12	14	16	99	58	14	15	16	1	2	12	18	19	50	46	11	15	18
36	98	24	1	6	9	86	74	7	9	10	33	87	3	9	10	81	20	1	3	9	9	13	3	6	7
37	85	40	11	13	14	97	80	6	12	14	69	3	6	8	13	50	83	7	11	13	61	43	8	9	15
38	10	39	18	19	20	50	43	15	18	19	83	40	11	15	16	16	83	12	13	18	28	98	16	17	20
39	27	86	5	8	9	49	48	2	7	8	19	34	2	7	8	24	39	2	4	6	45	58	2	5	7
40	25	47	10	12	13	87	91	8	12	13	46	9	6	8	14	75	28	10	11	14	93	48	9	12	13

Table A.2: Test Data I: Tests 6-10

$k =$	1	2	3	4	5	6	7	8	9	10	11
ϵ_i^k	1	2	3	4	5	6	7	8	9	10	11
p_i^k	0.040428	0.084224	0.140374	0.175467	0.175467	0.146223	0.104445	0.065278	0.036266	0.018133	0.013695

Table A.3: Poisson Demand Characteristics: $\mu_i = 5$

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ϵ_i^k	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
p_i^k	0.010336	0.018917	0.037833	0.063055	0.090079	0.112599	0.12511	0.12511	0.113736	0.09478	0.072908	0.052077	0.034718	0.021699	0.012764	0.014278

Table A.4: Poisson Demand Characteristics: $\mu_i = 10$

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ϵ_i^k	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
p_i^k	0.007632	0.01037	0.019444	0.032407	0.048611	0.066287	0.082859	0.095607	0.102436	0.102436	0.096034	0.084736	0.070613	0.055747	0.04181	0.029865	0.020362	0.01328	0.019465

Table A.5: Poisson Demand Characteristics: $\mu_i = 15$

		Test 1		Test 2		Test 3		Test 4		Test 5		Test 6		Test 7		Test 8		Test 9		Test 10	
i	μ_i	x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y
1	5	24	78	99	74	45	1	42	65	23	18	89	15	42	79	84	27	83	35	17	9
2	10	96	30	86	49	4	42	69	92	4	81	92	23	48	79	48	65	12	27	24	5
3	15	45	77	66	66	3	59	69	23	14	10	25	95	77	17	99	86	95	78	83	69
4	5	92	8	9	16	57	99	50	9	27	29	35	32	87	37	25	97	79	70	51	20
5	10	36	7	26	63	74	70	55	30	1	42	50	83	81	62	4	24	49	2	94	37
6	15	82	59	97	42	28	63	36	13	62	13	87	62	93	63	42	24	96	47	42	41
7	5	41	52	90	9	71	84	19	42	89	95	5	46	7	13	91	21	20	75	26	75
8	10	90	89	64	97	8	31	51	41	35	41	71	47	68	17	13	73	77	60	60	93
9	15	23	26	77	22	91	93	16	11	98	52	29	87	47	49	18	87	44	7	70	94
10	5	58	22	1	53	57	21	34	12	28	79	95	86	54	44	25	57	68	7	76	37
11	10	34	2	11	26	19	19	11	46	97	60	38	96	62	46	15	78	36	43	30	46
12	15	30	84	51	15	37	55	56	7	60	60	94	15	47	7	88	94	70	31	57	52
13	5	90	28	99	12	52	96	93	79	66	77	99	43	42	69	85	25	76	61	6	16
14	10	5	65	59	90	38	40	65	13	76	4	61	32	14	33	10	22	38	87	89	40
15	15	34	59	80	38	20	7	54	92	33	71	78	47	18	81	50	83	1	79	62	2
16	5	20	49	22	58	90	96	44	52	62	37	17	23	2	43	98	31	71	99	93	64
17	10	63	17	56	75	27	69	65	31	44	11	27	58	47	93	18	70	56	92	81	70
18	15	85	16	51	27	36	56	49	51	40	83	5	26	97	50	51	49	35	63	73	66
19	5	92	61	40	58	22	86	23	58	6	54	94	46	30	22	46	64	76	2	31	9
20	10	86	47	34	69	10	83	36	35	87	95	11	58	50	7	47	66	78	53	53	81
21	15	84	74	55	71	64	90	97	80	72	9	6	39	52	65	93	9	57	5	59	60
22	5	57	53	54	96	85	92	56	44	41	88	52	74	22	2	56	68	95	74	17	50
23	10	89	30	60	89	84	45	15	97	11	99	54	13	69	15	28	97	14	90	41	74
24	15	49	58	7	80	72	83	37	39	23	21	12	99	32	17	36	65	33	8	68	82
25	5	15	52	3	16	83	28	10	23	20	58	98	21	22	98	84	7	43	34	91	72
26	10	96	32	93	13	89	14	83	43	44	16	29	65	86	42	85	65	33	15	99	65
27	15	42	3	56	76	69	33	33	41	36	31	11	60	39	99	28	69	18	39	54	94
28	5	65	70	77	79	67	84	28	14	74	77	48	17	77	53	12	29	44	12	84	10
29	10	96	77	57	74	21	49	41	74	93	35	52	52	67	48	30	89	12	3	19	97
30	15	19	28	75	82	17	13	39	91	83	61	61	13	47	1	90	96	7	43	59	52
31	5	72	64	62	30	22	77	94	57	8	86	83	46	20	18	94	49	20	23	25	34
32	10	21	72	90	23	62	47	64	26	45	21	17	47	71	53	77	69	4	95	49	26
33	15	63	95	38	89	79	2	78	89	57	57	84	21	46	84	83	51	40	2	71	9
34	5	71	65	76	75	52	26	44	17	7	26	79	6	16	79	23	58	9	86	74	80
35	10	8	36	17	48	85	47	99	34	16	66	82	94	20	2	45	62	46	73	81	83
36	15	70	16	13	6	60	4	17	84	89	80	77	78	3	38	91	12	92	71	85	14
37	5	95	93	51	29	19	33	8	29	71	66	88	64	31	65	5	22	83	61	98	82
38	10	25	4	65	17	77	28	14	92	8	74	98	74	52	73	12	34	95	92	69	45
39	15	26	58	92	70	41	42	82	69	38	52	85	52	4	80	43	83	99	3	92	58
40	5	53	10	74	28	89	60	88	11	41	52	65	25	42	57	4	64	28	93	68	61

