The Pennsylvania State University

The Graduate School

The Mary Jean and Frank P. Smeal College of Business Administration

DEVELOPING STATE-DEPENDENT ROUTES FOR THE VEHICLE ROUTING PROBLEM UNDER UNCERTAINTY

A Thesis in

Business Administration

by

Surajit K. Das

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2004

We approve the thesis of Surajit K. Das.

Date of Signature

Donald Warsing
Assistant Professor of Supply Chain Management
Thesis Adviser
Chair of Committee

Piotr Berman
Associate Professor of Computer Science and Engineering

Douglas Thomas
Assistant Professor of Supply Chain Management

John Tyworth
Professor of Supply Chain Management
Chairperson, Supply Chain and Information Systems

Susan Xu
Professor of Management Science and Supply Chain Management

Abstract

This dissertation extends the study of an extensively studied problem in logistics, the Vehicle Routing Problem (VRP). We first model the variant of this problem that incorporates uncertain customer demands (VRPSD), in a single-vehicle environment, as a Markov Decision Problem (MDP). The advantage of using such a model lies in its ability to incorporate unfolding information—in this case the customer demand, revealed upon visiting the customer—as it is obtained; a focus of our research is utilization of such information. While MDP formulations are not new to the literature, our description of the properties of such formulations represents an enhancement of the existing literature. Furthermore, our computational experiments on VRPSD with a number of MDP-based solution procedures also establish the problem sizes that can effectively be addressed by MDP models, given the dimensional constraints and current computational ability.

Computational intractability, however, implies that exact techniques would be impracticable for large problems; accordingly, we next study some existing heuristic approaches to VRP and VRPSD. We modify these and demonstrate the improvements our modifications can bring about to available solutions, from the perspective either of solution quality in practice, or of guarantees on solution quality or computational time in theory. Apart from these modifications, we use network-flow techniques to construct an approximate solution for a special case of VRP. For VRPSD, in separate sets of computational experiments, we apply an MDP-based procedure and a heuristic tour-improvement procedure to an initial solution, and thereby improve solution quality with reasonable investment in computational resources.

Finally, we use the intuition and solution-development insight gained from the above studies to develop a heuristic approach to solve a real-world problem that to our knowledge has not been addressed in the literature before. This problem is actually a combination of two existing multiple-vehicle VRP variants: (a) a VRPSD that additionally incorporates stochasticity in customer presence, and that is formally referred to as VRPSCD, and (b) a (deterministic) VRP where

all vehicles are assumed to conduct delivery as well as pickup (also called backhaul) operations, formally referred to as VRPB. We call this problem the VRPSCDB.

Specifically, we modify existing "tabu search" (a neighborhood search approach that makes recently considered solutions inaccessible, or *tabu*, for a number of iterations) techniques for solving the VRPSCD, and apply them to VRPSCDB. We also compare our solutions with practice-based solutions, and we incorporate polynomial-time refinements that allow dynamic updates to vehicle routes as new information about customer presence and demands becomes available. Our computational experiments that incorporate such updating show encouraging results.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Our first section in this chapter outlines our research problem in context, providing references as needed for the definitions and descriptions introduced. (A detailed literature review is undertaken in the next chapter.) Later sections discuss the merits of the "dynamic" approach we intend to apply to the problem. The concluding section lays out the organization of the rest of the dissertation.

1.1 Research Problem in Context

An integral component of logistics is transportation, and a frequently arising situation in the transportation and distribution of commodities has usually been modeled as a Vehicle Routing Problem (VRP). In general, the VRP concerns the determination of a minimum-cost assignment of a number of vehicles to deliver goods to (or pick up goods from) a set of n customers while satisfying given constraints. Each of the vehicles is assigned to a route, which specifies an ordered subset of the customers, with each route starting and ending at a fixed point called the depot.

In their review paper, Gendreau et al. [8] identify the most common types of constraints in VRPs as capacity, duration and time window constraints. In general, the number of vehicles in a VRP could be a given constant or a decision variable. In this work, we initially consider a single-vehicle model and later expand our analysis to incorporate multiple vehicles. Further, we consider vehicle capacity(ies) given, and constant across vehicles in the latter case. Also, while capacities as constraints are integral to the entire work, duration and time window constraints are addressed only in parts thereof.

A VRP where one or more of the parameters are random variables is referred to as a stochastic VRP (SVRP). Some examples are the VRPSD (VRP with Stochasic Demands) and the VRPSCD

(VRP with Stochasic Customers and Demands). Application areas are manifold, starting with the refuelling of terminal tankage facilities from a refinery—the problem that gave birth to the SVRP in the academic literature, with the demand at the tanks a random variable (Charnes and Cooper [1]). Bertsimas [17] mentions strategic planning for serving a new market, delivery of money to branches or automatic teller machines of a central bank, distribution of packages from a post office and the design of a hot meals delivery system [53] as application areas for the SVRP, Secomandi [32] adds delivery of home heating oil, sludge disposal, and routing of forklifts in a cargo terminal, or in a warehouse to the list and Gendreau et al. [25] emphasize LTL trucking operations.

In particular, the problem that serves as the underlying motivation for this research concerns a company that operates both pickups and deliveries. Delivery amounts and locations are known *a priori*; however, pickup locations and quantities are random variables. Separate vehicles are not used for pickups, which must be delivered to the depot. Pickup requests are phoned in during the day, but the length of the routes necessitates vehicles starting on their "runs" well beforehand. The problem of integrating pickups and deliveries in an effective manner persists.

For greater generality, we assume that demands at all customers are unknown at the time the routes are constructed, and that demand at a given customer is revealed only when a vehicle actually arrives at the customer. Customer demands are, however, assumed to follow known probability distributions, and the demand at a given customer is assumed independent of the demand at any other customer.

Models incorporating both deliveries and pickups have been relatively scarce in the literature—as Casco et al. [63, p. 127] observe: "Very little has been written about this problem [i.e., the VRP with backhauls] in the Operations Research literature." The utility of such models can, however, be gauged from their further observation that combining deliveries and pickups for supermarkets led to a combined annual savings of $160 million. Since, moreover, there is no published work on the VRPB in a stochastic context, our problem appears to be significant from both a theoretical and a practical standpoint.

Our first objective, therefore, is to develop a reasonably realistic model for the above problem (we call it the VRPSCDB), and use it to generate a set of usable routes. A second objective also is to incorporate current information on the presence and demands of the customers as a route progresses, and thus our objective may be stated as follows:

To generate a set of a priori routes in a a multiple-vehicle, delivery and pickup environment that would be amenable to further improvement via a dynamic routing policy that incorporates new information concerning customer presence and demands as it is obtained.

To explain precisely the value of refining *a priori* routes, we take a brief look at the state of the art in the next section and provide a fuller appreciation of our research in context.

1.2 Prevalent Solution Approaches-pros and cons

Initial work on SVRP essentially attempted to transform the problem into deterministic programs that could be solved using known mathematical programming techniques—i.e., sought to subsume the stochasticity of the problem under *expected value*. A problem, however was (and is) the "NP-completeness" (Garey and Johnson [39]) of the imbedded Traveling Salesman Problem (TSP) in the VRP, an inherited trait that the VRP bequeaths in turn to its stochastic variants. This has meant that "exact" approaches are perforce limited in their applicability. As Cordeau et al. [54], citing Toth et al. [70], observe in the context of VRP in a recent work: "To this day, it seems that no exact algorithm is capable of consistently solving instances in excess of 50 customers." It was inevitable, therefore, that a concurrent approach focused on heuristic solutions.

A second drawback of the exact approaches, relying as they do on various *branch-and-cut* or *branch-and-bound* schemes, is an inability to provide guarantees (bounds) for computational time. Although heuristic approaches to the SVRP were in evidence earlier, "guaranteed" heuristic solutions were first introduced by Bertsimas et al. [16], Bertsimas [17], and Bertsimas et al. [26] through their *a priori* heuristic optimization techniques with *known* bounds, both as regards solution quality and (polynomial) computational time.

Important differences between the heuristic and exact approaches notwithstanding, common to both is the determination of *one*, and *only one*, solution that the user must make do with, for better or worse. Unfortunately, the solution provided by such a *static* (or fixed-route) policy in a stochastic context, unlike one provided by its deterministic counterpart, is "optimal" only in an expected value sense. Hence, while it can be expected to be the best in a *long term average* sense, it clearly need not be the best for all possible realizations of the random variable(s) in question (since each realization could potentially have a different "best" solution). Variability in actual performance of the routes is thus inevitable and could well be prohibitive in practice. We note here that using a criterion other than expected value (minimum variance, for instance) would still not detract from the argument against static policies—these still would provide *only one* solution.

A completely different philosophy, by contrast, underlies a *dynamic* (or state-dependent) policy, such as, for example, is embodied in a Markov Decision Problem (MDP) approach. The focus here is on determining the best course of action *dynamically*, as each *fresh realization of a random variable occurs*. In a VRPSD, for instance, apart from the invariant system parameters (e.g., the demand distributions at, and the locations of, the customers), the decision at a customer (whether to fulfill demand and where to go next) depends *only* upon the vehicle capacity available, the demand at that customer, and the remaining demand at any customers that might have been visited earlier (since vehicle capacity could be exhausted at a customer, multiple visits to a customer are possible). Given that this (variable) information is maintained, the order in which the customers were visited so far, or, the previous values of these variables are irrelevant to the current decision. Under these conditions, the essential MDP property of *memory-less-ness* would be satisfied, and choosing an action would move the system to another state through some (given) probabilistic transition rule. An MDP/DP (Dynamic Programming) approach could thus provide a fresh solution at every vehicle halt, that would be at least as good as one provided by another exact *a priori* technique.

In an ideal world, a dynamic approach could thus provide *continuous re-optimization*, and indeed, in many applications of MDP/DP, problem structure allows development of polynomial-time schemes of computation. Unfortunately, as discussed above, no such structure has so far been

discovered for the VRP, and any exact dynamic policy must similarly succumb to the "curse of dimensionality". In particular, this curse manifests itself during the computation of the "cost to go", i.e., the cost that would be incurred from a given stage onward. This computation, necessary at *every* stage of the typical MDP/DP recursion, would involve an exponential number of sets of customers in all in the case of VRP/SVRP, leading in turn to exponential computation time. (Running times for the various algorithms used in this research are discussed in detail in the body of our work.)

1.3 A Hybrid Approach

Given the above limitations, the natural question is: "Is a synthesis, incorporating good initial solution quality *as well as* productive utilization of unfolding information through a dynamic technique, at all possible?." A recent approach (Secomandi [33]) has in fact attempted just such a synthesis, with reasonably good results in guaranteed polynomial time for the single-vehicle, single-operation (i.e., either delivery or pickup, but not both) VRPSD (a twin-operation environment would have both). The essential idea is as follows: 1. Start with a heuristic *a priori* solution. 2. Using a proxy for the cost to go at each stage, dynamically amend the solution as the realizations of the random variables become known. This procedure is called *rolling out* the original heuristic solution. (While one could in theory also start with an exact solution, in view of the observation of Toth et al. [70] quoted above, it might not always be possible to obtain one.)

To sum up: (a) Whatever technique is used to obtain a solution, it might not guarantee an optimal solution, (b) incorporating unfolding information can only enhance solution quality, and (c) improvements in communications and computational power mean that improved solutions (better routes) as in (b) could be conveyed to the driver at every halt. Hence follows our effort at dynamic improvement after *a priori* route construction.

We note here the views of Secomandi [32, p. 6], who contends that differentiation between the VRPSD and VRPSCD is not necessary: "...in a real-time environment the two problems coincide"

(the assumption presumably being that on a given day, one would know beforehand the customers that require service). A glance at our motivating problem reveals otherwise, however, since the vehicles must commence their journeys before the pickup locations for the day are known, and hence the routes must initially be constructed under uncertainty in the presence of pickup customers.

In the interests of tractability, we will have to make some modifications to the problem stated above (as will be clarified at the appropriate places). We believe, however, that the modifications will not affect the essentials of the problem, and therefore, that our results will still remain applicable. For instance, in all the following chapters we assume that

- If multiple commodities are being transported, their carriage in the same vehicle does not pose safety hazards or cause other problems.
- In a multi-vehicle scenario, all the drivers are familiar with all the routes.
- The time for stoppages (distributions/collections) is negligible
- The actual distance traveled by a vehicle between any points on the route is precisely known and available to the central dispatcher.

In particular, some clarifications are called for regarding the assumption of negligible stoppage times, above. This assumption seems implicitly to have been made in all research that considers dynamic policies for the single-vehicle VRPSD that we are aware of, and it certainly appears reasonable in a pickup scenario. The "load shuffling" that a dynamic policy in real-life delivery situations would, however, be likely to entail means that its applicability in the latter scenario might involve the further assumptions that (a) the consignments meant for different customers are indistinguishable, or (b) a single, easily separable commodity is under transport. A third way to ensure that stoppage times are indeed negligible is to assume that there is always some free space available in the vehicle (as might well be the case for non-bulky items), so that load shuffling is not a time-consuming activity; and we will assume in our single-vehicle model that at least one of these three possibilities holds good, making our original assumption tenable.

The assumption about the insignificance of rearrangement time has not, however, been universally accepted in the context of the VRPB and primarily for this reason, researchers addressing this problem have more often than not sought to create (*a priori*) routes with all deliveries preceding any pickup. We address this issue in detail in Chapter 6 of this dissertation.

We note also the objection raised to the idea of re-optimization, in the context of the VRPSD, by Bertsimas [17]. This has to do with drivers' familiarity with routes, customers being used to particular timings, lack of adequate computational power, etc. While we hope to address the last objection adequately, the others undoubtedly remain valid, and we can only hope that the benefits of re-optimization would outweigh these objections. A comparison of the benefits with the costs that the additional strain on the system would impose, while an interesting topic in itself, is not addressed in this research.

1.4 Dissertation Layout

While our final objective remains as stated above, the problem is a complex one, and we adopt a stage-wise approach in tackling it.

Thus, we start, in Chapter 3, by addressing the "SD" component of the VRPSCDB, i.e., the VRPSD. Specifically, we model the single-vehicle, single-operation VRPSD as an MDP. While ours is not the first MDP formulation, our description of the properties of such formulations is more detailed than those available in the literature. We also undertake computational experiments with different operative policies that serve to establish the problem sizes we can hope to solve "exactly" via a dynamic (MDP) approach, given the state of the art.

Since computational limitations quickly become apparent, it is clear that heuristics must be used to address realistic problems. The reader might also recall our distinction between guaranteed and other solutions in Section 1.2. Since our focus (at least for the dynamic improvement procedures) is guaranteed solutions, in Chapter 4 we accordingly start with a well-known guaranteed heuristic solution procedure for VRP. We modify and extend this approach to single-vehicle

VRPSD. We also use network-flow techniques to construct an approximate solution for a special case of VRP. With certain assumptions relaxed (as we explain in Chapter 4), this technique represents a viable solution procedure for multi-vehicle VRPSD.

The insights gained from these chapters allow us, in Chapter 5, to modify the *rollout* procedure as applied by Secomandi [33] in two ways: (a) By incorporating an additional reoptimization routine in Secomandi's procedure, with computational times on the same order, and, (b) By incorporating one of the exact MDP techniques developed in Chapter 3.

Finally, in Chapter 6, we address our original motivating problem. We develop a model and suitably modify an existing tabu search heuristic, applied earlier by Gendreau et al. [25] to the multi-vehicle (two vehicles) VRPSCD to obtain our initial routes. We also compare these routes with the routes obtained by methods that simulate current industry practice, and then apply dynamic techniques to improve our *a priori* solutions.

Chapter 2

Literature Review

Our effort has been to make the following description chronological to the extent possible. Also, while we primarily focus on the literature that is of direct relevance to our work, some references to allied work become inevitable in order to maintain the flow of the description. Finally, more detailed descriptions of some approaches/techniques that we ourselves employ substantially, e.g., *rollout*, are provided in the concerning chapters.

2.1 Stochasticity enters the picture

The most quoted work on VRP undoubtedly is that of Clarke and Wright [3] (hereafter referred to as CW). In a single-depot, multi-vehicle setting, their algorithm starts by allotting each customer to a vehicle (a sufficient number of vehicles is assumed available) and computing the total distance traveled by all the vehicles. Next, savings are computed. A saving is the reduction in cost that results if any two customers that are not on the same route are joined together and thus placed on the same route. This means adding the cost of the arc joining these two nodes and subtracting the costs of the return arcs (one for each customer) joining the two nodes to the central depot. The algorithm proceeds to "join" customers in descending order of savings (i.e., the first two customers to be joined are the ones whose joining results in maximum reduction in total distance). Customers already on the same route (but not directly connected to one another) are not rejoined to one another. Initially, each customer has two connections available and each time a join takes place, the joining customers lose a connection each. The process continues for a vehicle until its capacity is exhausted and for the nodes until none has any usable connections left.

Predating *CW* by about four years was Charnes and Cooper's [1] seminal work on Chance-constrained programming (CCP), described in Chapter 1. Their formulation sought to determine supplies to keep the probability of failure in meeting demand below a certain specified threshold. In later work [2], the same authors considered the problem in [1] again and proposed methods to transform this problem into deterministic (though not necessarily linear) programs.

Taking the lead from the above, initial work on SVRP adhered to CCP models in one form or the other—a variation being to associate penalty costs with a route failure (i.e., an inability to cater to a customer demand due to vehicle capacity being exhausted). Subsequent modifications introduced Stochastic Programming with Recourse (SPR) models. The latter differ from CCP models essentially in their implicit assumption that a customer cannot be left dissatisfied and therefore that route failure at a customer must be followed by some "recourse" action to revisit him, immediately or at some later point of time. Thus, while the former at best assign penalties to failures, the latter specifically include the actual recourse costs (e.g., a return to the depot followed immediately by a second visit to the customer). The SPR model then attempts to find the route that minimizes the combined expected cost of the chosen route and the recourse.

The first heuristic algorithm for the VRPSD was proposed by Tillman [4] for the multiple depot case. It essentially extended *CW* to a case with more than one depot and with Poisson distributed customer demands. The problem also incorporated some additional restrictions, e.g., a limit on the number of miles traveled on a given route with penalties imposed on vehicles almost empty or filled over-capacity. Golden and Stewart [5] and Golden and Yee [6] both formulated VRPSD as a CCP with slightly differing modifications of *CW*. Also, the former considered Poisson distributed demands alone; the latter incorporated a larger number of distributions—along with the idea of using a larger vehicle capacity in *CW* based on the nature of the demand distribution—to provide a buffer against route failure, and as well, discussed the case of correlated demands.

Golden and Stewart [7] present a CC model and two recourse models. The first of the recourse models in [7] adds a penalty cost proportional to the probability of exceeding vehicle capacity. In the second, the penalty cost is proportional to the expected demand in excess of

the vehicle capacity. They also test two heuristics: one based on *CW*, the other (called GLM) on Lagrangian relaxation, and carry out a number of computations, including some for the case of correlated demands. In the latter, capacity constraints are moved into the objective function with a Lagrange multiplier for each constraint. A *3-opt* branch exchange procedure, an extension of *2-interchange* (procedures based on edge exchanges in the network) is used to find the local minimum to the resulting m-TSP. If the solution is not feasible to SVRP, the Lagrange multiplier associated with the largest demand route is increased, and the m-TSP is re-solved. In Dror and Trudeau [9], if a route failure occurs, the vehicle makes a back-and-forth trip to the depot from the point of failure and may also omit customers if vehicle capacity is below a computed threshold.

Laporte et al. [10] consider two variants of the SVRP—(a) In the "early information" case, all customer demands are known after the route is constructed but before the vehicle leaves. The returns to the depot and route breaks are planned in advance so as to minimize their expected cost. (b) For the "late information" case, breaks and failures coincide. In both cases, the planned routing sequence is assumed unaltered by failures or breaks. Savelsbergh and Goetschalkx [50] similarly present a two-stage SPR model in which one starts with a simple "savings type" initial solution in the first stage, and refines it in the second stage by including non-linear recourse actions that take the location of failure into account. Dror et al. [11] present a compendium of CCP and SPR models. They observe that the VRPSD should be formulated as a multi-stage, rather than as a two-stage SPR model, in the sense that whenever exact demands become known, a new set of decision variables can in principle be generated, and the number of such stages is at least equal to the number of customers in the VRPSD.

2.2 A new genre is born

The element of uncertainty in most of these early SVRP formulations was associated with customer demands (although some formulations also considered stochastic travel times for the TSP—e.g., Kao [51]). A new dimension was, however, added to the SVRP by Jaillet [13] and

Jezequel [14], with the introduction of the Probabilistic TSP (PTSP), where a customer is assumed to require service on a given day with only a given probability, rather than with certainty. This in turn led to the creation of the genres of VRPSC (VRP with stochastic customers)—e.g., in Jaillet et al. [15]—and VRPSCD (VRP with stochastic customers and demands)—e.g., in Gendreau et al. [24]. Jaillet showed that an *a priori* solution obtained by solving a deterministic TSP can be arbitrarily "bad" for the PTSP and that an optimal solution to a PTSP defined in a plane may cross itself (unlike in the deterministic TSP case). Jezequel also tests a number of heuristics for the PTSP using a nearest neighbor(NN) criterion.

Waters [12] considers cases where some customers do not need deliveries during a particular period (VRPSC). In such cases, for customers with "small" demands, "semi fixed" routes (which would skip absent customers) are obtained, essentially by clustering customers based on their distances from one another, and the probability that a customer would require a visit. This is done through an iterative improvement procedure. A number of separate procedures are used to select the customers for repositioning. Also, in later work, Jaillet et al. [15] specifically consider the PVRP (an extension of the PTSP with a single vehicle of finite capacity and customers having binary 0-1 demands) and derive several properties of the model. In their model, in a first stage, one determines a set of routes starting and ending at the depot and visiting each customer exactly once. A 0-demand situation at (absence of) a customer is known before actually visiting it; otherwise, the demand is revealed upon visiting the customer. In the second stage, the first stage routes are followed as planned with the following two exceptions: (1) any absent customer is skipped, and (2) whenever vehicle capacity is exceeded, the vehicle returns to the depot and resumes collections at the next present customer along its route.

Bertsimas et al. [16] provide a recursive expression for the value of the objective function for such a PVRP as well as bounds, asymptotic results and an analysis of several re-optimization policies. They also propose and analyze a number of greedy heuristics in an asymptotic sense. Bertsimas [17] generalizes the results in [16] to the case where customers have arbitrary discrete demands. Two of the properties derived in [17] that apply both to the VRPSD and VRPSC are

that the overall travel cost depends on the direction of travel, and that larger vehicle capacities may yield larger solution costs. *A priori* strategies for the PVRP are shown to be asymptotically optimal in the number of customers for customer locations uniformly distributed in the unit square; their average performance is also very close to the results obtained with the re-optimization strategy. Along with the precise benchmarks for comparison that they made available to researchers, the techniques used by the above authors also provided some advantages *vis-a-vis* (a) the exact approaches, in being unaffected by problem size, and (b) earlier heuristic approaches, in being able to provide guarantees on solution quality. A new computational alternative thus became available.

Laporte et al. [18] treat the depot location also as a decision variable and consider more general demand distributions. They present exact branch-and-cut algorithms for the chance-constrained version of the problem and for a bounded penalty model in which the expected recourse cost is structurally not allowed to exceed a certain percentage of the first stage solution value. They also provide computational results for problems with up to 30 vertices. Bastian et al. [19] reduce the single-vehicle VRPSD to the time-dependent TSP (TDTSP) by permitting only one trip between two locations in each time period. They present a Dynamic Programming (DP) algorithm as well as a penalty function model wherein every unit of unsatisfied demand incurs a penalty, as well as a CCP model that restricts the number of customers assignable to a vehicle.

Teodorovic et al. [20] propose a simulated annealing (SA) algorithm for the multi-vehicle VRPSD with i.i.d. uniform demands. SA starts with a feasible solution T at an initial temperature t. If a feasible solution T' in the neighborhood of T is better than T, T' replaces T in the next iteration. Otherwise, T' replaces T with probability $e^{-(|T'|-|T|)/t}$. After a predetermined number n of iterations, the temperature is appropriately reduced. If every feasible solution can reach any of the others in a finite number of transformations, and t is reduced at an appropriate rate, global optimality results with $\Pr \to 1$ as $n \to \infty$. The technique as applied to VRPSD starts by assigning a number of nodes to to each route. After this initial assignment, other nodes are assigned randomly to these routes. The expected length of the route-set is then calculated. The probability values (that are used to decide whether new solutions that are not better than the current one, are to

be considered or discarded) are based on random generation of a $U(0,1)$ variate; values of t are sequentially chosen from the set of temperatures. The process continues until the difference in route lengths obtained during successive iterations falls below some predetermined value. Some refinements are also proposed.

2.3 Further Refinements

Bouzaiene-Ayari et al. [21] state that returning to the depot in case of failure is just one of the possible recourse actions and that there is no guarantee of the optimality of this policy. As such, it might be preferable for the vehicle involved to move to a different customer and to let another vehicle serve the customer whose delivery could not be completed. They thus allow deliveries to be split and use a modification of the algorithm employed in Dror and Trudeau [9] wherein routes consisting of single customers are merged only if these really constitute different customers and not "fractions" of the same customers.

Laporte et al. [22] present two approaches for another VRP variant —VRP with Stochastic travel times. The key difference is that whereas the first approach incorporates an explicit expression for the recourse costs, the second uses an approximation and is thus simpler. Solutions are iteratively improved until global optimality can be proved. Both approaches in [22] are based on the integer L-shaped method (so called because of its similarity to the L-shaped method of Van Slyke and Wets [76] for continuous problems), introduced by Laporte et al. [23]. This branch-and-cut method computes a first stage solution using a lower bound on the cost of recourse. At any feasible and non-dominated solution, the expected cost of recourse is computed exactly and an optimality cut is generated. The effect of such a cut is to replace in the objective function the lower bound on the solution cost by its true value, or to force the branching scheme to move to another solution.

Bertsimas et al. [26] carry out computational comparisons of the *a priori* heuristics presented in [16] and [17] with the actual *a posteriori* re-optimized routes for the PTSP as well as the PVRP. For the starting TSP solution, they try combinations of the following heuristics and choose the one

that provides the best results. 1. The technique of "Space filling curves (*SFC*)" is applicable to a Euclidean TSP in the unit square. It transforms points (in our context, customer locations) to the unit interval and then constructs a tour based on the obtained ordering of the points in this interval. 2. "Radial sorting" takes as origin the center of mass of the n points and visits the points in order of their radial angle from this origin. The "2-interchange" local improvement heuristic for the PTSP is a generalization of the one used in TSP. The "1-shift" is a degenerate version of "3-interchange". It consists of trying all potential improvements resulting from shifting a single point to a different location in the tour. For the *a posteriori* values, the authors solve the resulting TSPs to near optimality using the *3-opt* procedure.

Bertsimas et al. [27] point out that in many of the applications in which VRPs arise there are significant stochastic and dynamic components to the problem. They consider models that integrate vehicle routing with other issues important to the firm, such as inventory control. Their asymptotic analysis indicates that in several types of problems, static vehicle routing methods, when properly adapted, can yield optimal or near-optimal policies for dynamic routing problems with uniform, stationary demands.

Hjorring et al. [28] follow an approach similar to that in [24], namely, they use the integer *L*-shaped method and add constraints dynamically to the problem to prevent sub-tours, while progressively forming tighter approximations of the costs due to route failure. The difference from the *L*-shaped method lies in their generation of "general" optimality cuts. In [24], each optimality cut imposes a bound on the route failure cost for only one feasible solution. Adding a "general" cut is shown (in general) to improve the lower bound on the total expected cost due to stock-out, thus serving to reduce the number of feasible solutions considered while preserving final optimality.

The idea in Yang et al. [29] is to consider a recourse policy different from that of returning to the depot whenever the vehicle runs out of stock; instead, the points at which the vehicle must return to restock are built into the route *a priori*. To compute the costs of different routes efficiently, the authors modify two well-known deterministic heuristics: *insertion* and *Or-opt* (a modified 3-opt procedure [41] that considers only a small percentage of the exchanges that would be considered

by the regular 3-opt). The authors present two heuristic procedures for the multi-vehicle VRPSD. In the first procedure, a single route is first found incorporating all the customers. Constraints on the lengths of individual routes are then incorporated, and a *Route-first-cluster-next* (RC) is then applied to partition the single route into sub-routes. In the second procedure, a *Cluster-first-route-second* (CR) procedure, based on an adaptation of the algorithm of Fisher et al [30] to the stochastic case is used. The clustering procedure is essentially based on the expected costs of the routes, obtained through a DP recursion.

2.4 Incorporating backhauls

Concurrently with the developments discussed so far, literature on VRPB started appearing in the mid-eighties. For the most part, solution approaches were modifications of techniques that had been used to solve more basic routing problems. To follow the categorization of Jacobs-Blecha et al. [71], these modifications can broadly be classified as follows.

1. *CR*. The classic example is the *SWEEP* algorithm of Gillet and Miller [47].

2. *RC*. Solve a TSP for all the customers (or separately for pickup and delivery customers if so desired), and then break the "big tours" into smaller ones. If initially, pickup and delivery customers were formed into two "big tours", patch the smaller routes so that each route comprises both types of customers.

3. Savings/Insertion, e.g., CW followed by Cheapest Insertion (CI).

4. Improvement/Exchange. Different heuristics allow different categories of exchanges e.g., intra- or inter-route exchanges.

5. Optimization-based techniques. The most popular starting point is the Generalized Assignment Problem (GAP) approach of Fisher et al. [30].

Deif and Bodin [59] start with CW but allow pickups only after delivery. Since this has the effect of creating short routes, however, they add a term that penalizes (and hence delays) the

connection of a pickup and a delivery location. A variant of this approach was proposed by Golden et al. [60] who first develop routes for all of the deliveries and then consider pickups for insertion in between. The insertion of pickups uses the procedure *CI*. (See Cook et al. [43] for details of *CI*). The latter authors also experiment with another variant in which the requirement that pickups must be at the end of a route is relaxed.

Goetschalckx et al. [61]'s *RC* procedure is based on *SFC*. Delivery and pickup locations in the plane are first mapped into the line and then clustered using (a) a greedy method and (b) the *K-median* method (K is the number of routes). In the latter method, after points have been mapped, the line is divided into K identical segments and the points closest to the mid-points of these segments are chosen as the median points. Vehicle capacity permitting, the point with the smallest coordinate among those that are not yet on the route, is assigned to the closest median point (route R). Otherwise, this point is assigned to route $R+1$. If all points cannot be placed on a route, K is increased by one and the clustering repeated.

Yano et al. [62] use a two-step exact procedure that consists of finding an initial feasible solution using a route-construction heuristic, followed by a set-covering routine that uses a branch-and-bound framework to find the optimal solution. The small number of locations per route (rarely exceeding four) makes an optimal solution achievable in their setting.

Casco et al. [63] propose a load-based backhaul insertion procedure. Essentially, they modify the procedure of Golden et al. [60] by (a) initial generation of *CW* routes with capacities smaller than actual vehicle capacities, and (b) filtering solutions through a set of multi-criteria rules that can help identify high-quality solutions. Goetschalkx and Jacobs-Blecha [64] develop an IP formulation for VRPB by extending the Fisher et al. [30] formulation to include pickup points. The model consists essentially of three subproblems. The first two subproblems correspond to the clustering decisions for the delivery and pickup customers which are independent *GAP*s. The third consists of independent single-route TSPs, each having one additional constraint that enforces completion of all deliveries before any pickup.

In subsequent work, Jacobs-Blecha and Goetschalkx [71] propose a new heuristic, called Linehaul-Backhaul (*LHBH*), based on *GAP*. It is in the first phase (initialization, i.e., providing a set of approximate locations for the route clusters) that *LHBH* differs substantially from their earlier heuristic, as follows. Radials are first located around the depot to represent each of the VRPB routes. The locations of, and distances between, all pickup and delivery customers are computed using a distance metric based on angular separation of these points. The facility location problem (with demand or pickup points viewed as existing facilities and cluster locations as new facilities) is then solved iteratively as a capacitated location-allocation problem with the single-source constraints relaxed.

Daganzo et al. [65] discuss the pros and cons of various routing schemes and quantify the performance of these schemes with simple distance formulas. The broad schemes describe implementable solutions, which can in turn be used to initialize fine-tunable algorithms. Toth et al. [67] propose an IP formulation incorporating a Lagrangian lower bound that is strengthened using the cutting-plane method. The Lagrangian lower bound is then combined with a lower bound obtained by dropping the capacity constraints, thus obtaining an overall bounding procedure. A branch-and-bound algorithm, reduction procedures, and dominance criteria are also described. Duhamel et al. [68] propose a tabu search heuristic with time windows. Mixed routes with both delivery and pickup customers are constructed. They consider both versions—i.e., one with deliveries always preceding pickups, and one with no such restrictions.

Mosheiov [69] presents two tour-partitioning heuristics and Salhi et al. [73] propose an insertion-based heuristic for the VRPB which considers insertion of more than one pickup location at a time, and may be applied to the multi-depot case as well as single-depot problems.

Toth et al. [71] propose a *CR* heuristic that is applicable to symmetric as well as asymmetric cost matrices. The approach is based on the lower bounds to a VRPB solution obtained through Lagrangian relaxation proposed by the authors in their [67]. The final set of feasible routes is built through a modified TSP heuristic, using inter-route and intra-route arc exchanges. Mingozzi et al. [72] propose an IP formulation of VRPB and compute a lower bound to the optimal solution

by combining different heuristic methods for solving the dual of the LP relaxation of the exact formulation. They also present an algorithm for the exact solution of the problem. Dethloff [74] applies an insertion heuristic based on the concept of "residual capacities" originally designed for the GPDP (the General Pickup and Delivery Problem) to the VRPB. (The GPDP essentially considers a set of origin and destination points, with the restriction that each origin has specific destinations. The fleet of service vehicles is located at the depot, and transport between a given origin-destination pair is usually not allowed to be split across vehicles. Enhanced versions of the problem also include time windows (for details, see Savelsbergh et al. [75])).

2.5 Recent Work on VRPSD

Secomandi [32] formulates the VRPSD as an SSPP (stochastic shortest path problem) and describes three models that result from differing assumptions about service policies. His formulation differs somewhat from earlier MDP models (the essential characteristics of those relevant to our work are described in greater detail in Chapters 3 and 5); in essence, however, his formulation retains the Markovian framework. The SSPP is described by Bertsekas et al. [34] as a generalization of the deterministic SPP, where the path traversed as well as its length are random variables; at each node we must choose a probability distribution over the set of successor nodes so as to reach a certain destination node with minimum expected cost. This destination node, according to Bertsekas et al. [35], corresponds to a cost-free termination state—upon reaching this state, the system remains there at no further cost. In a VRPSD context, the probability distributions relate to customer demands and the termination state corresponds to arrival at the depot with all customers fully served.

Faced, however, with the computational intractability associated with the above models, Secomandi [32, 33] finds inspiration from some key ideas of the newly emerging Neuro-Dynamic Programming (NDP) approach. As observed in Bertsekas et al. [35], NDP attempts to counter Bellman's "curse of dimensionality" by the use of parametric approximate representations of the cost

to go function obtained through simulation and function approximations. Typically, an approximate representation is chosen to enable easy computation of this cost. A good representation usually leads to a good heuristic policy.

According to Bertsekas et al. [35], following Tesauro and Galperin [36], if an already good base policy (BH)for the problem is known, it can be "rolled out" while computing the controls, i.e., while determining the best action to be taken at a particular stage in a dynamic environment, in order to obtain decisions better than in the original good policy[1]. This corresponds to executing a single policy improvement step on the states visited during execution of the *rollout* policy, as explained in Bertsekas [37]. Bertsekas et al. [38] present the theory of *rollout* algorithms for combinatorial optimization problems. In essence, the theory states that:

1. To determine a *rollout* policy, we must have a base policy that terminates, i.e., reaches a destination node starting from any given node. (A destination node is one that has no outgoing arcs.)

2. Given that a path-construction algorithm of the base policy yields a path terminating at a destination starting from any node, the corresponding *rollout* algorithm is terminating if the original base policy is sequentially consistent, meaning that the relative order of nodes in any subsequence is the same as in the original sequence.

To apply *rollout*, a given *a priori* solution to VRPSD must be adapted at each decision point to ensure that it always terminates from the current state of the system. Secomandi [33] proves that the Cyclic Heuristic (CH)proposed in Bertsimas [17] can be thus adapted and also that it is sequentially consistent. He then uses CH as the base policy for his dynamic reoptimization procedure (the reader will find further details of these policies in Chapters 4 and 5).

[1]The term *rollout policy* was first used by Tesauro et al. [36] in connection with simulation-based computer algorithms for playing backgammon.

Chapter 3

MDP Model, single vehicle

Dror et al. [11] proposed the first dynamic formulation, an MDP formulation, of the VRPSD. A later formulation by Dror [31] differs from the former in considering only Non-Preemptive (*NP*) service disciplines—i.e., those that, at any given customer, permit a move to the next customer only after service completion or route failure. Neither work, however, discusses model properties in detail, nor is computational experience provided in either.

These issues are addressed to an extent in the only (to our knowledge) follow-on work incorporating MDP formulations, by Secomandi [32,33]. Secomandi studies, and also provides computational results on, *strongly non-preemptive* (*SNP*) policies. An *SNP* policy permits a move to the next customer only after service completion. (If vehicle capacity gets exhausted in the process of serving a customer, the vehicle returns to the same customer after refilling/ offloading.)

While an MDP formulation is thus not a novel idea, some of the following issues that we intend to examine (for the first time, to our knowledge) should provide the justification for another investigation of MDP formulations.

- What are the properties of the solution under, and what would be an appropriate computational procedure for, a comprehensive (i.e., one that incorporates all possible states and actions) MDP formulation?

- What would be the costs/benefits of using different service policies—*SNP*, *NP* and others that incorporate preemption to some degree?

- How "large" are the problems that exact MDP approaches incorporating these policies could respectively handle? Dror [31, p. 440], for instance, is rather pessimistic "...[MDP approaches

can] solve three-node problems with a lot of computational effort." However, improvements in computational power should certainly have changed the situation as on date.

- Given the dimensional constraints, would it be possible to imbed an MDP-based procedure in a larger problem to obtain improved solutions *vis-a-vis* current practice?

We formally define the problem in the first section and provide an MDP model of the single-vehicle VRPSD in the second. In the next two sections we apply this model to a small network, one consisting of only two customers. The intuition developed from this study of the "small" network enables us to the describe the general model properties in the following section. In the final sections we carry out computational experiments with four different service policies and with a somewhat restricted action space. (This restriction does not affect the theoretical structure and properties established in Section 3.5, and the reasons for incorporating this restriction are elaborated in Section 3.6.)

