2009

A Guided Neighborhood Search Applied to the Split Delivery Vehicle Routing Problem

Rafael E. Aleman
Wright State University

A Guided Neighborhood Search Applied to the Split Delivery Vehicle Routing Problem

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

by

Rafael E. Aleman
M.S.Egr., Wright State University, 2005
B.S., Pontificia Universidad Javeriana, Colombia, 2001

2009
Wright State University

Wright State University
SCHOOL OF GRADUATE STUDIES

February 2, 2009

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Rafael E. Aleman ENTITLED A Guided Neighborhood Search Applied to the Split Delivery Vehicle Routing Problem BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy .

Raymond R. Hill, Ph.D.
Dissertation Director

Ramana Grandhi, Ph.D.
Director, Ph.D. in Engineering

Joseph F. Thomas, Jr. , Ph.D.
Dean, School of Graduate Studies

Committee on
Final Examination

Raymond R. Hill, Ph.D.

Xinhui Zhang, Ph.D.

James T. Moore, Ph.D.

George G. Polak, Ph.D.

Zhiqiang Wu, Ph.D.

ABSTRACT

Aleman, Rafael, Ph.D., Engineering Ph.D. Program, Department of Biomedical, Industrial and Human Factors Engineering, Wright State University, 2009. *A Guided Neighborhood Search Applied to the Split Delivery Vehicle Routing Problem.*

The classic vehicle routing problem considers the distribution of goods to geographically scattered customers from a central depot using a homogeneous fleet of vehicles with finite capacity. Each customer has a known demand and can be visited by exactly one vehicle. Each vehicle services the assigned customers in such a way that all customers are fully supplied and the total service does not exceed the vehicle capacity. In the split delivery vehicle routing problem, a customer can be visited by more than one vehicle, i.e., a customer demand can be split between various vehicles. Allowing split deliveries has been proven to potentially reduce the operational costs of the fleet.

This study efficiently solves the split delivery vehicle routing problem using three new approaches. In the first approach, the problem is solved in two stages. During the first stage, an initial solution is found by means of a greedy approach that can produce high quality solutions comparable to those obtained with existing sophisticated approaches. The greedy approach is based on a novel concept called the route angle control measure that helps to produce spatially thin routes and avoids crossing routes. In the second stage, this constructive approach is extended to an iterative approach using adaptive memory concepts, and then a variable neighborhood descent process is added to improve the solution obtained.

A new solution diversification scheme is presented in the second approach based on concentric rings centered at the depot that partitions the original problem. The resulting sub-problems are then solved using the greedy approach with route angle control measures. Different ring settings produce varied partitions and thus different solutions to the original problem are obtained and improved via a variable neighborhood descent.

The third approach is a learning procedure based on a set or population of solutions. Those solutions are used to find attractive attributes and construct new solutions within a tabu search framework. As the search progresses, the existing population evolves, better solutions are included in it whereas bad solutions are removed from it. The initial set is constructed using the greedy ap-

proach with the route angle control measure whereas new solutions are created using an adaptation of the well known savings algorithm of Clarke and Wright (1964) and improved by means of an enhanced version of the variable neighborhood descent process. The proposed approaches are tested on benchmark instances and results are compared with existing implementations.

Contents

List of Figures

List of Tables

Acknowledgement

Above all, thanks to God for giving me the opportunity to successfully complete my program and meet many valuable individuals and friends. This is a great opportunity to express my respect to my dissertation Director, Dr. Raymond Hill, who provided his supervision, expertise, and necessary support to pursue and successfully finish my doctoral dissertation journey. Thanks for treating myself as a student with strengths and weaknesses, but also as a person with academic and family responsibilities. Thanks for his confidence, guidance, and for always considering my point of view. I will not forget his professional ethics and respect to others. My sincere thanks go to all members in my committee, Dr. X. Zhang, Dr. J. Moore, Dr. G. Polack, and Dr. Z. Wu for helping in various ways to clarify and enrich my academic work. My gratitude also to Dr. X. Zhang and the Ph.D. in Engineering Program for providing me with scholarships when I really needed them. I want to sincerely thank Dr. Michel Gendreau for our talk in Montevideo, Uruguay, that encouraged and inspired me to continue my dissertation research.