Table A.6: Test Data II

Appendix B

The VRP-Based Test Problems

The following tests are made up of 2 sets of 40 vertices that are subsets (customers 1-40) of the two well known 50 and 75 customer VRP test problems given by Eilon *et al* [78]. The demands are identical to those in Test 6 of Test Data I for each specific vertex. Table B.1 displays the (x, y) coordinates for each vertex in the two tests.

	50 Customer VRP		75 Customer VRP	
	x	y	x	y
1	30	40	40	40
2	37	52	22	22
3	49	49	36	26
4	52	64	21	45
5	20	26	45	35
6	40	30	55	20
7	21	47	33	34
8	17	63	50	50
9	31	62	55	45
10	52	33	26	59
11	51	21	40	66
12	42	41	55	65
13	31	32	35	51
14	5	25	62	35
15	12	42	62	57
16	36	16	62	24
17	52	41	21	36
18	27	23	33	44
19	17	33	9	56
20	13	13	62	48
21	57	58	66	14
22	62	42	44	13
23	42	57	26	13
24	16	57	11	28
25	8	52	7	43
26	7	38	17	64
27	27	68	41	46
28	30	48	55	34
29	43	67	35	16
30	58	48	52	26
31	58	27	43	26
32	37	69	31	76
33	38	46	22	53
34	46	10	26	29
35	61	33	50	40
36	62	63	55	50
37	63	69	54	10
38	32	22	60	15
39	45	35	21	36
40	59	15	21	36

Table B.1: VRP-Based Tests: Vertex Sets

Appendix C

Results of the VRPSD Benchmark Test Problems

A range of test results, including optimal routes, computational times and optimal solution values from *Test Data I* and *Test Data II* are presented in Tables C.1 to C.4. The information provides a series of original benchmark tests for future reference.

Test	Q	Solution	RF^{z^*}	*cpu* (mins.)	Optimal Routes
1	144	439.2	48.1	11.8	1-5-4-6-8-13-12-7-15-3-11-9-14-16-2-10-1
2	154	346.1	46.0	1.6	1-11-4-3-9-15-2-16-13-10-14-6-7-5-8-12-1
3	150	305.3	46.8	4.9	1-10-14-13-15-12-16-11-2-8-5-9-4-6-3-7-1
4	144	364.3	47.0	36.5	1-10-5-12-2-14-8-15-11-3-13-4-16-7-6-9-1
5	160	387.8	45.0	5.5	1-4-12-13-9-15-2-16-10-8-3-11-5-14-6-7-1
6	165	336.7	48.0	36.4	1-3-11-5-10-7-15-4-6-2-14-8-9-13-12-16-1
7	160	382.3	46.2	1.0	1-8-7-10-11-14-9-4-16-15-13-6-12-3-2-5-1
8	150	408.1	44.5	21.5	1-2-15-9-5-3-16-14-6-10-12-4-11-7-13-8-1
9	158	320.0	48.0	50.4	1-10-7-5-9-14-11-8-2-6-15-4-16-13-3-12-1
10	156	272.8	48.4	0.4	1-8-5-3-11-6-7-16-9-2-15-13-4-14-10-12-1

Table C.1: Test Data I (Tests T2) Results: $n = 15$, $m = 1$, $U = 1.0$

Test	Q	Solution	RF^{z^*}	cpu (mins.)	Optimal Routes
1	72	505.1	92.1	664.8	1-10-15-7-12-13-8-6-4-5-1-2-3-11-9-14-16-1
2	77	430.6	61.4	193.9	1-13-10-14-6-7-5-8-12-1-11-9-2-16-15-3-4-1
3	75	385.9	86.1	225.8	1-2-8-5-9-4-6-3-7-1-14-13-15-12-16-11-10-1
4	72	465.6	73.0	2658.8	1-15-8-14-2-12-5-10-1-11-3-13-4-16-7-6-9-1
5	80	448.2	45.7	182.1	1-4-12-13-9-15-2-14-1-16-10-8-3-11-5-6-7-1
6	83	436.1	95.1	3278.1	1-6-4-15-7-10-5-11-3-16-1-13-2-14-8-9-12-1
7	80	529.4	98.5	1678.5	1-4-9-14-11-10-7-8-1-16-15-13-6-12-3-2-5-1
8	75	544.2	78.7	2006.0	1-12-10-6-14-4-11-7-13-8-1-15-16-3-5-9-2-1
9	79	443.0	67.0	2886.5	1-13-16-4-15-6-5-7-12-1-10-11-8-2-14-9-3-1
10	78	326.5	97.7	77.1	1-7-16-9-2-15-13-4-14-10-12-1-8-5-3-11-6-1