3.1 Problem Definition

We follow the generally accepted assumptions in single vehicle MDP-VRPSD formulations: Customer locations are known and fixed. A single vehicle of finite capacity Q based at a central depot serves n customers with product/s stored at the depot. In this chapter, we also assume delivery operations, with the understanding that the analysis applies equally to pickups. The problem is defined on a complete graph $G = (N, E)$, where $N = \{0, 1, \ldots, n\}$ is the set of nodes, i.e., customers and the depot (with 0 representing the depot), $E = \{(i, j) | i \neq j, i, j \in N\}$ is the set of edges connecting the nodes of the graph to each other, and c_{ij} the distance from node i to node j. The objective is to minimize the total distance traveled by the vehicle while satisfying demand at all the customers.

Let D_i be a random variable that represents demand at node i. We restrict D_i, $i \in N$ to integer values, define $D_0 \equiv 0$, and denote by $d_i^{\max}$ the maximum value that D_i can assume. We

also let $p_i(j) = \Pr\{D_i = j\}$, $F_i(j) = \Pr\{D_i \leq j\}$, and $\overline{F}_i(j) = \Pr\{D_i > j\} = 1 - F_i(j)$. We make the following additional assumptions:

- Individual demands cannot exceed the vehicle capacity, i.e., that $d_i^{\max} \leq Q$ for each $i \in N$.
- The triangle inequality holds, i.e., that $c_{ik} + c_{kj} \geq c_{ij}$ for every $i, j, k \in N$.
- The distance matrix is symmetric, i.e., that $c_{ij} = c_{ji}$ for every $i, j \in N$.

3.2 MDP formulation

We formulate the VRPSD as a Markov Decision Problem (MDP). (For the MDP terminology, our reference is Puterman [42].) We use i to denote the node which the vehicle is visiting currently, q to denote the current remaining capacity of the vehicle, and an n-dimensional vector δ to represent whether a given customer has been visited prior to the arrival of the vehicle at i. Specifically, $\delta_j = x$, where $x > Q$, denotes that customer j has not been visited earlier, and $\delta_j < x$ denotes at least one prior visit to j. In the latter case, i.e., $\delta_j < x$, δ_j represents the current unfulfilled demand at j.

We also define the sets $Y(\delta) = \{j|\delta_j < x\}$ and $\overline{Y}(\delta) = \{j|\delta_j = x\}$. $Y(\delta)$ thus denotes the set of customers visited at least once before and $\overline{Y}(\delta)$ the set of customers not visited so far.

An arbitrary action at customer i is represented by $A_i = (a_i, \nu_i)$ where $a_i \in \{0, ... \min(q, \delta_i)\}$ denotes the *service level* (amount delivered) at i and ν_i the next node to be visited. (If $i = 0$, $a_i = 0$ and the vehicle is replenished to its full capacity.) The decision to be made is the choice of the optimal action. We describe the system at a decision epoch, i.e., at the instant after the state of the system is observed and before this decision is made.

Let e_j be an n-dimensional vector with its jth component equal to 1 and all others equal to 0 and let $V(\cdot)$ denote the value function of the decision process. The optimality equations can then be written as follows.

If the vehicle is at the depot, we have

$$V(0,q,\delta) = \min\Bigg\{ \min_{j\in Y(\delta)} \Big\{ c_{oj} + V(j,Q,\delta) \Big\}, \min_{j\in \overline{Y}(\delta)} \Big\{ c_{oj} + \sum_{k=0}^{d_j^{\max}} p_j(k) V(j,Q,\delta - xe_j + ke_j) \Big\} \Bigg\} \tag{3.2.1}$$

where the first term within the outermost minimal operator is the minimum cost attainable if the vehicle visits a customer that has been visited before, and the second term is the minimum cost attainable if the vehicle visits a customer that has not been visited before. Note that, given $j \in \overline{Y}(\delta)$, the jth component of $(\delta - xe_j + ke_j)$ is k.

If the vehicle is at node $i \in N\backslash 0$, we have

$$V(i,q,\delta) = \min\Bigg\{ c_{oi} + V[0,Q,\delta - (\min(q,\delta_i)e_i)], \min_{j\in Y(\delta), a_i} \Big\{ c_{ij} + V(j, q-a_i, \delta - a_i e_i) \Big\},$$
$$\min_{j\in \overline{Y}(\delta), a_i} \Big\{ c_{ij} + \sum_{k=0}^{d_j^{\max}} p_j(k) V(j, q-a_i, \delta - a_i e_i - xe_j + ke_j) \Big\} \Bigg\}, \text{ where } a_i \in \{0, \ldots \min(q,\delta_i)\}. \tag{3.2.2}$$

In the above equations, the terms within the minimal operator denote respectively the minimum cost attainable corresponding to a visit to the depot, a visit to an already visited customer, and a visit to a customer not visited so far. The optimal action corresponds to the minimum value function.

3.3 The 2-customer network, fixed remaining capacity

As indicated earlier, in this section we consider a network with only two customers. For distinguishing between different actions we henceforth use $V[(i,Q,\delta), A_i]$ to denote the value function corresponding to a state-action pair at customer i. Also, since there is no ambiguity, we use d_i to denote the observed demand at customer i at the first visit.

Let us call our customers i and j and suppose that the vehicle visits customer i first. If, at i, the depot is chosen as the next node to visit, since $d_i \leq Q$, $a_i = d_i$ is the best service level.

Hence, equation (3.2.2) in expanded form can be written as follows.

$$V(i,Q,\delta) = \min\big\{\{\min\{V[(i,Q,\delta),(0,j)],\ldots,V[(i,Q,\delta),(d_i,j)]\}\},V[(i,Q,\delta),(d_i,0)]\big\} \tag{3.3.1}$$

The value function corresponding to $a_i = 0$ and $\nu_i = j$ can in turn be written as

$$V[(i,Q,\delta),(a_i,j)] = F_j(Q-d_i)(c_{0i}+2c_{ij}) + \overline{F}_j(Q-d_i)(c_{ij}+2c_{0i}+c_{0j}) \tag{3.3.2}$$

Similarly, the value function with $\nu_i = j$ and arbitrary a_i, $0 < a_i < d_i$ can be written as

$$\begin{aligned} V[(i,Q,\delta),(a_i,j)] &= F_j(Q-d_i)(c_{0i}+2c_{ij}) + [\overline{F}_j(Q-d_i) - \overline{F}_j(Q-a_i)](c_{ij}+2c_{0i}+c_{0j}) \\ &+\overline{F}_j(Q-a_i)(c_{ij}+c_{0i}+2c_{0j}) \end{aligned} \tag{3.3.3}$$

From (3.3.2) and (3.3.3) we note that, for $k \in \{i,j\}$, the strategy $a_k = 0$ dominates all other strategies a_k, $0 < a_k < d_k$. Hence, it suffices to consider $a_k = 0$ and $a_k = d_k$ in our network. For brevity, we henceforth denote service level $a_k = 0$ by $\overline{r}$ and the service level $\min\{q,\delta_k\}$ by r. In this section, we will also assume that i is the customer nearer from the depot, i.e., that $c_{0i} < c_{0j}$.

3.3.1 Value function at the first customer visited

In this subsection, we prove

PROPOSITION 3.3.1. *In a 2-customer network, the optimal action at a customer depends only on the observed demand at that customer (and on other state-invariant system parameters). All such optimal actions remain optimal for a certain range of values of the demand and do not retain their optimality outside this range. The optimal policy is thus of a threshold type.*

Proof.

CASE (A)—*i is visited first*:

$$V[(i,Q,\delta),(r,j)] = F_j(Q-d_i)(c_{0j}+c_{ij}) + \overline{F}_j(Q-d_i)(c_{ij}+3c_{0j}) \quad (3.3.4)$$

$$V[(i,Q,\delta),(\overline{r},j)] = F_j(Q-d_i)(c_{0i}+2c_{ij}) + \overline{F}_j(Q-d_i)(c_{ij}+2c_{0i}+c_{0j}) \quad (3.3.5)$$

$$V[(i,Q,\delta),(r,0)] = c_{0i}+2c_{0j} \quad (3.3.6)$$

Equations (3.3.4) through(3.3.6) represent the cost as a function of the action taken. Intuitively, if d_i is very high, it would be better to visit j via the depot, and if it is low, to visit j directly. The conditions under which partial service at i would be preferable to full service are not, however, obvious.

To determine these conditions, we first note that, by the triangle inequality, $(c_{0j}+c_{ij}) < (c_{ij}+3c_{0j})$ in (3.3.4) and $(c_{0i}+2c_{ij}) < (c_{ij}+2c_{0i}+c_{0j})$ in (3.3.5). Since $F_j(Q-d_i) \geq 0$, it therefore follows that $V[(i,Q,\delta),(r,j)]$ and $V[(i,Q,\delta),(\overline{r},j)]$ are both decreasing functions of $F_j(Q-d_i)$, which in turn is a decreasing function of d_i, and hence that $V[(i,..),(.)]$ is an increasing function of d_i.

Consequently, it follows that we can precisely determine the ranges of d_i for which the different actions become optimal by finding the points of intersection of the above equations. To specify these points of intersection precisely, we use $F_j^{tr}(Q-d_i)$, $F_j^{r}(Q-d_i)$ and $F_j^{\overline{r}}(Q-d_i)$ to denote, respectively, the values of $F(.)$ corresponding to the points of intersection of (3.3.4) and (3.3.5), (3.3.4) and (3.3.6), and (3.3.5) and (3.3.6). The terms corresponding to visiting j first, starting from the depot, are analogously defined by interchanging i and j.

The points of intersections are then given by:

$$F_j^{r}(Q-d_i) = (c_{0j}-c_{0i}+c_{ij})/2c_{0j} \quad (3.3.7)$$

$$F_j^{tr}(Q-d_i) = 2(c_{0j}-c_{0i})/(c_{0j}+c_{ij}-c_{0i}) \quad (3.3.8)$$

$$F_j^{\overline{r}}(Q-d_i) = (c_{0i}+c_{ij}-c_{0j})/(c_{0j}+c_{0i}-c_{ij}) \quad (3.3.9)$$

Hence, corresponding to the two sub-cases as indicated below, the ranges of d_i for which different actions are optimal can be specified as follows.

sub-case: $c_{ij} \geq c_{0j}$

(r, j) is optimal if $F_j(Q - d_i) \geq F_j^r(Q - d_i)$, i.e., if $d_i \leq d_i^r$ and otherwise, $(r, 0)$ is optimal.

sub-case: $c_{ij} < c_{0j}$

(a) If $F_j^{tr}(Q - d_i) \leq F_j^r(Q - d_i)$, i.e., if $d_i^{tr} \geq d_i^r$, the results are identical to those in the case $c_{ij} \geq c_{0j}$.

(b) Otherwise, we have the following results:

- $(r, 0)$ is optimal if $0 \leq F_{j,Q-d_i} \leq F_j^{\overline{r}}(Q - d_i)$, i.e., if $d_i \geq d_i^{\overline{r}}$
- $(\overline{r}, j)$ is optimal if $F_j^{\overline{r}}(Q - d_i) < F_j(Q - d_i) < F_j^{tr}(Q - d_i)$, i.e., if $d_i^{\overline{r}} \geq d_i \geq d_i^{tr}$.
- (r, j) is optimal if $F_j(Q - d_i) \geq F_j^{tr}(Q - d_i)$, i.e., if $0 \leq d_i^{tr}$.

Figure 3.1 shows the resulting value functions as functions of d_i if the customer nearer to the depot is visited first. (Note that $V[(.)(r, j)]$ and $V[(.)(\overline{r}, j)]$ are step functions of d_i rather than continuous functions, and they are shown as lines for ease of interpretation.)

CASE (B)—*j is visited first.* We have

$$V[(i, Q, \delta), (r, j)] = F_i(Q - d_j)(c_{0i} + c_{ij}) + \overline{F}_i(Q - d_j)(c_{ij} + 3c_{0i}) \tag{3.3.10}$$

$$V[(i, Q, \delta), (\overline{r}, j)] = F_i(Q - d_j)(c_{0j} + 2c_{ij}) + \overline{F}_i(Q - d_j)(c_{ij} + c_{0i} + 2c_{0j}) \tag{3.3.11}$$

$$V[(i, Q, \delta), (r, 0)] = c_{0j} + 2c_{0i} \tag{3.3.12}$$

Comparing (3.3.10) and (3.3.11), we see that $(\overline{r}, j)$ cannot be optimal.

Now, proceeding similarly as in CASE (A), we have

$$F_i^r(Q - d_j) = (c_{0i} - c_{0j} + c_{ij})/2c_{0i} \tag{3.3.13}$$

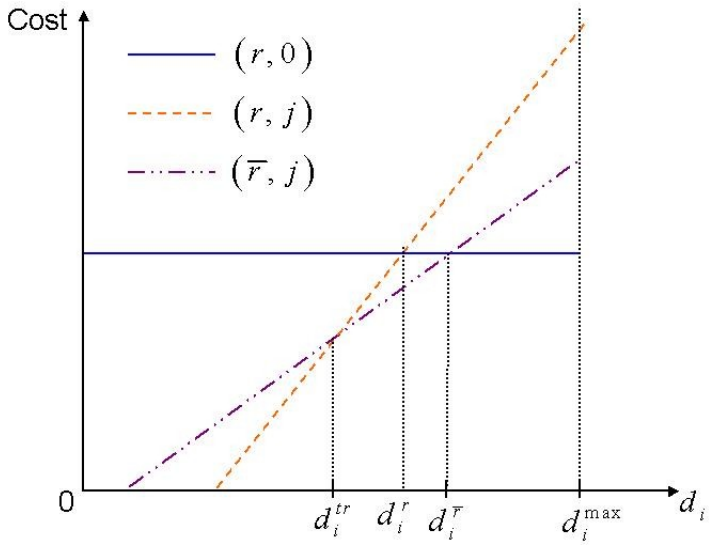

Fig. 3.1. Example of customer nearer to depot visited first

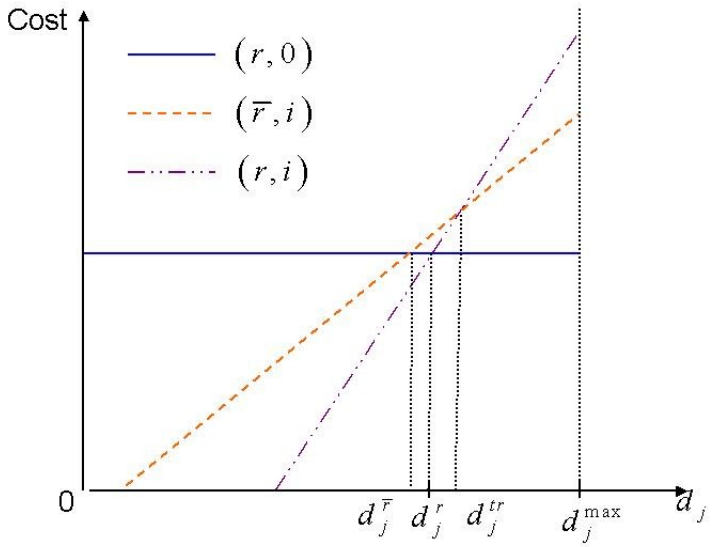

Fig. 3.2. Example of customer farther from depot visited first

Thus, if the farther customer is visited first, the optimal policy is $(r, 0)$ if $0 \leq F_i(Q - d_j) < F_i^r(Q - d_j)$, i.e., if $d_j \geq d_j^r$ and (r, i) otherwise. (Figure 3.2 shows the value functions if the customer farther from the depot is visited first.)

Furthermore, the ranges for the optimality of different actions are clearly disjoint in both cases.

3.3.2 Value function at the depot

Let us now determine the value function at the beginning of the decision process. We start by assuming i.i.d. customer demands. We then have

PROPOSITION 3.3.2. *In a 2-customer network with i.i.d. demands, the decision to visit the farther customer (from the depot) first is optimal.*

Proof. Since $c_{0i} < c_{0j}$ by assumption, from (3.3.7) and (3.3.13) we have

$$F_i^r(Q - d_j) = (c_{0i} - c_{0j} + c_{ij})/2c_{0i} < F_j^r(Q - d_i) = (c_{0j} - c_{0i} + c_{ij})/2c_{0j} \tag{3.3.14}$$

Similarly, from (3.3.7) and (3.3.9) we have

$$F_i^r(Q - d_j) < F_j^{\overline{r}}(Q - d_i) = (c_{0i} + c_{ij} - c_{0j})/(c_{0j} + c_{0i} - c_{ij}) \tag{3.3.15}$$

From our discussions in the previous subsection, we know that the values of $F(\cdot)$ correspond to values of the observed demands at the customers. Since D_i and D_j are i.i.d., (3.3.14) and (3.3.15) imply that $d_j^r \geq d_i^r$ and $d_j^r \geq d_i^{\overline{r}}$. In other words, if for any value d_i of D_i, (r, j) or $(\overline{r}, j)$ is optimal at customer i, then for an identical value d_j of D_j, (r, i) is optimal at customer j. Furthermore, since $F_i(Q - d_j) = F_j(Q - d_i)$ for any $d_i = d_j$, we can use F to denote either of these functions. Hence,

$$\begin{aligned}
&V[(i, Q, \delta), (r, j)] - V[(j, Q, \delta), (r, i)] \\
&= F_j(Q-d_i)(c_{0j}+c_{ij}) + \overline{F}_j(Q-d_i)(c_{ij}+3c_{0j}) - F_i(Q-d_j)(c_{0i}+c_{ij}) - \overline{F}_i(Q-d_j)(c_{ij}+3c_{0i}) \\
&= (3 - 2F)(c_{0j} - c_{0i})
\end{aligned}$$

Similarly,

$$V[(i,Q,\delta),(\overline{r},j)] - V[(j,Q,\delta),(r,i)] = c_{0j} - c_{0i} + F(c_{0i} + c_{ij} - c_{0j}), \text{ and,}$$

$$V[(i,Q,\delta),(r,0)] - V[(j,Q,\delta),(r,0)] = (c_{0j} - c_{0i})$$

For any given value d_i of D_i, the evaluation of $V(\cdot)$ when starting at the depot then gives

$$V[(0,Q,\delta),(.,i)] - V[(0,Q,\delta),(.,j)]$$
$$= (c_{0i} - c_{0j}) + (c_{0j} - c_{0i})(3 - 2F) \geq 0, \text{ if } 0 \leq d_i \leq d_i^{tr}$$
$$= (c_{0i} - c_{0j}) + (c_{0j} - c_{0i}) + F(c_{0i} + c_{ij} - c_{0j}) \geq 0, \text{ if } d_i^{tr} < d_i \leq d_i^{\overline{r}}$$
$$\geq 0, \text{ if } d_i^{\overline{r}} < d_i \leq d_j^{r}, \text{ since } V[(j,Q,\delta),(r,i)] \leq c_{0j} + 2c_{0i},$$
$$= 0, \text{ if } d_i > d_j^{r}$$

It follows that $E(V[(0,Q,\delta),(.,i)]) \geq E(V[(0,Q,\delta),(.,j)])$. Since the points of intersection of the different value functions obtained as explained above will not in general correspond to integer values of customer demand, we need to round off the demand values obtained. Hence, if the optimal policy is followed, the total expected distance is given by

$$V(i,Q,\delta) = c_{0j} + \sum_{k=d_j^{\min}}^{\left\lfloor d_j^{r} \right\rfloor} p_j(k)\Big\{c_{ij} + \big(3 - 2\sum_{u=d_i^{\min}}^{Q-k} p_i(u)\big)c_{0j}\Big\} + (c_{0j} + 2c_{0i}) \sum_{k=\left\lfloor d_j^{r}+1 \right\rfloor}^{k=d_j^{\max}} p_j(k) \tag{3.3.16}$$

Considering now the case where D_i and D_j are not identically distributed, Proposition 3.3.1 is still applicable, so that not fulfilling demand cannot be optimal at the farther customer, or at the nearer customer i if $c_{ij} > c_{0j}$. However,in general $F_j(Q - d_i) \neq F_i(Q - d_j)$, so that Proposition 3.3.2 is no longer applicable. Consequently, 3.3.16 must be modified as follows.

$$V(i,Q,\delta) = \min_{i,j \in N\setminus\{0\}, i \neq j} \Bigg[c_{0i} + \sum_{k=d_i^{\min}}^{\left\lfloor d_i^{tr} \right\rfloor} p_i(k)\Big\{c_{ij} + \big(3 - 2\sum_{u=d_j^{\min}}^{Q-k} p_j(u)\big)c_{0j}\Big\}$$
$$+ \sum_{k=\left\lfloor d_i^{tr}+1 \right\rfloor}^{\left\lfloor d_i^{\overline{r}} \right\rfloor} p_i(k)\Big\{c_{ij} + c_{0j} + 2c_{0i} - \sum_{u=d_j^{\min}}^{Q-k} p_j(u)\big(c_{0i} + c_{0j} - c_{ij}\big)\Big\} + (c_{0j} + 2c_{0i}) \sum_{k=\left\lfloor d_i^{\overline{r}}+1 \right\rfloor}^{d_i^{\max}} p_i(k)\Bigg] \tag{3.3.17}$$

The optimal customer to be visited first is the one corresponding to the lower cost.

3.4 The 2-customer network, variable remaining capacity

In this section, we continue with the 2-customer network, but now suppose that our VRPSD has at least 3 customers. Our 2-customer network is "imbedded" in this larger network and can thus be visited with remaining capacity $q \in \{0, Q\}$, rather than with Q, as was the case so far. Our primary objective in this section is to see to what extent the properties determined so far carry over to this "imbedded" 2-customer network; the importance of analyzing this network, however, lies also in the fact that it would serve as the base for developing an appropriate, recursive computational procedure for higher dimensional cases.

Specifically, in this section we assume that

1. The vehicle is at customer k and vehicle capacity permits complete service at k.
2. It has been decided to serve k fully (so that only a 2-customer network remains after serving k).
3. The remaining customers i and j have not been visited so far. (Reneging at k and revisits at i and j are addressed in later sections.)
4. We drop the assumption (made in Section 3.3) that i is the customer nearer from the depot, and instead assume that i is the customer visited first.

Our objective is to show that Proposition 3.3.1 is applicable to the imbedded 2-customer case with non-i.i.d. demands, and the value functions for such a network can be obtained by modifying the equations developed in Section 3.3.

We now show how the value functions for the imbedded 2-customer case with non-i.i.d. demands can be obtained.

We first define $\min\left(d_i^{\max}, d_j^{\max}\right) = \sigma$ and $\max\left(d_i^{\max}, d_j^{\max}\right) = \beta$. We also find it useful to condition on the value of remaining capacity q *after* serving k and carry out the analysis on the

basis of the four ranges into which q could fall. These are: $0 \leq q < \sigma$, $\sigma \leq q < \beta$, $\beta \leq q < \sigma + \beta$, and $q \geq \sigma + \beta$. We have

Case (A)

$q \geq \sigma + \beta$

If $c_{ik} + c_{0j} < c_{jk} + c_{0i}$, it is optimal to visit i, otherwise j.

Case (B)

$\beta \leq q < \sigma + \beta$

In this case, the points of intersections $F_j^{\cdot}(Q - d_i)$ can be obtained as in the proof of Proposition 3.3.1 since the conditions are identical (i.e., complete service of at least one customer is guaranteed). These give d_i^r, d_i^{tr} and $d_i^{\overline{r}}$, from which $V(\cdot)$ can be precisely determined. We choose the minimum-cost option and repeat the process for j, choosing the better of the two as the first to visit.

For the following cases, $q < d_i$ is possible, since the starting point is not the depot.

Case (C)

$\sigma \leq q < \beta$

sub-case: $d_i^{\max} = \sigma$.

$F_j^r(q - d_i)$ can be obtained as in Case (B)-identical conditions.

Also, we find that $V[(i, q, \delta), (\overline{r}, j)] \leq V[(i, q, \delta), (r, j)]$ if

$$F_j(q) - F_j(q - d_i) \geq (c_{0i} + c_{ij} - c_{0j})/(c_{0j} + c_{ij} - c_{0i}) \tag{3.4.1}$$

Similarly, $V[(i, q, \delta), (\overline{r}, j)] \geq V[(i, q, \delta), (r, 0)]$ if $c_{ij} \geq c_{0j}$ and $V[(\cdot), (\overline{r}, j)] \leq V[(\cdot), (r, 0)]$ if

$$F_j(q - d_i) \geq \{2c_{ij} - F_j(q)(c_{0j} + c_{ij} - c_{0i})\}/(c_{0i} + c_{0j} - c_{ij}) \tag{3.4.2}$$

We thus have the relations between the different $V[\cdot]$ functions, and solving these, we can get $F_j^{tr}(q - d_i)$ and $F_j^{\overline{r}}(q - d_i)$.

sub-case: $d_i^{\max} = \beta$.

If $d_i \leq q$, Case (B) is applicable, since at least one customer can be completely served. Otherwise, if either action (r, j) or $(\overline{r}, 0)$ is taken, $V[(i, q, \delta), (r, j)]$ or $V[(i, q, \delta), (\overline{r}, 0)]$ must be *at least* $2c_{0i} + c_{ij} + c_{0j}$, whereas $V[(i, q, \delta), (\overline{r}, j)] = 2c_{0i} + c_{ij} + c_{0j}$. Thus, action $(\overline{r}, j)$ is optimal.

Case(D)

$0 \leq q < \sigma$

sub-case: $d_i \leq q$

The analysis follows the Case(C) with $d_i^{\max} = \sigma$ (since the conditions are identical).

sub-case: $d_i > q$.

There are two possibilities:

(a) $q \leq Q - d_i$

In this case, $V[(i, q, \delta), (\overline{r}, 0)] \leq V[(i, q, \delta), (\overline{r}, j)]$ if

$$\begin{aligned} & c_{0i} + \sum_{d_j = d_j^{\min}}^{Q-d_i} p_j(d_j)(c_{ij} + c_{0j} + c_{0i}) + \sum_{d_j = Q-d_i+1}^{d_j^{\max}} p_j(d_j)(2c_{0j} + 2c_{0i}) \\ & \leq c_{ij} + \sum_{d_j = d_j^{\min}}^{q} p_j(d_j)(c_{0j} + 2c_{0i}) + \sum_{d_j = q+1}^{Q-d_i} p_j(d_j)(c_{ij} + c_{0i} + 2c_{0j}) \\ & + \sum_{d_j = Q-d_i+1}^{d_j^{\max}} p_j(d_j)(3c_{0j} + 2c_{0i}) \end{aligned} \tag{3.4.3}$$

(b) $q > Q - d_i$

Now, $V[(i, q, \delta), (\overline{r}, 0)] \leq V[(i, q, \delta), (\overline{r}, j)]$ if

$$\begin{aligned} & c_{0i} + \sum_{d_j = d_j^{\min}}^{Q-d_i} p_j(d_j)(c_{ij} + c_{0j} + c_{0i}) + \sum_{d_j = Q-d_i+1}^{d_j^{\max}} p_j(d_j)(2c_{0j} + 2c_{0i}) \\ & \leq c_{ij} + \sum_{d_j = d_j^{\min}}^{q} p_j(d_j)(c_{0j} + 2c_{0i}) + \sum_{d_j = q+1}^{d_j^{\max}} p_j(d_j)(3c_{0j} + 2c_{0i}) \end{aligned} \tag{3.4.4}$$

On the other hand, $(r,0)$ is optimal, i.e. $V[(i,q,\delta),(r,0)] \leq V[(i,q,\delta),(\overline{r},j)]$ if

$$c_{0i} + \sum_{d_j=d_j^{\min}}^{Q+q-d_i} p_j(d_j)(c_{ij}+c_{0j}+c_{0i}) + \sum_{d_j=Q+q-d_i+1}^{d_j^{\max}} p_j(d_j)(2c_{0j}+2c_{0i})$$
$$\leq c_{ij} + \sum_{d_j=d_j^{\min}}^{q} p_j(d_j)(c_{0j}+2c_{0i}) + \Pr\left\{(D_j > q) \cap (D_j + d_i \leq Q+q)\right\}(c_{ij}+c_{0i}+2c_{0j})$$
$$+ \Pr\left\{(D_j > q) \cap (D_j + d_i > Q+q)\right\}(2c_{0i}+3c_{0j})$$

Since $d_i \leq Q$ by assumption, this implies that $(r,0)$ is optimal if, given d_i,

$$\begin{aligned} &c_{0i} + \Pr(D_j \leq Q+q-d_i)(c_{ij}+c_{0j}+c_{0i}) + \Pr(D_j > Q+q-d_i)(2c_{0j}+2c_{0i}) \\ &\leq c_{ij} + \Pr(D_j \leq q)(c_{0j}+2c_{0i}) + \Pr\left\{(D_j > q) \cap (D_j + d_i \leq Q+q)\right\}(c_{ij}+c_{0i}+2c_{0j}) \\ &+ \Pr(D_j > Q+q-d_i)(2c_{0i}+3c_{0j}) \end{aligned} \tag{3.4.5}$$

Comparing coefficients term-wise in the last inequality, we find that it always holds. Similarly, we find that $V[(i,q,\delta),(r,0)] \leq V[(i,q,\delta),(\overline{r},0,)]$, and thus, as in the previous cases, by simultaneously solving pairwise the equations corresponding to a given action, we can determine the points of transition of optimality from one action to another.

3.5 Model Properties

With the background developed in the previous two sections, in this section, we formally state the properties of our model of Section 3.2. In essence, we find that the threshold structure observed in the simple networks that we consider in the earlier sections of this chapter carries over to a limited extent to general networks. Our first three lemmas of this section lead to Theorem 3.1, that summarizes this structure. The last three lemmas describe other properties that are useful from a computational perspective.

LEMMA 3.5.1. *$V(i,q,\delta)$ is non-increasing in remaining capacity q for every i.*

Proof. Suppose the vehicle reaches i at a given decision epoch, and including i, there are m customers still requiring service. Consider two feasible values q and q' of remaining vehicle capacity on arrival at i, where $q' < q$. For any given realization of demands at the m customers, any sequence followed starting with capacity q' can also be followed starting with q until the first arrival at the depot, and thereafter we start with capacity Q in both cases, so the sequence followed starting with q' can be reproduced starting with q. Conversely, since $q' < q$, not all feasible sequences starting with q' are necessarily feasible starting with q. Since i, m and q are all arbitrary, this must hold for all demand realizations.

Since the recursive formulas of Section 3.2 essentially compute, at each epoch, the lowest distance corresponding to each possible realization and then compute the expectation over all possible realizations, the lemma follows by linearity of expectation.

LEMMA 3.5.2. *Suppose on arrival at customer i with capacity q and observed demand $\delta_i = k \leq q$, $V[(i,q,\delta)],(k,0) \leq V[(i,q,\delta)],(k,j)$ for an arbitrary feasible choice of j. Then, this inequality will also hold for (a) any q', $q \geq q' \geq k$, for fixed δ and (b) for any value $k' \geq k$ of δ_i, with fixed q and the other components of δ unchanged.*

Proof.

(a) $V[(i,q,\delta),(k,j)] \leq V[(i,q',\delta),(k,j)]$ for any $q' \leq q$ by Lemma 3.5.1. On the other hand, $V[(i,q,\delta),(k,0)]$ is the same in both cases, since the vehicle starts from the depot with quantity Q.

(b) Increasing δ_i means that remaining capacity at j decreases if (k',j) is chosen at i. From Lemma 3.5.1 it therefore follows that $V(.)$ at j cannot decrease. Since this must hold for all feasible choices of j, $V[(i,q,\delta),(k',j)]$ cannot decrease either, for any $j \neq 0$. However, $V[(i,q,\delta),(k',0)] = V[(i,q,\delta),(k,0)]$.

LEMMA 3.5.3. *If i cannot be fully served and $V[(i,q,\delta),(\overline{r},0)] \leq V[(i,q,\delta),(\overline{r},j)]$, for given δ_i, q and $j \neq 0$, then this inequality will also hold for any $q' \leq q$ with fixed δ.*

Proof. As in the proof of Lemma 3.5.2, re-starting from the depot entails constant cost irrespective of remaining capacity, whereas $V[(i, q, \delta), (\overline{r}, j)]$ is non-decreasing in remaining capacity, by 3.5.1 and from arguments as in Lemma 3.5.2.

Recalling the figures of Section 3.3, the discussions in the preceding two sections, and from Lemma 3.5.1, we can visualize $V[(i, q, \delta), (a_i, j)]$, for given q, a_i and j, as a function mapping demand values δ_i at i to expected distance. Associated with each feasible (i, j) pair, we have a set of such functions, each function corresponding to a particular choice of a_i. For any δ_i, therefore, the minimum of the values $V[(i, ., .), (a_i, j)]$ over all feasible a_i corresponds to the optimal service decision if j is visited next. Next, we can minimize over all j to find the optimal ν_i. The "slope" of any given function would not, in general, be a constant but rather, could vary in an arbitrary manner. Thus, unlike the simple case studied in section 3.3, we need not any longer have unique points of intersection of the functions—the various V functions could have multiple intersections. So, while thresholds still exist, it is not guaranteed that once a service level loses optimality, it cannot be optimal again.

To sum up, we have

THEOREM 3.1. *Corresponding to each state of the system the optimal action depends on the values of the current components of the state vector (and on other state-invariant system parameters). Moreover, given the state vector, the action of visiting the depot after full-service or after no-service at a customer exhibits threshold behavior with respect to observed demand at that customer. Moreover, except as in accordance with these thresholds, no action can be eliminated a priori.*

An extension of Proposition 3.3.2 is the following

LEMMA 3.5.4. *Suppose that the vehicle visits customers in decreasing order of distance from the depot and reaches customer k with n customers (including k) yet to be served. If the remaining capacity is such that any subset of $n-1$ customers can be fully served, less than full service will not be optimal at k or at any succeeding customer.*

Proof. Suppose $a_i < \delta_i$ is served at some customer i while traversing the route from (and including) k onward. If it turns out at the last customer (say, j) that both i and j can be served, the distance required to be traveled from j is $c_{ij} + c_{0i}$. If, instead, $a_i = \delta_i$, the required distance is c_{0j}. If it turns out that both cannot collectively be served, the required distances are respectively $3c_{0j}$ and $c_{0j} + 2c_{oi}$. Since $c_{0i} \geq c_{0j}$ and i is arbitrary, the lemma follows.The following helps in state-space reduction.

LEMMA 3.5.5. *A second or later visit to customer j such that the optimal action at j is $a_j = 0$, cannot be part of an optimal sequence.*

Proof. Suppose at the current epoch, the vehicle is at customer i, j has been visited earlier, and $\delta_j \neq 0$. Then for any feasible $a_i = k$, the corresponding value function at j can be precisely determined at i (since $\delta_j \neq x$). If at j, the optimal $a_j = 0$, then by the triangle inequality we can proceed directly to the corresponding ν_j from i at a cost that cannot be higher. In other words, in an optimal sequence, $a_j = 0$ can occur at most once. The above lemma, in turn, leads to the following

LEMMA 3.5.6. *The recursive procedure used to evaluate the value functions must end finitely.*

Proof. From Lemma 3.5.5, if in an optimal sequence, a customer is visited more than once, at least partial service must take place at the second or any subsequent visit. Hence, no customer can be visited more than K times, where K is the maximum permissible customer demand, and thus, the total number of trips to customers cannot exceed $n(K + 1)$.

If no customer is visited a second time, and the service level is $a_\cdot = 0$ continuously, then if the vehicle is at customer i at decision epoch $n + 1$ (the vehicle is assumed to start from the depot at epoch 1), either at the next visit to customer j, j must be served at least partially, or, $a_i = \delta_i$ can be served followed by a trip to the depot. Thus every trip to depot reduces the number of customers by one, and hence the depot cannot be visited infinitely, either.

3.6 Service Disciplines and the Recursive algorithm

We turn in this section to computational aspects. Our analysis of Section 3.3 showed us that under certain conditions, it suffices to consider service levels r and $\overline{r}$ alone. While this observation cannot be generalized to the n-customer case, it provides an intuitive basis for restricting the service levels to r and $\overline{r}$ in our computational experiments. Let us denote by MDP2 our model with these restrictions incorporated and by MDP1 our original model of Section 3.2.

From a practical standpoint also, we feel that MDP2 is a more reasonable choice than MDP1. Following the optimal policy under either model, for instance, would in general require the customers to go through the delivery process a number of times for a given consignment—a course of action not likely to be very popular with them. Since, however, Lemma 3.5.5 assures us that revisiting a customer must involve *some* service, at a revisit under MDP2 the full remaining demand of the customer will be served unless vehicle capacity is exhausted. In other words, no service ($\overline{r}$) can occur at most once at any given customer under MDP2, and partial service would always be seen by the customer to be due to route failure (i.e., as beyond control). Hence, following MDP2 would be less annoying to customer j than following MDP1, which could theoretically involve as many as $d_j^{\max} + 1$ visits to j, in the worst case without route failure occurring on any visit (and thus causing the perception of deliberate denial of service). From a computational perspective, the reduction in action spaces under MDP2 will allow larger problem instances to be considered (as we make precise later in this section).

The reneging action presupposes that customers are not particular about delivery times as long as the delivery is made within the time horizon under consideration (generally, within a given day), and we will assume that this is so in considering policies that allow $\overline{r}$. Another implicit assumption made so far is that partial fulfillment is possible. One could, however, visualize a scenario where this assumption does not hold, e.g., if customer requirements were assumed to be for *sets* of items rather than for individual items, with the further assumption that the intended jobs at the customers could not be undertaken without the entire set being available. The consignee

in such cases might be averse to partial delivery and the driver would then have to try his luck elsewhere or to return to the depot.

The reader may recall, at this point, our reference to service policies NP and SNP at the beginning of this chapter. Inclusion of the action $\overline{r}$ permits us to define two more policies. Specifically, (a) if the action space of NP is augmented with the service level $\overline{r}$, the resulting policy will be called *RNP* and (b) if in (a) the additional restriction of only full service level at a customer is incorporated, the resulting policy will be called *CFRNP*. Note that under *CFRNP* a customer is served either completely or not at all.

Our focus will be to determine the limits of the feasibility of using these policies. It is clear that the enormity of the state space means that only "small" problems can be tackled (even with MDP2), but what is small? Furthermore, since an accuracy/computational time tradeoff is involved, one should also be able to get some idea of the problem dimensions corresponding to the greater viability of a given policy.

The essential differences between the different policies that we will use in our computational experiments can then be summarized as in Table 3.1.