I am pleased to thank my parents, Miguel Aleman and Celia Ricaurte, who have provided me with necessary assistance and for convincing me that I was smarter than I really was, and smart enough to finish my doctoral studies. I'm doing my best to follow your example. I should also thank my younger brother, Jesus M. Aleman, for staying with us and taking care of Gabriel and Miguel when needed. I would like to extend my thanks to my cousin, Dr. Juan C. Ricaurte, for his warm welcome in Los Angeles. I will never forget his affection as I felt it in an important moment in my life. That trip meant a lot to me. Many thanks to Los Reina for the friendship: Juan, Mariarosa "RoRo", Isabella, and Christina. Similarly, thanks to Dr. Francisco Alvarez and Begona Claveria.

I certainly would like to extend my thanks to my wife, Maria C. Berrocal. She has tolerated my

moodiness for the last long 3 years, brought me food to keep me going, not complained *much* when I sat in my sofa after a long day of writing just staring into space instead of chatting with her, juggled her schedule so I could stay in the school and do my research, took care of our kids and most household duties while I lived in my office, taught me to be patient and wait for my chance, and did not divorce me even though I have been a less than ideal husband during this long process. Thanks to our twin boys, Gabriel and Miguel, for making me a better human being and understand how hard it is to be a good father.

Lastly, I want to thank many people, friends, and colleagues who have not been explicitly mentioned, but have helped me to walk through this long and arduous process. Maybe it could not be possible without your support.

January 19, 2009

Dedicated to:

Gabriel and Miguel Aleman, my two angels.

Chapter 1

Introduction

The vehicle routing problem is a core problem in transportation, logistics, and supply chain management. Although it has been studied for almost 50 years, this problem is still under active investigation by practitioners and researchers. The optimization problem is to supply the demand of a number of customers with a fleet of vehicles with finite capacity. Usually, goods are delivered from a central depot to customers placing an order for those goods, while the operational cost of the fleet is minimized. Typically, the objective function corresponds to the total travel distance or total travel time. Sometimes, an additional set of constraints is used to establish an upper bound on the travel time of routes to avoid drivers exceeding the assigned work shift.

Real world situations involve other complexities that the classical VRP does not consider. Thus, various extensions to the VRP appear in the literature that try to incorporate more variables, more constraints, and model more realistic conditions. These extensions include but are not limited to: VRP with multiple depots (MDVRP) where a company can have several depots from which customers are supplied; periodic VRP (PVRP) where the planning horizon is composed of several periods, usually days, and customers have fixed daily demands; VRP with split deliveries (SDVRP) where various vehicles can serve a customer; stochastic VRP (SVRP) where some elements in the problem (including travel time, customer demand, customer presence) are stochastic; VRP with pick-ups and deliveries (VRPPD) where vehicles can pick-up goods at the customers for return to

the depot; VRP with time windows (VRPTW) where a time interval is associated to each customer wherein the delivery has to be made; and VRP with backhauls (VRPB) which is similar to the VRPPD, but in the case of VRPB all deliveries in a route are made before the pick-ups in order to avoid rearranging the loads.

The VRP has been studied both in a deterministic and stochastic version and a wide variety of techniques have been used to solve each of them. These techniquees include exact algorithms, classical heuristics, such as constructive and saving algorithms, single-route and multiple-route improvement algorithms, sweep algorithms, petal algorithms, sequential route-building algorithms, cluster-first route-second algorithms, route-first cluster-second, and matching algorithms, and meta-heuristics, such as simulated annealing, tabu search, genetic algorithms, and ant systems.

The SDVRP is an alternative to the classic VRP, which allows one to potentially reduce the operational costs of the fleet of vehicles. However, there is not much in the literature on the SDVRP. In fact, only a few heuristic algorithms have been developed to solve the problem. This dissertation applies meta-heuristic methods to solve the SDVRP. In these methods, solutions are constructed through a simple and effective greedy algorithm that assign customers to the routes under construction based on the route angles. This mechanism favors spatially thin routes. A motivation for this angle evaluation can be found in commercial trucking where drivers from a common depot prefer to not travel far among customers and dislike crossing routes with other drivers.

Constructed solutions are subsequently improved following the philosophy of the variable neighborhood search proposed by Mladenović and Hansen (1997). The defining characteristic of a variable neighborhood search is changing the neighborhood structure to avoid local optimum. Given a set of pre-defined neighborhoods, the solution space is explored within a local search while systematically changing the neighborhood of the current solution. The search stops when the current solution is a local optimum for all the pre-defined neighborhoods.