Table C.2: Test Data I (Tests T2) Results: $n = 15$, $m = 2$, $U = 1.0$

Test	Q	Solution	RF^{z^*}	cpu (mins.)	Optimal Routes
1	150	375.7	47.4	4.9	1-14-16-15-7-9-11-5-10-4-13-2-6-8-3-12-1
2	150	400.9	47.4	52.1	1-8-14-3-5-16-10-11-4-12-9-7-13-15-2-6-1
3	150	358.3	47.4	18.8	1-15-11-8-2-3-6-13-4-16-9-7-5-12-14-10-1
4	150	318.8	47.4	37.5	1-15-2-13-8-5-3-14-12-4-6-10-9-11-7-16-1
5	150	401.1	47.4	4.6	1-3-5-2-10-15-12-13-7-11-9-14-6-16-8-4-1
6	150	355.0	47.4	15.7	1-15-8-14-4-16-7-9-3-11-5-10-6-13-2-12-1
7	150	325.4	47.4	1.7	1-15-16-14-7-12-8-3-4-6-5-11-10-9-13-2-1
8	150	356.4	47.4	1.8	1-6-14-5-10-8-11-9-4-15-2-12-3-16-7-13-1
9	150	378.4	47.4	2.9	1-6-8-13-4-3-16-14-7-15-11-2-9-5-10-12-1
10	150	366.4	47.4	7.2	1-2-15-4-6-12-10-14-5-16-3-9-8-7-11-13-1

Table C.3: Test Data II (Tests T3) Results: $n = 15$, $m = 1$, $U = 1.0$

Test	Q	Solution	RF^{z^*}	cpu (mins.)	Optimal Routes
1	75	478.7	85.0	2389.6	1-10-4-13-2-6-8-3-12-1-14-16-9-11-5-7-15-1
2	75	525.7	85.0	2558.2	1-12-4-11-10-16-5-3-14-8-1-6-13-7-9-15-2-1
3	75	522.7	94.1	2866.7	1-13-4-16-9-7-5-10-1-14-12-6-3-2-8-11-15-1
4	75	413.4	68.8	2411.1	1-11-7-9-10-6-5-8-16-1-4-12-14-3-13-2-15-1
5	75	498.2	68.8	798.6	1-6-14-9-11-7-13-8-4-1-16-12-15-10-2-5-3-1
6	75	495.9	100.0	2866.7	1-5-11-3-9-7-16-4-14-8-15-6-10-13-1-2-12-1
7	75	431.2	68.8	1312.7	1-10-11-8-3-4-6-5-2-1-15-16-14-7-12-9-13-1
8	75	485.1	68.8	2088.0	1-4-9-11-8-10-5-14-6-1-2-15-12-3-16-7-13-1
9	75	473.1	68.8	709.4	1-11-14-7-15-2-9-5-10-12-1-8-13-4-16-3-6-1
10	75	498.6	100.0	2866.7	1-11-7-8-9-3-16-12-6-4-10-14-5-15-2-1-13-1

Table C.4: Test Data II (Tests T3) Results: $n = 15$, $m = 2$, $U = 1.0$

Bibliography

[1] Y. Agarwal. A set partitioning approach to vehicle routing. Paper presented at TIMS/ORSA Conference, Boston, USA, 1985.

[2] Y. Agarwal, K. Mathur, and H. Salkin. A set partitioning based exact algorithm for the vehicle routing problem. *Networks*, 19:731–749, 1989.

[3] K. Altinkemer and B. Gavish. Heuristics for equal weight delivery problems with constant error guarantees. *Transportation Science*, 24:294–297, 1990.

[4] K. Altinkemer and B. Gavish. Parallel savings based heuristics for the delivery problem. *Operations Research*, 39:456–469, 1991.

[5] G. Andreatta and L. Romeo. Stochastic shortest paths with recourse. *Networks*, 18:193–204, 1988.

[6] P. Augerat, J. Belenguer, E. Benavent, A. Corbean, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vrp. Technical Report RR949-M, Artemis-Imag, Grenoble, France, 1995.

[7] I. Averbakh and O. Berman. Probabilistic sales-delivery man and sales-delivery facility location problems on a tree. *Transportation Science*, 29:184–197, 1995.

[8] E. Balas and P. Toth. Branch and bound methods. In E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Schmoys, editors, *The travelling salesman problem*, chapter 10. John Wiley and Sons Ltd., New York, USA, 1985.

[9] M. Balinski and R. Quandt. On an integer program for a delivery problem. *Operations Research*, 12:300–304, 1964.

[10] M. Ball, A. Assad, and L. Bodin. Planning for truck fleet size in the presence of common carrier option. *Decision Sciences*, 14:103–120, 1982.

[11] M. Ball, J. Hagstrom, and J. Provan. Threshold reliability of networks with small failure sets. *Networks*, 25:101–115, 1995.

[12] R. Barr, F. Glover, and D. Klingman. The alternating basis algorithm for assignment problems. *Mathematical Programming*, 13:1–13, 1977.

[13] C. Bastian and A. Kan. The stochastic vehicle routing problem revisited. *European Journal of Operational Research*, 56:407–412, 1992.

[14] J. Beasley. Computer based road maps. *International Journal of Physical Distribution*, 12:52–56, 1982.

[15] J. Beasley. Route first cluster second methods for vehicle routing. *Omega*, 11:403–408, 1983.

[16] M. Bellmore and J. Malone. Pathology of travelling-salesman subtour-elimination algorithms. *Operations Research*, 19:278–307, 1971.

[17] J. Benders. Partitioning procedures for solving mixed-variable programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[18] W. Benton and M. Rosetti. The vehicle scheduling problem with intermittent customer demands. *Computers and Operations Research*, 19:521–531, 1992.