Table 3.1.
Characteristics of different policies

POLICY	allowed actions	ν_i, $q < \delta_i$
SNP	r	i
NP	r	any feasible
CFRNP	r only if $\delta_i \leq q$, $\overline{r}$	any feasible
RNP	$r, \overline{r}$	any feasible

We next look at the time complexity of these policies and consider first DP applied to the (deterministic) VRP. Using the backward recursion technique, at the last (nth) stage, when no customer remains to be served, there would be $\binom{n}{0}$ choices of customers to be visited. At stage $n-1$, there would be $\binom{n}{1}$ choices. In general, the number of choices at the ith stage would be $\binom{n}{i}$. Within each of these choices (sets of customers), any member could be visited first. The computations would be carried out for the $Q+1$ possible values of q. Since $\sum_{i=0}^{n} \binom{n}{i} = 2^n$ and there are n stages in all, the running time of the algorithm would thus be $O(n^2 2^n (Q+1))$.

The problem in extending the above analysis to VRPSD is that the number (T) of decision epochs is now a random variable. We observe, however, that the randomness in T arises from two factors: (a) a customer may be visited more than once, and, (b) reneging. Factor (b) is not relevant for *SNP* or *NP*. Furthermore, *SNP* does not consider proceeding to the next customer until the current one is fully served, in effect removing the randomness in T—there is precisely one stage for each customer. The running time of DP-VRPSD under *SNP* would thus be $O(Kn^2 2^n (Q+1))$.

For *NP*, additionally, the next customer to be visited after route failure is decided by performing a minimization over the entire state space and thus we would in general have a number of customers with partially fulfilled demand awaiting service. The total number of demand realizations (states) *NP* would consider would thus be $(K+1)^n$, instead of K; and its running time is thus $O((K+1)^n n^2 2^n (Q+1))$—the factor $K+1$ arising because for each customer, one state corresponds to unknown demand.

The states in *RNP* are the same as in *NP* (and somewhat less in *CFRNP* because customers do not have partial demands—but for uniformity, we ignore this difference). Considering next the effect of factor (b) on these policies, from Lemma 3.5.5 it follows that the $(\overline{r}, .)$ action can be taken at most n times in continuation. The running time of DP-VRPSD for these policies is thus $O((K+1)^n n^3 2^n (Q+1))$. (The additional computational time required due to inclusion of the $\overline{r}$ action is subsumed under the "constant" term in the expression.)

Under MDP1, the additional computational time required cannot be subsumed under the constant term, since all possible values for a_i at node i must be considered, and the number of such

possible values would depend on δ_i as well as on q. In the worst case, this number could be K, so that the worst-case running time of MDP1 would be greater than that of *RNP* by a factor of K.

We note, however, that while the different policies essentially operate on reduced state and action spaces (with reference to the sets of all feasible states and actions), they all are based on the observed demand realizations. The Markov nature of the state transitions thus remains unaffected and hence our observations about the threshold structure carry over to all the policies. The threshold computations can therefore be incorporated for each (and would be expected to reduce the running times for all the policies in practice).

We now discuss the details of the basic recursive procedure that would be applied. (We essentially discuss *RNP*; for the others, the procedures are straightforward modifications.) To emphasize that the value functions are obtained by minimization over the actions r and $\overline{r}$ at a given customer, we write

$$V(i, q_t, \delta) = \min\{V[(i, q_t, \delta), (r, .)], V[(i, q_t, \delta), (\overline{r}, .)]\} \tag{3.6.1}$$

(where the subscript on q references the decision epoch, with the depot visited at epoch 1). From equation (3.2.2) we know that

- If $\nu_i = 0$, $q_{t+1} = Q$. Otherwise,
- Under action $\overline{r}$, $q_{t+1} = q_t$.
- Under action r, $q_{t+1} = \delta_i - q_t$ if $q_t \geq \delta_i$, and $q_{t+1} = 0$, otherwise.

We now consider the three-customer case with the vehicle starting from the depot and no customer visited previously. Since we have $d_i \leq Q$ for each $i \in N$, the first r action will reduce the problem to a two-customer problem, the solution of which has been discussed in previous sections. The first terms in each of the following equations refer to minimization over the actions r which reduce the problem to a two-customer subproblem, and are hence shown only concisely. We will thus focus on the $\overline{r}$ action and follow the state transitions only till the problem has been reduced

to a 2-customer sub-problem. Also, it is possible that if $q = d_k$ for some k, r at k can be followed by a visit to another customer u rather than the depot (if $\Pr\{D_u = 0\}$ is "high"). To avoid more complicated expressions, we will, however, assume that this is never the case, and that if $q = d_k$, the r action is always followed by a visit to the depot.

We also add a superscript to $V(\cdot)$ to indicate the dimensionality of the current sub-problem, and for clarity, we also explicitly show the nodes $\{i|\delta_i \neq x\}$ that have known values of demands. (For instance, V^3 on the L.H.S. of (3.6.2) denotes a 3-customer problem, and $\{i\}$ on the R.H.S. of the same equation denotes that the demand at customer i is known.)

Letting i be the first customer visited (at epoch 2), with k denoting observed demand, and recalling that the first inner parenthesis on the R.H.S. refers to the r action, we have

$$V^3\left(i, Q, \delta | Y(\delta) = \{\emptyset\}\right) = \min\left\{\min_u\{.\}, \min_u\left\{c_{iu} + \sum_k p_u(k) V^3[u, q_3, \delta | Y(\delta) = \{i\}]\right\}\right\} \quad (3.6.2)$$

where $u \in N\backslash(\{i\} \cup \{0\})$. If $\overline{r}$ is chosen at i, and j is the second customer visited,

$$V^3[j, q_3, \delta | Y(\delta) = \{i\}] = \min\left\{\min_u\{.\}, \min_u\left\{c_{ju} + \sum_k p_u(k) V^3[u, q_4, \delta | Y(\delta) = \{i, j\}]\right\}\right\} \quad (3.6.3)$$

where $u \in N\backslash(\{j\} \cup \{0\})$. If $\overline{r}$ is chosen at j, and the third customer visited (v) is the same as i, by Lemma 3.5.5, we need be concerned only with the r action at v. If $v \neq i$,

$$V^3[v, q_4, \delta | Y(\delta) = \{i, j\}] = \min\left\{\min_u\{.\}, \min_u\left\{c_{vu} + \sum_k p_u(k) V^3[u, q_5, \delta | Y(\delta) = \{i, j, k\}]\right\}\right\} \quad (3.6.4)$$

where$u \in N\backslash(\{v\} \cup \{0\})$. Now, if the $\overline{r}$ action is chosen yet again at v, one of the previously visited customers must be visited a second time, and by Lemma 3.5.5, the $\overline{r}$ action at that customer need not be considered.

The above shows how the solutions of the two-customer sub-problems can be used to solve the three-customer case. In a similar way, one can analyze an "imbedded" three-customer case,

i.e., one with the three-customer system a sub-system in a larger network. The basic difference in such a case is that we can no longer assume that the first r action will reduce the dimensionality of the problem, so the analysis requires some additional steps. (We provide the details in Appendix A.)

Recursively, the principle can thus be applied to $n-1$-customer sub-problems to solve n-customer problems, with the properties derived in Section 3.5 used where applicable.

Though we do not perform computational experiments with MDP1, we note that the above technique can similarly be extended to it. Thus, in equation (3.6.1), the minimization under MDP1 would consider $\min\{q, \delta_j\}+1$ actions when visits to customer j are considered. Considering equations (3.6.2) through (3.6.4), the analogous equations—as long as the problem remains a three-customer problem, upon revisits to customer j, will consider the actions $a_j = \{1, ..., \min\{q, \delta_j\}\}$, where δ_j will decrease in value with each revisit. Among these, the action corresponding to the minimum expected value will be chosen, instead of the action r alone being considered at the first revisit and no revisits allowed after the first, as in MDP2. The problem will thus still eventually be reduced to a two-customer subproblem, but not, as in MDP2, at the latest on the second visit to a customer. Instead, all the "less than full service" (LFS) actions taken at a customer would have to be followed through all combinations of feasible LFS actions at successive customers until the problem is so reduced. The longest such sequence of actions that would be considered for customer j will require $d_j^{\max}+1$ visits to j to reduce the problem accordingly. The procedure for higher-dimensional cases would be analogous.

The essence of the recursive procedure: (a) all revisits must involve service, (b) use of thresholds where applicable, and (c) use of lower-dimensional cases to solve higher-dimensional ones, would remain the same in both.

3.7 Computational Experiments-Methodology

Our computational algorithms essentially apply the equations of Section 3.6 as well as equation (3.2.2), and use the properties established in Sections 3.3 through 3.5 wherever applicable. Thus, the algorithms begin by considering all two-customer cases to determine the threshold values of the demands that correspond to the different optimal actions. They next use backward recursion to determine the optimal decisions and the corresponding optimal distances for all k-customer-to-go cases up to $k = n$. (The algorithms for all the policies are based on these equations, but the state and decision spaces are defined differently in each. Thus, the algorithms for SNP and NP do not include the $\overline{r}$ action. The details of each algorithm are explained at length as comments in the codes.)

We first explain the relevant terms used by Gendreau et al. [24] (and followed by Secomandi [32]) in their computational experiments, based on an exact approach. Thus, filling coefficient (rate) $\overline{f}$ is defined as $\sum_{i=1}^{n} E[D_i]/mQ$, so that the expected number of route failures $\overline{f}' = \max\{0, \overline{f} - 1\}$ (m represents the number of vehicles and thus $m = 1$ in our experiments). We evaluate the effectiveness of the different policies by computing the expected distances traversed under each on a number of problem instances, and in keeping with Secomandi's motivating work, we (a) distinguish between experimental categories by the values of n and $\overline{f}'$ chosen, and determine Q from the equation for $\overline{f}'$ above, rounding up if necessary, (b) represent customer demand distributions by $U(\underline{d}, \overline{d})$, where $(\underline{d}, \overline{d}) \in \{(1, 3), (2, 4), (3, 5)\}$, with each distribution having an equal probability of being selected for any given customer, and (c) position customer locations within a unit square in $\mathbf{R}^2$ by randomly generating the x and y coordinates corresponding to each location.

Our simulation experiments, like Secomandi's, subject each $(n, Q, \overline{f}')$ combination to a number of different problem instances. First, for each n, we generate a set of customer locations. For each $(n, Q, \overline{f}')$ category there are *replication* times *iteration* problem instances, where a replication corresponds to a set of customer locations and an iteration to an assignment of one of the three demand distributions described above randomly to these customers (locations).

Regarding depot location, Secomandi [32, p. 43] observes: "In a multi vehicle situation,...the depot would be located somewhere in a median location with respect to the customers...[we have already decided] which nodes should be served by each vehicle. Then,...[if we consider any one of these routes, its customers should] form a cloud of points fanning out from the depot...looking at Figure 3, page 150 [of the work of Gendreau et al. [24]], our assumption on depot location seems to hold in a stochastic setting as well." Secomandi's depot, following these observations, is located at $(0, 0)$, and his unit square is $[0, 1]^2$.

While Secomandi's arguments are undoubtedly reasonable, a closer look at the shapes of the optimal routes in the figure alluded to by Secomandi (partially reproduced in Chapter 6 for the reader's convenience) indicates a sufficient degree of overlap between the routes to suggest the benefits of simultaneous reoptimization across routes. We feel, therefore, that a reoptimization policy that considers each vehicle route individually, to the exclusion of the others, would be more suited to a single-vehicle, than to a multi-vehicle environment. If the depot is indeed situated at a "median" location, this should mean proximity to the center, rather than to a corner of the region in the single-vehicle environment. Accordingly, while we retain Secomandi's depot location and other parameters, the vertices of our unit square, unlike Secomandi's, are at $(0.5, 0.5)$, $(-0.5, 0.5)$, $(-0.5, -0.5)$ and $(0.5, -0.5)$.

In addition, again following Secomandi, we perform two sets of experiments: *SC* (small number of customers) and *SLC* (somewhat larger number of customers). The above descriptions, choices and parameters apply to both. We first describe and detail *SC* (the details of *SLC* are provided in the next section).

For *SC*, $n \in \{5, ..., 8\}$, and $\overline{f}' \in \{0.75, 1.25, 1.75\}$. We carry out 5 replications and 20 iterations per replication, i.e., we consider 100 problem instances (except for $n = 8$, and the deviation is explained in detail below) for each $(n, Q, \overline{f}')$ combination (Table 3.2); and report the average expected distance under each policy for these 100 instances. While the exponential nature of the algorithms perforce limits the number of replications and iterations, improved computational

power enables us to undertake more of them than Secomandi (one replication with 10 iterations per replication for both SC and SLC).

Table 3.2.
Customers, vehicle capacities and $\overline{f}'$ for SC

	$n = 5$	$n = 6$	$n = 7$	$n = 8$
$Q, \overline{f}'$	9, 0.75	10, 0.75	12, 0.75	14, 0.75
$Q, \overline{f}'$	7, 1.25	8, 1.25	9, 1.25	11, 1.25
$Q, \overline{f}'$	5, 1.75	7, 1.75	8, 1.75	9, 1.75

3.8 Computational Experiments-Analysis of Results

Clearly, SNP would be the least burdensome computationally among the policies we consider, and in SC, we use it as the base to examine the improvements that the others bring about, as well as the additional computational effort they require. The algorithms were implemented in Microsoft Visual C++ 6.0 on a Pentium Intel x86 1GHz PC with 325 MB RAM. The RAM capacity unfortunately proved inadequate for completing the 100 runs planned for the case $n = 8$ for RNP and NP—some of the randomly generated customer demands turned out to be "too large"—thus forcing reduction in sample size[1]. The solution values obtained in this case are, however, more or less consistent with the trends observed in the $n < 8$ cases and so we have included these, with the obvious caveat that these might not be entirely representative. The results of SC are summarized in Table 3.3. (Improvements are with respect to SNP and a '*' indicates deterioration.)

[1]A 2 GB RAM Pentium 3 machine with the same CPU speed was used for this case to avoid using disk memory and the consequent disproportionate increase in running times, and as many iterations were carried out as did not entail disk use—which turned out to be 1 replication of 17 iterations. Inadequate memory capacity also prevented extension of the inter-policy comparisons beyond the $n = 8$ case.

Table 3.3.
Expected Distances, *SC*

		expected distance				% improvement vs. *SNP*			
	Q	SNP	$CFRNP$	NP	RNP		$CFRNP$	NP	RNP
$n = 5$	9	2.5606	2.5630	2.5541	2.5538		0.09*	0.26	0.27
	7	2.8374	2.8602	2.8263	2.8248		0.80*	0.39	0.45
	5	3.3153	3.3613	3.2847	3.2837		1.39*	0.92	0.95
						average	0.76*	0.52	0.55
$n = 6$	10	2.8254	2.8320	2.8204	2.8198		0.23*	0.18	0.20
	8	3.0858	3.0964	3.0761	3.0750		0.34*	0.31	0.35
	7	3.2920	3.3153	3.2771	3.2740		0.71*	0.45	0.55
						average	0.43*	0.31	0.37
$n = 7$	12	2.8660	2.8718	2.8630	2.8624		0.20*	0.11	0.13
	9	3.1941	3.2038	3.1853	3.1837		0.30*	0.27	0.33
	8	3.3728	3.3867	3.3568	3.3547		0.41*	0.48	0.54
						average	0.31*	0.29	0.33
$n = 8$	14	3.2871	3.2920	3.2867	3.2867		0.14*	0.01	0.01
	11	3.4906	3.4946	3.4874	3.4872		0.12*	0.09	0.10
	9	3.7227	3.7243	3.6991	3.6982		0.04*	0.63	0.66
						average	0.10*	0.25	0.26

In essence, the running times are in the order $SNP < CFRNP < NP < RNP$, while the solution values are in the order $RNP < NP < SNP < CFRNP$. As far as $CFRNP$ is concerned, it therefore appears that it would be useful only if its assumptions were satisfied, i.e., if partial service at a customer were not allowed. In other words, the benefits of re-optimization in case $q < \delta_i$ at customer i under $CFRNP$ are apparently outweighed by the effects of having to return to the depot with non-zero residual capacity, a contingency that does not arise under any of the other policies.

Regarding NP and RNP, the percentage improvements they bring about *vis-a-vis* SNP are positively correlated with $\overline{f}'$. Intuitively, the consequences of not re-optimizing fully after route

Table 3.4.
CPU times in seconds per run, *SC*

	$n = 5$					$n = 6$			
Q	SNP	$CFRNP$	NP	RNP	Q	SNP	$CFRNP$	NP	RNP
9	< 0.1	< 0.1	0.1	0.1	10	< 0.1	0.3	0.9	1.1
7	< 0.1	< 0.1	0.1	0.1	8	< 0.1	0.3	0.8	1.0
5	< 0.1	< 0.1	0.1	0.1	7	< 0.1	0.3	0.7	0.8
	$n = 7$					$n = 8$			
Q	SNP	$CFRNP$	NP	RNP	Q	SNP	$CFRNP$	NP	RNP
12	< 0.1	3.2	12.7	15.3	14	< 0.1	24.8	75.2	89.6
9	< 0.1	2.4	9.1	11.1	11	< 0.1	19.6	60.1	71.6
8	< 0.1	2.2	8.0	9.8	9	< 0.1	16.2	49.5	59.8

failure, as in *SNP*, should be more apparent when there are a larger number of returns to the depot, and this is what the results seem to indicate. It seems, therefore, that NP and RNP are more likely to be useful for higher $\overline{f}'$.

The differences between NP and RNP themselves, while evident in 11 of the 12 cases considered, do not appear significant. Experiments with different assumed parameters (in particular, non-uniform demand distributions that could, intuitively speaking, provide a more asymmetrical setting) are needed, however, before the utility of reneging can be ruled out.

In a similar way, the *average* percentages of improvement provided by NP and RNP *vis-a-vis* SNP do not seem worth the computational effort; however, the *individual* percentage differences show more variability if we consider the individual replications or iterations. While the detailed results are not shown here, for the case $(n, Q, \overline{f}') = (7, 12, 0.75)$, for instance, the maximum percentage difference is 0.68% whereas the overall % difference is 0.13% (Table 3.3). Similarly, while $CFRNP$ always does worse than SNP on the average, for the case $(n, Q) = (8, 9)$, it does better on 8 of the 17 iterations. This suggests that in actual applications, NP/RNP would provide less or more improvement depending on problem parameters.

Furthermore, since we have considered only relatively small values of n and only one demand distribution, the better solution quality obtained through NP/RNP should at least provide the motivation for further experiments with the "more exact" policies. While such experiments would of course depend upon the availability of increased computational power, our experiments can in the meantime hopefully provide some guidelines about their computational feasibility (Table 3.4).

Coming now to *SLC*, we consider only *SNP* and summarize in Table 3.5 the parameters and results of the earlier and the current efforts. Specifically, the differences between these are: For customer demands *we continue with the procedure as in SC*, since it adds a further dimension of randomness, and choose constant $Q = 15$, for comparability. (Secomandi [32] uses only one demand distribution for each (n, Q) pair). Moreover, in these instances, comparison with the other exact policies is obviously not possible and the primary purpose therefore is testing computational viability. Consequently, only 4 replications and 10 iterations/replication are used[2].

Our computation times are much lower (though Secomandi's Pentium PC was only a 233 MHz version) and indicate that *SNP* does offer *some* prospects for extension of an exact MDP technique to slightly larger instances. Since, moreover, at least within our experimental limitations, *SNP* appears to be fairly competitive with the more exact policies as regards solution quality also, it seems plausible that the attempt at such an extension would be worthwhile. Our next step, therefore, is to take a "heuristic" look at what *SNP* could realistically hope to accomplish in actual practice. The crippling factor is the term 2^n in the complexity and not much can be done about it. Consequently, our only hope lies in the manageability of K and Q, and we now focus on the computational limits of these parameters. To do this, we visualize the algorithmic steps during an actual implementation and arbitrarily consider the k customer case, $k < n$. At this stage, the values of the $k-1$ customer case would have been determined. The algorithm would consider $\binom{n}{k}$ sets and in each set s evaluate in turn $V(\cdot)$ corresponding to visiting first each of the k customers in s. For a given $i \in s$ the operation would involve the comparison of (r, j) and $(r, 0)$ for all

[2]For $n = 19$ the 2 GB machine was used.

Table 3.5.
Experimental parameters and results-*SLC*

	Secomandi				Das			
n	Q	D_i	$\overline{f}'$	$CPUsecs./run$	Q	$\overline{f}'$	$expdist.$	$CPUsecs./run$
9	8	$U(0,4)$	1.25	506.7	15	0.8	3.33	< 0.1
10	8	$U(0,4)$	1.50	1338.3	15	1.0	3.77	0.1
11	6	$U(1,3)$	2.67	1653.9	15	1.2	3.80	0.1
12	5	$U(1,3)$	3.80	3551.9	15	1.4	3.92	0.3
13	–	–	–	–	15	1.6	4.34	0.8
14	–	–	–	–	15	1.8	4.47	1.8
15	–	–	–	–	15	2.0	4.71	4.1
16	–	–	–	–	15	2.2	4.97	9.4
17	–	–	–	–	15	2.4	5.28	20.4
18	–	–	–	–	15	2.6	5.40	49.7
19	–	–	–	–	15	2.8	5.61	122.3

$j \in s\backslash\{i\}$, $q \in \{0, ..., Q\}$, which involves $k-1$ customer cases. When the case $q = Q$ is considered, computations corresponding to all values in $\{1, ..., K\}$ would be necessary. When the cases $q < Q$ are considered, however, it would be necessary only to consider the *actual* demand values that D_i can assume. (If $q < \delta_i$, the customer will be immediately revisited with modified demand less than K, and with $q = Q$, the $V(\cdot)$ corresponding to which would already have been computed.)

Now it is often the case that practitioners use only the three categories *High*, *Medium* and *Low* to describe demands; even assuming the availability of accurate historical data, it might in fact be difficult to distinguish between small variations in demand. Let us then assume that each customer demand can take only three values, as in our experiments. The running time is then $O(Kn^2 2^n)$ or $O(3n^2 2^n Q)$. Since $Q > K$ by assumption, the latter is the greater and accordingly is taken as the worst-case running time. Our *SLC* results indicate a growth rate in running time (RT) somewhat less than 2.5 for constant Q. As a first approximation, extrapolating this growth rate (assuming adequate RAM capacity—not very difficult for a company to acquire), and noting

that dependence on Q is linear, we find that for $n = 19$ and $Q = 100$, RT would be around 10 minutes. Thus $n = 18$ should take about 4 minutes and so on downwards. Computations on run, as demands unfold, should then be feasible if it is (reasonably) assumed that vehicle halts would be more than the running times.

The above discussion thus indicates that exact MDP policies can be useful in "small" instances (and the consistently rising curve of computational ability in tandem with parallel computation should continue to extend the frontier). On the flip side, one must simultaneously conclude that given present-day technology, no exact policy would be computationally viable in generating optimal solutions if used in unalloyed form on large, say, $n > 20$ problems. More significant, consequently, is the question: what sort of "alloying" applied to *SNP* would be tractable computationally and rewarding result-wise? Such alloying must involve the use of heuristic/approximation techniques. We turn, therefore, to a study of such techniques in Chapter 4, and in Chapter 5 perform computational experiments that imbed *SNP* within a heuristic technique and are thus able to address larger problems.

Chapter 4

Bounded a-priori solutions, VRP and VRPSD

The focus of this chapter, as discussed earlier, is "good" and bounded *a-priori* solutions. In Section 4.1.1 we briefly recapitulate the heuristic for multi-vehicle VRP that provides the best-known approximation ratio of 2.5 to the optimal solution. Sections 4.1.2 through 4.1.4 discuss our modifications that lead to strongly polynomial computational time bounds for this heuristic solution, while retaining the same approximation ratio. Section 4.1.5 treats a special case that provides intuition for an improved approximation ratio. Section 4.2 develops this intuition to obtain such an improved ratio for VRP/VRPSD under certain additional assumptions. Finally, Section 4.3 extends our modified heuristic of Section 4.1 to the stochastic VRP with a single vehicle.

4.1 VRP-Multiple Vehicles

4.1.1 Recapitulation of Haimovich et al. [46]

Define $X = N\backslash\{0\}$, $X = \{1, \ldots, n\}$, and assume that each $i \in X$ has (deterministic) unit demand. X is served by m vehicles each of capacity Q, where $m = \lceil n/Q \rceil$. Define $X_j^0 = X_j \cup \{0\}$, where X_j denotes the set of customers that a VRP solution assigns to the j^{th} vehicle. Denote by $R^*(N)$ and $T^*(S)$ the total distances traveled in the optimal VRP solution and in the optimal TSP solution on a set S, respectively. To be consistent with the notation of these authors, we also define $\ell_i = c_{0i}$ for each $i \in X$, and $\overline{\ell} = \sum_{i \in X} \ell_i / n$.

We then have the following bounds for the optimal VRP solution

$$T^*\left(X_j^0\right) \geq 2 \max_{i \in X_j} \{\ell_i\} \geq 2 \frac{\sum_{i \in X_j} \ell_i}{|X_j|} \geq (2/Q) \sum_{i \in X_j} \ell_i \tag{4.1.1}$$

and hence that,

$$R^*(N) = \sum_j T^*\left(X_j^0\right) \geq (2/Q)\sum \ell_i = 2n\bar{\ell}/Q \tag{4.1.2}$$

The Optimal Tour Partition (OTP) heuristic proposed by these authors breaks the optimal TSP tour through X into disjoint segments, none incorporating more than than Q customers, and the endpoints of each segment are connected to the depot by a "radial" edge (i.e., an edge of the form ℓ_i). Suppose the TSP tour is, in order, $1, 2, \ldots, n$. The process is repeated for each of the n tours $i, i+1 \ldots n, 1, 2, \ldots, i-1$, where $i \in \{1, \ldots, n\}$, thus obtaining n feasible solutions. Since the cumulative length of the n solutions is $2m\sum \ell_i + (n-m)T^*(X)$, and the best of these (denoted by $R^H(N)$) can be no more than their average, we have

$$R^*(N) \leq R^H(N) \leq 2m\bar{\ell} + (1-(m/n))\,T^*(X) \tag{4.1.3}$$

Since $R^*(N) \geq T^*(X)$ by the triangle inequality, the above equations then imply that

$$\max\left\{(2n/Q)\,\bar{\ell}, T^*(X)\right\} \leq R^*(N) \leq 2\lceil n/Q \rceil\,\bar{\ell} + (1-(m/n))\,T^*(X) \tag{4.1.4}$$

Considering now the usual situation in practice where we have a heuristic τ-optimal (i.e., $1+\tau$ times optimal), rather than an exact, solution to the TSP, (4.1.3) can be rewritten

$$R^H(N) \leq 2m\bar{\ell} + (1-(m/n))\,(1+\tau)\,T^*(X) \tag{4.1.5}$$

The worst case ratio can then be obtained as follows. From (4.1.4) and (4.1.5) we have

Case: $(2n/Q)\,\overline{\ell} \geq T^*(X)$

$$\begin{aligned} R^H(.)/R^*(.) &\leq \left(2m\overline{\ell} + (1-(m/n))(1+\tau)(2n/Q)\,\overline{\ell}\right)/(2n/Q)\,\overline{\ell} \\ &= \left(m + (1-(m/n))(1+\tau)(n/Q)\right)/(n/Q) \\ &= 1+\tau+m(Q-1-\tau)/n \\ &= 1+\tau+\lceil n/Q\rceil/(n/(Q-1-\tau)) \end{aligned} \tag{4.1.6}$$

Case: An identical result is obtained if we assume $(2n/Q)\,\overline{\ell} \leq T^*(X)$.

4.1.2 A Modification—Unit Customer demands

If $\lceil n/Q\rceil = n/Q$, the ratio $R^H(.)/R^*(.) \leq 1+\tau+(Q-1-\tau)/Q = 2+\tau-(1+\tau)/Q$. If we also assume that the best-known value of τ (=1/2) [45] is used, $R^H(.)/R^*(.)$ is at most $2.5-[3/(2Q)]$. Haimovich et al. [46] do not, however, provide a better bound than that given by (4.1.6) if $\lceil n/Q\rceil \neq n/Q$. In fact, Charikar et al. [48] refer to the OTP as a 3-approximation heuristic, but it is possible for the worst case ratio to exceed 3. Thus, if we take n=20 and Q=19, (4.1.6) gives a value of 1+0.5+2(17.5/20)=3.25.

To obtain a ratio of $2.5-[3/(2Q)]$ even when $\lceil n/Q\rceil \neq n/Q$ (i.e., in all cases), let us consider the following modification to OTP (we call the modified heuristic OTPD).

1. Assume that $\lceil n/Q\rceil \neq n/Q$ and let u be the smallest integer such that $\lceil (n+u)/Q\rceil = (n+u)/Q$.
2. Let $n+u=\nu$, so that $m=\lceil n/Q\rceil = \nu/Q$.
3. Obtain a τ-optimal heuristic TSP solution on N (e.g., via the Christofides algorithm [45]).
4. Create the set X' from X by adding to the latter u customers, $\{n+1, n+2, \ldots, n+u\}$, each located at the depot and each with unit demand. These "spurious" customers are inserted

between the last customer on our TSP tour and the depot. (Note that the insertion does not violate any capacity restrictions.)

5. Apply OTP with ν iterations instead of n.

LEMMA 4.1.1. *Procedure OTPD provides a solution with approximation ratio of 2.5 to the optimal VRP solution in linear computational time.*

Proof. Since the optimal VRP solution remains unchanged and since any solution that serves all customers in X can also serve all customers in X' without traversing additional distance, (4.1.3) can now be written

$$R^*(N) \leq R^H\left(X'\right) \leq 2m\overline{\ell}^* + (1-(m/\nu))\,T^*(N) \tag{4.1.7}$$

where $\overline{\ell}^*$ is the average over all ν radii, i.e. $\overline{\ell}^* = \left(\sum_i^{\nu} \ell\right)/\nu$.

Next, by the triangle inequality, we have $R^*(N) \geq T^*(N)$. This can be seen as follows. If there is only one vehicle route, $R^*(N) = T^*(N)$. Otherwise, we can concatenate the m vehicle routes by joining the last customer on the ith route, $i \in \{1,\ldots,m\}$, with the first on the $(i+1)$st where $m+1 \equiv 1$. By the triangle inequality, the result is no larger than $R^*(N)$. Since the resulting configuration is a TSP tour, $R^*(N) \geq T^*(N)$.

Then, from this result and (4.1.7), equation (4.1.4) can be modified as under

$$\max\left\{(2n/Q)\,\overline{\ell}, T^*(N)\right\} \leq R^*(N) \leq 2m\overline{\ell}^* + (1-(m/\nu))\,T^*(N) \tag{4.1.8}$$

By construction, we have $(2n/Q)\,\overline{\ell} = (2\nu/Q)\,\overline{\ell}^*$. Now proceeding exactly as in the derivation of (4.1.6) with $\overline{\ell}^*$ replacing $\overline{\ell}$ and ν replacing n, we are led to exactly the same result. Further, since $\lceil \nu/Q \rceil = \nu/Q$, we have our ratio of 2.5.

In executing OTPD, moreover, there can be at most Q distinct groupings of the customers, since in the $(Q+1)th$ run, we obtain precisely the configuration obtained in the first run (i.e., every customer has precisely the same predecessor and successor *on its route* and thus any configuration

evaluated by OTPD repeats after every Q runs). Hence OTPD need execute no more than Q iterations, each with $O(m)$ operations. The running time is thus $O(mQ) = O(\nu Q/Q) = O(\nu) = O(n)$, since $\nu < 2n$. (We assume the availability of a starting TSP solution—otherwise running the Christofides algorithm [45] would take $O\left(n^3\right)$). On the other hand, the original OTP heuristic would run in $O(nm) = O(n\lceil n/Q \rceil) = O(n((n/Q)+1)) = O\left(n^2/Q\right)$ (and $O(n)$ only in the special case where $\lceil n/Q \rceil = n/Q$, and appropriate checking conditions were included in the algorithm).

Also, if in any segment (vehicle route) we have the situation $\{a_i, \ldots, a_k, \sigma, \sigma, \sigma, a_u, \ldots, a_w\}$ where an a refers to an "actual" and a σ to a "spurious" customer, we could further improve solution quality by incorporating in the algorithm a provision to remove the edges $a_k\sigma$ and σa_u, and include $a_k a_u$. Since this can happen in at most one segment per iteration, the additional work would involve only one comparison and 3 addition/subtraction operations in an iteration, and the order of the running time would not be affected. Let us call this provision Z.

We note also that although OTP examines a larger number of (distinct) solutions than OTPD does, the solution space of OTPD is not a subset of that of OTP. For instance, if we have the customers $\{a, b, c, d, e, f\}$ and Q=4, $\{\{a, b, c\}, \{d, e, f\}\}$ is a solution that would be examined by OTPD but not by OTP.

Also, while in later work, Haimovich et al. [49] proposed a version of OTP (we call it OTPM) that provides a ratio of $2.5 - [3/(2Q)]$ irrespective of whether $\lceil n/Q \rceil = n/Q$ or not, OTPD operates differently and also allows generalization to the stochastic VRP (as we show in Section 4.3). Further, with the inclusion of provision Z, OTPD would in practice be expected to find a better solution than OTPM (with the same running time bounds), as the following example demonstrates.

Let the customers be $\{a, b, c, d, e, f, g\}$ and suppose that $Q = 4$. The solutions examined by OTPM would be $\{a\}, \{b, c, d, e\}, \{f, g\}$; $\{a, b\}, \{c, d, e, f\}, \{g\}$; $\{a, b, c\}$, $\{d, e, f, g\}$; and $\{a, b, c, d\}, \{e, f, g\}$. By contrast, OTPD would use only 2 vehicles in all solutions and, with "spurious" trips to the depot eliminated (as per provision Z), would do better than OTPM.

4.1.3 Arbitrary demands

We next consider customers with arbitrary, bounded integer demands, and with more than one vehicle allowed to visit a customer having demand d_i. We treat every customer i as d_i customers, each with unit demand and connected to the depot with an edge of length ℓ_i so that, in the optimal (or τ-optimal) TSP solution, we:

1. Replace customer i with d_i copies $i_1, i_2, \ldots, i_{d_i}$.
2. Introduce the edges $\left(i_k, i_{k+1}\right), k \in \{1, \ldots, d_i - 1\}, d_i \geq 2$, each with cost 0.
3. Assuming an arbitrary (say clockwise) orientation of the TSP tour, replace each edge (i, j) of the original network with edge $\left(i_{d_i}, j_1\right)$ if $i \neq 0$ and $j \neq 0$, edge $\left(i_{d_i}, 0\right)$ if $j = 0$, and edge $(0, j_1)$ if $i = 0$. The cost of such a newly introduced edge remains c_{ij}, and the total number of customers now is $n' = \sum_i d_i$.
4. As in the previous section, introduce u copies of an additional spurious customer σ located at the depot and add them to the network as per the rules above.

The number of vehicle routes is now given by $m = \left\lceil n'/Q \right\rceil = \nu'/Q$, where

$$\nu' = n' + u \tag{4.1.9}$$

If $\nu' = n'$, a direct application of (4.1.1) through (4.1.6) with n' replacing n gives us the desired bound. Otherwise, we apply OTPD, and follow the steps as in the unit demand case with ν' and n' replacing ν and n respectively, to get the same ratio of 2.5.

Q runs of the algorithm thus again suffice, and the running time is consequently $O(mQ) = O\left(v'\right) = O\left(n'\right) = O(nK)$ (where K is the maximum permissible demand value). Alternatively, we could assume that $d_i < Q \ \forall \ i \in \{1, \ldots, n\}$. (If this condition is not satisfied for any d_i, while running OTPD we can assume that the demand is $d_i' = d_i \bmod Q$ and add $\ell_i \lfloor d_i/Q \rfloor$ to each of the Q solutions. If $d_i = jQ$, with j an integer, and in any route, i is entirely served by j vehicles, we

have the same result. If not, by the triangle inequality, we can obtain a better solution by removing i from the tour and connecting the immediate neighbors of i.) We thus have an $O(nQ)$ algorithm, since now the number of routes cannot exceed n.

4.1.4 Polynomial time bound, arbitrary demands

If K or Q were "small" relative to n, we would obtain practically a linear time algorithm. If, however, either $Q > \log n$ or $K > \log n$, (in what follows, it is understood that logarithms are to the base 2), we can obtain the better upper bound of $O(n \log n)$. This bound, moreover, is independent of Q or K and is thus strongly polynomial. The idea is as follows.

When we perform a fresh iteration of OTPD, the new configuration obtained would not represent a solution distinct from the previous one if none of the segment endpoints change. (This possibility does not arise in the unit demand case.) Hence, a candidate configuration should be evaluated only if there is such a change *vis-a-vis* the last evaluated configuration. By keeping track of the changes as they occur, we would be in a position to determine when (and what) next change(s) would occur and thus skip unnecessary searches or computations. This will enable us to obtain an $O(n \log n)$ time bound. The procedure is summarized below.

1. As in Section 4.1.3, assume WLOG that $d_i < Q$ for every $i \in \{1, 2, \cdots, n\}$.

2. If $\sum_i d_i$ is not an integer multiple of Q, add an $(n+1)th$ customer, with adequate demand and located at the depot, to the set X, creating the set X'.

3. Find the best possible TSP tour through the set X or X', as the case may be.

4. Assume clockwise tour orientation, and number the customers accordingly.

5. Create customer objects with the following data fields: predecessor and successor in the TSP tour (the successor of $n+1$—or n—as the case may be, is 1), d_i, as well as *priority* and *repeat* fields.

The last two fields are initiallized as follows.

$sum \leftarrow d[1]; \quad priority[1] \leftarrow \ -1$

for (customer i=2 to n+ 1) *do*

$sum \leftarrow sum + d[i]$;

if($sum > Q$) *then*

$sum \leftarrow priority[i] \leftarrow (sum - Q); \quad repeat[i] \leftarrow 1;$

else if ($sum = Q$) *then*

$sum \leftarrow 0; \quad priority[i] \leftarrow 1; \quad repeat[i] \leftarrow 0;$

else

$priority[i] \leftarrow \ -1; \quad repeat[i] \leftarrow \ -1;$

$sum \leftarrow d[1]; \quad priority[1] \leftarrow \ -1$

In words, non-dummy (i.e., greater than 0) priorities are assigned in only the following two cases: (a) if i_1 is served by vehicle j and i_{d_i} by vehicle $j+1$ (if $j = m$, $j + 1 \equiv 1$), the priority is given by the portion of the demand that carries over into route $j + 1$. Thus if α_i units are being supplied by vehicle j, the priority of i would be $d_i - \alpha_i$, and (b) if i_{d_i} is the trailing endpoint of a route, i would have priority 1.

Let *VRPval* be the value of the initial solution (its computation takes $O(n)$ time since there can be at most n routes). Next, form a min-priority queue of the customers with priority greater than 0 and perform the following operations.

$best \leftarrow VRPval$

while (total number of deletions $\leq n$+1)

perform a series of deleteMin operations on priority queue until priority of object deleted remains the same as *value*, where *value* is the priority of the first object deleted (a)

for (each object i deleted from queue) *do* (b)

if ($repeat[i]$=0)

if (d[succ[i]]=1)

$VRPval \leftarrow VRPval + c[0][succ[succ[i]]] - c[0][i];$

$repeat[succ[i]] \leftarrow 0; \quad priority[succ[i]] \leftarrow value + 1;$

else

$VRPval \leftarrow VRPval + c[0][succ[i]] - c[0][i];$

$repeat[succ[i]] \leftarrow 1; \quad priority[succ[i]] \leftarrow value + d[i] - 1;$

insert $succ[i]$ in priority queue;

else

$VRPval \leftarrow VRPval + c[0][succ[i]] - c[0][i];$

$repeat[i] \leftarrow 0; \quad priority[i] \leftarrow value + 1;$

insert i in priority queue;

for loop ends

if $(VRPval < best)$

$best \leftarrow VRPval$

while loop ends

Output *best*

LEMMA 4.1.2. *The above procedure runs in $O(n \log n)$ time and requires $O(n)$ space.*

Proof. The operations at line (b) involve no more than n objects. Each object can be placed in the priority queue at most twice and is removed exactly once. This is because a customer is deleted only if it becomes a trailing endpoint at any iteration. It cannot become a trailing endpoint again until Q more iterations. But since each remaining customer is now at a distance $1, 2, \ldots, Q-1$ from a trailing endpoint, all of these customers must become trailing endpoints and be deleted before i becomes a trailing endpoint again. Since we allow only $n + 1$ deletions, no object can be deleted twice. Also, since the object with minimum priority is always at the first index in the queue, after each deleteMin we need check only one index to determine whether to delete the next object. Thus the checking operations at (a) add at most n constant-time operations. The initial binary heap can be built in $O(n)$ and each subsequent deleteMin or insertion takes $O(\log n)$. For every object

accessed, only constant time operations are performed. Moreover, no array used is of size larger than $O(n)$. Hence we need $O(n \log n)$ time and $O(n)$ space.

While the optimal VRP solution obtainable might be improved (since a larger number of solutions would be examined) by allowing a customer to be served by more than one vehicle (even when $d_i < Q \ \forall i$), as compared to the case where such splitting of demands is not allowed, (4.1.1) and (4.1.2) (and consequently (4.1.4)) are independent of the procedure employed to obtain such a solution. Hence, for the performance ratio of 2.5 to hold, it suffices to make the "splitting allowed" assumption only for OTPD, irrespective of the procedure followed in obtaining the optimal solution. However, as observed by Haimovich et al. [46], not allowing splitting introduces bin-packing features into the VRP and OTPD does not guarantee feasible solutions.

4.1.5 Equal Radial Distances

The bounds on solution values in the preceding sections were based on the idea of combining the bounds for "radial" edges and "tour" edges. If special conditions exist with regard to any of these sets of edges, can bounds be improved? This is a question we attempt to answer in this section—the intuition being that the analysis would be a helpful first step in lowering bounds for the general case. Specifically, we will assume that all customers are equidistant (with common distance ℓ) from the depot.

Consider first the following straightforward heuristic procedure to construct a VRP solution in this special case. (We use Kruskal's procedure for obtaining a Minimum Spanning Tree (MST) connecting a given set of nodes—for a lucid exposition, see Cook et al. [43].) We also scale down the vehicle capacity and all customer demands by a factor of Q.

Heuristic Procedure: Start procedure KRUSKAL on X. Stop when the minimum feasible edge length is ℓ (no edge e with $c_e \geq \ell$ is included). Suppose at this stage there are k forests. Consider an arbitrary forest $\mathcal{F}_i$. Denote by d_i the sum of the (scaled) demands of its nodes, by E_i its edge set and by T_i the sum of the lengths of the edges in E_i. Duplicate E_i. The resulting

graph is connected and all its nodes have even degree, i.e. it is Eulerian. Connect the depot to the two nodes at the ends of the longest path in $\mathcal{F}_i$ and remove one edge (of the two edges that now exist) between adjacent nodes on this longest path. The graph remains Eulerian. A traversal of this graph visits all customers in $\mathcal{F}_i$ as well as the depot and incurs cost no more than $2\ell + 2T_i - \lambda_i$, where λ_i is the maximum path length in $\mathcal{F}_i$.

Vehicle i then follows route i until all nodes in $\mathcal{F}_i$ have been served. If route failure occurs at customer j, the vehicle returns to the depot and next proceeds to j or $j+1$ as necessary.

We now bound the ratio of $R^H(\cdot)/R^*(\cdot)$. Our method is to apportion the total cost of $R^*(\cdot)$ among our heuristic vehicle routes and then to carry out routewise comparisons.

The ith route can have at most $\lfloor d_i \rfloor$ route failures ($\lfloor d_i \rfloor - 1$ if $\lfloor d_i \rfloor = d_i$)—vehicle capacity being exactly exhausted after serving the last customer on the route is not considered a failure. Hence, not considering any tour edges saved through trips to the depot,

$$R^H(X_i^0) \leq 2\ell \lceil d_i \rceil + 2T_i - \lambda_i \tag{4.1.10}$$

Consider now the optimal set of routes. There must be at least $2\sum_{i=1}^{k} \lfloor d_i \rfloor$ radial edges. Moreover, each customer must be visited at least once. The total cost of serving the customers in $\mathcal{F}_i$ (howsoever they are clustered or sequenced in the optimal solution) must be at least T_i. Accordingly, we can apportion $\ell \lfloor d_i \rfloor + T_i$ of the optimal cost to the ith route. If a route in the optimal solution visits exactly the customers in $\mathcal{F}_i$, then ℓ must be added to the apportionment *return to the depot*. Otherwise, the optimal solution must have at least one edge from $u \in \mathcal{F}_i$ to $v \in \mathcal{F}_j$, $i \neq j$ with $c_{uv} \geq \ell$ (the way we ran KRUSKAL ensures this), and the apportionment to $\mathcal{F}_i$ is again at least ℓ. Moreover, this also implies that the edge $(0, v)$ is not included in the optimal solution and so in allocating $\left\lfloor d_j \right\rfloor$ radial edges to $\mathcal{F}_j$ we will not be counting twice. Hence, the apportionment to $\mathcal{F}_i$ must be at least $\ell \lceil d_i \rceil + T_i$. This must hold for all forests and so from equation (4.1.10) we have $R^H(\cdot)/R^*(\cdot) \leq 2$.

4.2 Better approximation ratio, special case, multi-vehicle VRP

In this section, we attempt to obtain an approximation ratio less than 2.5 for another special case of multi-vehicle VRP, as explained below. We start with a general LP formulation of the multi-vehicle VRP.

4.2.1 Linear program

Let x_e denote the number of times the vehicle traverses edge e. For a given set $S \subset N$ we denote by $\delta(S)$ the *cut* of S, i.e., the set of edges which have only one end in S. We also denote by $x(\delta(S))$ the sum $\sum (x_e : e \in \delta(S))$ Our LP may then be stated as follows.

$$\text{Minimize} \sum (c_e x_e : e \in E) \tag{4.2.1}$$

subject to:

$$x_e \geq 0 \text{ for each } e \in E \tag{4.2.2}$$

$$x(\delta(A)) \geq 2\lceil \sum_{i \in A} d_i \rceil \text{ for each } A \subseteq \{1, \ldots, n\} \tag{4.2.3}$$

Our objective is to find a set of vehicle routes whose cumulative length is at most twice as large as the optimum solution to the above linear program. While the above formulation neither incorporates vehicle capacities, nor specifically provides an IP formulation, it does provide the lower bound necessary for a factor 2 approximation. For convenience, we scale down the vehicle capacity and all customer demands as in Section 4.1.5.

Also, we restrict ourselves to a VRP that incorporates the following additional assumption:

Customer i is allowed to have only "large" (say, $d_i = 1 - n^{-1}$), or "small" (say, $d_i = n^{-2}$) demands. Consequently, (a) no two "large demand" customers can be served by a single vehicle, and (b) a vehicle that serves a given "large demand" customer can additionally serve any number of "small demand" customers.

We alternatively refer to the respective customers as "1"-demand customers (or "delivery points"), and "0"-demand customers (or "visit points").

4.2.2 Zero demands

We first describe our solution procedure for the case where all d_i are restricted to small values. The VRP now reduces to a TSP, and the LP formulation of Section 4.2.1 is accordingly modified as under:

$$\text{Minimize} \sum (c_e x_e : e \in E) \tag{4.2.4}$$

subject to:

$$0 \leq x_e \leq 1 \text{ for each } e \in E \tag{4.2.5}$$

$$x(\delta(A)) \geq 2 \text{ for each } \emptyset \subset A \subset N \tag{4.2.6}$$

Following the notation of Cook et al. [43] we denote by y_A the dual variable corresponding to the constraint equation on the cut of the set A. The dual of the above LP can then be written as:

$$\text{Maximize} \sum (2y_A : \emptyset \subset A \subset N) \tag{4.2.7}$$

subject to:

$$\sum (y_A : e \in \delta(A) : \emptyset \subset A \subset N) \leq c_e \text{ for each } e \in E \tag{4.2.8}$$

$$y_A \geq 0 \text{ for each } \phi \subset A \subset N \tag{4.2.9}$$

Our solution approach is one used earlier, for instance, by Agrawal et al. [77] and Goemans et. al. [78]. Like the primal-dual simplex method, these approaches start with a dual feasible solution that is not necessarily basic. The idea is to grow balls around sets of nodes, starting with each node

as a singleton set, and with set A having radius y_A, the vector y being initialized to 0 (note that this is a feasible solution). Balls grow until two balls collide, at which stage the two corresponding sets of nodes are merged and converted into a single ball, and the process restarts. Essentially, the process of ball-growing adds a constant to all respective dual variables, and thus "consumes" portions of the edges. Since the corresponding sets of nodes are separated by the edges of the network, any collision of balls corresponds to some edge $e \in E$ being fully "consumed". Hence, the corresponding sets of nodes are no longer separated and are therefore merged. Equivalently, a "collision" can be viewed as the corresponding dual constraint becoming "tight", i.e., becoming exactly satisfied. In turn, this results in an edge being added to the spanning tree defined on the set of nodes.

We apply the above procedure as follows. Define $T(A)$ as the set of edges of the spanning tree contained in A and denote the reduced cost (or *residual length*) $\overline{c}_e$ of edge e by $c_e - \sum(y_A : e \in \delta(A) : \phi \subset A \subset N)$ for every $e \in E$. A *tight* edge is an edge e for which $\overline{c}_e = 0$. At any time during the execution of our algorithm, the nodes are partitioned into *active* and *inactive* sets. An active set is one that is still "growing", and an inactive set is one that has stopped growing. For convenience, we will henceforth refer to the dual variable corresponding to a set as its *coefficient*. Formally, the algorithmic steps are:

Initialization. All sets are singletons, the depot forms an inactive set, and the customers form active sets. y_A is initialized to 0 and $T(A)$ to $\emptyset$ for all A as defined in (4.2.7) through (4.2.9).

Iteration. For each edge we define priority $\pi(e)$: if e intersects a active sets (where $a \in \{0, 1, 2\}$) then $\pi(e) = \overline{c_e}/a$; in particular, if $a = 0$, $\pi(e) = \infty$. We pick an edge e with the minimal priority $\rho = \pi(e)$.

If $\rho = \infty$, we terminate (note that this corresponds to the inclusion of all nodes in the spanning tree).

Otherwise, we add ρ to every coefficient y_B corresponding to active set B. Let us denote by A_1 and A_2 the two sets intersected. Two cases arise:

1. If both A_1 and A_2 are active, we replace them with a new active set $A = A_1 \cup A_2$ and add edge e to $T(A)$.

2. If only A_1 is active, we make it inactive and add e to $T(A_1)$.

Once set A has been created in the first case, or A_1 has become inactive in the second, we make no further changes to the values of y_{A_1} or y_{A_2}, or of y_{A_1}, respectively.

We also add two copies of e to our primal solution.

Analysis. For each set A in the partition maintained by our algorithm, we define

- $LB(A) = 2\sum(y_B : B \subset A)$
- $cost(A) = 2\sum_{e \in T(A)} c_e$.

Our sets satisfy the following invariants.

1. If A is active, $T(A)$ is a spanning tree of A;

2. if $\mathcal{A}$ is the family of inactive sets, then $\bigcup_{A \in \mathcal{A}} T(A)$ is the spanning tree of $\bigcup_{A \in \mathcal{A}} A$;

3. For any two sets A_1 and A_2 that belong to the current partition, and any $e \in \delta(A_1) \cap \delta(A_2)$, there exists ℓ such that $c_e - \overline{c}_e \leq 2\ell$. If A_1 and A_2 are both active we have $c_e - \overline{c}_e = 2\ell$;

4. $cost(A) \leq 2\,LB(A)$ for every inactive set A;

5. $cost(A) = 2\,LB(A) - 4\ell$ for every active set A.

Regarding item (3), intuitively, the difference $c_e - \overline{c}_e$ corresponds to the "consumed" portion of an edge and thus ℓ gives the "radius" of the two balls on either side of the edge. The radius of a set stops increasing as soon as it becomes inactive. Hence, if the edge e connects two active sets, the consumed portion is equally apportioned to the two radii. Otherwise, it is not equally apportioned, with the active set having the greater apportionment. Items (4) and (5) hold trivially at the commencement of the algorithm.

We next discuss preservation of the invariants. There are two cases:

1. The status of an active set A_1 changes to inactive.

2. A new active set A is created consequent to the union $A_1 \cup A_2$

An increase in a coefficient at an iteration is equivalent to increasing ℓ by ρ and putting $c_e = 0$. Hence, corresponding respectively to the two cases above:

1. We increase $cost(A_1)$ by $2c_e \leq 2(\ell + \ell) = 4\ell$. Thus $cost(A_1) \leq LB(A_1) - 4\ell + 4\ell = LB(A_1)$;

2. Now, $cost(A) = cost(A_1) + cost(A_2) + 2c_e$, where we can argue that $c_e = 2\ell$.

Note that while primal feasibility is not guaranteed by our algorithm, dual feasibility is maintained at every step of its execution, since $\overline{c}_e \geq 0$ and $y_A \geq 0$ respectively for every edge and coefficient defined as in equations. When the algorithm terminates every ball has the depot on its perimeter and every node is connected to at least one other node with a tight edge. We thus have a spanning tree that consists of tight edges. Our primal solution uses each edge of the spanning tree twice, so we have an Euler tour. Since dual feasibility is always maintained, we thus have a factor-2 solution to the LP.

4.2.3 One demands

We next consider the case where all d_i are restricted to large values. The following changes are now required in the LP constraints of Section 4.2.2

$$x(\delta(S)) \geq 2|S| \text{ for each } S \subseteq N \backslash 0 \tag{4.2.10}$$

$$x_e \geq 0 \text{ for each } e \in E \tag{4.2.11}$$

The optimum solution now is a collection of trips that visit exactly one customer each. We can prove optimality of this solution via complementary slackness. Assume that the sequence $c_{01}, c_{02}, \ldots, c_{0n}$ is non-decreasing. Define $A(i) = \{i + 1, \ldots, n\}$. Consider the n constraints corresponding to the sets $A(i)$, $i = 0, \ldots, n - 1$ to be satisfied with equality in the primal solution.

The choice $y_{A_i} = c_{0,i+1} - c_{0i}$ for the dual variables corresponding to these sets and $y_{A_i} = 0$ for all other sets satisfies the corresponding dual constraints exactly, provides an objective function value $2\sum_i c_{0i}$, and thus satisfies complementary slackness. The dual is feasible as the L.H.S. in the other dual constraints (i.e., those that are not tight) is of the form $c_{0j} - c_{0i}$ whereas the R.H.S. is of the form c_{ij} where $i < j$. $c_{0j} - c_{0i} \leq c_{ij}$ holds by the triangle inequality.

Our solution thus has a path to the depot from each customer that consists of tight edges—each path in this case happens to consist of only one edge.

4.2.4 Zero-one demands

In this section we combine the ideas of the previous two subsections and discuss our solution procedure with every d_i allowed to take either a large, or a small value. We henceforth refer to the set of delivery points as L, with $|L| = k$, and to element i of this set as l_i. Our objective is to find k tours, each tour connecting the depot with exactly one $l_i \in L$, and with each visit point included in exactly one such tour. Our tours incorporate two paths between every l_i and the depot, a "to" path and a "from" path. A path between the depot and a given l_i is either direct (i.e., via the radial edge connecting the two), or, is via sets consisting of visit points. In turn, each such path may also include side-tours that take off from some point on the path, visit a set of visit points, and then return to the same point. Such paths are shortest paths between their endpoints as measured in residual lengths; to construct such paths, we proceed in two phases.

Phase 1 starts with singleton sets, as in the 0-demand case, with the sets corresponding to the depot and to 1-demand customers made inactive. In this phase, we perform unions of the sets and increase set coefficients. In Phase 2, we "undo" part of this activity, i.e., we reduce set coefficients in some of the dual constraints to obtain a sufficient number of shortest paths. Specifically, for every l_i, we attempt to find two paths to the depot consisting of tight edges, such that every set that has positive y_A lies on at most one such path. We also need the following property: If a set defined in Phase 1 does not lie on one of these paths (i.e., it is on a side-tour), then its coefficient in Phase 2 remains as established in Phase 1. In conjunction with the invariant relation between $cost(A)$ and

$LB(A)$ established in Section 4.2.2, this property implies that we have a factor-2 approximation, as shown below.

Consider first the following graph: Nodes are maximal sets among those that have positive y_A after Phase 2, and they lie on "to" and "from" paths, with the edges having residual lengths. Our collection of paths then forms an optimal solution to the 1-demand case.

In the above graph, let us now relax the assumption that the nodes are points; rather, the "node" corresponding to set A is the spanning tree $T(A)$ with $cost(A) + 2y_A \leq LB(A)$. We modify our path as follows.

- If $A = \{u\}$, we replace edges adjacent to A with edges adjacent to u, and increase the lengths of these edges by y_A. This step introduces cost and lower bound components of $2y_A$ each.
- If $A = A_1 \cup A_2$, we replace A with nodes A_1 and A_2, and the two edges adjacent to A with two edges adjacent to A_1 or A_2, depending upon the endpoints of the edges. We also increase the length of these edges by by y_A, and add edge (A_1, A_2) of residual length 0.

We next relax the assumption that an arbitrary set A must lie on a path. There are two possibilities:

1. A is a sibling of B (i.e., A and B have a common parent in the implicit binary tree of sets formed during Phase 1). Then B must have a point on a path. Inductively, there exist side-tours of this path that traverse the entire set B. We then have an edge from A to B of length at most $2y_A$. We add to our solution two copies of this edge as well as two copies of $T(A)$, and the cost of the added edges is within $2LB(A)$. Consequently, all the nodes in A are now visited by side-tour on the path through B.
2. A has no sibling. $2LB(A)$ then covers the cost of a tour that visits all the nodes in A and one node not in A. We add this tour as a side-tour, as in Case 1, above.

Each relaxation of our original assumptions results in an augmentation of our initial paths to tours that visit all the points, and the increase in cost corresponding to every such augmentation is covered by the corresponding component of the lower bound.

4.2.4.1 A network flow formulation for Phase 2

We note that Phase 1 ends with a collection of inactive sets. Henceforth, we use $\mathcal{A}$ to denote the set of maximal sets with positive coefficients in this collection and use A^j to denote the jth element of $\mathcal{A}$. We start by forming network $NW1$, that has the following nodes and edges:

1. Source node s, sink node t (the depot).

2. A node l_i for each $l_i \in L$ with an edge from s to l_i having capacity 2. The set of such edges will be called E_2.

3. For every $A^j \in \mathcal{A}$, we have nodes A^j_{in}, A^j_{out} and an edge $(A^j_{\text{in}}, A^j_{\text{out}})$ with capacity 1. The set of such edges will be called E_1.

4. Let S be the set of edges on shortest paths from s to t. For every edge $e \in S$ not described in (2) or (3), we create a network edge with unbounded capacity. If e starts in A^j, the new edge starts at A^j_{out} and if e ends in A^j, the new edge ends at A^j_{in}. If e neither starts nor ends in any A^j, the endpoints of the new edge are the same as for e. (Such an edge e connects a delivery point to the depot.) The set of such edges will be called E_∞.

We denote by V the set of nodes and by E the set of edges of network $NW1$ and use $z \in \mathbf{R}^E$ to denote a vector of flows in $NW1$. Given z, we denote by $f_z(v)$ the net flow (i.e., the excess of inflow over outflow) into node v. Stated as a max-flow problem, our problem then is to

$$\text{Maximize } f_z(t) \tag{4.2.12}$$

subject to:

$$f_z(v) = 0 \text{ for each } v \in V \backslash \{s, t\} \tag{4.2.13}$$

$$0 \leq z_e \leq i \text{ for each } e \in E_i \tag{4.2.14}$$

Finding a max-flow in $NW1$ is equivalent to finding a minimum cut $\mathcal{C}$ in $NW1$. (The capacity of cut $\mathcal{C}$ will be denoted by $|\mathcal{C}|$.) If $|\mathcal{C}| = 2k$, we have $2k$ paths with no two paths traversing the same edge $(A^i_{\text{in}}, A^i_{\text{out}})$, and with a pair of paths to t for each l_i. In other words, we have a collection of shortest paths with no two paths traversing the same set $A^i \in \mathcal{A}$. (Note that traversal of the *set* A^i, corresponds, in the flow formulation, to a flow through the *edge* $(A^i_{\text{in}}, A^i_{\text{out}})$.)

If $|\mathcal{C}| \neq 2k$ we iterate the following process. Suppose that $|\mathcal{C}| = 2k - u$, $u > 0$, with a edges from E_2 and $2k - 2a - u$ edges from E_1. Then, for each $(A^i_{\text{in}}, A^i_{\text{out}}) \in \mathcal{C} \cap E_1$ we reduce y_{A^i} by the maximum Δ such that either (a) some y_{A^i} is reduced to 0, at which stage we change $\mathcal{A}$, or (b) a new shortest path edge appears, at which stage we change E; in either case, we repeat the iteration with a new network.

The reduction in y_{A^i} results in each iteration in a decrease in the lower bound by $[4(k - a) - 2u]\Delta$. The residual distance to t does not decrease for any l_i, since no y_{A^i} is increased. For the $k - a$ delivery points l_i such that $(s, l_i) \notin \mathcal{C}$, each path to t must traverse at least one edge from $\mathcal{C} \cap E_1$, and hence a set A^i with decreased y_{A^i}. Consequently, $k - a$ residual distances increase by 2Δ, increasing the lower bound by $4(k - a)\Delta$. Hence, there is a net increase in the lower bound by $2u\Delta$. Since the lower bound can be increased only a finite number of times, we will eventually have a minimum cut of capacity $2k$.

However, if all the visit points are to be visited with these tours, sets that are not traversed by the shortest paths (i.e., that are on side-tours) must have sufficient LBs to "pay" for the respective side-tours. This implies the requirement that the coefficients of sets that are not traversed by a path are not reduced, which would entail adding to the max-flow formulation (4.2.12) through (4.2.14)

the constraint

$$z_e = 1 \text{ for each } e \in E_1^R \tag{4.2.15}$$

where E_1^R and E_1^{NR} respectively consist of edges $(A^i_{\text{in}}, A^i_{\text{out}})$ corresponding to sets A^i with reduced and non-reduced y_{A^i}, and where $E_1 = E_1^R \cup E_1^{NR}$.

4.2.4.2 Network Transformation

To see if we can satisfy this additional requirement, we consider a flow problem in a transformed network $NW2$. We create $NW2$ from $NW1$ by (a) introducing a new sink node t' and an edge (t, t') of capacity $2k$, denoted by E_{2k}, and (b) replacing each edge $(A^i_{\text{in}}, A^i_{\text{out}}) \in E_1^R$ with two edges (s, A^i_{out}) and (A^i_{in}, t'), each of unit capacity, the set of such edges being denoted by E_1^{R2}. $NW2$ thus has node-set $V \cup \{t'\}$, and edge-set $E' = (E \backslash E_1^R) \cup E_1^{R2} \cup E_{2k}$.

We use, for all nodes A^j with reduced y_{A^j}, $\mathcal{O}$ and $\mathcal{I}$ to denote respectively the set of nodes of the form A^j_{out} and A^j_{in}. The max-flow formulation (4.2.12) through (4.2.14) is applicable to $NW2$ with V', E' and t' replacing V, E and t respectively.

Consider next the following questions:

1. Does a flow of value $2|E_2|$, satisfying equation (4.2.15) exist in $NW1$? and,
2. Does a flow of value $2|E_2| + |E_1^{R2}|/2$ exist in $NW2$?

We now prove

LEMMA 4.2.1. *Questions 1 and 2 are equivalent, and a solution for $NW2$ can be transformed in polynomial time into a solution for $NW1$.*

*Proof.*Let $|E_2| = k$ and $|E_1^{R2}| = 2v$. The desired flows in both networks can be decomposed into paths (integer flows with value 1). Assume that the edges of E_1^R traversed by the s to t path P in $NW1$ are $(A^j_{\text{in}}, A^j_{\text{out}})$ for $j = 1, \ldots, i$, in that order. P is then the concatenation $P_0(A^1_{\text{in}}, A^1_{\text{out}})P_1 \ldots P_{i-1}(A^i_{\text{in}}, A^i_{\text{out}})P_i$, where P_{j-1} represents a sub-path connecting two nodes on the path (note that P_0 begins at s and P_i ends at t). The corresponding s to t' paths in $NW2$ are $P_0(A^1_{\text{in}}, t')$, $(s, A^j_{\text{out}})P_i$, and $(s, A_{\text{out}}{}^j)P_j(A^{j+1}_{\text{in}}, t')$, for $j = 1, \ldots, i-1$. Hence, a path in $NW1$

that utilizes i edges from E_1^R translates to $i+1$ paths in $NW2$. Consequently, a flow of $2k$ in $NW1$ is equivalent to a flow of $2k+v$ in $NW2$.

For the converse, we first note that since $\delta(s) = 2k+v$ and $\delta(t') = 2k+v$, a flow in $NW2$ with value $2k+v$ utilizes every edge (s,i) with $i \in \mathcal{O}$ and every edge (j,t') with $j \in \mathcal{I}$. Hence, this flow is composed of a unit flow in each of $2k+v$ paths $P_1, \ldots, P_{2k+v}$. Let us now collapse path P_i, $i \in \{1, \ldots, 2k+v\}$ to supernode $\overline{P}_i$ and define a graph $(\mathcal{P}, \mathcal{E})$ where $\mathcal{P} = \{\overline{P}_1, \ldots, \overline{P}_{2k+v}\}$, and $(\overline{P}_i, \overline{P}_j) \in \mathcal{E}$ if, for some k, (a) the last edge of P_i is (A_{in}^k, t'), and (b) the first edge of P_j is (s, A_{out}^k). We make the following observations about graph $(\mathcal{P}, \mathcal{E})$.

i. If $(\overline{P}_i, \overline{P}_k), (\overline{P}_j, \overline{P}_k) \in \mathcal{E}$, then $\overline{P}_i \neq \overline{P}_j$ (otherwise, we would need to have a flow of value 2 through some (A_{in}^u, t')).

ii. If $(\overline{P}_k, \overline{P}_i), (\overline{P}_k, \overline{P}_j) \in \mathcal{E}$, then $\overline{P}_i = \overline{P}_j$ (otherwise we would need to have a flow of value 2 through some (s, A_{out}^u). From (i) and (ii) it follows that $(\mathcal{P}, \mathcal{E})$ is a collection of node-disjoint paths.

iii. If $(\overline{P}_i, \overline{P}_j), (\overline{P}_j, \overline{P}_k) \in \mathcal{E}$, then P_j and P_k respectively start with (s, A_{out}^u) and (s, A_{out}^w) for some u and w, and the residual distance of A^u to t' is larger than that of A^w. This is because traversals through these sets in $NW1$ involve positive residual lengths and on a shortest path, consecutive reduced sets must have strictly decreasing distances to the depot. Hence, $(\mathcal{P}, \mathcal{E})$ contains no cycles.

iv. If path $P_i : \overline{P}_i \in \mathcal{P}$ starts with some edge (s, A_{out}^k) for $A_{\text{out}}^k \in \mathcal{O}$, then another path $P_j : \overline{P}_j \in \mathcal{P}$ ends with (A_{in}^k, t'), so $\overline{P}_i$ cannot start a path that is a component in $(\mathcal{P}, \mathcal{E})$. Similarly, the last path in such a component cannot end in (A_{in}^k, t').

Next consider a connected component of $(\mathcal{P}, \mathcal{E})$, a path $\overline{P}_1, \ldots, \overline{P}_u$. This path of paths can be converted to a single path in $NW1$ as follows. For $j = 1, \ldots u-1$, remove the last (A_{in}^i, t') from P_j and the first (s, A_{out}^i) from P_{j+1}, and replace these two edges with $(A_{\text{in}}^i, A_{\text{out}}^i, t)$. Consequently, we obtain $2k$ paths in $NW2$, eliminate all the new edges that are included in the paths as a result

of the transformation of $NW1$ into $NW2$, and use all v edges from E_1^R.

Finally, if we have a negative answer for Question 2 (posed immediately before Lemma 4.2.1), we can improve lower bounds and iterate phase 2, as shown below.

Because a flow with value $2k + v$ does not exist, we have $|\mathcal{C}| = 2k + v - u$, $u > 0$. Suppose that $\mathcal{C}$ contains a edges from s to L.

We change y_{A^i} using the following rules (if no rule applies, there will be no change):

(π_1) if $(A^i_{\text{in}}, t') \in \mathcal{C}$ and $(s, A^i_{\text{out}}) \in \mathcal{C}$, decrease y_{A^i} by $\Delta/2$;

(π_2) if $(A^i_{\text{in}}, A^i_{\text{out}}) \in \mathcal{C}$ (i.e., if A^i is a set with non-reduced coefficient and is in the cut), decrease y_{A^i} by $\Delta/2$;

(ω_s) if $(A^i_{\text{in}}, t') \in \mathcal{C}$ and $(s, A^i_{\text{out}}) \notin \mathcal{C}$, no change;

(ω_t) if $(A^i_{\text{in}}, t') \notin \mathcal{C}$ and $(s, A^i_{\text{out}}) \in \mathcal{C}$, no change;

(μ) if $(A^i_{\text{in}}, t') \notin \mathcal{C}$ and $(s, A^i_{\text{out}}) \notin \mathcal{C}$, increase y_{A^i} by $\Delta/2$,

For simplicity, in the following we use π to denote application of either rule π_1 or π_2. (In such cases, the conclusions are the same irrespective of whether π_1 or π_2 is applied). Similar considerations apply to the use of ω to denote either ω_s or ω_t. We distinguish two cases.

CASE(1): $(t, t') \notin \mathcal{C}$

LEMMA 4.2.2. *Consider a shortest path P from an arbitrary demand point $l_i \in L$ to the depot, according to residual lengths. Let $\eta_P(\alpha)$ be the number of sets on this path to which rule α was applied. Then*

i. $\eta_P(\pi) - \eta_P(\mu) \geq 0$;

ii. if $(s, l_i) \notin \mathcal{C}$ *then* $\eta_P(\pi) - \eta_P(\mu) > 0$.

Proof. We will refer to a set to which rule α was applied as an α-set. Figure 4.1 depicts pictorially the connections to the source and sink of various sets to which the rules stated above would be applied (edges of $\mathcal{C}$ are "cut" with thick gray lines).

Consider a path as in the statement of the lemma. Such a path has in general the following objects: a starting node from L, an end node t, and the sets α to which the rules, above, are applied after network transformation.

Considering NW2, we see that object $i \in \{\mu, \omega_s\}$ cannot be followed by object $j \in \{\mu, \omega_t, t\}$. Moreover, if $(s, l_i) \notin \mathcal{C}$, $l_i \in L$ cannot be followed by object $j \in \{\mu, \omega_t, t\}$ either.

This is because any violation of the above prohibitions would imply a path from s to t' in NW2 with an edge in $\mathcal{C}$, which is not possible, given that $\mathcal{C}$ is a cut. For example, if ω_s is followed by t', then for a certain ω_s-set A^i, in NW1 we have a path from A^i to t that does not pass through any objects to which our rules are applicable. In NW2 this maps into a path from A^i_{out} to t without any edges from $\mathcal{C}$; because A^i is an ω_s-set, in NW2 we also have edge $(s, A^i_{\text{out}}) \notin \mathcal{C}$. Together with edge (t, t') we obtain a path from s to t' without any edges from $\mathcal{C}$.

Figure 4.2 shows the possible sequences that we can encounter in terms of a flow diagram; our claim follows by an easy inspection.

Lemma 4.2.2 implies that:

- The residual distance from any $l_i \in L$ to t never decreases.
- If edge (s, l_i) is not in $\mathcal{C}$, this distance will increase by $\Delta/2$

Hence, the changes of the various components of the lower bound can be summarized as follows.

I. $k - a$ delivery points have residual distance to t increased by Δ, i.e., contribute to an increase in lower bound by $2(k - a)\Delta$.

II. Because $|E_1^R| = v$, we have $\eta(\pi_1) + \eta(\omega) + \eta(\mu) = v$.

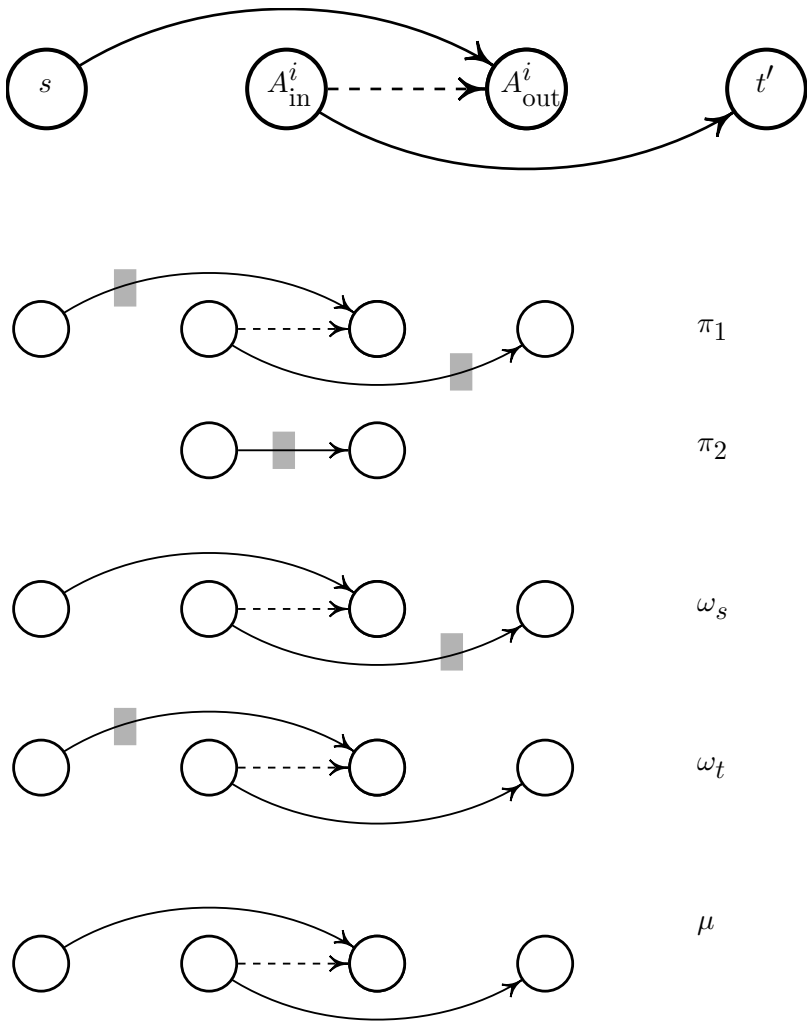

Fig. 4.1. Creating NW2 and cases of $\mathcal{C} \cap E_1'$

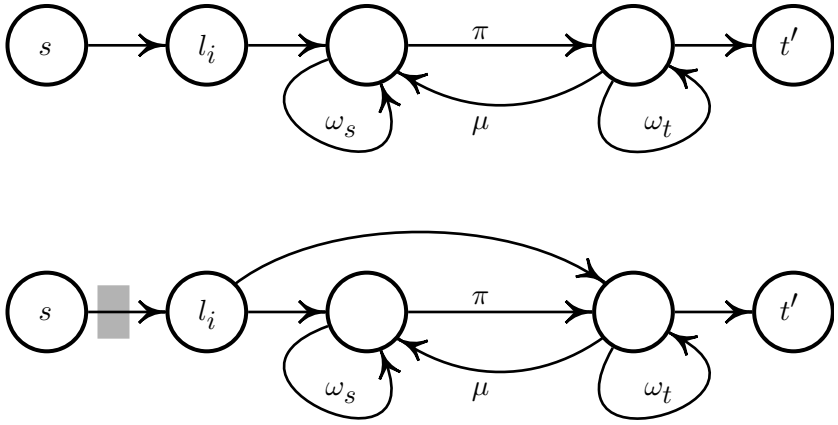

Fig. 4.2. Possible paths from L to t'

III. Respectively, the numbers of edges from $\mathcal{C} \cap E_1$ associated with each π_1-set, π_2-set and ω-set are 2, 1, and 1 (see figure 4.1). Hence, $2\eta(\pi_1) + \eta(\pi_2) + \eta(\omega) = 2(k-a) + v - u$.

IV. Subtracting the equation in II from that in III, we get $\eta(\pi) - \eta(\mu) = 2(k-a) - u$.

V. We recall that application of rule μ to set A_i corresponds to increasing y_{A_i} and of rule π, to decreasing y_{A_i}. Hence, from IV, the sum of y_A^is decreases by $[2(k-a) - u]\Delta$.

VI. From I and V, the total increase in lower bound is $u\Delta$.

CASE(2): $(t, t') \in \mathcal{C}$

1. We decrease the estimates of $\eta_P(\pi) - \eta_P(\mu)$ in Lemma 4.2.2 by 1, and consequently, the estimated increase in lower bound corresponding to (I) above by $2k\Delta$.

2. We decrease the estimate of the size of $C \cap E_1$ by $2k$, which decreases the estimate of $\eta(\pi) - \eta(\mu)$ by $2k$, and in turn, decreases the estimate corresponding to (III) above by $2k$.