[19] C. Berge. *Graphs et hypergraphes.* Dunod, Paris, France, 1970.

[20] A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling problems. *Computers and Operations Research*, 19:521–531, 1987.

[21] D. Bertsimas. *Probabilistic combinatorial optimisation problems.* PhD thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, USA, 1988.

[22] D. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40:574–585, 1992.

[23] D. Bertsimas, P. Chervi, and M. Peterson. Computational approaches to stochastic vehicle routing problems. *Transportation Science*, 29:342–352, 1995.

[24] D. Bertsimas and L. Howell. Further results on the probabilistic travelling salesman problem. *European Journal of Operational Research*, 65:68–95, 1993.

[25] D. Bertsimas, P. Jaillet, and A. Odoni. A priori optimisation. *OR*, 38:1019–1033, 1990.

[26] D. Bertsimas and G. Van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research*, 39:601–615, 1991.

[27] D. Bertsimas and G. Van Ryzin. Stochastic and dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles. *Operations Research*, 41:60–76, 1993.

[28] D. Bertsimas and G. Van Ryzin. Stochastic and dynamic vehicle routing with general demand and interarrival time distributions. *Advances in Applied Probability*, 25:947–978, 1993.

[29] D. Bertsimas and D. Simchi-Levi. A new generation of vehicle routing research: Robust algorithms, addressing uncertainty. *Operations Research*, 44:286–304, 1996.

[30] A.J.M. Beulens. Robust vehicle routing dss and road networks on a european scale? *Annals of Operations Research*, 55:489–509, 1995.

[31] J. Birge and F. Louveaux. *Introduction to stochastic programming.* Springer-Verlag, New York, USA, 1997.

[32] J. Birge and F. Louveaux. *Introduction to stochastic programming*, chapter 9, pages 285–329. Springer-Verlag, New York, USA, 1997.

[33] L. Bodin. Twenty years of routing and scheduling. *Operations Research*, 38:571–579, 1991.

[34] L. Bodin and B. L. Golden. Classification in vehicle routing and scheduling. *Networks*, 11:97–108, 1981.

[35] L. Bodin, B. L. Golden, A. A. Assad, and M. O. Ball. Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research*, 10:63–212, 1983.

[36] B. Bouzaiene-Ayari, M. Dror, and G. Laporte. Vehicle routing with stochastic demands and split deliveries. *Foundations of Computing and Decision Sciences*, 18:63–69, 1993.

[37] J. Bramel and D. Simchi-Levi. A location based heuristic for general routing problems. *Operations Research*, 43:649–660, 1995.

[38] R.L. Carraway, T.L. Morin, and H. Moskowitz. Generalized dynamic programming for stochastic combinatorial optimisation. *Operations Research*, 37:819–829, 1989.

[39] D. Cattrysse and L. Van Wassenhove. A survey of algorithms for the generalised assignment problem. *European Journal of Operational Research*, 72:167–174, 1992.

[40] I. Chao, B. Golden, and E. Wasil. An improved heuristic for the period vehicle routing problem. *Networks*, 26:24–44, 1995.

[41] A. Charnes and W. Cooper. Chance-constrained programming. *Management Science*, 5:73–79, 1959.

[42] N. Christofides. The vehicle routing problem. *RAIRO*, 10:55–70, 1976.

[43] N. Christofides. The vehicle routing problem. In E. Lawler, J. Lenstra, A. Kan, and D. Schmoys, editors, *The travelling salesman problem: A guided tour of combinatorial optimisation*, chapter 12, pages 431–448. Wiley, Chichester, UK, 1985.

[44] N. Christofides and S. Eilon. An algorithm for the vehicle dispatching problem. *Operational Research Quarterly*, 20:309–318, 1969.

[45] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial optimisation*, pages 315–338. Wiley, Chichester, UK, 1979.

[46] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem based on spanning trees and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981.

[47] N. Christofides, A. Mingozzi, and P. Toth. State space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11:145–164, 1981.

[48] V. Chvatal. Edmonds polytopes and weakly hamiltonian graphs. *Mathematical Programming*, 5:29–40, 1973.

[49] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[50] T. Cook and R. Russell. A simulation and statistical analysis of stochastic vehicle routing with timing constraints. *Decision Sciences*, 9:673–687, 1978.

[51] G. Cornuejois and F. Harche. Polyhedral study of the capacitated vehicle routing problem. Management Science Research Report No. 553, Carnegie Mellon University, Pittsburgh, USA, 1989.

[52] J. S. Croucher. A note on the stochastic shortest route problem. *Naval Research Logistics Quarterly*, 25:729–732, 1978.

[53] H. P. Crowder and M. W. Padberg. Large-scale symmetric travelling salesman problems. *Management Science*, 26:495–509, 1980.

[54] W. Cunningham and A. Marsh. A primal algorithm for optimum matching. *Math. Program. Study*, 8:50–72, 1978.

[55] G. B. Danzig, D. Fulkerson, and S. Johnson. Solution of a large scale travelling salesman problem. *Operations Research*, 2:393–410, 1954.

[56] G. B. Danzig and K. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

[57] M. Dempster. Introduction to stochastic programming. In M. Dempster, editor, *Stochastic programming*, pages 3–59. Academic Press, New York, USA, 1980.

[58] U. Derigs. A shortest augmenting path method for solving minimal perfect matching problems. *Networks*, 11:379–390, 1981.

[59] U. Derigs. Solving the non-bipartite matching problem via shortest path techniques. *Annals of Operations Research*, 13:225–261, 1988.