3. Identical computations as in CASE (1) then give the same results.

Let us now consider computational complexity. At every iteration, either the lower bound is increased, or we get a solution within factor-2. In each iteration, we solve either one or two flow

problems in standard form, which have known polynomial time bounds. It remains to bound the number of increases in lower bound. We scale all distances to be integers and scale the initial lower bound to be n^2. Our solution can consist of at most n distances, and the number of iterations can be at most n^2. Hence, the error introduced by rounding is $\{n(LB)/n^2\}$. Hence, the relative error is $(n+1)/n$. Consequently, in polynomial time, our approximation ratio is $2(n+1)/n$

As Bertsimas [17, p. 575]) notes, there are instances of multi-vehicle VRPSD where "...the actual demand is known before the tour starts—customers call or the operator calls them or in the case of package deliveries the addresses are known...[in such cases] it is clearly better to use a re-optimization strategy...[but] practical considerations [might] make the re-optimization strategy less attractive (computing resources not available, it is time-consuming even if resources are available)." The polynomial computational time hence makes our procedure of this section a viable computational procedure for exactly such a problem.

With our additional assumptions, the applicability of our procedure is admittedly limited; however, the methodology appears promising, since to our knowledge, it leads to the first demonstration of an approximation ratio less than 2.5, hitherto unknown even for a special case. The extension of the procedure to the general multi-vehicle VRP (i.e., one where customers are allowed to have arbitrary demands) is thus a promising area of research.

4.3 VRP with stochastic customer demands, single vehicle

In this section, we modify OTPD to obtain performance guarantees for the expected and worst-case solution ratios to single-vehicle VRPSD, with a given customer invariably assumed to have non-zero demand. We also assume the availability of a function rand() that can generate a random integer uniformly in $\{1, Q\}$. Our modification to OTPD consists in randomizing the vehicle capacity on starting the route, as explained below.

4.3.1 Expected ratio

Let us denote by f the first customer on the τ-optimal TSP tour through N, assuming a clockwise orientation starting with the depot, and by e the last. We follow procedure *Stochastic Tour* (ST): 1. $remCap = rand()$. 2. Start with customer f and vehicle capacity $remCap$. 3. Follow the TSP tour until all nodes have been served, making trips to the depot in the event of route failure at any node i, returning to i, or to $i + 1$ as necessary.

We will show that an execution of ST is equivalent to performing an iteration of OTPD (recall that we perform Q iterations of OTPD), and that randomization ensures that these iterations are performed equiprobably. Consequently, in expectation, the same performance guarantee holds in following ST as in applying OTPD to the VRPSD with complete foreknowledge of customer demands. (In the deterministic version, the best of the Q OTPD solutions is chosen as the route to follow. However, the performance guarantee for the best solution is the same as that for the average of the Q solutions.)

To establish the equivalence between ST and OTPD, we consider a given realization of VRPSD. As in Section 4.1.3, OTPD would be applied to a network with (a) d_i copies of customer i and u copies of σ, with the vehicle traversing the m segments sequentially, and (b) σ inserted between the depot and e, so that $c_{0\sigma} = 0$, $c_{\sigma f} = c_{0f}$, and $c_{\sigma e} = c_{0e}$.

In considering the corresponding application of ST to this network, we must take into account the following two factors, occasioned by the stochastic setting of ST.

1. In Sections 4.1.2 and 4.1.3, we saw that insertion of a spurious customer is not required if $\nu = n$ or $\nu' = n'$ respectively, as equations (4.1.1) through (4.1.6)—with copies of a customer made in the arbitrary demands case—directly lead to the desired approximation ratio. However, in the stochastic case, it cannot be ascertained *a priori* whether the artifice would be required or not. Hence, we consider σ always present and requiring service, even when $u = d_\sigma = 0$. (This assumption does not violate any capacity restrictions.)

2. Provision Z of Section 4.1.2 cannot be invoked. (Since at least one σ must always be in the same segment as at least one of e or f, the distance $c_{0e} + c_{0f}$ must invariably be considered traversed, rather than c_{ef}.)

Let us denote by OTPS the version of OTPD that we obtain after incorporating these modifications. Regarding the second modification, the bounds of Sections 4.1.2 and 4.1.3 are independent of Z and hence OTPS has the same performance guarantee as OTPD. Regarding the first, we show below that it does not affect the performance guarantee of OTPS *vis-a-vis* OTPD. We now consider the cases where $d_\sigma = 0$ and $d_\sigma \neq 0$, in turn.

Assume a particular demand realization k, and let $remCap = q$. ST is equivalent to starting at the $Q - q + 1th$ customer (call it j) in the first segment of the original TSP tour with capacity Q under OTPS. This is because the edges forming part of the OTPS solution starting with j and those traversed under ST with $remCap = q$ would be identical from j through e and again from f onward to the final return to the depot. The only difference would be in the edges traversed between e and f. Under OTPS, these would add up to $c_{e\sigma} + c_{\sigma f}$, and under ST, to $c_{e\sigma} + c_{\sigma 0} + c_{0f}$. But these two distances are exactly the same!

The bounding procedures we have used are all based on computing the cumulative length (CL) of the ν' (or n') solutions and then using the average length of these solutions (which is the same if only the first Q solutions are considered, as we have seen) to bound the best solution. The CL of the Q OTPS solutions with the first starting with a visit to f and the Qth, with a visit to the Qth customer in the first segment, is $2\sum \ell_i + (Q-1)T^*(N)$. The reasoning is parallel to that of Haimovich et al. [46]: the first term reflects the fact that each customer is once first and once last on its route and the second term, that each edge in the TSP tour (i.e., a "tour" edge) is excluded once and therefore present $Q - 1$ times. If $d_\sigma \neq 0$, these observations are still applicable, CL is correctly computed, and their bounding procedure remains applicable to the average of the Q OTPS solutions.

If $d_\sigma = 0$, and we consider only Q solutions, however, the following deviations arise. (a) e is never the last visited, (b) σ is never the first visited, and (c) the tour edge (e, σ) is never excluded. To account for these deviations, and noting that if $d_\sigma = 0$, there are $Q + 1$ customers in the mth segment, the $Q + 1th$ being σ, we rewrite (4.1.1) as

$$T^*\left(X_m^0\right) \geq 2 \max_{i \in X_m} \{\ell_i\} \geq 2\frac{\sum\limits_{i \in X_j \backslash \{\sigma\}} \ell_i}{|X_m \backslash \{\sigma\}|} = (2/Q) \sum_{i \in X_m \backslash \{\sigma\}} \ell_i = (2/Q) \sum_{i \in X_m} \ell_i \qquad (4.3.1)$$

Consequently, the subsequent analysis of Section 4.1.1 (and its extension to customers with arbitrary demands), must also hold, and hence a ratio of $2.5 - [3/(2Q)]$ holds even when a spurious $Q + 1th$ customer is added. Next, considering the first of these Q solutions, since $c_{e\sigma} = c_{0e}$, we can take the *radial* edge c_{0e} as present in this solution, and the *tour* edge $c_{e\sigma}$ as excluded. Now, noting that $c_{\sigma 0} = 0$, we add $2c_{\sigma 0} = 0$ to the sum of the first Q solutions (the $Q + 1th$ solution would have σ immediately preceding f in a segment and thus be identical to the first solution) and find the CL to be $2\sum \ell_i + (Q-1)T^*(N)$, as required.

Since any value of $remCap$ corresponds to ST following one of the Q solutions, randomization ensures that all possible Q solutions *for realization* k under OTPS are implemented equiprobably under ST. If M realizations are possible in all, the expected distance of travel given realization i, $E[R_{VRP}|i]$, where $i \in \{1, \ldots, M\}$, would thus be the same as the average of the Q OTPS solutions, given i, and would therefore be bounded above by 2.5. Since this would hold for each i, it follows from linearity of expectation that $E[R_{VRP}]$, the expected distance traveled over all $i \in \{1, \ldots, M\}$ would also adhere to this upper bound.

4.3.2 Worst-case ratio

As we saw in Section 4.3.1, given a VRPSD realization, ST chooses one of the Q solutions that OTPS would generate. In this section, we attempt to find the worst-case solution ratio

obtainable in following ST, compared to the best OTPS solution. We consider an arbitrary VRPSD realization that requires m vehicle routes for complete fulfillment and we:

1. Denote by $\ell^*_{k,l}$ and $\ell^*_{k,t}$ respectively the leading and trailing radii (i.e. the radii used for going to the customer and to the depot respectively) utilized in the kth route by the best OTPS solution for the given realization, where $k \in \{1..., m\}$.

2. Denote by $\ell_{k,l}$ and $\ell_{k,t}$ the corresponding radii utilized by an arbitrary OTPS solution.

3. Number customers 1 through m', the latter defined as in (4.1.9).

4. Denote by (k, bt) and (k, wt) respectively the customers at which the kth return to the depot occurs for the best and the arbitrary tours respectively.

5. Similarly denote by (k, bl) and (k, wl) the corresponding customers at which the vehicle starts the kth route.

6. Arbitrarily assume that $(1, bt)$ is numbered lower than $(1, wt)$. (The derivation of the approximation ratio in the contrary case is similar to our derivation below.)

By the triangle inequality, we then have

$$\ell^*_{k,t} - \ell_{k,l} \leq \sum_{u=(k,wl)}^{u=(k,bt)-1} c_{u,u+1} \tag{4.3.2}$$

and, similarly,

$$\ell^*_{k+1,l} - \ell_{k,t} \leq \sum_{u=(k+1,bl)}^{u=(k,wt)-1} c_{u,u+1} \tag{4.3.3}$$

with the understanding that if $k = m$, $k + 1 = 1$. Summing up the L.H.S. and R.H.S. of the above two equations over all k, we get $\sum_{k=1}^{m} \left(\ell^*_{k,l} + \ell^*_{k,t}\right) - \sum_{k=1}^{m} \left(\ell_{k,l} + \ell_{k,t}\right) \leq T^* \left(X'\right) = T^* (N)$, where the R.H.S. follows because the the inter-node distances that are summed up can add up at most to the entire tour. With this bound on the maximum difference between the best and the worst

routes produced by OTPS, we obtain, following the procedure of the derivation of (4.1.6), and with $(2n/Q)\,\overline{\ell} \geq T^*(N)$,

$$\begin{aligned} R^H(\cdot)/R^*(\cdot) &\leq \left(2l\overline{\ell} + (2-(l/n))(1+\tau)(2n/q)\,\overline{\ell}\right)/(2n/q)\,\overline{\ell} \\ &= (l + (2-(l/n))(1+\tau)(n/q))/(n/q) \\ &= 1 + 2(1+\tau) - (1/q)(1+\tau) \end{aligned} \tag{4.3.4}$$

Since ST cannot lead to a traversal of distance greater than that traversed by the arbitrary OTPS solution, this gives us a bound of 4. (Again, a similar proof is applicable to the other case.)

We note that our bounds appear to be improvements on the expected ratio of $2.5 + O(1/n)$ (for the case of i.i.d. demands) and $Q+1$ (for the case of non-i.i.d. demands) obtained by Bertsimas [17] for the *CH*, his adaptation of OTP to the stochastic case, with the same assumptions about d_i as ours. (*CH*, like OTP, starts with a "good" initial TSP tour on $N\backslash 0$ and determines the best of the n tours that result from starting the tour in turn with each $i \in N\backslash 0$, comparing the different tours on the basis of their *expected* lengths. *CH*, however, considers only the single-vehicle VRPSD). Moreover, with the assumption that $K \leq Q$, CH requires $O(n^3K^2)$ time, and this bound was improved later to $O(n^2KQ)$ by Gendreau et al. [24]. Our procedure takes $O(nK)$ or $O(nQ)$ as in Section 4.1.3; or, $O(n \log n)$ as in Section 4.1.4.

With the insights obtained from our study, in this chapter, of heuristic techniques with guarantees for VRP and VRPSD, we now proceed to apply the insights from this chapter and the previous one in the construction of improved solutions to VRPSD. Our solution procedures aim to retain the Markovian framework while utilizing approximate techniques that would be computationally viable. This is the focus of the next chapter.

Chapter 5

Dynamic Policy, Single Vehicle

While procedure ST of Chapter 4 provides better bounds than CH, in this chapter our focus is dynamic improvement. Since Secomandi [33], however, uses CH as a base for applying the *rollout* technique (described in Chapters 1 and 2) to VRPSD, for better comparability we retain CH as the base for evaluating dynamic improvement. Section 5.1, accordingly, provides a description of CH and *rollout* as applied to it. In section 5.2, we discuss our proposed modifications to this procedure. Section 5.3 details the methodology adopted in the subsequent computational experiments, while section 5.4 analyzes their results. Finally, Section 5.5 combines the *rollout* philosophy with an exact MDP approach discussed in Chapter 3.

5.1 Base Heuristic and the Rollout Procedure

CH starts with the first customer on the best tour found by it and follows the tour, performing return trips to the depot when route failure occurs. A particular set of demand realizations would lead to a particular path (called "trajectory" by Secomandi [33]) being followed in serving the customers corresponding to any such set. The expectation over all such paths gives us the expected length traversed. Considering now the vehicle on arrival at customer i, and before the demand at i is observed, the maximum possible number of trajectories is K^{i-1}. Each such trajectory would result in a particular value of q (including replenishments to vehicle capacity that were necessitated) on arrival at i. The computational requirement thus seems exponential; Bertsimas [17]'s recursive algorithm, however, computes the probabilities of occurrence of the different q in polynomial time, and thus enables the expected length of the entire tour to be computed in

$O(n^2K^2)$ time. Bertsimas et al. [26] further enhance this procedure with preventive anticipatory trips to the depot when vehicle capacity is "close" to exhaustion.

By modifying Bertsimas' computational scheme, Gendreau et al. [24] obtain the better bound of $O(nKQ)$. This bound is better than $O(n^2K^2)$ because in non-trivial cases, $nK > Q$. (If $nK \leq Q$, the problem reduces to a TSP.) Secomandi [33] essentially uses a variant of the latter algorithm, conditioning on the value of remaining vehicle capacity *after* service completion rather than before, as the earlier researchers had done. We denote by q_i this remaining capacity at i and henceforth in this work use the notation $p_{ij} = \Pr\{D_i = j\}$. Secomandi [33] computes the expected length of the remaining tour commencing at $i+1$ (the customers assumed serially numbered according to their positions on the route), denoted by $E\left[L_{i+1}\right]$, assuming that $i+1$ is visited with (a) q_i, i.e., directly (option 1) and (b) Q, if a preventive trip is made to the depot at i (option 0), choosing the lower-cost option. Thus,

$$E\left[L_i\right] = \gamma_i(q_i) \tag{5.1.1}$$

where $\gamma_n(q_n) = c_{on}$, $\gamma_i(q_i) = \min\{\gamma_i^0(q_i), \gamma_i^1(q_i)\}$, and

$$\gamma_i^1(q_i) = c_{i,i+1} + \sum_{j=0}^{q_i} \gamma_{i+1}\left(q_i - j\right) p_{i+1,j} + \sum_{j=q_i+1}^{d_{i+1}^{\max}} \Big[2c_{0,i+1} + \gamma_{i+1}\left(Q + q_i - j\right)\Big] p_{i+1,j} \tag{5.1.2}$$

$$\gamma_i^0(q_i) = c_{0,i} + c_{0,i+1} + \sum_{j=0}^{d_{i+1}^{\max}} \gamma_{i+1}\left(Q - j\right) p_{i+1,j} \tag{5.1.3}$$

When the last customer n has been served, the remaining length is clearly c_{on}; otherwise, recursively at each customer i, the algorithm chooses between going to $i+1$ via options 0 and 1. $\gamma_\cdot^0(\cdot)$ thus represents the cost associated with the former action, $\gamma_\cdot^1(\cdot)$ that associated with the latter, and $\gamma_u(q)$ is the expected cost to go, customer u onward, if the capacity remaining after serving u fully is q. Since, moreover, $d_\cdot^{\max} \leq K$, $q_i \leq Q$ and there are no more than n customers, the running time is $O(nKQ)$.

CH, however, chooses only the *starting* customer; for the rest, it follows the TSP tour. What if we choose the others afresh, too? This is precisely what *rollout* does. Specifically, in *static rollout* (*SR*), at i, for each value of q_i, we compute the cost of going to $j \in Y_i$, where Y_i denotes the set of unvisited customers at i, via the two options, choosing the lower-cost option. Summing up over all q_i (recall from above that the probabilities of the different q_i occurring are known now) gives us the expected cost of going to j. Minimization over all j's gives the best choice for $i+1$, and thus we change the customers positioned at the second, third...$(n-1)$th places on the tour, fixing one incumbent in each iteration. If, instead of considering the probabilities of the different q_i occurring, we operate with *only one* value of q_i, the capacity that actually remains after serving i in a dynamic procedure (as in an actual vehicle run), we would have *dynamic rollout* (*DR*). To be precise, the $(i+1)$th customer in the latter is determined according to the minimization: $i+1 = \arg\min\Big\{\min\big([V(1)],[V(0)]\big)\Big\}$ where

$$[V\,(1)] = \min_{j \in Y_i} \left\{ c_{ij} + \sum_{k=0}^{q_i} p_{j,k} E\left[L_j | q_j = q_i - k\right] + \sum_{q_i+1}^{d_j^{\max}} p_{j,k}\left[2c_{0j} + E\left[L_j | q_j = q_i + Q - k\right]\right] \right\} \tag{5.1.4}$$

$$[V\,(0)] = \min_{j \in Y_i} \left\{ c_{0i} + c_{0j} + \sum_{k=0}^{d_j^{\max}} p_{j,k} E\left[L_j | q_j = Q - k\right] \right\} \tag{5.1.5}$$

where 1. $E[L_j|\cdot]$ is the expected length of the tour that starts at j and thereafter follows the TSP ordering, skipping customers with satisfied demands (such a tour is called a *cyclic permutation* of the original) and 2. $V(\cdot)$ is the best expected cost to go of the remaining route after service completion at i under the two options. For *static rollout* (*SR*), the cost corresponding to a visit to any $j \in Y_i$ would be given by the weighted average of the costs corresponding to each $q_i \in \{0, Q\}$, the weighting factor being the probability of occurrence of each such q_i. Since the cost to go is approximated by $E[\cdot]$, and less than n^2 tours are evaluated in all, the use of approximations thus enables a total running time of $O(n^3KQ)$.

In Secomandi's experiments (that we designate SE), the results obtained by applying DR to the solutions provided by (a) CH and (b) SR are compared with those obtained by using the unvarnished versions of the latter two. The most stringent comparison for DR would clearly be with SR, and Secomandi [33]'s results (Table 5.1), bear out this assertion.

Table 5.1.
Performance of DR *vs.* CH and SR, SE

	expected distance		percentage improvement		
$n, Q, \bar{f}'$	CH	SR	SR applied to CH	DR appl. to CH	DR appl. to SR
(20, 91, 0.75)	5.33	5.29	0.75	3.81	0.94
(20, 71, 1.25)	5.83	5.81	0.34	2.13	0.73
(20, 58, 1.75)	6.62	6.62	0	1.74	0.93
(40, 183, 0.75)	7.28	7.16	1.65	1.91	1.27
(40, 142, 1.25)	8.00	7.93	0.88	4.18	0.24
(40, 116, 1.75)	8.82	8.80	0.23	3.48	1.42
(60, 274, 0.75)	8.55	8.38	1.99	1.75	1.85
(60, 213, 1.25)	9.32	9.20	1.29	2.94	1.38
(60, 175, 1.75)	10.01	9.89	1.2	3.50	1.58
Average	7.75	7.67	1.03	2.90	1.24

5.2 Proposed modifications to the Procedure

Our main concern, however, is to incorporate *dynamism* into our policies and by this token, the *really valid* comparison of DR would also be with SR, since the latter is, after all, also an *a priori* solution. The above results therefore serve as our motivation to improve the dynamic procedure further.

For the initial TSP tour, *CH* applies the Nearest Neighbor (*NN*) heuristic augmented with the 2-*opt* tour improvement procedure [41] to the set $N\backslash 0$. (A technical clarification—Secomandi [33] uses the term 2-*int* whereas Bertsimas et al. [26] use 2-*opt*. The process wherein we consider in turn each non-adjacent pair of edges in an existing tour, delete these edges and reconnect the two resulting paths-there is a unique way to do this-to obtain a tour of lower cost, is called 2-*interchange* or 2-*int*. If no such deletion/recombination can result in an improved tour, the tour is called 2-*opt*. We assume that 2-*int* is carried out until a 2-*opt* tour is available.)

The neighbors of any given node are thus essentially chosen on the basis of proximity. This TSP ordering is also retained during the course of *rollout*. During the course of *rollout*, however, once a node has been fixed at some position and thus eliminated from the set of the remaining nodes, the set of available neighbors of these nodes also changes. Consequently, at a given iteration of *rollout*, the current ordering of the remaining nodes might not remain the best possible ordering that NN+2-*opt* would determine for them. Obtaining a fresh TSP tour on the remaining customers at every iteration of *rollout* therefore offers the possibility of improved solutions.

Our "reoptimize the TSP tour" routine is the Lin-Kernighan (*LK*) procedure of Lin et al. [44]. Empirical evidence adduced by Cook et al. [43] suggests that it can give results even better than 2-*opt*. In brief, while in 2-*opt*, in any given iteration, one endpoint of the edges being exchanged remains fixed, in *LK* an edge exchange is not decided on the basis of immediate benefit but rather, the incoming node can serve as a fresh starting point for further exchanges. As Reinelt [40, p. 124] describes *LK*: "...sometimes a modification slightly increasing the tour length can open up new possibilities for achieving considerable improvement afterwards. The basic principle is to build complicated tour modifications that are composed of simple moves where not all of these moves necessarily have to decrease the tour length." It is thus an r-*opt* procedure where r is a variable and the terminating criteria are designed differently.

In what follows, we use *BH* to denote an arbitrary base heuristic, *RH* to denote the heuristic that applies *rollout* to *BH*, and *RR* to denote our proposed *reoptimization with rollout* procedure,

that can be stated as follows.

Go to the first customer as decided by *SR* or *CH*. (The first customer is same in both).

Let T denote the current sequence of remaining customers.

for (every customer $i \in \{1, \ldots, n-1\}$)

Choose $i+1$ according to equations 5.1.4 and 5.1.5.

Let j be the best choice for $i+1$, with associated expected cost x. (A)

if $(Q \geq \sum_{u \in Y_i} d_u^{\max})$ $(B1)$

Run LK on the set $Y_i \cup \{0\}$ and let y be the value returned by LK be y.

Let the value returned by LK be y.

else $(B2)$

Run LK on the set Y_i and let T' be the TSP tour returned.

Apply CH to T' to determine the best starting customer.

Let this customer be k with associated expected cost y.

if $(c_{0i} + y < x)$

if $(B2$ holds)

Replace T by cyclic permutation of T' starting with k, follow sequence i-depot-k.

else

Add $(c_{0i} + y)$ to the total distance traveled so far and end program.

else

Replace T by sequence starting with the next unserviced customer after j.

Follow sequence i-depot-j or sequence i-j as determined in A.

for loop ends.

The properties of the procedures we use are summarized below.

PROPOSITION 5.2.1. *The expected distance traveled in following the sequence generated by any of the rollout procedures as applied to* BH *can be no more than that traveled in following the route provided by* BH *itself.*

Proof. Define $E_1\left[L_i^{\cdot}|(\cdot)\right]$ and $E_2\left[L_i^{\cdot}|(\cdot)\right]$ as the expected distance traveled up to and including service of customer i and the distance traveled subsequent to this, respectively; and $E\left[L^{\cdot}\right]$ as the total expected distance, where the superscript on L denotes the particular heuristic used. By linearity of expectation we have $E\left[L^{\cdot}\right] = E_1\left[L_i^{\cdot}|(\cdot)\right] + E_2\left[L_i^{\cdot}|(\cdot)\right]$. So long as DR follows the same sequence as BH, the expected costs remain same. If DR's choice of i is different from that of BH, from (5.1.4) and (5.1.5) it follows that $E_2\left[L_{i-1}^{DR}|(.)\right] < E_2\left[L_{i-1}^{BH}|(.)\right]$ and hence that $E\left[L^{DR}\right] < E\left[L^{BH}\right]$. Now DR will follow the new sequence (starting with the new incumbent at $i+1$) and this inequality will continue to hold as long as there is no further modification, since BH follows a fixed sequence and $E[L^{BH}]$ is thus a constant. Any modification by DR at a subsequent iteration can only further decrease $E_2\left[L_{\cdot}^{DR}|(\cdot)\right]$ and in turn $E[L^{DR}(\cdot)]$ compared to this new sequence, so the inequality will still hold. Identical arguments hold if SR is substituted for DR

Next consider RR. If it finds a better choice for i than DR, we have $E_2\left[L_{i-1}^{RR}|(\cdot)\right] < E_2\left[L_{i-1}^{DR}|(\cdot)\right] \leq E_2\left[L_{i-1}^{BH}|(\cdot)\right]$ and hence $E\left[L^{RR}\right] < E\left[L^{DR}\right] \leq E\left[L^{BH}\right]$. Any further modification introduced by RR can only reduce the expected distance of its associated route vis-a-vis BH's route.

PROPOSITION 5.2.2. *Discounting the running time of* LK, RR *generates a sequence of nodes corresponding to a trajectory in* $O(n^3KQ)$.

Proof. We have explained above the bound of $O(n^3KQ)$ for DR as obtained by Secomandi. The additional work done in RR is applying CH at every customer and this adds no more than $O(n^3KQ)$ to the running time.

LEMMA 5.2.1. *Decisions 0 and 1 follow a threshold structure in remaining capacity.*

Proof. Proved earlier by Bertsimas and Secomandi, and also by us in Chapter 3. *RR* does not change the structure—it merely substitutes one *a-priori* sequence for another.

We note that Proposition 5.2.1 holds with *CH* substituted for *BH* and *SR* the rollout procedure. This bears out our earlier observation that the most stringent comparison for the dynamic policies would be with *SR* functioning as *BH*.

A result similar to Proposition 5.2.1 is proved by Secomandi [32, 33] for *rollout*. His proof, however, is based on the properties of *rollout* algorithms derived by Bertsekas et al. [38], while ours proceeds directly from first principles. Also, the reason for not including the running time of *LK* in Proposition 5.2.2 is elaborated upon in Section 5.4. Finally, the performances of *RR* and *DR* cannot be directly compared in theory, since they would (in general) be operating on different base sequences. This leads us to empirical analysis, the subject of the next section.

5.3 Computational Experiments-methodology

All of our terminology and several of our experimental parameters are as in Section 3.7; the reader may like to glance at that section at this point. As in Chapter 3, we compare the expected distances traversed under each policy on a number of problem instances, but now, to assess these expected distances, we use simulation (as Secomandi [33, p. 800] aptly observes: "exact computation of the expected length of the *rollout* policy from the initial to the final state becomes computationally challenging...[so] simulation is employed..."). For the simulation experiments of this chapter, therefore, following Secomandi, while problem instances are still differentiated by n and $\bar{f}'$, a problem instance is defined differently.

Specifically, a problem instance, now referred to as a *replication*, corresponds to a particular set of customer locations, each customer with an associated demand distribution, whereas an iteration corresponds to a particular set of demand realizations at the customers, also randomly generated on the basis of the associated demand distributions at the different customers. The

customer demand distributions are $U(\underline{d}, \overline{d})$, where $(\underline{d}, \overline{d}) \in \{(1,5),(6,10),(11,15)\}$, with each distribution having an equal probability of being selected for any given customer. The experimental region is the unit square.

We subject each $(n, Q, \bar{f}')$ combination to 10 replications and estimate the expected distances by carrying out 200 *iterations* for each replication. For the number of iterations used, we follow the convention established by Bertsimas et al. [26, p. 347] in determining the expected route length under *CH*, where they justify their choice with the observation that "...[number of iterations] no more than 200 was necessary; increases to 500 had essentially no effect on sample means". (Secomandi uses 5 replications and 200 iterations per replication.)

We next discuss our choice of $\bar{f}'$. Secomandi [33]'s choices are $\bar{f}' \in \{0.75, 1.25, 1.75\}$. However, Bertsimas et al. [26]'s experiments consider a much wider range of $\bar{f}'$ (though the authors do not specifically mention this). For instance, problem numbers 10, 12, 14 and 15 in Table 2 of their work have $n \in \{40, 50, 50, 50\}$. The corresponding demand distributions are $U(0, 24)$, $U(1, 10)$, $N(12, 4)$, $U(0, 24)$; the vehicle capacity is 50 in each case. A simple calculation gives $\bar{f}'$ as 8.6, 4.0, 11 and 11 respectively. (For details, please see the original.)

Apart from the example set by Bertsimas et al. [26], our choices for $\bar{f}'$ also incorporate the following considerations. For reasons of capital costs and ease of maneuverability (increasingly congested roads make the latter an important factor), companies may prefer smaller trucks. As Swan [58, p. 56] observes: "Regional LTL companies may use trailers which are smaller than the longest allowed...[which is] economical for...[several] reasons. First, all equipment is used for both over-the-road and pickup and delivery, so standardization is a requirement. Second, the shorter trailer is easier to use in pickup and delivery because it can be maneuvered easier on city streets...." We therefore feel a larger range of $\bar{f}'$ would be reasonable and thus our choices are 0.75 to 2.75, in increments of 0.5. Also, in keeping with increased computing capability, we include $n = 80$ and $n = 100$ in addition to Secomandi [33]'s choices of $n \in \{20, 40, 60\}$.

Regarding the experimental region, Bertsimas et al. [26] and Secomandi [33]'s choice is $[0, 1]^2$, with Secomandi's depot located at $(0, 0)$. Gendreau et al. [24] choose depot location randomly.

Bertsimas et al. do not locate the depot randomly, but neither do they specifically mention the location; from the context, it appears to be $(0, 0)$. Since a company usually has some choices in establishing a depot, we assume a fixed depot location at $(0, 0)$ and retain the unit square as our region, but construct the unit square somewhat differently.

Specifically, we perform two sets of experiments, $DA1$ and $DA2$. Respectively, these have the depot at the center and the bottom left corner of the unit square. The unit square in $DA2$ is thus $[0, 1]^2$, whereas that in $DA1$ has vertices as in Chapter 3. Our choices are based on the following observations.

The ability of *rollout* to provide lower-cost routes depends on reordering the customers to change the probabilities that different *tour edges*—inter-customer distances—and *radial edges*—edges connecting the depot to a customer—are used. Since there are $O(n^2)$ tour edges and only $O(n)$ radial edges, the average difference between the tour edge lengths would tend to be less than that between radial edge lengths, and hence radial edges would have a greater impact on expected distances. Since the variation in radial edge lengths would be less on average in $DA1$ as compared to those in $DA2$, chances of improvement must accordingly be lower.

In a sense then, a centrally located depot represents a more severe test for the different improvement schemes considered above than the "depot at a corner", and our two sets of experiments should serve to verify this intuition empirically. As well, we should be able to obtain an idea of the impact of depot positioning on the performance of our policy (since we would be considering the two extreme cases).

The algorithms were implemented in Microsoft Visual C++ 6.0 on a Pentium Intel x86 1GHz PC with 325 MB RAM. (We do not report the running times of CH and SR since these are static policies and thus not relevant to re-optimization on run.) We compare SR with CH, and compare the results obtained by applying DR and RR respectively to SR, the best available *a priori* solution. Thus, for DR and RR, SR serves as the BH (once a solution has been obtained via SR, it becomes a *fixed* sequence).

The columns $CH(LK)$ in Tables 5.2 and 5.3 detail the expected distances traveled in $DA1$ and $DA2$ respectively, in following the sequence generated by CH with the starting solution obtained *via* $NN + LK$. Our objective in obtaining these results is to check whether these are significantly lower than those obtained by initializing CH with a TSP solution obtained *via* NN+2-*opt*; i.e., whether a better starting solution is a more plausible explanatory factor for the improvements we expect to obtain by using RR, than re-optimization *per se*. The figures in each cell in Tables 5.2 and 5.3 represent the averages over the 10 replications performed for that particular $(n, Q, \vec{f}')$ combination.

5.4 Analysis of results

We observe that

- While all all versions of *rollout* provide improvement, RR in particular does better than DR on *every* $(n, Q, \vec{f}')$ combination in both experiments, and on an average, more than doubles the percentage improvement *vis-a-vis* SR.

- The computation times of RR (59 seconds for the largest instance-i.e. $n = 100$ and $Q = 457$) appear manageable and are on the same order (corresponding time: 34 seconds) as those of DR. Thus, while SR should always be useful in generating an initial solution, from a *dynamic* perspective, DR might profitably be replaced by an RR-type improvement scheme that considered more options.

- The increased values of all the three averages (last row, column "% improvement") in Table 5.3 as compared to the corresponding entries in Table 5.2 validate our intuition about the "central" and "corner-located" depots, and as well, demonstrate the impact that depot positioning can have on the degree of improvement dynamic techniques can bring about.

- As Arora [57, p. 554] observes: "Versions of the Lin-Kernighan run-after careful tuning-in nearly *linear* time (author's emphasis) and often produce tours whose cost is within a few

Table 5.2.
Performance of *RR-DA*1

	expected distance					% improvement			*seconds/iter*	
	base:CH			base:SR		vs. CH	vs. SR			
$(n, Q, \bar{f}')$	$CH(LK)$	CH	SR	DR	RR	SR	DR	RR	DR	RR
(20, 91, 0.75)	4.4698	4.3743	4.3422	4.3406	4.3389	0.73	0.04	0.08	< 0.1	0.1
(20, 71, 1.25)	4.8551	4.7882	4.7468	4.7133	4.6894	0.87	0.70	1.21	< 0.1	0.1
(20, 58, 1.75)	5.1626	5.0896	5.0308	5.0026	4.9771	1.16	0.56	1.07	< 0.1	0.1
(20, 49, 2.25)	5.5295	5.4815	5.3946	5.3670	5.3412	1.58	0.51	0.99	< 0.1	< 0.1
(20, 43, 2.75)	5.8825	5.8557	5.7680	5.7296	5.7083	1.50	0.66	1.03	< 0.1	< 0.1
(40, 183, 0.75)	5.7595	5.7424	5.6358	5.6261	5.6141	1.86	0.17	0.38	0.7	1.3
(40, 142, 1.25)	6.0798	6.1022	5.9957	5.9634	5.9341	1.74	0.54	1.03	0.6	1.1
(40, 116, 1.75)	6.4210	6.4685	6.3588	6.3188	6.2797	1.70	0.63	1.24	0.5	0.9
(40, 98, 2.25)	6.8278	6.7770	6.6763	6.6344	6.5963	1.49	0.63	1.20	0.4	0.7
(40, 85, 2.75)	7.1178	7.0765	7.0013	6.9725	6.9165	1.06	0.41	1.21	0.3	0.6
(60, 274, 0.75)	7.0703	7.0264	6.9707	6.9576	6.9462	0.79	0.19	0.35	4.0	6.9
(60, 213, 1.25)	7.4824	7.3997	7.3023	7.2914	7.2276	1.32	0.15	1.02	3.0	5.4
(60, 175, 1.75)	7.7398	7.7088	7.6721	7.6406	7.5764	0.48	0.41	1.25	2.3	4.5
(60, 148, 2.25)	8.0408	7.9772	7.8186	7.7712	7.7458	1.99	0.61	0.93	1.9	3.8
(60, 128, 2.75)	8.4918	8.3467	8.1973	8.1353	8.0990	1.79	0.76	1.20	1.7	3.3
(80, 366, 0.75)	8.0317	7.9703	7.9057	7.8974	7.8689	0.81	0.10	0.47	13	22.4
(80, 284, 1.25)	8.4230	8.3517	8.2311	8.2076	8.1547	1.44	0.29	0.93	9.9	17.5
(80, 233, 1.75)	8.7371	8.6476	8.5393	8.5114	8.4360	1.25	0.33	1.21	7.8	14.4
(80, 197, 2.25)	8.9451	8.9288	8.8435	8.7651	8.7322	0.96	0.89	1.26	6.6	12.2
(80, 171, 2.75)	9.2208	9.2505	9.1059	9.0350	8.9557	1.56	0.78	1.65	5.7	10.5
(100, 457, 0.75)	8.8574	8.8839	8.8417	8.8412	8.8001	0.48	0.01	0.47	34.5	59.0
(100, 356, 1.25)	9.1704	9.1397	9.0366	8.9822	8.9333	1.13	0.60	1.14	26.0	45.8
(100, 299, 1.75)	9.4791	9.4586	9.3973	9.3704	9.2431	0.65	0.29	1.64	20.8	38.9
(100, 246, 2.25)	9.8636	9.8989	9.7643	9.7177	9.5615	1.36	0.48	2.08	17.0	31.3
(100, 213, 2.75)	10.1992	10.1230	10.0165	9.9899	9.8585	1.05	0.27	1.58	14.9	26.9
Average	7.5143	7.4747				1.23	0.44	1.06		

percent of the optimum." The first part of this observation (the *time* aspect) explains why the running time of RR is never more than twice that of DR. Essentially, it is the contribution from running CH at every vehicle halt ($O(n^2KQ)$ at each halt and thus $O(n^3KQ)$ overall) that seems to account for the increased running time, so that LK can be subsumed under this time ($B2$ in the algorithm of section 5.2 indicates the use of CH).

- The maximum improvement brought about to CH by the cumulative effects of static and dynamic policies is roughly 2.9(=1.52+1.36)%. This figure does not seem very high; we must,

Table 5.3.
Performance of *RR-DA*2

	expected distance					% improvement			*seconds/iter*	
	base:CH			base:SR		vs. CH	vs. SR			
$(n, Q, \bar{f}')$	$CH(LK)$	CH	SR	DR	RR	SR	DR	RR	DR	RR
(20, 91, 0.75)	5.9415	5.9039	5.7731	5.7398	5.7014	2.21	0.58	1.24	< 0.1	0.1
(20, 71, 1.25)	6.7817	6.6866	6.5973	6.5454	6.5170	1.34	0.79	1.22	< 0.1	0.1
(20, 58, 1.75)	7.6242	7.5571	7.4746	7.4233	7.3860	1.09	0.69	1.19	< 0.1	0.1
(20, 49, 2.25)	8.4724	8.5093	8.3845	8.3317	8.2946	1.47	0.63	1.07	< 0.1	< 0.1
(20, 43, 2.75)	9.1753	9.1426	9.0679	9.0106	8.9960	0.82	0.63	0.79	< 0.1	< 0.1
(40, 183, 0.75)	7.1492	7.1162	6.8826	6.8634	6.8039	3.28	0.28	1.14	0.7	1.3
(40, 142, 1.25)	8.0273	8.0086	7.8375	7.7887	7.7031	2.14	0.62	1.71	0.6	1.1
(40, 116, 1.75)	8.8119	8.6870	8.6019	8.5448	8.4736	0.98	0.66	1.49	0.5	0.9
(40, 98, 2.25)	9.7600	9.7009	9.5426	9.4867	9.3701	1.63	0.59	1.81	0.4	0.7
(40, 85, 2.75)	10.418	10.3403	10.2600	10.1647	10.1198	0.78	0.93	1.37	0.3	0.6
(60, 274, 0.75)	8.4635	8.3517	8.3005	8.2539	8.1971	0.61	0.56	1.25	3.9	6.9
(60, 213, 1.25)	9.3309	9.1134	8.9938	8.9401	8.8550	1.31	0.60	1.54	3.0	5.4
(60, 175, 1.75)	9.9884	9.9281	9.7852	9.7167	9.5896	1.44	0.70	2.00	2.5	4.5
(60, 148, 2.25)	10.8351	10.7474	10.5207	10.4605	10.3711	2.11	0.57	1.42	2.1	3.8
(60, 128, 2.75)	11.7163	11.6041	11.4332	11.3386	11.2771	1.47	0.83	1.37	1.8	3.4
(80, 366, 0.75)	9.2200	9.0866	8.9121	8.8793	8.8079	1.92	0.37	1.17	12.9	22.4
(80, 284, 1.25)	10.2736	10.0404	9.8271	9.7731	9.7464	2.12	0.55	0.82	9.8	17.4
(80, 233, 1.75)	10.8944	10.7883	10.6345	10.5975	10.4883	1.43	0.35	1.37	8.5	14.4
(80, 197, 2.25)	11.5296	11.5739	11.4173	11.2867	11.2177	1.35	1.14	1.75	6.7	12.2
(80, 171, 2.75)	12.2635	12.2992	12.1833	12.1095	12.0641	1.94	0.61	0.98	5.8	11.3
(100, 457, 0.75)	9.9841	9.79773	9.68693	9.6769	9.5622	1.13	0.10	1.29	34.1	58.2
(100, 356, 1.25)	10.8816	10.8528	10.7061	10.6556	10.5206	1.35	0.47	1.73	26.6	46.5
(100, 299, 1.75)	11.5998	11.5309	11.2731	11.2275	11.1617	2.24	0.40	0.99	21.4	38.9
(100, 246, 2.25)	12.3813	12.5255	12.2965	12.2296	12.0853	1.83	0.54	1.72	16.6	32.1
(100, 213, 2.75)	13.3556	13.2637	13.1183	13.0246	12.9148	1.10	0.71	1.55	15.2	27.5
Average	9.7952	9.7263				1.52	0.60	1.36		

remember, however, that CH by itself is known to provide fairly good solution quality—for instance, Bertsimas et al. [26]'s results are within 3% of (and in some cases better than) the *deterministic* solutions obtained with complete *a-priori* knowledge of demand realizations through the CW [3].

- Comparing the last rows of Tables 5.2 and 5.3 (columns CH and $CHLK$), we find that CH under $NN + LK$ provides noticeably higher solution values than CH under NN+2-*opt*; the evidence thus argues in favor of attributing the improvements under RR to re-optimization.

- Coming to the second part (i.e., the *quality* aspect) of Arora's observations, above, the apparent inability of LK to find a better initial solution than 2-*opt* seems due to our inability to "tune" LK appropriately. However, RR does reasonably well even with this limitation. It is understood, moreover, that software that can better tune LK is available, and so, *even better results should be possible in practice* under RR, both as regards running time and solution quality. It is also worth pointing out here that Arora's observations about tuning do not necessarily apply to all tour improvement procedures-the code for 2-*opt*, for instance, is a fairly straightforward affair, and our code would probably not be noticeably inferior as compared to a professional version.

Next, to get some idea of the underlying trends in the results across categories, we first categorize the average results n-wise and $\bar{f}'$-wise (Table 5.4). Secomandi [33, p. 801] observes that "...average % improvements...appear to be more substantial with larger instances...", and indeed, all the improvements in SE do seem to have a strong positive correlation with n. In both $DA1$

Table 5.4.
Average % improvements across n and $\bar{f}'$ for SE, $DA1$, and $DA2$ results

		n					$\bar{f}'$				
		20	40	60	80	100	0.75	1.25	1.75	2.25	2.75
SE	SR	0.36	0.92	1.49	–	–	1.46	0.84	0.48	–	–
	DR	0.87	0.98	1.60	–	–	1.35	0.78	1.31	–	–
DAM	SR	1.17	1.57	1.27	1.21	0.93	0.93	1.30	1.05	1.47	1.39
	DR	0.50	0.48	0.42	0.48	0.33	0.10	0.46	0.44	0.62	0.58
	RR	0.88	1.01	0.95	1.10	1.38	0.35	1.07	1.28	1.29	1.33
DAC	SR	1.39	1.76	1.39	1.55	1.53	1.83	1.65	1.43	1.68	1.02
	DR	0.66	0.62	0.65	0.60	0.45	0.38	0.61	0.56	0.70	0.74
	RR	1.1	1.5	1.52	1.22	1.46	1.22	1.41	1.41	1.55	1.21

and $DA2$, however, we find that the % improvements with respect to n show a declining trend *ab-initio* under DR, whereas under RR they exhibit almost monotonic increase in $DA1$, and positive correlation in $DA2$. It seems, therefore, that RR would be relatively more effective than DR for larger n. % improvements under SR appear to be roughly bell-shaped in $DA1$; in $DA2$, although the trend is not equally clear, they appear to be concave. Intuitively, increasing n should cause greater improvements (since more alternatives become available) *but equally*, "tapering off" should also occur after a point (since more and more nodes are being accommodated in the same space). This seems to explain the concavity of the improvements under SR in our experiments, which consider larger numbers of customers, but at the same time, RR seems better equipped to compensate for this "falling off" of improvements than DR, and thus maintains an increasing trend longer.

While variation with fill rate does not seem to follow a trend in SE or under SR in $DA2$, in the others there is some evidence of larger percentage improvement with higher $\bar{f}'$. This also is intuitively plausible, since a higher value of $\bar{f}'$ means more preventive trips to the depot, and, consequently, more chances of a change in sequence. Moreover, RR and DR seem equally affected by this factor, so it seems that RR would retain its advantage *vis-a-vis* DR across different $\bar{f}'$. Of course, more experience with the different policies would be needed before definitive statements could be made about the trends.

Our concluding observation has to do with some other differences noticed between SE and $DA2$, identical cases as regards depot positioning within the unit square. First, while SE roughly provides an average improvement of 2.27 (=1.03+1.24)% across *CH-SR-DR*, the average across the corresponding categories (i.e., considering only Secomandi's $(n, Q, \bar{f}'$ sets)) in DAC is 2.21%, so the findings are consistent overall. However, SR seems to do better in $DA2$ (as well as in $DA1$) than in SE, while the reverse (the difference appears significant) is the case with DR. Given the large number of iterations used in all the experiments, *prima facie* this anomaly is unexpected. On closer inspection, we feel, however, that this is probably due to differences in the random number generation (packages, streams used). In particular, for the smaller values of n, the generated locations could significantly affect the results. (With a larger number of customers, the distribution

of customers in the assigned space would tend to be more or less identical across streams.) As an illustration, the results of *SR*, extracted from the different experiments (Table 5.5), do provide testimony that the gap between the expected distances traversed in the two cases reduces as n increases and thus we are led to conclude that, while the best solution quality should correspond to *RR*, the exact quantum of improvement brought about by any of the procedures would depend on a number of factors, e.g., depot and customer locations, n and $\bar{f}'$.

Table 5.5.
Comparison of expected distances under *SR*

	$n = 20$			$n = 40$			$n = 60$		
Q	91	71	58	183	142	116	274	213	175
SE	5.29	5.81	6.62	7.16	7.93	8.80	8.38	9.20	9.89
$DA1$	5.773	6.597	7.474	6.882	7.837	8.601	8.300	8.993	9.785

5.5 Dynamic improvement incorporating *SNP*

The reader may recall our discussion in Section 3.8 of the problem sizes that would be amenable to *SNP*, the least time-consuming among the exact policies considered therein, and thus the exact policy of choice for answering the question raised at that point: If problem parameters happen to fall outside the computational feasibility limits of an exact MDP approach, could the latter at least be imbedded in the solution procedure of this larger problem to achieve better results? Our objective in this section is to attempt to answer this question by using the insights from the preceding sections of this chapter to combine approximate and exact techniques through our procedure "Approximate-Exact" (*APEX*).

The approximation is provided by *CH*, through the initial solution. As in the case of the *rollout* techniques, this initial solution also serves as the base with which we compare the refined solution. We let ν denote the number of nodes that the *SNP* algorithm can effectively address for a given problem instance without either increasing computational time beyond viability or violating memory capacity constraints. *APEX* then works as follows.

1. Obtain the *CH* solution for all n customers. This is the current best route.
2. Repeat the following steps for $i = 1$ to $i = n$.
   (a) Note the demand realization at the customer at position $i-1$ (if $i = 1$, $d_{i-1} = 0$).
   (b) Run the *SNP* algorithm on the set of customers at positions i to $\min\{n, i+\nu-1\}$ on the current best route.
   (c) If $n > i+\nu-1$ in step (b), incorporate the expected cost of traversal of the remaining route comprising of $n' = n-(i+\nu-1)$ customers according to equations (5.5.1) through (5.5.3).
   (d) Assign position i according to equations (5.5.4) and (5.5.5).
   (e) If the incumbent at position i changes as a result of execution of step (d), place the earlier incumbent at the position vacated by the new incumbent.

If $n' > 0$, we denote by ι the first among the n' customers on the remaining (i.e., the original *CH*) route at any given execution of step 2(b) above, and by j an arbitrary customer among the ν considered by *SNP*. Then, with $\gamma_j(q_j) = \min\{\gamma_j^0(q_j), \gamma_j^1(q_j)\}$ as in Section 5.1, we have

$$\gamma_j^1(q_j) = c_{j\iota} + \sum_{k=0}^{q_j} \gamma_\iota\left(q_j - k\right) p_{\iota k} + \sum_{k=q_j+1}^{d_\iota^{\max}} \left[2c_{0\iota} + \gamma_\iota\left(Q + q_j - k\right)\right] p_{\iota k} \tag{5.5.1}$$

$$\gamma_j^0(q_j) = c_{0j} + c_{0\iota} + \sum_{k=0}^{d_\iota^{\max}} \gamma_\iota\left(Q - k\right) p_{\iota k} \tag{5.5.2}$$

The recursive algorithm for *SNP* (as for the other exact policies), however, conditions on the expected cost to go *on arrival* at a customer (rather than *after* service, as in *rollout*). Denoting, then, by q^j the remaining capacity on arrival at j, and by $\Gamma(\cdot)$ the corresponding expected cost, we have

$$\Gamma_j(q^j) = \sum_{k=0}^{q^j} \gamma_j \left(q^j - k\right) p_{jk} + \sum_{k=q^j+1}^{d_j^{\max}} \left[2c_{0j} + \gamma_j \left(Q + q^j - k\right)\right] p_{jk} \tag{5.5.3}$$

$\Gamma(\cdot)$ thus enables us to compute the solutions of one-customer subproblems under SNP, and in turn, higher-dimensional subproblems. Next, on the lines of Section 5.1, we have $i = \arg\min\Big\{\min\big([V(1)], [V(0)]\big)\Big\}$, where $i-1$ references the customer at that position, and where

$$V(1) = \min_{j \in Y_{i-1}} \left\{ c_{i-1,j} + \sum_{k=0}^{d_j^{\max}} p_{jk} E_{SNP}\left[\lambda_j | k, q^j = q_i\right] \right\} \tag{5.5.4}$$

$$V(0) = \min_{j \in Y_{i-1}} \left\{ c_{0,i-1} + c_{0j} + \sum_{k=0}^{d_j^{\max}} p_{jk} E_{SNP}\left[\lambda_j | k, q^j = Q\right] \right\} \tag{5.5.5}$$

where λ_j denotes the cost to go if the sequence of traversal is $(i-1)$-j-ι, with the CH sequence followed from customer ι onward.

Since *APEX* retains the remaining CH sequence in its tail, by arguments similar to those in the proof of Proposition 5.2.1, it follows that it would provide expected cost no worse than CH. However, as before, the performance of *APEX* cannot theoretically be compared with those of the *rollout* techniques; hence empirical comparison is necessary.

We retain the experimental region and depot location of $DA1$ along with the number of replications. The policy comparisons are also essentially the same as in $DA1$, with the results of *APEX* also being included now. Specifically, the improvements obtained by *APEX vis-a-vis CH* are compared with those obtained by *DR* and *RR vis-a-vis CH*. (To obtain the latter values, the two-stage improvements, e.g., *CH-SR-DR* are combined into a single improvement.) Keeping

in view the additional computational effort involved, we drop some of the values of n and $\bar{f}'$; we thus choose $n \in \{20, 40, 60\}$ and $\bar{f}' \in \{0.75, 1.25, 1.75\}$—the same as in Secomandi's original experiments. Also, after some preliminary trials, we found $\nu = 15$ to be a good choice keeping in view computer-memory constraints.

We note also that as soon as the remaining route contains ν customers, i.e., as soon as $n' = 0$, we have an *exact* (within the limitations of *SNP*) solution for the *entire remaining route*, without the approximation involved in "tagging on" the "tail of the route" via (5.5.1) through (5.5.3). Hence, after this stage, *SNP* computations need not be repeated any longer; the optimal actions corresponding to all possible states that are obtained when running *SNP* with $n' = 0$ can be stored once for all, and used in deciding where to go next at the subsequent customers. However, memory capacity permits us to take the storage option only when the number of customers left is 11, so we carry on the computations for a further four stages before storing the results.

Finally, while the computational time of an *individual iteration* of *APEX* (i.e., deciding all n positions on a route for a given set of demand realizations) remains practicable, that of 2000 iterations would not, at least for the higher values of n (as the table below shows). Hence, while there still are 200 iterations for each replication under $n = 20$, the numbers of iterations per replication are 60 and 20 respectively for $n = 40$ and $n = 60$. The number of iterations for these higher values of n are thus inadequate to estimate accurately the expected distances, according to Bertsimas et al. [26]'s criterion noted in Section 5.3; the improvements that *APEX* brings about for these values of n, however, appear consistent with the improvements for the case $n = 20$. Also, a look at the corresponding entries of Table 5.2 for the results under the other policies should convince the reader that there are essentially no differences in the expected distances traversed even with the number of iterations reduced.

Two sets of computational times for *APEX* are shown in our results (Table 5.6). The first, *iter*, denotes the running time for an iteration. These running times are much greater than those for the *rollout* techniques; noting, however, that we suppose these experiments to be performed during vehicle halts, the column *halt* roughly indicates the time taken for one such computation

(i.e., the average computational time at a vehicle halt). Since *SNP* is the dominant contributor to running time, we obtain this latter time by dividing the entry under *iter* by the number of vehicle halts at which *SNP* is invoked (ignoring *SNP* running times for the cases where it is applied to less than ν customers, since reduction in problem dimension by one essentially halves the running time).

Table 5.6.
Performance of *APEX*

	expected distance					% improvement vs CH			time(secs)	
$n, Q, \bar{f}'$	*CH*	*SR*	*DR*	*RR*	*APEX*	*DR*	*RR*	*APEX*	*iter*	*halt*
(20, 91, 0.75)	4.374	4.342	4.341	4.339	4.222	0.77	0.81	3.47	132	26.3
(20, 71, 1.25)	4.799	4.747	4.713	4.689	4.628	1.56	2.06	3.34	103	20.5
(20, 58, 1.75)	5.090	5.031	5.003	4.977	4.903	1.71	2.21	3.67	81	16.2
(40, 183, 0.75)	5.742	5.633	5.622	5.615	5.550	2.09	2.22	3.36	1449	57.9
(40, 142, 1.25)	6.102	5.995	5.963	5.935	5.939	2.29	2.73	2.67	1319	52.8
(40, 116, 1.75)	6.468	6.366	6.325	6.296	6.192	2.22	2.66	4.28	893	35.7
(60, 274, 0.75)	7.026	6.973	6.956	6.944	6.823	1.00	1.18	2.90	4272	94.9
(60, 213, 1.25)	7.400	7.303	7.293	7.226	7.080	1.45	2.34	4.31	3266	72.6
(60, 175, 1.75)	7.709	7.677	7.648	7.584	7.407	0.79	1.62	3.92	2652	58.9
Average						1.54	1.98	3.55		

We observe that:

- The percentage improvements obtained by *APEX* are superior to those obtained via the "purely approximate" techniques, and the improvements appear to be significant.

- The improvements obtained by *APEX*, both compared to *CH*, and compared to the dynamic *rollout* techniques, are maximum in the case $n = 60$, which seems to be a positive indication as far as its applicability to larger problems is concerned.

- The computational times at a vehicle halt, while much more than in the approximate techniques, still appear to be viable.

Chapter 6

Multiple Vehicles-Dynamic Policy

In this chapter, devoted to VRPSCDB, we start with a brief description of previous work of relevance, and lay out the intuition behind our proposed approach—the focus of Section 6.1. Section 6.2 provides a description of our basic model and the procedure for evaluating the cost of *a priori* routes. Section 6.3 describes the heuristic that we use to generate *a priori* routes. Later sections compare the obtained solutions with solutions obtained via approaches that in our opinion simulate current industry practice, and also apply dynamic improvement techniques.

6.1 The Background

We assume that

- There are m vehicles, each of capacity Q, to cater to customer demands.
- Delivery customers (δ) require daily visits, whereas pickup (π) customer i requires service on a given day with probability ϕ_i.
- Pickup locations for the day are not known at the time when the vehicles commence their runs.
- There is a cut off time (W) beyond which pickup requests cannot be entertained. (If a pick up request is phoned in before the cutoff time, it must be fulfilled.)
- Each vehicle must in general cater to both types of customers, who can be visited in any order, except that pickup customers cannot in general be visited before information regarding their presence for the day is available.

- There is an upper limit (OL) on the number of hours a driver may work per day without incurring overtime (OT), and an upper limit (SL) on the hours a driver may travel in all (overtime travel inclusive). The first upper limit is due to contractual obligations; the second is a consequence of safety requirements, and once it is reached, the vehicle must stop altogether and await a fresh driver.

In the real-world problem that served as one of the motivations for this research, delivery amounts are known with certainty and pickup amounts are known when the pickup requests are made. In the interests of generality, however, we assume that both delivery and pickup amounts are random variables, and that the actual quantum of demand for any customer is known only on arrival at that customer.

We also assume—to avoid more unwieldy expressions—that the demand of a customer that is present is always non-zero. Also, the time at which it is actually known whether a given pickup location has a demand on a particular day is often a random variable in practice; however, in the interests of model tractability, we assume availability of this information for all π customers at time W.

We next address the issue of one of our implicit model assumptions that is critical to our development, namely, that pickups and deliveries can be made in any order.

6.1.1 Validity of "Any-order-service" Assumption

In the context of VRPB, Jacobs-Blecha and Goetschalkx [66, p. 1] rule out such an assumption, stating that: "...all deliveries must be made on each route before any pickups can be made...the vehicles are rear-loaded, and rearrangement of the loads...at the delivery points is not deemed economical or feasible." While, however, the stochasticity of our problem argues in favor of allowing mixing of δ and π locations, their observation might not always hold even in a (deterministic) VRPB, as it does not in one of the three case studies detailed by Casco et al. [63]. The latter authors, while conceding [p. 136] that "...many vehicles are rear-loaded...", also observe that the

"deliveries first" dictum is often a consequence of other anticipated problems at pickups "...such as lack of a loading dock" [rather than to apprehensions about onboard rearrangement *per se*]. Since, as they further observe "In most distribution operations, deliveries receive higher priority than pickups...[and] customers...expect timely deliveries", it is inevitable that "Management insists that the vast majority of units on a route be delivered before pickups are considered."

For cases where the last-mentioned restriction is in fact enforced by management, Casco et al. [63, p. 136] suggest a set of rules whereby one might persuade management to consider a relaxation (and thereby allow consideration of other potentially better solutions), e.g.:

1. ...[Any service] vehicle should deliver 60% of its units before accepting any backhaul.

2. If a delivery follows a backhaul...accept the backhaul as long as no more than 80% of the vehicle is full after doing so...reserve [some] vehicle capacity for 'load shuffling'..."[so that rearrangement time does not remain a factor any longer].

They also add: "These rules are somewhat ad hoc, but, to a large extent...follow what common sense would dictate."

Our inclusion of time windows (TW) thus ensures that a rule in the spirit of the first rule above is implicitly followed (a good route must necessarily visit *some* δ customers before arriving at a π location) and, though we have tried to keep our model "general purpose", such a provision is specifically incorporated in the accompanying algorithms.

We next note that in a twin-operation context(TOC), unlike in the single-operation context (SOC), a start (or re-start after route failure) from the depot would not necessarily occur with Q units of deliverables; rather, the starting quantity also becomes a decision variable, and in case of a tie, our model chooses the minimum possible deliverable quantity. Consequently, since, (a) real-world routes have substantially fewer pickups than deliveries (a situation our computational experiments attempt to replicate) and, (b) space would become available due to the first rule being followed as in the preceding paragraph, one would expect available vehicle capacity to permit fast

and easily arranged "coexistence" of δ and π consignments in general. Casco et al. [68]'s second rule is thus also honored in substance.

6.1.2 Some possible solution methodologies

Given, then, that our assumptions can reasonably be justified, our first concern is the development of *a priori* routes for VRPSCDB, and a possible approach is to adapt one of the techniques described in Chapter 2 for the VRPB. As far as exact VRPB techniques are concerned, however, we are aware of only one—one that, moreover, is severely restricted in applicability, as discussed in Chapter 2.

We could also adapt a heuristic VRPB technique; a problem with this approach, however, is that extensions of techniques successful in deterministic cases to the corresponding stochastic versions have not always been equally successful. As Gendreau et al. [25, p. 470] observe: "Traditional heuristic approaches to deterministic VRPs, and in particular those based on geometric arguments such as tour partitioning, are likely to produce highly sub-optimal solutions [to the SVRP]." (Since we will frequently refer to Gendreau and co-authors, we will from now on refer to them as GEA.) For instance, in GEA [24, p. 150, Fig. 3], the authors present the optimal sets of routes delivered by their exact algorithm on a small ($n = 8, m = 2$) instance of the VRPSCD for nine randomly generated realizations of customer presence and demands. The routes crisscross in five of these realizations, clearly indicating that on such a problem, deterministic tour partitioning might not generate optimal solutions since, by its very nature, the technique cannot generate crisscrossing routes.

6.1.3 Rationale of the proposed approach

It seems clear, therefore, that stochasticity must be incorporated *ab initio* in the process of developing solutions for problems with stochastic elements. Among exact approaches that thus incorporate stochasticity, the literature reveals, as in VRPB, only one exact algorithm for the multi-vehicle VRPSCD: that of GEA [24], which is based in turn on the so-called Integer L-shaped

method of Laporte et al. [23]. This algorithm is used to solve to optimality different variants of SVRP. In particular, for the VRPSCD, results are reported only for the 2-vehicle case with $n = 46$, and, as the authors themselves concede (p. 151, GEA [24]), the ability to solve a given problem instance exactly is limited by dimensional constraints.

In the same vein, the experiments of Chapter 3 indicate the dimensional constraints that exact approaches suffer from, and the difficulty is likely to be compounded with the inclusion of two types of customers, TW, and other problem complexities. Consequently, our focus is the development of a heuristic base solution. We have seen in Chapter 5 that *a priori* solutions, even if "very good", could nevertheless be improved by a dose of "dynamism" and therefore can hope that a heuristic solution improved by dynamism would merit consideration by practitioners.

Among heuristic approaches for the single-vehicle case we have *CH* (an excellent heuristic result-wise, as discussed in Chapter 5). Its extension to the multi-vehicle case, however, would imply the assumption that a vehicle re-starting from the depot is in effect the second vehicle starting its run. In practical situations involving multiple vehicles, however, as in our case, such vehicles must commence their journeys at more or less the same time and thus their itineraries must be decided beforehand. *CH* must therefore be ruled out for the starting solution; the utility of its DP philosophy remains, however, and is duly exploited in our dynamic improvement procedure.

Other multi-vehicle heuristics have generally involved adaptation of well known deterministic heuristics, e.g. Stewart et. al. [7] used *CW* and Waters [12] used the *Sweep* algorithm of Gillett and Miller [47]. While some, like *Sweep*, are essentially geometric, and thus best avoided in light of GEA [25]'s observation above, judgment of the solution quality offered by others has consequently been imprecise at best.

Precise comparisons with the exact results in GEA [24] were, however, carried out by GEA [25], who developed a tabu search[1] heuristic called *Tabustoch* for VRPSCD (also the only *heuristic* approach to the VRPSCD), obtaining an average deviation of 0.38 % from optimality for the problems to which *Tabustoch* found a solution within a pre-determined time for a number of iterations. Although their procedure allows for any number of vehicles, results are reported, as in the exact case, only for two vehicles. Still, the results indicate the effectiveness of *Tabustoch*, and it accordingly serves as our starting point.

In Chapter 2, we described two heuristic procedures for the multi-vehicle VRPSD proposed by Yang et al. [29]. These authors also compared their heuristic solutions with lower-bound solutions based on the LP relaxation of a set-partitioning formulation of multi-vehicle VRPSD. However, their heuristics do not permit stochasticity in customer presence.

Also significant is the fact that so far, no procedure for multiple-vehicle SVRPs has incorporated dynamism, and Secomandi [33, p. 801], rightly observes that: "Improvement of the proposed method [*rollout*] to encompass multiple vehicles is a topic for further research"; this is the focus of Section 6.5. As pointed out in Chapter 3, furthermore, the shapes of the optimal routes (Figure 5.1) clearly imply the possibilities of larger benefits with a dynamic policy that addresses the *interactions* between the different routes, i.e., simultaneous reoptimization across routes, rather than reoptimization of every route in isolation; this is the focus of Section 6.6. In fact, *Tabustoch* does not consider another dimension of the problem important in real life, namely, the *time* taken for traversal of a given route (or its proxy, route length). Our "simultaneous reoptimization" of Section 6.6, on the other hand, takes the route lengths of the individual routes into consideration dynamically, focusing on reducing overall costs by reducing drivers' OT. This is accomplished by allowing a vehicle to "hop" from one route to another if certain conditions are satisfied. Our work thus addresses both of these apparent lacunae.

[1]Tabu search, as used today, originated with the work of Glover [55] and Hansen [56]. It is essentially a neighborhood search procedure that has worked well for deterministic VRP's. The name *Tabu* comes from the fact that recently considered solutions are made inaccessible (or Tabu) for a number of iterations. An excellent exposition of the technique can be found in Glover et al. [52].

6.2 The Model

On arrival at a customer i, we denote by q_d the deliverable amount on the vehicle, by q_p the amount picked up since the immediately preceding start from the depot, by q_t the total amount on the vehicle (thus $q_t = q_d + q_p$), and by q_a the amount of available empty space on the vehicle (thus $Q = q_t + q_a$). Also, $d_i \leq Q$ now can refer to either pickup or delivery, and in case of route failure or at route commencement, for each customer that might be visited next, there now is an optimal starting deliverable quantity that minimizes the expected cost of the rest of the route (whereas this quantity was invariably assumed to be Q in *SOC*). As well, a route failure now occurs under different conditions than in *SOC*. Specifically, route failure now means (a) inability to serve current customer completely, and/or (b) vehicle capacity inadequate to serve the next customer *at all*. For instance, if $q_d = 0$ after fully serving a δ, under *SOC* route failure would occur, whereas now, if the next location were a π, route failure would additionally require the condition $q_a = 0$.

6.2.1 *A priori* routes, TW not included

Given a set of *a priori* routes, we next explain how we determine the expected costs of their traversal. To start with, we assume that only vehicle capacity constraints are involved (TW is included in the formulation later in this section; *OL* and *SL* in the next). With these stipulations, we assume the following simple strategy in the traversal of any route in the route-set: follow the route, returning to the depot and restarting upon route failure. The traversal strategy, assuming knowledge about the *type* of the customer j being visited next (i.e., whether π or δ) may then be stated as follows (where "1" denotes the action of following the sequence i-j, "0" the sequence i-0-j, and "2" the sequence i-0-i-j):

At a δ customer

if ($d_i < q_d$)

if next is π

if $q_a + q_d - d_i > 0$ action 1

else action 0

else action 1

else if($d_i > q_d$) action 2

else

if next is δ action 0

else

if $q_a + q_d > 0$ action 1

else action 0

At a π customer

if ($d_i > q_a$) action 2

else if (next is δ)

if $q_d > 0$ action 1

else action 0

else

if ($d_i = q_a$) action 0

else action 1

In a stochastic scenario, of course, there would be a probability associated with a given customer being visited next, and the above can then be considered conditional actions taken given that a particular type of customer is next.

Following GEA [24], we conceive of the expected cost of traversal of a route as having two components, the expected direct cost (dc), assuming no recourse action (i.e., traversal with a vehicle of infinite capacity), and the expected recourse cost (rc). A recourse cost is incurred at i if route failure occurs, and is given by $2c_{0i}$ if i must be revisited (i.e., if residual demand remains at i), and by $e_{ij} = c_{0i} + c_{0j} - c_{ij}$, otherwise. We assume that route i comprises n^i customers, numbered in order 1 through n^i, with $n^i + 1 \equiv 0 \; \forall i$ (the superscripts refer to the route); and we define $\beta_i^k(q_t, q_d)$

as the *rc* of the remaining portion of route k if the remaining route starts with i, with q_t the current total quantity on board at i, of which q_d is deliverable. The expected cost of a set of m routes is then given by

$$T = \sum_{i=1}^{m} \tau^i = \sum_{i=1}^{m} \left(\sum_{j=0}^{n^i} \sum_{k=j+1}^{n^i+1} c^i_{jk} \bar{\phi}^i_{jk} + \beta^i_1(q_t, q_d) \right) \tag{6.2.1}$$

where $\bar{\phi}^i_{jk} = \phi^i_j \phi^i_k$ if $k = j+1$, and $\bar{\phi}^i_{jk} = \phi^i_j \phi^i_k \prod_{h=j+1}^{k-1} 1 - \phi_h$, if $k > j+1$. Also, T represents the total expected cost of the set of routes and τ^i, that of route i. The correctness of the expression for *dc* (the first term in the rightmost equality in 6.2.1) is proven in GEA [24]. The expression for *rc* (the second term in the same equality), must, however, be developed afresh since the earlier model considers only one type of operation.

A note about notation is in order here. The equations for *rc* that we develop refer to any route $k \in \{1, ..., m\}$. Strictly speaking, therefore, we should use i^k to denote customer i on route k and $\beta^k_i(q_t, q_d)$ to denote the further *rc* at i. However, to simplify the notation, we omit the superscript k, assuming its presence, and with the understanding that the analysis refers to any route $k \in \{1, ..., m\}$. Thus we will use i and $\beta_i(q_t, q_d)$ instead.

The computation of *rc* (i.e., $\beta(\cdot)$) in (6.2.1) is carried out through backward recursion, stage-wise. We define a stage as a *position* in the *a priori* sequence (where the context is unambiguous, we also use i to denote the customer at position i). Starting with recourse costs at stage n^k on route k, we determine the recourse cost at stage 1^k (stage 0, corresponding to the depot, does not add any recourse cost).

This brings us to the crucial difference between *SOC* and *TOC* evident in our algorithm above: the action at customer i depends *both* on the vehicle *capacity* and the *type* of the customer that would be visited next; accordingly: 1. We define ψ_i as the probability that a δ is visited next. If $\psi_i \neq 0$, we call this δ, ν_i. 2. If i is π, we define ξ_i as the *rc* at i if *at least* one π occurs before ν_i.

Also, as noted earlier, if route failure occurs at i, the optimal starting q_d under TOC must be specifically computed. Hence, if route failure occurs at i: 1. We define $q_d^*(j,q)$ as the optimal starting deliverable quantity on arrival at j, where j is the first of the remaining customers *present*, and where q is the maximum possible value that q_d can have at j, and, 2. We define $\alpha_i(t)$ as the *rc* of at i if the supply-demand gap at i is t.

Since a π customer may or may not be present in a given realization, it is convenient also to define $\chi_i(q_t,q_d)$ as the *rc* at i, *given that i is present* and assuming arrival at i with (q_t,q_d). Thus for a δ, we must have $\chi_i(q_t,q_d)=\beta_i(q_t,q_d)$, but for a π, in general, $\chi_i(q_t,q_d)\neq\beta_i(q_t,q_d)$ since $\chi_i(\cdot)$ is computed assuming i is present and the effect of $\phi_i\neq 1$ must therefore be incorporated for computation of $\beta_i(\cdot)$. We then have

PROPOSITION 6.2.1. *If i is π, then (a) if i+1 is also π, $\beta_i(q_t,q_d)=\phi_i\chi_i(q_t,q_d)+(1-\phi_i)\beta_{i+1}(q_t,q_d)$, and (b) otherwise, $\beta_i(q_t,q_d)=\phi_i\chi_i(q_t,q_d)$.*

Proof. If i is π, $\chi_i(\cdot)$ must be weighted by ϕ_i to obtain i's contribution to *rc*. If $i+1$ is π then with probability $(1-\phi_i)$ the recourse costs will be those of stage $i+1$. If $i+1$ is δ, then in computing *rc* at $i-1$, we would add the expected recourse costs incurred in going to $i+1$ directly, weighted by probability ψ_{i-1}. Hence we do not include this cost in $\beta_i(\cdot)$.

We note that expressions like $\alpha(\cdot)$ and $\xi(\cdot)$ not only will allow us to make our equations more compact in the following, but also are indicative of the actual mechanics of the computational procedure, enabling polynomial time bounds (as we prove later).

We first consider computation of $\alpha_i(t)$. If $t=0$, we have $q=Q$, and otherwise: $q=Q+q_d-d_i$ if i is δ, and $q=Q+q_a-d_i$, if i is π. We then have

PROPOSITION 6.2.2. *Given that i is present* $\alpha_i(0)=\sum_{j=i+1}^{j=n} (\bar{\phi}_{ij}/\phi_i)e_{ij}\chi_j(q_d^*(j,Q),q_d^*(j,Q))$, *where* $q_d^*(j,Q)=\arg\min_{q\in(0,Q)}\chi_j(q,q)$

Proof. $\bar{\phi}_{0j}$ gives the probability that i and j are both present and j is the first customer to be visited. Division by ϕ_i is required since we assume that i is present. Since the vehicle starts from the

depot, $q_p = 0$, and thus $q_t = q_d$ for any value of q_d that we choose. The minimization determines the optimal starting value of q_d for each j and the summation thus computes the expectation of the recourse cost.

Similar reasoning also gives us

PROPOSITION 6.2.3. *Given that i is present, $\alpha_i(t) = 2c_{0i} + \sum_{j=i+1}^{j=n^{\cdot}} (\bar{\phi}_{ij}/\phi_i)\chi_j(q_d^*(j,v) + z, q_d^*(j,v))$, where, if (a) i is δ, $v = Q + q_d - d_i$, $z = 0$, $t = d_i - q_d$ and $q_d^*(j,v) = \arg\min_{q \in (0,v)} \chi_j(q,q)$, and, if i is π, (b) $v = Q + q_a - d_i$, $z = t = d_i - q_a$, and $q_d^*(j,v) = \arg\min_{q \in (0,v)} \chi_j(q+t,q)$*

Proof. If i is π, t is the picked-up amount that carries over to $i+1$. The limits of the minimization and the corresponding resultant states of nature are accordingly modified.

We next illustrate our earlier statement about the computations of $\alpha_{\cdot}(\cdot)$ etc. being possible in polynomial time. For instance, ξ_i, ψ_i and ν_i can be computed in $O(n)$ time at any stage i, where $n = \sum_{i=1}^{m} n^i$, as follows:

$prod \leftarrow 1$, $\xi_i \leftarrow 0$ (Initialize)

While the next customer j is not δ *do*

$\xi_i \leftarrow \xi_i + prod * \phi_j \chi_j(q_d^*(j,Q), q_d^*(j,Q))$ (Computations if j is the first customer visited)

$prod \leftarrow prod * (1 - \phi_j)$ (If j is not present)

$j \leftarrow$ next customer

if no δ encountered, $\psi_i \leftarrow 0$

else $\psi_i \leftarrow prod$

if $(\psi_i \neq 0)$ $\nu_i \leftarrow j$

We are now in a position to state the equations for rc at any customer in general:

PROPOSITION 6.2.4. *At the last stage on a route, represented by customer i, if (a) i is δ, $\beta_i(.,q_d) = \chi_i(.,q_d) = 2c_{0i}\sum_{d_i>q_d} p_{i,d_i}$ and, (b) if i is π, $\chi_i(q_t,.) = 2c_{0i}\sum_{d_i>Q-q_t} p_{i,d_i}$ and $\beta_i(q_t,.) = 2\phi_i c_{0i}\sum_{d_i>Q-q_t} p_{i,d_i}$*

Proof. For a δ, a visit to depot becomes necessary if $d_i > q_d$ irrespective of q_p, whereas for a π, it becomes necessary if $d_i > q_a$ irrespective of q_d. For a π, moreover, this cost is incurred with the probability that i is present.