[60] M. Desrochers, J. Desrosiers, and M.M. Solomon. A new optimisation algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1987.

[61] M. Desrochers, J. Lenstra, M. Savelsbergh, and F. Soumis. Vehicle routing with time windows: optimisation and approximation. In B. Golden and A. Assad, editors, *Vehicle routing: Methods and studies*, pages 65–84. Elsevier Science Publishers B.V., North-Holland, Amsterdam, Holland, 1988.

[62] M. Desrochers, J. Lenstra, and M. Savelsburgh. A classification scheme for vehicle routing and scheduling problems. *Journal of Operational Research*, 46:322–332, 1990.

[63] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network routing: Handbooks in operations research and management science 8*, pages 35–139. North-Holland Publishing, Amsterdam, Holland, 1995.

[64] J. Desrosiers, Y. Dumas, and F. Soumis. Routing with time windows by column generation. *Networks*, 14:545–565, 1984.

[65] J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematics and Management Science*, 6:301–325, 1986.

[66] E. Dijkstra. A note on two problems in connection with graphs. *Numer. Math*, 1:269–271, 1959.

[67] J. Dongarra. Performance of various computers using standard linear equations software. *Computer Architecture News*, 20:22–24, 1992.

[68] M. Dror. Modelling vehicle routing with uncertain demands as a stochastic program: Properties of the corresponding solution. *European Journal of Operational Research*, 64:432–441, 1993.

[69] M. Dror, G. Laporte, and F. Louveaux. Vehicle routing with stochastic demands and restricted failures. *ZOR - Methods and Models of Operations Research*, 37:273–283, 1993.

[70] M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation Science*, 23:166–176, 1989.

[71] M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50:239–254, 1994.

[72] M. Dror and P. Trudeau. Stochastic vehicle routing with modified savings algorithm. *European Journal of Operational Research*, 23:228–235, 1986.

[73] M. Dror and P. Trudeau. Savings by split delivery routing. *Transportation Science*, 23:141–145, 1989.

[74] M. Dror and P. Trudeau. Split delivery routing. *Naval Research Logistics*, 37:383–402, 1990.

[75] Y. Dumas, J. Desrosiers, and F. Soumis. The pick-up and delivery problem with time windows. *European Journal of Operational Research*, 54:57–22, 1991.

[76] J. Edmonds. Maximum matching and a polyhedron with 0,1, vertices. *J. Res. Natl. Bur. Stand.*, 69B:125–130, 1965.

[77] J. Edmonds. Paths, trees and flowers. *Can. J. Math.*, 17:449–467, 1965.

[78] S. Eilon, C. Watson-Gandy, and N. Christofides. *Distribution management.* Griffin, London, UK, 1971.

[79] H. Eiselt, M. Gendreau, and G. Laporte. Location of facilities on a network subject to a single-edge failure. *Networks*, 22:231–246, 1992.

[80] Y. Ermoliev. Stochastic quasigradient methods. In Y. Ermoliev and R. Wets, editors, *Numerical techniques for stochastic optimisation*, pages 141–186. Springer-Verlag, Berlin, Germany, 1988.

[81] M. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642, 1994.

[82] M. Fisher. Vehicle routing. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network routing: Handbooks in operations research and management science 8*, pages 1–33. North-Holland Publishing, Amsterdam, Holland, 1995.

[83] M. Fisher and R. Jaikumar. A decomposition algorithm for large-scale vehicle routing. Working Paper 78-11-05, Department of Decision Sciences, University of Pennsylvania, USA, 1981.

[84] M. Fisher and R. Jaikumar. A generalised assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

[85] B. Fleischmann. Linear programming approaches to travelling salesman and vehicle scheduling problems. Paper presented at the XI International Symposium on Mathematical Programming, 1982.

[86] B. Fleischmann. A cutting planes procedure for the travelling salesman problem on road networks. Research Report QM-02-83, University of Hamburg, Germany, 1983.

[87] B. Foster and D. Ryan. An integer programming approach to the vehicle scheduling problem. *Operational Research Quarterly*, 27:367–384, 1976.

[88] H. Frank. Optimum locations on a graph with probabilistic demands. *Operations Research*, 14:409–421, 1966.

[89] H. Frank. Shortest paths in probabilistic graphs. *Operations Research*, 17:583–599, 1969.

[90] A. M. Frieze and G. R. Grimmett. The shortest path problem for graphs with random arc lengths. *Discrete Applied Mathematics*, 10:57–77, 1985.

[91] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the theory of NP-completeness.* W.H. Freeman and Co., San Francisco, USA, 1979.

[92] T. J. Gaskell. Bases for vehicle fleet scheduling. *Operational Research Quarterly*, 18:281–287, 1967.

[93] M. Gaudioso and G. Paletta. A heuristic for the periodic vehicle routing problem. *Transportation Science*, 26:86–92, 1992.

[94] B. Gavish and S. Graves. The travelling salesman problem and related problems. Working paper No. 7905, Graduate School of Management, University of Rochester, USA, 1979.

[95] B. Gavish and S. Graves. Scheduling and routing in transportation and distributions systems: Formulations and new relaxations. Working paper, Graduate School of Management, University of Rochester, USA, 1982.

[96] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40:1276–1290, 1994.

[97] M. Gendreau, G. Laporte, and R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29:143–155, 1995.

[98] M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88:3–12, 1996.

[99] M. Gendreau, G. Laporte, and R. Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44:469–477, 1996.

[100] M. Gendreau, R. Séguin, J. Pelleu, and P. Soriano. Tabu search diversification strategies for a stochastic vehicle routing problem. Paper presented at INFORMS Spring'97-Managing Services in the Next Millennium, San Diego, USA, 1997.

[101] B. Gillet and J. Johnson. Multiterminal vehicle dispatch algorithm. *Omega*, 4:711–718, 1976.

[102] B. Gillet and L. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22:340–349, 1974.

[103] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 5:522–549, 1986.

[104] F. Glover. Tabu search: A tutorial. *Interfaces*, 20:74–94, 1990.

[105] B. Golden and A. Assad. *Vehicle routing: Methods and studies.* North-Holland Publishing, Amsterdam, Holland, 1988.

[106] B. Golden, A. Assad, L. Levy, and F. Gheysens. The fleet size and mix problem. Management Science and Statistics Working Paper No.82-020, University of Maryland at College Park, MD, 1981.

[107] B. Golden, T. Magnanti, and H. Nguyen. Implementing vehicle routing algorithms. *Networks*, 7:113–148, 1977.

[108] B. Golden and W. Stewart. Vehicle routing with probabilistic demands. *Computer Science and Statistics: Tenth Annual Symposium on the Interface*, 503:252–259, 1978.

[109] B. Golden and J. Yee. A framework for probabilistic vehicle routing. *AIIE Transactions*, 11:109–112, 1979.

[110] M. Gondran and M. Minoux. *Graphs and Algorithms.* Wiley, Chichester, UK, 1984.

[111] M. Grotschel and M. Padberg. On the symmetric travelling salesman problem: I-ii. *Mathematical Programming*, 16:265–302, 1979.

[112] M. Guinard and M. Rosenwein. An improved dual based heuristic for the generalized assignment problem. *Operations Research*, 17:658–663, 1989.

[113] E. Hadjiconstantinou and R. Baldacci. A multi-depot vehicle routing problem arising in the utilities sector. JORS (forthcoming).

[114] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest path relaxations. *Annals of Operations Research*, 61:345–358, 1995.

[115] E. Hadjiconstantinou and D. Roberts. Routing under uncertainty: An application in the scheduling of field service engineers. In P. Toth and D. Vigo, editors, *The vehicle routing problem.* SIAM. (forthcoming).

[116] M. Haimovich and A. H. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10:527–542, 1985.

[117] R. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20:182–188, 1986.

[118] R. Hassin and E. Zemel. On shortest paths in graphs with random weights. *Mathematics of Operations Research*, 10:557–564, 1985.

[119] J. Higle and S. Sen. Stochastic decomposition: An algorithm for two stage linear programs with recourse. *Mathematics of Operations Research*, 16:650–669, 1991.

[120] K. Hoffman and M. Padberg. Solving airline crew-scheduling problems by branch-and-cut. *Management Science*, 39:657–682, 1993.

[121] D. Houck, J. Picard, M. Queyranne, and R. Vemuganti. The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *Opsearch*, 17:93–109, 1980.

[122] H. Ishii and T. Nishida. Stochastic bottleneck spanning tree problem. *Networks*, 13:443–449, 1983.

[123] H. Ishii and S. Shiode. Chance constrained bottleneck spanning tree problem. *Annals of Operations Research*, 56:177–187, 1995.

[124] H. Ishii, S. Shiode, and T. Hishida. Chance constrained spanning tree problem. *Journal of the Operational Research Society of Japan*, 24:147–157, 1981.

[125] P. Jaillet. *Probabilistic travelling salesman problems.* PhD thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, USA, 1985.

[126] P. Jaillet. A priori solution of a travelling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36:929–936, 1988.

[127] P. Jaillet. Analysis of probabilistic combinatorial optimisation problems in euclidean spaces. *European Journal of Operational Research*, 39:71–78, 1989.

[128] P. Jaillet and A. Odoni. The probabilistic vehicle routing problem. In B. Golden and A. Assad, editors, *Vehicle routing: Methods and studies*, pages 293–318. Elsevier Science Publishers B.V., North-Holland, Amsterdam, Holland, 1988.

[129] A. Jezequel. Probabilistic vehicle routing problems. Master's thesis, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA, 1985.

[130] N. Kadaba, K. Nygard, and P. Juell. Integration of adaptive machine learning and knowledge-based systems for routing and scheduling applications. *Expert Systems Applications*, 2:15–27, 1991.

[131] P. Kall. Stochastic programming. *European Journal of Operational Research*, 10:125–130, 1982.

[132] P. Kall and S. Wallace. *Stochastic Programming.* John Wiley, Chichester, UK, 1994.

[133] E.P.C. Kao. A preference order dynamic program for a stochastic travelling salesman problem. *Operations Research*, 26:1033–1045, 1978.

[134] R. Karp. Om the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[135] S. Kirkpatrick, C. Gellat, and M. Vecchi. Optimisation by simulated annealing. *Science*, 220:671–680, 1983.

[136] A.W.J. Kolen, A.H.G. Rinnooy Kan, and H.W.J.M. Trienekens. Vehicle routing with time windows. *Operations Research*, 35:266–273, 1987.

[137] H. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics*, 2:83–97, 1955.

[138] V. Lambert, G. Laporte, and F. Louveaux. Designing collection routes through bank branches. *Computers and Operations Research*, 20:783–791, 1993.

[139] A. Land and S. Powell. *Fortran codes for mathematical programming: Linear, quadratic and discrete.* Wiley, New York, USA, 1973.