Similarly, at any stage other than the last, equations 6.2.2 through 6.2.7, dealing respectively with the cases: (a) i and $i+1$ are both δ (6.2.2); (b) i is δ and $i+1$ is π (6.2.3); (c) i and $i+1$ are both π with $q_d \neq 0$ (6.2.4); (d) i is π and $i+1$ is δ with $q_d \neq 0$ (6.2.5); (e) i and $i+1$ are both π with $q_d = 0$ (6.2.6); (f) i is π and $i+1$ is δ with $q_d = 0$ (6.2.7), denote the expected recourse costs at stage i (for the reader's convenience, we provide a glossary of frequently used terms at the end of this section).

$$\beta_i(q_t,q_d) = \chi_i(q_t,q_d) = p_{i,q_d}\alpha_i(0) + \sum_{d_i<q_d} p_{i,d_i}\beta_{i+1}(q_t-d_i, q_d-d_i) + \sum_{d_i>q_d} p_{i,d_i}\alpha_i(d_i-q_d) \tag{6.2.2}$$

$$\begin{aligned}\beta_i(q_t,q_d) = \chi_i(q_t,q_d) &= p_{i,q_d}[\psi_i(e_{i,\nu_i} + \beta_{\nu_i}(q_d^*(\nu_i,Q), q_d^*(\nu_i,Q))) + \beta_{i+1}(q_t-d_i,0)] \\ &+ \sum_{d_i<q_d} p_{i,d_i}[\psi_i\beta_{\nu_i}(q_t-d_i,q_d-d_i) + \beta_{i+1}(q_t-d_i,q_d-d_i)] + \sum_{d_i>q_d} p_{i,d_i}\alpha_i(d_i-q_d)\end{aligned} \tag{6.2.3}$$

$$\begin{aligned}\chi_i(q_t,q_d) = &\sum_{d_i<q_a} p_{i,d_i}[\psi_i\beta_{\nu_i}(q_t+d_i,q_d) + \beta_{i+1}(q_t+d_i,q_d)] \\ &+p_{i,q_a}(\xi_i + \psi_i\beta_{\nu_i}(Q,q_d)) + \sum_{d_i>q_a} p_{i,d_i}\alpha_i(d_i-q_a)\end{aligned} \tag{6.2.4}$$

$$\chi_i(q_t,q_d) = \sum_{d_i\leq q_a} p_{i,d_i}\beta_{i+1}(q_t+d_i,q_d)) + \sum_{d_i>q_a} p_{i,d_i}\alpha_i(d_i-q_a) \tag{6.2.5}$$

$$\chi_i(q_t,0) = \sum_{d_i<q_a} p_{i,d_i}[\psi_i(e_{i,\nu_i} + \beta_{\nu_i}(q_d^*(\nu_i,Q), q_d^*(\nu_i,Q))) + \beta_{i+1}(q_t+d_i,0)] \\ +p_{i,q_a}\alpha_i(0) + \sum_{d_i>q_a} p_{i,d_i}\alpha_i(d_i-q_a) \tag{6.2.6}$$

$$\chi_i(q_t,0) = \sum_{d_i\le q_a} p_{i,d_i}[e_{i,\nu_i} + \beta_{i+1}(q_d^*(\nu_i,Q), q_d^*(\nu_i,Q))] + \sum_{d_i>q_a} p_{i,d_i}\alpha_i(d_i-q_a) \tag{6.2.7}$$

All the equations follow in a straightforward manner from our greedy strategy-namely, follow the route and do not visit the depot unless necessary. Thus, e.g., considering (6.2.3), if $d_i = q_d$, then with probability ψ_i we include the costs of going to ν_i via the depot and otherwise (since $d_i = q_d$ means that $q_a \neq 0$ after i is served), the recourse costs are those of proceeding to *some* π; if $d_i > q_d$, then we revisit i with the optimal q_d, and if $d_i < q_d$, with probability ψ_i we visit ν_i directly, and otherwise visit a π directly. Similarly, considering (6.2.7), since $i+1$ is δ and $q_d = 0$, the vehicle cannot go forward without visiting the depot, so the only question is whether it revisits i or not.

6.2.2 A priori routes, time windows included

We next assess the impact of time windows on the VRPSCD. As a concession to practicality, we assume that the first customer on any route is invariably a δ (and our heuristic also incorporates this assumption). As long as the next customer on the *a-priori* route is a δ, there is no risk of a wasted trip in following this route. Such a risk arises only when for the first time, it so happens at a δ customer that the next customer is a π, and W has not elapsed as of that time. (We refer to this first stochastic, or π customer on the *a priori* route, as σ). Note that this is the only situation where the inclusion of W in the problem makes a difference to the strategy to be followed. We will suppose that such a situation occurs at δ customer i. We next define $\rho_l(q)$ as the probability that the vehicle arrives at customer l with $q_d = q$, having travelled an expected distance $\lambda_l(q)$ ($q_p = 0$ since we have dealt only with δ customers so far). Assuming that customer j follows customer l, and considering only the route up to i, we set the initial values $\rho_1\{q_d^*(1,Q)\} = 1$ and

$\lambda_1\{q_d^*(1,Q)\} = c_{01}$. $\rho_j(q)$ can then be computed *via* the following recursive equations.

$$\rho_j(q) = \sum_k \rho_l(q+k)p_{lk} + \sum_k \rho_l(k)p_{lk}|q = q_d * (j,Q) + \sum_k [\sum_{v=0}^{k-1} \rho_l(k-v)p_{lk}|q = q_d * (j,v)] \quad (6.2.8)$$

The expected distance traversed corresponding to any non-zero $\rho_j(q)$ is similarly given by

$$\lambda_j(q) = \left(\begin{array}{l} \sum\limits_k p_{lk}\rho_l(q+k)[\{\lambda_l(q+k)/\rho_l(q+k)\} + c_{lj}] \\ + \sum\limits_k p_{lk}\rho_l(k)[\{\lambda_l(k)/\rho_l(k)\} + c_{0l} + c_{0j}]|q = q_d^*(j,Q) \\ + \sum\limits_k [\sum\limits_{v=0}^{k-1} p_{lk}\rho_l(k-v)[\{\lambda_l(k-v)/\rho_l(k-v)\} + 2c_{0l} + c_{lj}]|q = q_d^*(j,v)] \end{array} \right) / \rho_j(q) \quad (6.2.9)$$

Equation (6.2.8) conditions on the values of q, i.e., q_d at l that result in particular values of q at j. The computation $\lambda_l(\cdot)/\rho_l(\cdot)$ in (6.2.9) removes the weighting factor from $\lambda_l(\cdot)$. By linearity of expectation, the resulting term is the distance travelled up to i if the vehicle reached i with q units.

Two options are available at i: "0", i.e., wait till W before proceeding further, and, "1", i.e., proceed immediately to σ. We note that "1" might, e.g., be an attractive option if ϕ_σ is high. We next evaluate the consequences of following each option. Recalling that τ^k denotes the total cost of route k, we add a subscript (0 or 1) to τ^k to indicate the option followed. We assume that under option 0, if $d_i > q$, the vehicle would proceed to the depot and wait there until W, and otherwise, it would wait at i (note that if $d_i = q$, a trip to the depot might not be necessary). Now there arises another problem. W as defined is in time units. To study its impact on route cost, it must therefore be converted into equivalent distance units. We describe this conversion process in Section 6.4—for now, we assume that W has been thus converted, and refers to a distance. Then, with the understanding that the terms involving W are 0 if negative, we have

$$\tau_0 = \sum_{j=0}^{n'} \sum_{k=j+1}^{n'+1} c_{jk}\bar{\phi}_{jk} + \chi_1(q_t, q_d) + \sum_{q=0}^{Q} [(W - \lambda_i(q)) \sum_{d_i \le q} p_{i,d_i} + (W - \lambda_i(q) - c_{0i}) \sum_{d_i > q} p_{i,d_i}] \quad (6.2.10)$$

Considering next option 1, if $d_i \leq q$ the vehicle proceeds to σ, otherwise, for any $\lambda_i(q)$, there are three possibilities:

- (a) $\lambda_i(q) + c_{0i} \geq W$. Proceed as if there were no TW.
- (b)$\lambda_i(q) + c_{0i} < W \leq \lambda_i(q) + 2c_{0i}$. On return to i, go to the next in the sequence that is actually present.
- (c) $\lambda_i(q) + 2c_{0i} < W$. Proceed to σ.

If $d_i \leq q$, or cases (a) or (b) occur, option 1 affects dc also (since now the edge (i, σ) is traversed with certainty) and thus the modified dc, i onward, denoted by κ_i, is given by

$$\kappa_i = c_{i\sigma} - \sum_{j=\sigma}^{n^{\cdot}+1} \bar{\phi}_{ij} c_{ij} + ((1/\phi_\sigma) - 1) \sum_{j=\sigma+1}^{n^{\cdot}+1} \bar{\phi}_{\sigma j} c_{\sigma j} \tag{6.2.11}$$

Next, letting $t = Q + q - d_i$ we note the following two possibilities under case (c): $q_d^*(\sigma, t) \neq 0$ (c1), and $q_d^*(\sigma, t) = 0$ (c2). Then, with $z = W - \lambda_i(q) - 2c_{0i} - c_{i\sigma}$, we have

$$\begin{aligned} \beta_i(q, q|W) &= p_{i,q}(\psi_i(e_{\sigma,\nu_i} + \beta_{\nu_i}(q_d^*(\nu_i, Q), q_d^*(\nu_i, Q))) + \beta_\sigma(q - d_i, 0) + \kappa_i + W - \lambda_i(q) - c_{i\sigma}) \\ &+ \sum_{d_i > q_d} p_{i,d_i} x + \sum_{d_i < q} p_{i,d_i} [\psi_i \beta_{\nu_i}(q - d_i, q - d_i) + \beta_\sigma(q - d_i, q - d_i) + \kappa_i + W - \lambda_i(q) - c_{i\sigma}] \end{aligned} \tag{6.2.12}$$

where $x = \alpha_i(d_i - q)$ under case(a), $x = 2c_{0i} + \sum_{j=\sigma}^{j=n^{\cdot}} (\bar{\phi}_{ij}/\phi_i)\chi_j(q_d^*(\sigma, t), q_d^*(\sigma, t))$ under case (b), $x = \psi_i \beta_{\nu_i}(q_d^*(\sigma, t), q_d^*(t)) + \beta_\sigma(q_d^*(\sigma, t), q_d^*(\sigma, t)) + z$ under case(c1) and $x = \psi_i[e_{\sigma,\nu_i} + \beta_{\nu_i}(q_d^*(\nu_i, Q), q_d^*(\nu_i, Q))) + \beta_\sigma(q_d^*(\sigma, t), q_d^*(\sigma, t)] + z$ under case(c2). Consequently, we have

$$\tau_1 = \sum_{q=0}^{Q} \rho_i(q)(\lambda_i(q) + w(q)) \tag{6.2.13}$$

We note here that option 1 could be generalized to more than one such visit to a π customer, presence unknown as yet, in the case where the time at which $\phi(\cdot)$ became known were different

across customers, or even a random variable. By our assumptions, however, the vehicle must wait at σ till W if it reaches there before W (since the presence of σ is unknown till W). We also note that, with the inclusion of W, our (greedy) strategy must also be modified as follows:

1. Follow the *a priori* route, returning to the depot and re-starting in case of route failure.
2. If W has not elapsed on arrival at δ customer i, and the next customer on the *a priori* route is a π, act as per *option* "0" or "1" as determined to be best.

Finally, we establish computational complexity for the above procedure

LEMMA 6.2.1. *The time complexity of computing the expected cost of all the routes is* $O(n^2+nKQ^2)$

Proof. At each customer, in computing $q_d^*(\cdot)$ for each value of the supply-demand gap, we consider Q values of q_d and for each value of q_d, consider at most K distinct values that demand can take. Thus, the complexity of this computation is $O(KQ)$. (Procedurally, the computations can be repeated at most KQ times—equivalently, we say that KQ cells are accessed). The computations of α_i and ν_i (as shown earlier) are $O(n)$ each. From similar considerations of the number of cells accessed, the computations of $\beta_i(\cdot)$ and $\chi_i(\cdot)$ can take at most $O(KQ^2)$ each. The forward recursion used in assessing the impact of TW (equations 6.2.8 and 6.2.9) considers $O(Q)$ cells at each customer. The computation of x under option 1 (shown following equation 6.2.9) takes $O(n)$ and of $w(q)$, $O(KQ)$. Since the maximum number of cells accessed at a customer is thus $O(KQ^2)$, the total number of customers in all the routes is n, and only constant time operations are performed for each cell accessed, the computation of rc can take at most $O(nKQ^2)$. The computation of direct costs takes $O(n^2)$ (proved by GEA [24] earlier—also follows from the cell count). The lemma follows.

6.3 The base heuristic

In describing our heuristic, that we call *Tabutwin*, we adopt the notation used by GEA [25] wherever possible, and first explain their terms. Next, we describe how we modify their procedure.

Table 6.1.
Glossary of frequently used terms

notation	explanation
$\beta_i(q_t, q_d)$	Recourse cost of the route at stage i
$\chi_i(q_t, q_d)$	Recourse cost of the route at stage i assuming customer i is present
κ_i	Direct cost of the route, node i onward
ϕ_i	Probability of presence of customer i
ψ_i	Probability that a δ customer is visited next
ν_i	The first δ customer on the route after customer i
ξ_i	If i is a π, recourse cost at i if *at least* one π occurs before ν_i
$q_d^*(j, q)$	Optimal deliverable quantity on arrival at j, with q maximum possible value
$\alpha_i(t)$	The recourse cost of the route at i if the supply-demand gap at i is t

6.3.1 *Tabustoch* revisited

A proposed solution to the problem assigns to each edge (i, j) a value $x_{ij} \in \{0, 1, 2\}$ depending upon the number of times the edge is used in the solution (2 on a single customer route). x denotes the vector x_{ij}. T is now treated as a function of x and hence referred to as $T(x)$. The neighborhood $N(u, v, x)$ of a solution x is the set of solutions reachable by removing in turn one of u randomly selected customers, and inserting it either immediately before, or immediately after one of its v nearest neighbors. If a node is moved at iteration t, its further movement is *tabu* until iteration $t+\theta$, where θ is randomly selected in the interval $\{n-5, ..., n\}$. (Empirically, random tabu durations have been shown to reduce the probability of cycling, and 5 is the figure recommended by earlier researchers; both are accordingly adopted herein.)

Customers are numbered serially 1 to n, and S_k denotes the set of customers on route k. We use $[i^-]^t$ to denote the predecessor of i at iteration t and $[i^+]^t$ for its successor. Both heuristics performs a number of predetermined iterations MAX and output the best route found, the basis of evaluation of competing sets of routes being expected costs.

It would be prohibitively expensive to compute the cost of all candidate routes encountered, so proxies must be used for preliminary screening. Empirically, GEA [25] found the best proxy for the change in dc to be $A(i,j,l) = \phi_j(c_{ij} + c_{jl} - c_{il})$ if j is inserted between i and l. We think this measure would be applicable in our case, too, since the "type" of a customer does not affect direct cost, and therefore adopt it without modification.

For rc, GEA [25] use r_i^k, the change in recourse cost of route k caused by removal of customer i from k, to estimate the change that would occur in the rc's of both routes if i were inserted into route $j \neq k$ (the computation of insertion cost, as compared to removal cost, is costlier by a factor of n, so this means considerable time savings). The maximum change μ^k in rc of route k caused by removal of a customer from k is maintained, and in estimating the impact of insertion of i into k, μ^k is weighted by i's weight. This weight is obtained by dividing i's average demand, multiplied by ϕ_i, by the sum of the corresponding values for all customers. These removal costs, along with a penalty term (ϵ) employed if the routes are infeasible, i.e., if the sum of the number of routes found is not m, are periodically updated.

6.3.2 The modified heuristic—*Tabutwin*

The above ideas are all intuitively reasonable, and have worked well in *SOC*. We feel, therefore that these ideas can be used, and next formally describe how we adapt them.

Our first observation is that since the types of the customers are now different, the comparison of relative weights should be within the same type. To indicate this differentiation, we add a δ or a π to the notation of the preceding paragraph and also use $\eta(i,t)$ for the route to which i belongs at iteration t. Thus, e.g., $\mu^k(\delta)$ indicates the maximum change in the recourse cost caused by the removal of a δ customer from this route. We also use ω for the weights. Thus we have $\mu^k(\delta) = \max_{i \in S_k | \phi_{i=1}} r_i^k$, $\mu^k(\pi) = \max_{i \in S_k | \phi_i \neq 1} r_i^k$ and $\omega_i = \sum_{d_i=1}^{K} p_{i,d_i} d_i / \sum_{j|\phi_j=1} \sum_{d_j=1}^{K} p_{j,d_j} d_j$ if i is δ,

and $\omega_i = \sum_{d_i=1}^{K} \phi_i p_{i,d_i} d_i / \sum_{j|\phi_j \neq 1} \sum_{d_j=1}^{K} \phi_j p_{j,d_j} d_j$ otherwise, $\forall k \in \{1, ..., m\}$. Then, if $\eta(i,t) = \eta(j,t)$,

$$\Delta(i,j,t) = A([j^-]^t, i, j) - A([i^-]^t, i, [i^+]^t) \tag{6.3.1}$$

and, otherwise,

$$\Delta(i,j,t) = A([j^-]^t, i, j) - A([i^-]^t, i, [i^+]^t) + \mu^{\eta(i,t)}(\cdot)\omega_i - r_i^{\eta(i,t)} \tag{6.3.2}$$

where Other terms used are inf for the number of infeasible solutions in the last 10 iterations, $\bar{F}$ for the approximate change in cost caused by a candidate move (as computed by the use of the proxies), F for the *actual* cost of a set of routes (as computed by the equations of section 6.2), F^* for the best value of F observed so far (feasible or not), and where

$$\bar{F}(i,j,t) = \Delta(i,j,t) + \epsilon\{1/(|S^t(j)| + 1) - 1/|S^t(i)|\} \tag{6.3.3}$$

with the weighting of ϵ being designed to encourage moves to larger routes.

Our second point of departure from *Tabustoch* lies in the use of penalties. GEA [25] use additional penalty functions to weight F in case of infeasibility. We do not, since we feel that our use of W, OL and SL will be sufficient to drive the heuristic to feasibility; we thus use ϵ only for $\bar{F}$. In particular, the differences lie in that *Tabutwin* (a) evaluates the consequences of both options "0" and "1" for each route, (b) excludes any route the total distance of which goes beyond SL and (c) multiplies $(\tau(\cdot) - OL)$ by an overtime factor (otf).

Next, the best $TEST$ number of solutions are obtained through proxies at each iteration in both heuristics, but whereas GEA [25] use $\bar{F}$ for a first stage screening of these $TEST$ solutions and then determine F for only the screened solutions, we compute exactly the cost of each of the $TEST$ solutions. Finally, GEA [25] evaluate the route cost assuming in turn traversal in either direction, but given the existence of TW, such an exercise would be meaningless for *Tabutwin*. (There are

also some differences in the parameters used, which we will point out at the appropriate place, and some differences in the way the heuristics choose candidates, specified as comments in the coded algorithms.)

Keeping in view the resulting complexity of the model, we do not incorporate preventive anticipatory trips to the depot (as was done in Chapter 5) in finding the *a priori* solution—following *Tabutwin* in this particular aspect. (Such trips are, however, incorporated when we use dynamic techniques to improve the starting solution.) Also, while *Tabutwin*, like its predecessor *Tabustoch*, is based essentially on neighborhood search, the inclusion of W in *Tabutwin* tends to restrict the search space. Hence, each iteration (i.e., finding and evaluating potential solutions) of *Tabutwin* goes through two phases: (a) Find solutions based purely on proximity of nodes, assuming $W = 0$. (b) Include W in evaluating these solutions.

For purposes of the heuristic, m, τ, i and j are all variable and depend upon the particular solution under consideration by the algorithm; the description of *Tabutwin*, below, refers to them accordingly.

Create n to-and-from vehicle routes, single customer on each.

Initialize F^* to the total cost of all routes and T to INFINITY

for(all values of t from 1 to MAX)

for(each i among the u randomly chosen customers)

  for(each j among the v neighbors of i)

    Determine $\bar{F}(i, j, x)$, allowing no π to be earlier than third on a route

Rank the $\bar{F}(.,.,.)$'s obtained in ascending order of costs, retaining no more than 2^*TEST, and with only the first $TEST$ allowed to be Tabu

for(each k of the $k^t \leq 2 * TEST$ best moves obtained at iteration t)

  $F(i(k), j(k), t) \leftarrow \sum_{l=1}^{m(k)} [\tau^l]^*$, where $[\tau^l]^* = \min\{\tau_0^l, \tau_1^l\}$

if best $F(\cdot)$ improves upon F^*, replace F^*

Among the $F(\cdot)$ note best tabu and best non-tabu $F(\cdot)$ separately

if feasible and T-improving, $T \leftarrow$ better of the two

To determine starting x for $t+1$ *do*

if F^* was improved, choose the x corresponding to the best F

else if a non tabu solution is available

Choose the x corresponding to the best F among non-tabu solutions

else if a solution is available at all

Choose the x corresponding to the best among them

else continue with existing x

update inf

if(t mod 10 =0) update ϵ, $\mu_{\cdot}(\cdot)$ and $r_{\cdot}$

end for

GEA [25] start with $\epsilon = 1$ and after every tenth iteration, $\epsilon \leftarrow \epsilon * 2^{inf/5-1}$. We do the same, except that we set some upper and lower limits for the values attainable for ϵ, to prevent ϵ becoming "too high" or "too low" and thereby vitiating the values of $\bar{F}$. We follow GEA [25] in using $TEST = 5$, but whereas GEA [25] use $v = 5$, we use $v = 6$, because of the extra complications introduced by TW; we also let u vary instead of using $u = 5m$ as recommended by GEA [25]. (More details on this aspect are provided in Section 5.7.) Note that the complexity of computing removal cost of nodes is $O(n^2KQ^2)$, so the updates (last line of the algorithm above) are carried out only once every 10 iterations.

6.4 Validating the heuristic

As mentioned earlier, GEA [25] compared some of the *Tabustoch* solutions with their earlier exact solutions, but since we cannot do the same, we must test *Tabutwin* solutions with those obtained by using other prevalent methods.

6.4.1 A base for comparing *Tabutwin*

A simple heuristic for the (deterministic) VRP is the *CW*. This has since been supplanted (certainly in academia, if not in the field) by more sophisticated heuristics, in particular, by the so-called "meta-heuristics". The latter in general provide superior (5-6 % lower cost) solutions. The (relative) simplicity and low running times of *CW*, mean, however, that it is still widely used in practice. (Meta-heuristics involve considerably greater time—in some cases, even 1000 times more—for details, please see Cordeau et al. [54].) We therefore use *CW* solutions as a base to compare with *Tabutwin* solutions.

Since *CW* needs actual demand values rather than probabilities, we must resort to simulation. Specifically, this would involve generating a set of customer demands assuming a probability distribution for the demands. Next, the costs of traversing the set of routes obtained via *CW* with the generated set of demands would be compared with those incurred in following the corresponding routes generated by *Tabutwin*.

A second problem with using *CW* is that it cannot handle two types of operations. We solve this problem in the following way. First, we apply *CW* to the set of δ customers. After *CW* has generated a set of routes for these customers, we add the π's to the routes generated, by using the technique "Cheapest Insertion" (*CI*). Essentially, this procedure works as follows: Suppose k routes have been generated in all and the customer currently at the end of route k is i_k. Then, iteratively for each remaining j in the list of π customers, we make the (i_k, j) connection that minimizes increase in total cost of the routes. (See Cook et al. [47] for more details about the procedure). Algorithmically,

Determine *CW* routes for the δ customers.
while(a π customer awaits accommodation in a route) *do*
  for(each such π customer j) *do*
    for(each route k) *do*

Determine increase in total route cost if j is connected to the customer currently at end of route k, disallowing the connection if the cost of k goes beyond SL

Choose the connection that gives the minimum increase in cost and update the lists

end while

The insertion of pickups at route-ends follows the practice of the majority of researchers on VRPB. To our knowledge, the *a priori* routes followed in practice usually do not make specific provision for the πs; the presence of TW therefore suggests that an algorithm similar in spirit would be the only practicable alternative to the vehicle controller. Thus while our route-construction procedure might not look like a sophisticated way to accommodate the πs, it probably would not be greatly inferior to current logistics practice in general.

For the corresponding routes generated by *Tabutwin*, since customer presence is completely known at W, we compute the $q_d^*(\cdot., \cdot)$, valid once W has elapsed, at all customers on the *actually realized* route, as well as at those on the *a priori* route (without affecting order of running time). Essentially, at any customer on any route, we take action according to the greedy strategy as stated in section 6.2, substituting the actual for the *a priori* route at W, and using the appropriate $q_d^*(\cdot., \cdot)$ to re-start upon route failure, depending upon whether the route is in the pre-W or the post-W phase. (If a route has no π, there is no distinction.) A detailed algorithmic description of the procedure is provided in Appendix B.1).

6.4.2 Choice of "external" parameters

The next problem is setting W, OL and OL, ie., the "external" parameters, appropriately. We assume an exact correspondence between time and distance, i.e., (1 time unit=1 distance unit). We can do this WLOG since any multiplicative factor would maintain intact the relative lengths of the routes. At this point, we note that with no constraints except those of vehicle capacity (i.e., with no constraints on the expected length of an individual route), while we would no doubt obtain the least *total* expected route length, there would be nothing to prevent an individual route

becoming "too long" or "too short" in terms of constraints like OL or SL. These constraints enforce some sort of "balance" between the routes by reducing the chances of unduly long or short routes.

Next, we note that the minimum route length corresponding to a set of nodes would be given by a TSP solution on this set. Hence, we first apply procedure LK to N, assuming $\phi_i = 1$ for each i. The solution represents a lower bound on total distance, at least within the limitations of LK. Introducing the actual ϕ_i's, we next compute the dc of the sequence provided by LK—a better lower bound for the *expected* route length.

We combine the ideas of the preceding two paragraphs as follows. We multiply dc by M/m with M allowed to vary. Intuitively, division by m corresponds to splitting the total distance equally among the routes, and multiplication by M then simulates the increase in length due to the constraints (vehicle capacity as well as external constraints). Also, given that *Tabutwin* is not guaranteed to find a solution to every problem instance within a given computational time (nor, for that matter, is *Tabustoch*), we perform more than 10 (our chosen number of) replications (i.e., we consider more than 10 problem instances) to ensure that we have a reasonable number of sets of routes. M is then finally chosen on the basis of its efficacy in producing valid routes and M/m is taken as OL.

It is clear that for a given problem instance, better ways of determining an appropriate OL could be found, but our problem is that we must perform all our experiments in an identical setting in order to avail the benefits of simulation packages—hence the need for this roundabout estimation procedure.

W and SL can next be obtained by multiplying OL by suitable factors *twf* and *slf* respectively. We chose $twf = 0.4$ and $slf = 1.2$. While these figures are based on our discussions with practitioners, we do not claim any sanctity for them, and the reader is free to think of them as arbitrary (though reasonable). We note also that since CW does not allow route failure, we cannot incorporate into it the constraint of exactly m vehicle routes and we would expect to get CW solutions providing more routes than m, with a CW route being longer in length than a *Tabutwin* route on average. The choice of *twf* entails also the additional consideration that a higher value for

it would add extra distance to the individual CW routes (since W would not have elapsed when we wanted to add the π's to the former). Since we are interested in testing *Tabutwin* severely, this would not do at all!

As regards the portion of the distance traversed that falls between SL and OL, the choice is straightforward—we multiply it by $otf = 1.5$, the accepted industry standard where applicable. We note again that, given the discussion above, the likelihood of a CW route incurring overtime (OT) is not very high—another indication of *Tabutwin*'s comparative disadvantage.

6.5 The dynamic improvement procedure

In this section we focus on using DP/*rollout* techniques to improve the *a priori* routes. Since a given route is independent of the others, these techniques can be applied to each of the routes individually as in Chapter 5, and this is our objective in this section. While the methodology of Chapter 5 remains relevant, the corresponding equations and the algorithm require modifications owing to the differences in assumptions. The latter (along with their consequences) can be summed up as follows.

(a) Action is now conditional on q_d with one type of customer, and on q_a with the other. Accordingly, the $\gamma(\cdot)$ of Chapter 5 now has two arguments, q_t and q_d. (The superscript on γ—then as now—refers to action choice, i.e. "0" for going to the next customer via the depot and "1" for going there directly, and *not* to the route.) As in the earlier sections of this chapter, our subsequent analysis is understood to refer to any of the set of routes.

(b) *Rollout* does not incorporate stochasticity in customer presence. While it is possible in principle to modify the equations to take this factor into account, it would complicate the procedure much more, and, moreover, many of the options would not be viable because if the algorithm, in the earlier stages of a run, chose a π as the element to be visited next, one would have to wait until W either at the current or at the next customer. To avoid these contingencies, we will apply

rollout only after W. (We will see later that the utility of DP/*rollout* is in evidence even with these constraints.)

(c) At the customer (ι) at which *rollout* is being performed, dc and rc would have to be determined separately for each choice of $j \in Y_\iota$ as a possible successor to ι. Specifically, if revisiting ι becomes necessary, the optimal delivery quantity q_d^* with which ι must be revisited must be computed separately for each such choice of j. This additional computation is not necessary while evaluating route costs at any other node on the route yet to be traversed, since, from (b) above, the successor (and hence q_d^*) at that node would already be known precisely.

Denoting, then, by $\gamma_i(q_t, q_d)$ the cost to go if (q_t, q_d) is the state of nature *after* fully serving i, by s the customer immediately following $i \neq \iota$, if $s = 0$ we have $\gamma_i(q_t, q_d) = c_{0i}$ for all pairs (q_t, q_d). If $s \neq 0$, $\gamma_i(\cdot)$ is determined according to the minimization

$$\gamma_i(\cdot) = \min\{\gamma_i^0(\cdot), \gamma_i^1(\cdot)\} \tag{6.5.1}$$

where, respectively for the cases with s a δ or a π, $\gamma_i^1(\cdot)$ is given by

$$\gamma_i^1(q_t, q_d) = c_{is} + \sum_{q=0}^{q_d} \gamma_s(q_t - q, q_d - q)p_{s,q} + \sum_{q=q_d+1}^{d_s^{\max}} [2c_{0s} + \gamma_s(q_d^*(s, v), q_d^*(s, v))]p_{s,q} \tag{6.5.2}$$

$$\gamma_i^1(q_t, q_d) = c_{is} + \sum_{q=0}^{q_a} \gamma_s(q_t + q, q_d)p_{s,q} + \sum_{q=q_a+1}^{d_s^{\max}} [2c_{0s} + \gamma_s(q_d^*(s, v) + u, q_d^*(s, v))]p_{s,q} \tag{6.5.3}$$

where $v = Q + q_d - q$ if s is a δ and $v = Q + q_a - q$, $u = q - q_a$, if s is a π.
Similarly, the corresponding equations for the action 0 are

$$\gamma_i^0(q_t, q_d) = c_{0i} + c_{0s} + \sum_{q=0}^{d_s^{\max}} \gamma_s(q_d^*(Q) - q, q_d^*(Q) - q)p_{s,q} \tag{6.5.4}$$

and

$$\gamma_i^0(q_t, q_d) = c_{0i} + c_{0s} + \sum_{q=0}^{d_s^{\max}} \gamma_s(q_d^*(s, Q) + q, q_d^*(Q)) p_{s,q} \tag{6.5.5}$$

We note that, as opposed to its use in Section 6.2, "0" now denotes a preventive, rather than a forced, trip to the depot, and hence $q_d^*(j, Q)$ is computed somewhat differently. Specifically, since a preventive trip to the depot can always be made at s, the domain of $q_d^*(s, Q)$ does not include those values of q_d that would require revisiting s. Thus, respectively for the cases that s is δ or π, we have

$$q_d^*(s, Q) = \underset{q \in \{d_s^{\max}, Q\}}{\arg\min} \sum_{d_s = d_s^{\min}}^{d_s^{\max}} p_{s,d_s} \gamma_s(q - d_s, q - d_s) \tag{6.5.6}$$

$$q_d^*(s, Q) = \underset{q \in \{0, Q - d_s^{\max}\}}{\arg\min} \sum_{d_s = d_s^{\min}}^{d_s^{\max}} p_{s,d_s} \gamma_s(q + d_s, q) \tag{6.5.7}$$

Then, in light of observation (c) above, we have

PROPOSITION 6.5.1. *If ι is revisited, $q_d^*(j, u) = \underset{q \in \{l, u\}}{\arg\min} \sum_{d_j = d_j^{\min}}^{d_j^{\max}} p_{j,d_j} \gamma_j(w + q - d_j, q - d_j)$ if ι is a δ and $q_d^*(j, u) = \underset{q \in \{l, u\}}{\arg\min} \sum_{d_j = d_j^{\min}}^{d_j^{\max}} p_{j,d_j} \gamma_j(w + q + d_j, q)$ if ι is a π, where, if ι and j are respectively of type δ, δ; δ, π; π, δ; and π, π; we have $w = 0, l = d_j^{\max}, u = Q + q_d - d_\iota$; $w = 0, l = 0, u = \min\{Q + q_d - d_\iota, Q - d_j^{\max}\}$; $w = q_d + q_p + d_\iota - Q, l = d_j^{\max}, u = Q - w$; and $w = q_d + q_p + d_\iota - Q, l = 0, u = Q - w - d_j^{\max}$;*

Proof. The expressions are obtained by establishing the domain of the values of $q_d^*(\cdot)$ given the state of nature at ι and a particular realization of D_ι assuming that revisits at j are excluded from consideration.

We recall that we conceive the remaining customers at any stage as arranged in a cycle, that the traversal of this cycle can be started at any point on it, and that *rollout* chooses the optimal

starting point. Defining, then, L_j as the expected distance traveled if traversal starts with customer j, we have

$$L_j = \gamma_j(q_t, q_d) \tag{6.5.8}$$

with (q_t, q_d) the state of nature after fully serving j, and with the understanding that traversal continues till the last remaining customer is fully served. The next customer to be visited is then the one that minimizes $\min\{E^1_\iota(q_t, q_d), E^0_\iota(q_t, q_d)\}$ where $E^1_\iota(\cdot)$ and $E^0_\iota(\cdot)$ represent the optimal distance in case the action 1 or 0, respectively, is chosen at ι and the minimization is performed over Y_ι. Depending upon the type of $j \in Y_\iota$ under consideration, if ι does not require a second visit, the appropriate equations are on the lines of (6.5.2) through (6.5.5). Thus, eg., if j is π, we have

$$E^1_\iota(q_t, q_d) = \min_{j \in Y_\iota} \Big\{ c_{\iota j} + \sum_{q=0}^{q_a} p_{jq} L_j(q_t + q, q_d) + \sum_{q=q_a+1}^{d_j^{\max}} p_{jq} [2c_{0j} + L_j(q^*_d(j, v) + u, q^*_d(j, v))] \Big\} \tag{6.5.9}$$

and

$$E^0_\iota(q_t, q_d) = \min_{j \in Y_\iota} \Big\{ c_{0\iota} + c_{0j} + \sum_{q=0}^{d_j^{\max}} L_j(q^*_d(j, Q) + q, q^*_d(j, Q)) \Big\} \tag{6.5.10}$$

with v and u defined as in (6.5.3). If, on the other hand, ι requires a revisit, the choice of the next customer minimizes $E^1_\iota(\cdot)$ after ι is revisited, and we have, if ι is δ,

$$E^1_\iota(q_t, q_d) = 2c_{0\iota} + \min_{j \in Y_\iota} \Big\{ c_{\iota j} + \sum_{q=0}^{d_j^{\max}} p_{jq} L_j(q^*_d(j, u) - q, q^*_d(j, u) - q) \Big\} \tag{6.5.11}$$

and otherwise,

$$E^1_\iota(q_t, q_d) = 2c_{0\iota} + \min_{j \in Y_\iota} \Big\{ c_{\iota j} + \sum_{q=0}^{d_j^{\max}} p_{jq} L_j(q^*_d(j, u) + q, q^*_d(j, u)) \Big\} \tag{6.5.12}$$

with $q_d^*(j,u)$ defined as in Proposition 6.5.1. With the equations set up, we now apply *rollout*. The procedure is analogous to that used in generating the actual *Tabutwin* routes (Section 6.4, Appendix B.1). In brief, (a) routes that have no π undergo *rollout* starting with the first customer, and (b) for the others, *rollout* commences when W has been traveled. Thus, before rollout has started, we follow the greedy strategy of Section 6.2 and afterwards, at any customer, we perform *rollout* and proceed to the next customer as decided by equations (6.5.11) and (6.5.12)—or their counterparts. It is understood that the occurrence of W coincides with the substitution of the *a priori* route with the actual one, and it is on the latter that *rollout* is performed. We apply both *DR* and *RR* to *BH*(*Tabutwin*) and bound the running time of our procedure as follows.

LEMMA 6.5.1. *The time complexity of* DR *or* RR *applied to* Tabutwin *is* $O(n^3KQ^2)$

Proof. In section 6.2 we established the time complexity of determining the length of a single route. Neither *DR* nor *RR* requires the computation of *dc* and can be performed at most n times on at most n customers each time, the lemma follows.

As in Chapter 5, the running time of *LK* is not included in the computational complexity given by the above lemma, but as we saw earlier, *LK* roughly adds only linear time. We also have, analogous to Chapter 5, thresholds for the action "0", as follows

LEMMA 6.5.2. *At any customer, for given* q_t, $\gamma_i(q_t, q_d)$ *is non-increasing in* q_d

Proof. If $i+1$ is a δ, clearly if $q_d \geq d_i$ both actions 0 and 1 will always be possible, otherwise not. If $i+1$ is a π, only q_t matters, and γ cannot increase with q_d.
Similar reasoning gives us

LEMMA 6.5.3. *At any customer, for given* q_d, $\gamma_i(q_t, q_d)$ *is non-decreasing in* q_t

The last two lemmas can then be taken advantage of during computations at any given customer, in deciding for which values of q_d the action "0" is optimal for given q_t.

6.6 Dynamic improvement, route interaction allowed

As pointed out in the introductory section of this chapter, the current section is concerned with dynamic improvements based on a consideration of more than one route simultaneously, rather than improvements that consider each route in isolation, as in the previous section. Also, we focus on only one type of "simultaneous reoptimization"—namely, the reduction of OT, as we elaborate in the following.

The routes designed by a company would presumably seek to utilize a driver's time optimally. Such routes would have expected route distance close to the distance a driver would travel within his stipulated time. (It would be reasonable to assume that the criterion adopted would give greater weight to expected, rather than to maximum possible route distance; otherwise, too many drivers would have to be hired.) It would then be quite possible for a vehicle to encounter "heavy" demands and make more than the expected number of trips to the depot before completing its route, thus causing, at best, delay in customer service and in more severe cases, overtime payments or even vehicle shut-downs, as explained in Chapter 1.

If, simultaneously, another vehicle encounters "low" demands and quickly finishes its route, it would be helpful to use the second vehicle for assistance on the first route. While it is evident that doing this would in general not reduce the total distance traveled, the strategy might well provide lower overall costs by seeking to minimize drivers' overtime (OT), rates for which are typically much higher than fuel costs or normal wages. Our procedure of Section 6.5, while it might improve each route individually, would fail to provide such assistance to a "distressed" route, and needs augmentation in order to be able to do so.

Such augmentation could be done in a number of ways, e.g.: (a) The vehicle sent to a distressed route would not necessarily be among the ones that are currently free—we could evaluate also the consequences of using a vehicle that would become free later, but is potentially a better choice on the basis of proximity to the affected route and/or larger remaining capacity. (b) The starting and/or ending points of, and/or number of customers to be served by, the vehicle being

sent as relief could be decision variables. (c) A distressed route could be plied by a number of vehicles (if more than one had become free) and there could be a number of interchanges of vehicles between the routes, the procedure performing, so to say, some sort of dynamic re-optimization on the entire set of customers.

Keeping in view, however, the programming complexity involved in the above schemes, we will restrict ourselves to simpler alternatives. Specifically, we will assume that:

1. At any given time, a single route can be worked on by at most two vehicles.
2. An "SOS" can be sent by a distressed route (or initiated by the central dispatcher) only for the last customer on any such route.
3. A vehicle can be sent as relief only after it has completed its original route.
4. The process of assistance is initiated at the first instant that both a "donor" and an "acceptor" are available.
5. If computations show possible benefits (i.e. reduction in expected OT) assistance must commence immediately—we do not assess if a vehicle becoming free later could provide greater benefits.
6. After a vehicle has been sent as relief, no subsequent SOS from any other route is honored.
7. When a vehicle becomes free (i.e. it has completely served the last customer on its original route) it waits at that customer till the time that it would be possible for it to return to the depot without incurring OT (unless its services are requisitioned earlier).