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

[141] G. Laporte and F. Louveaux. Formulations and bounds for the stochastic capacitated vehicle routing with uncertain supplies. Working paper. Ecole des Hautes Etudes de Montreal, Montreal University, Canada, 1987.

[142] G. Laporte and F. Louveaux. Formulations and bounds for the stochastic capacitated vehicle routing with uncertain supplies. In J.J. Gabszewicz, J. F. Richard, and L. A. Wolsey, editors, *Economic Decision Making: Games, Econometrics and Approximate Algorithms*, pages 443–455. North-Holland, Amsterdam, Holland, 1990.

[143] G. Laporte and F. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142, 1993.

[144] G. Laporte, F. Louveaux, and L. Van Hamme. Exact solution to a location problem with stochastic demands. *Transportation Science*, 28:95–103, 1994.

[145] G. Laporte, F. Louveaux, and H. Mercure. Models and exact solutions for a class of stochastic location-routing problems. *European Journal of Operational Research*, 39:71–78, 1989.

[146] G. Laporte, F. Louveaux, and H. Mercure. The vehicle routing problem with stochastic travel times. *Transportation Science*, 26:161–170, 1992.

[147] G. Laporte, F. Louveaux, and H. Mercure. A priori optimisation of the probabilistic travelling salesman problem. *Operations Research*, 42:543–549, 1994.

[148] G. Laporte, H. Mercure, and Y. Norbert. An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16:33–46, 1986.

[149] G. Laporte, Y. Norbert, and D. Arpin. An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6:293–310, 1986.

[150] G. Laporte, Y. Norbert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33:1050–1073, 1985.

[151] G. Laporte, Y. Norbert, and S. Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science*, 22:161–172, 1988.

[152] R. Larson. Transporting sludge to the 106-mile site: An inventory/routing model for fleet sizing and logistics system design. *Transportation Science*, 22:186–198, 1988.

[153] K. Lee. *Algorithms for routing problems in distribution.* Ph.D Thesis, Dept. of Management Science, Imperial College, University of London, UK, 1990.

[154] H. Lewis and C. Papadimitriou. *Elements of the theory of Computation.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.

[155] C. Li and D. Simchi-Levi. Analysis of heuristics for the multidepot capacitated vehicle routing problem. *ORSA Journal of Computing*, 2:64–73, 1992.

[156] C. Li, D. Simchi-Levi, and M. Desrochers. On the distance constrained vehicle routing problem. *Operations Research*, 40:790–800, 1990.

[157] S. Lin. Computer solution of the travelling salesman problem. *Bell Systems Technology Journal*, 44:2245–2269, 1965.

[158] F. Louveaux. Discrete stochastic location models. *Annals of Operations Research*, 6:23–34, 1986.

[159] A. Lucena. *Exact solution approaches for the vehicle routing problem.* Ph.D Thesis, Dept. of Management Science, Imperial College, University of London, UK, 1986.

[160] L. Magnanti. Combinatorial optimisation and vehicle fleet planning. *Networks*, 11:179–213, 1981.

[161] R. B. Marchandani. Shortest distance and reliability of probabilistic networks. *Computers and Operations Research*, 4:347–355, 1976.

[162] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulation of travelling salesman problems. *Journal of ACM*, 7:326–329, 1969.

[163] A. Mingozzi, R. Baldacci, and S. Giorgi. An exact method for the vehicle routing problem with backhauls. Transportation Science (forthcoming).

[164] A. Mingozzi, N. Christofides, and E. Hadjiconstantinou. An exact algorithm for the vehicle routing problem based on the set partitioning formulations. Internal Report. Dept. of Mathematics,, University of Bologna, Italy, 1994.

[165] R. Mole. A survey of local delivery vehicle routing methodology. *Journal of the Operational Research Society*, 30:245–252, 1979.

[166] C. Noon, G. You, and T. Chan. A fast lower bound for the minimum cost perfect 2-matching linear program. *American Journal of Mathematical and Management Sciences*, 13:357–370, 1993.

[167] National Council of Physical Distribution Management Study. Measuring productivity in physical distribution. prepared by A.T. Kearney, Inc., Chicago, IL, USA., 1978.

[168] I. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operational Research*, 41:421–451, 1993.

[169] C. Papadimitriou and K. Steiglitz. *Combinatorial optimisation: Algorithms and complexity.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1982.

[170] J. Popović. Vehicle routing in the case of uncertain demand : A bayesian approach. *Transportation Planning and Technology*, 19:19–29, 1995.

[171] J. Potvin, G. Lapalme, and J. Rousseau. Alto: A computer system for the design of vehicle routing algorithms. *Computers and Operations Research*, 16:451–470, 1989.

[172] W. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network routing: Handbooks in operations research and management science 8*, pages 141–295. North-Holland Publishing, Amsterdam, Holland, 1995.

[173] H. N. Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14:130–154, 1980.

[174] H. N. Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17:351–360, 1983.

[175] H. N. Psaraftis. k-interchange procedures for local search in a precedence constrained routing problem. *European Journal of Operational Research*, 13:391–402, 1983.

[176] C. Reeves. *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific, UK, 1993.

[177] D. Roberts and E. Hadjiconstantinou. The vehicle routing problem with stochastic demands: An overview. Working paper. Imperial College Management School, London SW7 2AZ, UK, 1996.

[178] D. Roberts and E. Hadjiconstantinou. Algorithmic developments in stochastic vehicle routing. Paper presented at The Symposium on Operations Research (SOR97), Jena, Germany, 1997.

[179] D. Roberts and E. Hadjiconstantinou. An application of stochastic vehicle routing to the scheduling of maintenance engineers. Paper presented at The Tenth Meeting of the European Chapter on Combinatorial Optimisation (ECCO X), Tenerife, Spain, 1997.