A problem with our strategy is the following. During the determination of $q_d^*(\cdot,\cdot)$, in case of a tie, the minimum q_d is chosen. Thus, if only customer i, a δ, remains to be visited and the vehicle is starting from the depot, $q_d^*(i,Q)$ would be d_i^{max}, even though, for instance, $q_d^*(i,Q) = Q$ would give the same results. This makes sense from a practical standpoint (the vehicle carrying the minimum possible load), but for our current purpose this might be a disadvantage. We therefore

add the following amendment—if at the last customer i on its original route, the vehicle needs to revisit i, it comes back from the depot with a quantity such that after service is completed, $q_d = K$. (Since in practice, as also in our computational experiments, $Q \geq 2K$, this is always possible.) Note that this does not affect our earlier algorithms in any way.

Our procedure for "dynamic route hopping" (DRH) follows the basic steps of DR, with the difference that we keep track of potential "donors" and "acceptors" and allow the donor vehicles to move to acceptor routes if the move is potentially beneficial. Thus, rather than tracking the total costs of each route *per se*, we check only whether DRH reduces the cumulative OT of the routes. (For a detailed algorithmic description of the procedure, see Appendix B.2.) Clearly, from an expenditure point of view, if OL is not exceeded, the exact distance traveled on a route does not matter very much (with our assumption about fuel costs, above). Thus, while actual distances traveled might go up, the company would still gain from an overall cost viewpoint.

The focus of DRH is thus different from the approach in the previous section, where it was distance minimization *per se*, and the two would thus be applicable in somewhat different contexts. The applicability of each would of course be decided by management. Also, the approaches might or might not be complementary; we assume that they are, and apply DRH to the results of DR (we do not use RR due to programming complexity).

While DRH could obviously be improved by adopting some of the more complex schemes mentioned above, our intended demonstration of its utility in lowering costs could still be undertaken with this simpler version, since the principle remains the same. Since—as we show in our experiments—such a dynamic approach can be applied in real-time, the availability of relatively inexpensive software sustains our hope that the approach would find acceptance among at least the "forward-looking" companies, and that the more complex schemes would also be tried out sooner or later.

6.7 Computational experiments

Our computational experiments broadly follow the same pattern as in Chapter 5. (To a significant extent, so does our terminology.) There still are a number of replications (problem instances) for each $(n, Q, \bar{f}')$ combination. We choose $m \in \{2, 4, 6, 8\}$, extending GEA [25]'s choice of $m = 2$. We choose n with a view to having roughly "balanced" routes on any problem instance. Thus, corresponding respectively to the values $\{m = 2, 4, 6, 8\}$, we assume the values $\{20, 40, 60, 80\}$ for the total numbers of δ customers (so that every route would have 10 δs on an average). Now adding four π customers per route, we arrive finally at the figures of $n \in \{28, 56, 84, 112\}$ respectively.

A word on the relative proportions of δs and πs: in the three case studies mentioned in Casco et al. [64] (Section 6.1), the percentage of pickups was approximately 10-30%, 50%, and 10-20%, averaging about 28%. (The corresponding figure is below 20% in "our" company.) While our experiments attempt to replicate these proportions, we allow more randomness and do not fix the exact *numbers* of δs and πs; instead, we fix only their relative proportions. Thus, we generate pickup customers on the assumption that Pr{customer is π}= 2/7, maintaining (on an average) the relative proportions of Casco et al. [64], as well as equity across routes. The ϕ_is are generated from a uniform distribution, and customers are located randomly in the unit square (centered on the depot, which is positioned at (0,0)). The demand distributions for both δs and πs are chosen in the same way as they were for a customer in Chapter 5. Dedicated random number streams are used for generating the different parameters.

For obtaining suitable values of Q for our experiments, we start, as in chapter 5, with $\bar{f}'$, but now restrict ourselves to the three smaller values, i.e., {0.75, 1.25 and 1.75}. The reader may recall that corresponding to these values of $\bar{f}'$, $Q = \{91, 71, 58\}$ for $n = 20$ in Chapter 5. Since we have 10 δ's per route on an average, these values now translate roughly to the values of 46, 36 and 29 respectively for Q for a *single* vehicle, for all choices of m (rounding up where necessary). By our assumption that the first customer on a route is never a δ, and the lower proportion of

πs as compared to δs, the additional space created on the vehicle as deliveries are effected would plausibly suffice to accommodate the πs, and as such, $\bar{f}'$ would remain roughly the same as before. Since some of our data structures assume a minimum size of 30(=$2K$), the value $Q = 29$ is revised upward to 30.

As indicated in Section 6.3, we were not able to obtain 10 sets of routes for each $(n, Q, \bar{f}')$ combination; we include the number of replications (R) carried out (i.e., problem instances considered) under each category in Table 6.2. Also, as stated in Section 6.3, we allowed the number of customers chosen as "seeds" in an iteration of *Tabutwin* (i.e., u in Section 6.3) to vary, rather than keeping this number constant at $5m$, as was done by GEA [25]. Specifically, for $(n, Q) = \{(112, 36), (112, 30), (56, 36)\}$, the numbers of seeds used were 35, 33 and 15 respectively. (Our trials with different values for u seem to indicate that the choice of u is an aspect that also requires further research.) All our algorithms weight $\tau(\cdot) - OL$ by *otf* and no algorithm allows a route length to exceed SL.

We limited the CPU time allowed to *Tabutwin* by fixing the number of iterations it was allowed to perform for each problem instance considered. Roughly, the number of iterations were chosen to ensure that the running time in finding solutions for an (n, Q) combination would not exceed two hours. (This time limit was exceeded in only two cases-Table 6.2.) *Tabutwin* found solutions in 70.5% of the problem instances considered.

Table *Tabutwin* summarizes the first phase of the experiments. The initial columns of this table show the number of "fruitful" replications R (i.e., the number of problem instances in which a solution was obtained by *Tabutwin*) for each (n, Q) category, along with the corresponding values of M. We next tabulate the average expected distances as determined by *Tabutwin* for each category (i.e., the averages over R replications). Finally, for given realizations of $\phi(\cdot)$ and D_i, the table compares the distances of the routes generated in following the *Tabutwin* sequence, with (a) the expected *Tabutwin* distances and (b) the corresponding routes generated *via* CW+CI. (The vehicles required, i.e., the number of routes generated under CW+CI are also shown.) As in chapter 5, the averages over 200 iterations (i.e., realizations of ϕ_i and D_i) serve as "proxies" for

the corresponding expected distances in a replication. All our inter-policy comparisons, similarly, are carried out by comparing the average distances across all replications policy-wise. We observe

Table 6.2.
Generation and validation of routes

			total distance			rts. used		% diff of actual			time
	R	M	TT		CW	TT	CW	*vs. TT*	*vs. CW*		mins.
(n, Q)			expected	actual					dist	rts	TT
(28, 46)	10	1.35	6.643	6.965	7.262	2	4.51	4.63	-4.26	126	93.4
(28, 36)	10	1.45	7.386	7.634	7.529	2	5.55	3.25	1.38	178	59.6
(28, 30)	10	1.55	8.336	8.733	8.194	2	6.60	4.55	6.17	230	41.3
(56, 46)	10	1.6	11.748	11.884	12.503	4	7.76	1.15	-5.21	94	135
(56, 36)	9	1.8	14.112	14.222	13.315	4	9.77	0.78	6.38	144	74.4
(56, 30)	9	1.94	15.885	16.420	14.652	4	11.81	3.26	10.77	195	59.2
(84, 46)	10	1.8	16.900	17.052	16.218	6	10.90	0.89	4.89	82	148
(84, 36)	8	2.01	19.459	19.702	17.293	6	13.97	1.23	12.23	133	84.2
(84, 30)	8	2.22	22.194	22.710	19.469	6	16.46	2.27	14.27	174	61.5
(112, 46)	9	2.11	23.216	23.718	20.825	8	14.57	2.12	12.20	160	113
(112, 36)	9	2.4	27.348	28.124	23.923	8	19.41	2.76	14.94	143	77.1
(112, 30)	8	2.6	32.149	32.905	25.508	8	23.25	2.30	20.19	191	62.4
Average								2.43	7.83	154	

that:

- The difference between the expected distances predicted by *Tabutwin* and the expected distances as obtained by averaging the "actual" distances traversed appear reasonable in view of the large number of random elements involved.

- Keeping in view the much larger number of vehicles used by CW (on an average 2.54 times those used in traversing the $Tabutwin$ routes), the average increase in total route distances (of only 7.83%) under the latter seem to indicate *Tabutwin*'s efficacy.

- *Tabutwin* appears to do relatively better for lower values of $\bar{f}'$, which is intuitive, since a higher $\bar{f}'$ implies a larger number of trips to the depot by the *same* vehicles, whereas CW is always at liberty to use a fresh vehicle.

In the second, or dynamic, phase of our experiments, we first compare the distances traversed under *Tabutwin* with the distances that result if the dynamic improvement techniques, i.e., DR or RR, are incorporated. Next, we augment DR with DRH. As indicated in Section 6.6, while using DRH, we compute only the savings in OT. Specifically, in any iteration, the percentage improvement (i.e., reduction) in OT is taken on the base of the total OT for that iteration, considering all routes, that would have resulted if the routes had been determined with DR incorporated in the procedure, but not DRH. Computational times for DRH are not indicated—in any case, by the nature of the procedure used, these cannot exceed the computational time of the corresponding run under DR. Table 6.3 summarizes these experiments ("succ" in this table stands for "successful", i.e., that the redeployment of the vehicle resulted in a reduction of OT). We observe that:

- The improvements brought about by the two versions of *rollout* are encouraging, and indicate that the efficacy of such dynamic techniques is not restricted to SOC.

- While RR does better than DR as expected, the differences between the two do not appear as significant as in Chapter 5. However, it must be remembered that *rollout* is now being performed over relatively smaller sets of nodes and hence the chances of RR finding better node sequences *vis-a-vis* DR are reduced.

- Even with the restrictions imposed on the applicability of DRH, both the number of "assistance programs" initiated, and the improvements obtained, indicate that "simultaneous reoptimization" could usefully be applied in tandem with "individual reoptimization".

Table 6.3.
Dynamic improvement of routes

	expected distance			% improvement		*DRH*			time	
				vs. act		rts. changed		% impr, OT	sec/iter	
(n, Q, m)	act	*DR*	*RR*	*DR*	*RR*	total	succ	*vs. DR*	*DR*	*RR*
(28, 46, 2)	6.965	6.731	6.729	3.36	3.39	3	3	53.49	< 0.1	< 0.1
(28, 36, 2)	7.634	7.283	7.283	4.60	4.60	14	14	49.23	< 0.1	< 0.1
(28, 30, 2)	8.733	8.193	8.189	6.18	6.23	8	8	26.74	< 0.1	< 0.1
(56, 46, 4)	11.884	11.706	11.680	1.50	1.72	99	96	17.80	0.1	0.1
(56, 36, 4)	14.222	13.663	13.604	3.93	4.35	60	57	31.42	< 0.1	0.1
(56, 30, 4)	16.420	15.275	15.213	6.97	7.35	25	25	11.93	< 0.1	< 0.1
(84, 46, 6)	17.052	16.600	16.549	2.65	2.95	93	90	10.42	0.2	0.2
(84, 36, 6)	19.702	18.916	18.843	3.99	4.36	38	37	6.15	0.1	0.1
(84, 30, 6)	22.710	21.489	21.410	5.37	5.72	70	68	9.33	0.1	0.1
(112, 46, 8)	23.718	22.980	22.952	3.11	3.23	136	133	11.01	0.2	0.3
(112, 36, 8)	28.124	26.708	26.615	5.03	5.37	97	92	5.37	0.1	0.2
(112, 30, 8)	32.905	30.708	30.531	6.68	7.22	74	71	10.64	0.1	0.1
Average				4.45	4.71					

- The computational times appear viable.

Chapter 7

Conclusion

Our first section in this chapter outlines the contributions made by the current work. The second section lays out areas where further research might be fruitful.

7.1 Contributions

In Chapter 3,

- Our description of the properties of a comprehensive MDP model for the single-vehicle VRPSD represents the first attempt to describe exactly these properties.
- We also developed a recursive procedure to compute solutions to VRPSD based on this model, using four different operative policies for serving customers. Our computational experiments establish the current computational limits of exact solution approaches to single-vehicle VRPSD *via* MDP.

In Chapter 4,

- We modified the heuristic with the best-known approximation ratio of 2.5 *vis-a-vis* the optimal solution for the multi-vehicle VRP to obtain strongly polynomial computational time bounds with the same approximation ratio.
- We extended our heuristic to the single-vehicle VRPSD, obtaining improved bounds as regards both solution value and computational time, compared to the best bounds currently known.
- We applied the primal-dual methodology to obtain a factor-2 solution to a special case of VRP/VRPSD.

In Chapter 5,

- We amended the only existing dynamic solution procedure to VRPSD, applying the augmented procedure to a larger set of problems than considered in the motivating work. We obtained a percentage quantum of improvement *vis-a-vis* the same *a-priori* solution that more than doubles the improvement obtainable by using the earlier dynamic approach, with computation time on the same order.

- We provided an alternative (and simpler) proof of the "improving" nature of such dynamic procedures.

- We applied the augmented procedure to a larger set of problems than considered in the motivating work, and obtained a percentage quantum of improvement *vis-a-vis* the same *a-priori* solution that more than doubles the improvement obtainable by using the earlier dynamic approach, with computation time on the same order.

- Additionally, we developed an algorithm to improve the *a-priori* solution dynamically by using an exact MDP technique, that results in lower-cost solutions than obtainable through the use of approximations alone. The computational experiments based on this algorithm showed that the approach is computationally viable.

In Chapter 6,

- We incorporated both pickup and delivery in the context of SVRP.

- We incorporated time windows and limits on route lengths in the context of VRPSCD.

- We modified an existing heuristic to obtain *a-priori* solutions to VRPSCDB.

We validated our heuristic *Tabutwin* by comparing its solutions with those obtained *via* techniques that we believe simulate existing industry practice. The quality of *Tabutwin* solutions compares favorably with the solutions obtained *via* use of the latter. We also developed polynomial-time computational refinements that allow dynamic updates to the obtained vehicle routes as

new information about customer presence and demands becomes available by (a) changing the sequencing of customers on a route based on current information and (b) allowing a vehicle to move from one route to another if this move helps in curtailing the cumulative overtime (OT) of the two routes. The quantum of dynamic improvements is encouraging, and the time actually observed on the computations appears reasonable.

7.2 Future research

We list below, chapter-wise, the areas where we hope our work will motivate further research.

In Chapter 3,

- Computational experiments with different demand distributions and with larger values for the number of customers and vehicle capacities, that could enable more general conclusions to be drawn about the quantum of improvements that exact policies are capable of providing.

Keeping in view the exponential nature of the exact algorithms, the feasibility of performing such experiments would be strongly linked to the availability of adequate computational power. However, the consistent improvements in CPU speeds indicate that such experiments should be possible sooner or later.

In Chapter 4,

- The extension of our network-flow methodology to the general multi-vehicle VRP, (i.e., one where customers are allowed to have arbitrary demands).

In Chapter 5,

- As observed by several researchers, e.g. Dror et al. [11], Savelsbergh et al. [51], route *direction* also matters in VRPSD (e.g., following the sequence 1-2-3 might not lead to the same expected costs as following 3-2-1). Thus, we could evaluate also the cost of the candidate routes assuming reverse traversal. This would however, essentially double the running time, and we have therefore not considered such traversal. As more computational power becomes available, however, examination of this alternative could also be included in the search procedure.

- Since the objective of *rollout* techniques is to find improving, rather than optimal solutions, it might not be necessary to evaluate all the possible options in each iteration. We could thus consider a subset of the neighbors of a node that had been finalized in the immediately preceding iteration, rather than the entire set, chosen according to a suitable measure of proximity. Since, however, our running times appear reasonable, we have not explored this line of research further. On the other hand, if n is significantly larger than in our experiments, determining such subsets might well become a meaningful exercise.

In Chapter 6,

- As mentioned in the body of this chapter, we have considered only a small number of the different ways in which re-optimization across routes could be effected in a multi-vehicle scenario. Exploration of the other possibilities appears to present chances of a greater quantum of improvement in reduction of OT.

- If the time spent on load shuffling is consequential, the additional time spent by the driver in performing this activity could be incorporated either through the use of penalties or by estimating the actual time and adding it (or its distance equivalent) to the solution. A model that incorporates these factors would have greater generality.

Appendix A

MDP Recursion with 3 customers

Our "imbedded" three-customer case has customers y, i and j constituting a sub-system in a larger network. We assume these customers unvisited, and all others fully served, as of epoch t. We also let $q_t < d_y$.

The analysis of the different categories described in Section 3.3 then proceeds as follows.

Case (A)

$q \geq \sigma + \beta$

$$V^3[y, q_t, \delta | Y(\delta) = \{\emptyset\}] = \min_u \left\{ \kappa + \sum_k p_u(k) V^3[u, q_{t+1}, \delta | Y(\delta) = \{y\}] \right\} \tag{A.0.1}$$

where $u \in \{y, i, j\}$. Also, if $u = y$, we have $\kappa = 2c_{0y}$, $q_{t+1} = Q$, and $\delta_y = d_y - q_t$. Otherwise, we have $\kappa = c_{uy}$; $q_{t+1} = Q$ if action $(r, 0)$ is taken at y, and $q_{t+1} = q_t$ otherwise. Clearly, a two-customer subproblem results if either i or j is served; otherwise, we have a three-customer problem with starting quantity Q, a case that has already been considered.

It will also be convenient, in order to avoid ambiguity in the following, to use $V_0^u[(.), (.)]$ to denote the optimal solution to a u-customer subproblem starting at the depot, i.e., to denote $V^u(0, Q, \delta)$. This notation will allow us to explicitly indicate the remaining demand at a customer. Specifically, the first parenthesis in $V_0^u[(.), (.)]$ will denote the remaining demand at customer k, as follows: $d_i + d_j - q$ is denoted by k, $q - d_i$ by $k^{(1)}$ and $d_i - q$ by $k^{(2)}$. The second parenthesis will contain the customers with known demands but *not fulfilled at all*, i.e., customer $i \in Y(\delta) | \delta_i = d_i$. Assuming a starting visit to i, we then have

Case (B)

$\beta \leq q < \sigma + \beta$

The equations involving $F_j^r(q-d_i)$, $F_j^{\overline{r}}(q-d_i)$ and $F_j^{tr}(q-d_i)$ now are as follows

$$V[(i,q,\delta),(r,j)] = c_{ij} + c_{0j} + 2F_j(q-d_i)c_{oy} + \overline{F}_j(q-d_i)V_0^2[(j),\{y\}] \tag{A.0.2}$$

$$V[(i,q,\delta),(\overline{r},j)] = 2c_{ij} + c_{0i} + 2F_j(q-d_i)c_{oy} + \overline{F}_j(q-d_i)V_0^2[(i),\{y\}] \tag{A.0.3}$$

$$V[(i,q,\delta),(r,0)] = c_{0i} + V_0^2[(y),\{\emptyset\}] \tag{A.0.4}$$

With $F_j^{\cdot}(q-d_i)$ and $F_i^{\cdot}(q-d_j)$ known (as earlier, by finding the points of intersections of the above equations), we obtain the optimal action for a given q, and this process can similarly be repeated for each q within the stipulated range.

Case (C)

$$\sigma \le q < \beta$$

If $d_i^{\max} = \sigma$, $V[(i,q,\delta),(r,0)]$ and $V[(i,q,\delta),(r,j)]$ are as in Case (B), and,

$$\begin{aligned} V[(i,q,\delta),(\overline{r},j)] &= (\overline{F}_j(q-d_i) - \overline{F}_j(q))\min\{c_{0i} + c_{ij} + V_0^2[(i),\{y\}], c_{0j} + V_0^2[\{\emptyset\},\{i,y\}]\} \\ &+ c_{ij} + F_j(q-d_i)\left(c_{0i} + c_{ij} + 2c_{oy}\right) + \overline{F}_j(q)\{c_{0j} + V_0^2[\left(j^{(2)}\right),\{i,y\}]\} \end{aligned} \tag{A.0.5}$$

If $d_i^{\max} = \beta$, then if $d_i \le q$, the equations are as in Case (B), and otherwise, we get

$$V[(i,q,\delta),(r,0)] = c_{0i} + V_0^3[\left(i^{(1)}\right),\{y\}] \tag{A.0.6}$$

$$V[(i,q,\delta),(\overline{r},j)] = c_{ij} + \min\left\{c_{0i} + c_{ij} + V_0^2[(i),\{y\}], c_{0j} + V_0^2[\{\emptyset\},\{i,y\}]\right\} \tag{A.0.7}$$

Case (D)

$$0 \le q < \sigma$$

If $d_i \leq q$, the results are as in Case(C) with $d_i^{\max} = \sigma$. Otherwise, we have, $V[(i,q,\delta),(r,0)] = c_{0i} + V_0^3[\left(i^{(1)}\right),\{y\}]$, and,

$$\begin{aligned} V[(i,q,\delta),(\overline{r},j)] &= c_{ij} + F_j(q)\min\{c_{0i} + c_{ij} + V_0^2[(i),\{y\}], c_{0j} + V_0^2[\{\emptyset\},\{i,y\}]\} \\ &+ F_j(q)\{c_{0j} + V_0^3[\left(j^{(2)}\right),\{i,y\}]\} \end{aligned} \tag{A.0.8}$$

The concerning equations would be unwieldy in higher-dimensional cases; however, in principle they could still be obtained as above. However, as noted in Section 3.5, for these cases, we might not have unique points of intersection, so piecewise analysis might become necessary. The same will hold for MDP1, except that the number of functions considered would be larger.

Appendix B

Multi-vehicle algorithms

B.1 Actual *Tabutwin* routes

Let $fnr(\cdot)$ (follow normal rules) denote the action of following the greedy strategy of Section 6.2, where $(\cdot)$ includes "a" (pre-W) and "b" (post-W) phases. Let C^k denote the current cost of route k. Our procedure is then as follows.

Randomly generate actual presence of all π's and demands for all present

Determine $q_d^*(.,.)$ for each customer on each *actual* route, per procedure of Section 6.2

Start route k with $q_d^*(1^k, Q)$, set $i^k \leftarrow 1^k$ and $C^k \leftarrow c_{0,i^k}$, all per *a priori* route.

for each route k, while a customer still requires service *do*

if W has not elapsed at i

if $i+1$ is δ, fnr(a)

else

if best option for this route is 0

if i does not need a repeat visit, add $W - C^k$ to C^k

else $C^k \leftarrow C^k + c_{0i}$, add $W - C^k$ to C^k

fnr(b)

else

if i does not need a repeat visit

$C^k \leftarrow C^k + c_{i,\sigma^k}$, add $W - C^k$ to C^k, thereafter fnr(b)

else

$C^k \leftarrow C^k + c_{0i}$

if $C^k \geq W$, revisit i with $q_d^*(\cdot)$ corresponding to *actual* $i+1$, then fnr(b)

else

Revisit i with $q_d^*(\cdot)$ corresponding to *a priori* $i+1$

if $C^k + c_{0i} < W$, proceed to *a priori* $i+1$, wait there till W if not yet W

else, proceed to *actual* $i+1$

fnr(b)

else

Switch to *fnr*(b) if not already done so, thereafter *fnr*(b)

Update C^k

end *for*

Under option 1, if on arrival at the corresponding location, σ is not found to be present, the algorithm assumes a demand realization of 0, and, if a route has no π, *fnr*(a) is applicable throughout the vehicle's run on that route.

B.2 Dynamic Route Hopping

for each route *do*

if the route has no π determine the first customer through *rollout*

else first customer is the first per *a priori* route

Initialize each route with distance from depot to first and put all distances in list LIST

while a customer still requires service *do*

Sort LIST

$k \leftarrow$ route corresponding to minimum distance in LIST, $curr \leftarrow C^k$

$i \leftarrow$ the current customer on k

if i is last on k and C^k+distance traveled to complete service at i+$c_{0i} < OL$

Put i in donor list

else if i is second last and $C^k + [E_i^{\cdot}(.,.)]^* > OL$

Put i in acceptor list

else

Go to next customer on k per $fnr(\cdot)$ or DR

if donor and acceptor lists are both non-empty

for each donor-acceptor pair

Compute total costs of assistance based on the (q_t, q_d) configuration of the pair and assuming current distance as *curr* for both routes

Choose the pair that minimizes total OT for the two routes considered

if computation indicates assistance process will be viable

Update accordingly and end program

else

Update cost and other parameters of k and continue while loop

References

[1] Charnes, A. and W. Cooper. 1959. Chance Constrained Programming. *Management Science*, 6, 73-79.

[2] Charnes, A. and W. Cooper. 1963. Deterministic equivalents for optimizing and satisficing under chance constraints *Operations Research*, 11, 18-39.

[3] Clarke, G. and J. W. Wright. 1964. Scheduling of Vehicles from a Central Depot to a number of Delivery Points. *Operations Research*, 12, 4, 568-81.

[4] Tillman, Frank A. 1969. The Multiple Terminal Delivery Problem with Probabilistic Demands *Transportation Science*, 3, 3, 192-204.

[5] Golden, Bruce L. and W. R. Stewart, jr. 1977, April 14-15. Vehicle routing with probabilistic demands *Proc of the tenth annual symposium on the interface of computer science and statistics* Gaithersburg, Maryland.

[6] Golden, Bruce L. and James R. Yee. June 1979. A Framework for Probabilistic Vehicle Routing. *AIEE Transactions.*

[7] Stewart, W. R., jr. and Bruce L. Golden. 1983. Stochastic Vehicle Routing: A comprehensive Approach. *European Journal of Operational Research*, 14, 371-385.

[8] Gendreau, M, Gilbert Laporte and Rene Seguin. 1996. Stochastic Vehicle Routing. *European Journal of Operational Research*, 88, 3-12.

[9] Dror, M. and Pierre Trudeau. 1986. Stochastic Vehicle Routing with modified savings algorithm. *European Journal of Operational Research*, 23, 228-235.

[10] Laporte, G. and F. Louveaux. 1990. Formulations and Bounds for the Stochastic CVRP with uncertain Supplies. *Economic Decision Making: Games, Econometrics and Optimization.* J.J.Gabszewicz, J.F. Richard and L.A. Wolsey (Eds.) North-Holland, Amsterdam.

[11] Dror, M., Gilbert Laporte and Pierre Trudeau. 1989. Vehicle Routing with Stochastic Demands: Properties and Solution Frameworks *Transportation Science* 23, 3.

[12] Waters, C.D.J. 1989. Vehicle Scheduling Problems with Uncertainty and Omitted Customers *Journal of the Operational Research Society* 40, 12, 1099-1108.

[13] Jaillet, P. 1987. Stochastic Routing Problems. *Stochastics in Combinatorial Optimization.* G. Andreatta, F. Mason and P. Serafini (Eds.). World Scientific, New Jersey.

[14] Jezequel, A. 1985. Probabilistic Vehicle Routing Problems. M. Sc. Dissertation, Dept of Civil Engineering, MIT, Cambridge.

[15] Jaillet, P. and A. R. Odoni. 1988. The probabilistic Vehicle Routing Problem. *Vehicle Routing: Methods and Studies* B. L. Golden and A. A. Assad (Eds.). North-Holland, Amsterdam, The Netherlands.

[16] Bertsimas, Dimitris J., P. Jaillet and A. R. Odoni. 1990. A priori Optimization. *Operations Research*, 38, 6, 1019-1033.

[17] Bertsimas, Dimitris J. May-June 1992. A Vehicle Routing Problem with Stochastic Demand. *Operations Research*, 40, 3, 574-585.

[18] Laporte, G., F. Louveaux and H. Mercure. 1989. Models and Exact Solutions for a class of Stochastic Location Routing Problems *European Journal of Operational Research*, 39, 71-78.

[19] Bastian, C. and A. H. G. Rinnooy Kan. 1992. The Stochastic Vehicle Routing Problem Revisited. *European Journal of Operational Research*, 56, 407-412.

[20] Teodorovic, D. and G. Pavkovic. 1992. A Simulated Annealing Technique Approach to the VRP in the case of Stochastic Demands *Trans. Planning and Tech.* 16, 261-273.

[21] Bouzaiene-Ayari, B., M. Dror and G. Laporte. 1993. Vehicle Routing with Stochastic Demands and Split Deliveries *Foundations of Computing and Decision Sciences* 18, 2, 63-69.

[22] Laporte, G., F. Louveaux and H. Mercure. 1992. The Vehicle Routing Problem with Stochastic Travel times. *Transportation Science*, 26, 161-170.

[23] Laporte, G. and Francois V. Louveaux. 1993. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* 13, 133-142.

[24] Gendreau, M., G. Laporte and Rene Seguin. 1995. An Exact Algorithm for the VRP with Stochastic Demands and Customers. *Transportation Science* 29, 2.

[25] Gendreau, M., G. Laporte and Rene Seguin. May-June 1996. A Tabu Search Heuristic for the VRP with Stochastic Demands and Customers. *Operations Research* 44, 3.

[26] Bertsimas, Dimitris J., Philippe Chervi and Michael Peterson. 1995. Computational Approaches to Stochastic Vehicle Routing Problems. *Transportation Science* 29, 4.

[27] Bertsimas, Dimitris J. and David Simchi-Levi. March-April 1996. A New Generation of Vehicle Routing Research: Robust Algorithms, Addressing Uncertainty. *Operations Research*, 44, 2.

[28] Hjorring, C. and J. Holt. 1999. New Optimality Cuts for a single vehicle Stochastic Routing Problem. *Annals of Operations Research*, 86, 569-84.

[29] Yang, W. H, K. Mathur and R. H. Ballou. Stochastic Vehicle Routing with restocking *Transportation Science* 34, 99-112.

[30] Fisher, M. L. and R. Jaikumar. 1981. A Generalized Assignment Heuristic for Vehicle Routing *Networks* 11, 109-124.

[31] Dror, M. 1993. Modelling Vehicle Routing with uncertain demands as a Stochastic Program: Properties of the corresponding Solution. *European Journal of Operational Research*, 64, 432-441.

[32] Secomandi, N. 1998. Exact and Heuristic DP algorithms for the VRP with Stochastic Demands. Ph.D. Dissertation, Dept of Decision and Info Sciences, University of Houston, Texas.

[33] Secomandi, N. Sept-Oct 2001. A Rollout Policy for the VRP with Stochastic Demands. *Operations Research*, 49, 5, 796-802.

[34] Bertsekas, Dimitri P. and John N. Tsitsiklis. 1991. An Analysis of Stochastic Shortest Path Problems *Mathematics of Operations Research*, 16,580-595.

[35] Bertsekas, Dimitri P. and John N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

[36] Tesauro, G. and G. R. Galperin. 1996. On line Policy Improvement using Monte Carlo search. *Proceedings of the Neural Inform. Process. Systems Conference* Denver, CO.

[37] Bertsekas, Dimitri P. 1997. Differential Training of Rollout Policies. *Proc. 35th Allerton Conference on Comm., Control and Comput.* Allerton Park, IL.

[38] Bertsekas, Dimitri P., John N. Tsitsiklis and Cynara Wu. 1997. Rollout Algorithms for Combinatorial Optimization. *Journal of Heuristics*, 3 245-262.

[39] Garey, M. R. and D. S. Johnson. 1979. *Computers and Intractability: A guide to the Theory of NP Completeness*. Freeman.

[40] Reinelt, Gerhard. 1994. *The Traveling Salesman: Computational Solutions for TSP applications*. Springer Verlag.

[41] Golden, B. L. and W. R. Stewart. 1985. Empirical Analysis of Heuristics. *The Traveling Salesman problem: A guided Tour of Combinatorial Optimization* E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan. Wiley.

[42] Puterman, Martin. 1994. *Markov Decision Processes*. Wiley.

[43] Cook, William J., W. H. Cunningham, W. R. Pulleyblank, and Alexander Schrijver. 1998. *Combinatorial Optimization*. Wiley.

[44] Lin, S. and B. Kernighan. 1973. An effective heuristic algorithm for the TSP. *Operations Research*, 21, 498-516.

[45] Christofides, N. 1976. Worst Case Analysis of a new heuristic for the TSP. Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA.

[46] Haimovich, M. and A. H. G. Rinnooy Kan. 1985. Bounds and Heuristics for the Capacitated Routing Problem *Mathematics of Operations Research* 10, 527-542.

[47] Gillett, Billy E. and Leland L. Miller. 1974. A heuristic algorithm for the vehicle dispatch problem *Operations Research*, 22, 340-349.

[48] Charikar, M., S. Khuller and B. Raghavachari. 2001. Algorithms for Capacitated Vehicle Routing. *SIAM Journal of Computing*, 31, 3, 665-682.

[49] Haimovich, M., A. H. G. Rinnooy Kan and L. Stougie. 1988. Analysis of heuristics for Vehicle Routing Problems. *Vehicle Routing: Methods and Studies*. B. L. Golden and A. A. Assad (Eds.). North-Holland, Amsterdam, The Netherlands.

[50] Savelsbergh, M. W. P. and M. Goetschalkx. 1995. A comparison of the efficiency of fixed versus variable routes *Journal of Business Logistics*. 16, 163-187.

[51] Kao, E. P. C. 1978. A preference order dynamic program for a stochastic traveling salesman problem. *Operations Research*, 26, 1033-1045.

[52] Glover, Fred and Manuel Laguna. 1997. *Tabu Search* Kluwer Academic Publishers, Boston.

[53] Bartholdi, J. J., L. K. Platzman, C. R. Lee and W. H. Warden. 1983. A minimal technology routing system for Meals on Wheels. *Interfaces*, 13, 1-8.

[54] Cordeau, J-F, M. Gendreau, J-Y Potvin and F. Semet. 2002. A guide to Vehicle Routing Heuristics. *Journal of the Operational Research Society*. 53, 512-522.

[55] Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, 5, 533-549.

[56] Hansen, P. 1986. The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming. *Congress on Numerical Methods in Combinatorial Optimization*, Capri.

[57] Arora, Sanjeev. 1997. Nearly Linear Time Approximation Schemes for Euclidean TSP and other Geometric Problems. *Foundations of Computer Science, IEEE*.

[58] Swan, P. F. 1997. The effect of changes in Operations on LTL Motor Carrier Productivity and Survival. Ph.D. Dissertation. Department of Business Administration, University of Michigan.

[59] Deif, I. and L. Bodin. 1984. Extension of the Clarke and Wright algorithm for solving the Vehicle Routing Problem with backhauling. *Proceedings of the Babson conference on software uses in Transportation and Logistics Management*. A. E. Kidder (ed.), Babson Park, MA, 75-96.

[60] Golden, B., E. Baker, G. Alfaro and J. Schaffer. 1985. The Vehicle Routing Problem with Backhauls: Two Approaches. Proceedings of the Twenty-First Annual Meeting of S. E. TIMS. R. D. Hammesfahr (ed.). Myrtle Beach, SC, 90-92.

[61] Goetschalkx, M. and C. Horsley. 1986. The Vehicle Routing Problem with backhauls. *Working Paper, Material Handling Research Center*, Dept. of Industrial and Systems Engineering, Georgia Institute of Technology.

[62] Yano, C., T. Chan, L. Richter, T. Cutler, K. Murthy and D. Mcgettigan. 1987. Vehicle Routing at Quality Stores. *Interfaces*, 17, 52-63.

[63] Casco, Daniel O., B. L. Golden and Edward A. Wasil. 1988. Analysis of heuristics for Vehicle with Backhauls: Models, Algorithms and Case Studies *Vehicle Routing: Methods and Studies*. B. L. Golden and A. A. Assad (Eds.). North-Holland, Amsterdam, The Netherlands.

[64] Goetschalkx, M. and C. Jacobs-Blecha. 1989. The Vehicle Routing Problem with backhauls. *European Journal of Operational Research*, 42, 39-51

[65] Daganzo, C. F. and R. W. Hall. 1993. Routing model for pickups and deliveries: no capacity restrictions on the secondary items. *Transportation Science*, 27, 315-329

[66] Jacobs-Blecha, C. and M. Goetschalkx. 1993. The vehicle routing problem with backhauls: Properties and solution algorithms. *Technical Report MHRC-TR-88-13, Georgia Instiute of Technology, Atlanta*

[67] Toth, P. and D. Vigo. 1997. Exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31, 372-385

[68] Duhamel, C., J.-Y. Potvin, J.-M. Rousseau. 1997. Tabu search heuristic for the vehicle routing problem with backhauls and time windows. *Transportation Science*, 31, 49-59

[69] Mosheiov, G. 1998. Vehicle routing with pick-up and delivery: Tour-partitioning heuristics. Computers and Industrial Engineering. 34, 669-684

[70] Toth, P. and D. Vigo. 1998. Exact solution of the Vehicle Routing Problem *Fleet Management and Logistics*. T.G. Crainic and G.Laporte (Eds.). Kluwer: Boston, 1-31.

[71] Toth, P. and D. Vigo. 1999. Heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, 113, 528-543

[72] Mingozzi, A., S. Giorgi and R. Baldacci. 1999. Exact method for the vehicle routing problem with backhauls. *Transportation Science*, 33, 315-329

[73] Salhi, S., and G. Nagy. 1999. Cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50, 1034-1042

[74] Dethloff, J. 2002. Relation between vehicle routing problems: An insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls. *Journal of the Operational Research Society*, 53, 115-118

[75] Savelsbergh, M. W. P. and M. Sol. 1995. The General Pickup and Delivery Problem. *Transportation Science*, 29, 17-29.

[76] Van Slyke, R. and R. J-B. Wets. 1969. L-shaped linear programs with application to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17, 638-663

[77] Agrawal, A., P. Klein and R. Ravi. 1991. When Trees Collide: An approximation algorithm for the generalized Steiner problem on networks. *Proceedings of 23rd Annual ACM Symposium on Theory of Computing*, 134-144

[78] Goemans, M. X. and D. P. Williamson. 1995. A General Approximation Technique For Constrained Forest Problems. *SIAM Journal of Computing*, 24, 296-317

Vita

Surajit Das received the B.E. degree in Electronics and Communication Engineering from the I.I.T. Roorkee, India. He also received the P.G.D.M. from the I.I.M. Calcutta, India in 1982. He enrolled in the Ph. D. program in Business Logistics (since renamed the Department of Supply Chain and Information Systems) at the Smeal College, Pennsylvania State University in Fall 2000.