[180] D. Roberts and E. Hadjiconstantinou. A new exact algorithm for the vehicle routing problem with stochastic demands. Paper presented at INFORMS Spring'97-Managing Services in the Next Millennium, San Diego, USA, 1997.

[181] D. Roberts and E. Hadjiconstantinou. Algorithmic developments in stochastic vehicle routing. In P. Kischka, H. Lorenz, U. Derigs, W. Domschke, P. Kleinschmidt, and R. Moehring, editors, *Operations Research Proceedings 1997*, pages 156–161. Springer-Verlag, Berlin, Germany, 1998.

[182] D. Roberts and E. Hadjiconstantinou. A computational approach to the vehicle routing problem with stochastic demands. Paper presented at Computational Engineering in Systems Applications (CESA98), Hammammet, Tunisia, 1998.

[183] D. Roberts and E. Hadjiconstantinou. A computational approach to the vehicle routing problem with stochastic demands. In P. Borne, M. Ksouri, and A. El Kamel, editors, *Computational Engineering in Systems Applications*, pages 139–144. IEEE, 1998.

[184] R. Russell. An effective heuristic for th m-tour travelling salesman problem with some side conditions. *Operations Research*, 25:517–524, 1977.

[185] R. Russell and D. Gribbin. A multiphase approach to the period routing problem. *Networks*, 21:747–765, 1991.

[186] M. Savelsberg. Local search in routing problems with time windows. *Operations Research*, 4:285–305, 1985.

[187] R. Seguin. *Stochastic vehicle routing problems.* PhD thesis, CRT, Montreal University, Canada, 1994. (In French).

[188] E. Sigal, A. Pritsker, and J. Solberg. The stochastic shortest route problem. *Operations Research*, 28:1122–1129, 1980.

[189] R. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[190] E. Sniedovich. Analysis of a preference order travelling salesman problem. *Operations Research*, 29:1234–1237, 1981.

[191] M. Solomon and J. Desrosiers. Time window constrained routing and scheduling problems. *Transportation Science*, 22:1–33, 1988.

[192] W. Stewart. *New algorithms for deterministic and stochastic vehicle routing problems.* PhD thesis, University of Maryland at College Park, MD, USA, 1981.

[193] W. Stewart and B. Golden. A vehicle routing algorithm based on lagrange multipliers. In *Proc. AIDS 1979 Annual Convention, New Orleans*, pages 108–110, 1979.

[194] W. Stewart and B. Golden. Stochastic vehicle routing : A comprehensive approach. *European Journal of Operational Research*, 14:371–385, 1983.

[195] E. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–673, 1993.

[196] D. Teodorović, E. Krcmar-Nozić, and G. Pavković. The mixed fleet stochastic routing problem. *Transportation Planning and Technology*, 19:31–43, 1995.

[197] D. Teodorović and G. Pavković. A simulated annealing technique approach to the vrp in the case of stochastic demand. *Transportation Planning and Technology*, 16:261–273, 1992.

[198] D. Teodorović and G. Pavković. The fuzzy set theory approach to the vehicle routing problem when demand at nodes is uncertain. *Fuzzy Sets and Systems*, 82:307–317, 1996.

[199] P. Thompson. *Local search algorithms for vehicle routing and other combinatorial problems.* PhD thesis, Massachusetts Institute of Technology, USA, 1988.

[200] M. Thornton. *Matching problems in graphs.* PhD thesis, Imperial College, University of London, 1988.

[201] F. Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3:192–204, 1969.

[202] F. Tillman and H. Cochran. A heuristic approach for solving the delivery problem. *Journal of Industrial Engineering*, 19:354–358, 1968.

[203] N. Tomizawa. On some techniques useful for solution of transportation network problems. *Networks*, 1:179–194, 1972.

[204] P. Toth and D. Vigo. An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31:133–143, 1997.

[205] P. Trudeau and M. Dror. Stochastic inventory routing: Route design with stockouts and route failures. *Transportation Science*, 26:171–183, 1992.

[206] S. W. Wallace. Bounding the expected time-cost curve for a stochastic pert network from below. *Operations Research Letters*, 8:89–94, 1989.

[207] C. Waters. Vehicle scheduling problems with uncertainty and omitted customers. *Journal of the Operational Research Society*, 40:1099–1108, 1989.

[208] C. Waters and G. Brodie. Realistic sizes for routing problems. *Journal of the Operational Research Society*, 38:565–566, 1987.

[209] J. R. Weaver and R. L. Church. Computational procedures for location problems on stochastic networks. *Transportation Science*, 17:168–180, 1983.

[210] C. Wets. Stochastic programming: Solution techniques and approximation schemes. In A. Bachem, M. Grotchel, and B. Korte, editors, *Mathematical programming: State of the art 1982*, pages 566–603. Springer-Verlag, Berlin, 1983.

[211] B. Williams. Vehicle scheduling: proximity priority searching. *Journal of the Operational Research Society*, 33:961–966, 1982.

[212] A. Wren. *Computers in transport planning and operation.* Ian Allan, London, UK, 1971.

[213] A. Wren and A. Holloday. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operations Research Quarterly*, 23:333–344, 1972.

[214] J. Yee and B. Golden. A note on determining operating strategies for probabilistic vehicle routing. *Naval Research Logistics Quarterly*, 27:159–163, 1980.

[215] P. Yellow. A computational modification to the savings method of vehicle scheduling. *Operations Research Quarterly*, 21:281–283, 1970.