Some variable neighborhood searches have been applied to routing problems. Bräysy (2003) proposes a reactive variable neighborhood search that modifies select parameters and changes the

objective function to avoid local optimality. The method is applied successfully to the VRPTW and provided four new best-solutions. Polacek et al. (2004) use variable neighborhood search to solve the multi-depot VRPTW (MDVRPTW). The algorithm outperforms a tabu search, found 10 new best-solutions, and demonstrates some superiority solving large real-world problems. Kytöjoki et al. (2007) use a variable neighborhood descent to solve large-scale VRPs and accept non-improving solutions by penalizing certain solution features. High quality solutions are found for problems involving up to 20,000 customers.

Initial solutions are constructed with a greedy algorithm based on an insertion method. Customers are added to a list and sorted according to their distance to the depot. They are then inserted into the solution at the lowest possible cost. A customer can be inserted to initiate a new route or to modify an existing route. In the latter case, a customer is inserted into the cheapest position in the route. However, this insertion method produces poor solutions when the triangular inequality favors the customer insertion into existing routes producing spread routes instead of initialization of new routes. Thus, a new mechanism based on the spatial distribution of routes is used to penalize the insertion of customers producing spread routes. This mechanism uses a fixed threshold value to determine when a route is too wide and penalizes its insertion. The incorporation of this mechanism provides high quality solutions compared to the best known SDVRP solutions in the literature at a low computational time.

This dissertation presents new search techniques for practitioners and researchers solving SDVRPs. For practitioners, using simple yet effective solution techniques allow operators and managers to efficiently use the fleet of vehicles. The quality of solutions obtained is comparable to the quality of those obtained with existing sophisticated techniques. Computational times are substantially lower than those of existing techniques. For researchers, the development of meta-heuristics based on new diversification and vocabulary building techniques represents an advance in variable neighborhood searches.

This dissertation provides multiple contributions. Part I provides a focused yet thorough liter-

ature review on the VRP, SDVRP, and VRPSC. The review includes problem properties, models, and solution algorithms. A new problem classification scheme is also presented in Part I useful for categorizing modern routing problems. This classification scheme is based on three criteria: staticism/dynamism of the problem parameters, the knowledge of the information relevant to the design of its solution, and the method to model the unknown data. Part II presents new algorithmic contributions, including: 1) a novel constructive approach, an iterative constructive approach that uses adaptive memory concepts, and a variable neighborhood descent found in Aleman et al. (2007); 2) a new solution diversification scheme found in Aleman et al. (2008) based on concentric rings centered at the depot that partitions the original problem and solves the resulting problems using a constructive approach; and 3) a population-based search approach, called tabu search with vocabulary building approach, that constructs an initial solution set and then uses the set of solutions to find attractive solution attributes with which to construct new solutions and evolve the set.

This document is organized as follows. First, a literature review relevant to this research is found in Chapters 2 to 4. Chapter 2 contains a review on representative existing techniques to solve the classic VRP. Chapter 3 reviews existing research and search methods to solve the SDVRP, the primary focus of this research. Chapter 4 contains concepts and existing studies on routing problems with stochastic customers. Second, the proposed approaches to efficiently solve the SDVRP are presented in Chapters 5 to 7. Chapter 5 describes a greedy approach with a novel route angle control measure, an iterative approach using adaptive memory concepts, a variable neighborhood descent process based on the standard customer shift and swap adapted to handle split deliveries plus a new operator that introduces split deliveries into the solution in order to reduce the objective function value. Chapter 6 provides a new solution diversification scheme based on concentric rings centered at the depot to partition the original problem, solve the resulting-subproblems independently, produce a complete solution, and then improve it using the variable neighborhood descent described in Chapter 5. Chapter 7 outlines a learning procedure that uses a population of solutions to derive information used to find attractive attributes and a tabu search framework to generate new solutions. Finally, a summary of this dissertation is provided in Chapter 8.

Part I

Background Literature

Chapter 2

Vehicle Routing Problem (VRP): Solution Techniques

2.1 Problem Definition

The vehicle routing problem (VRP) was first formulated by Dantzig and Ramser (1959) and is a core problem in transportation, logistics, and supply chain management. It is also sometimes called the capacited vehicle routing problem (CVRP) or truck dispatching problem. The VRP involves a fleet of vehicles with fixed characteristics (i.e., speed, capacity, etc.) and a set of geographically scattered delivery points (i.e., cities, warehouses, schools, customers, etc.) with fixed demands for transporting goods between a unique depot and specified delivery points. The VRP is defined on an undirected graph $G = (V, E)$ where $V = \{0, 1, ..., n\}$ is the set of $n + 1$ nodes of the graph, and $E = \{(i, j) : i, j \in V,\ i < j\}$ is the set of edges. Node 0 represents a depot where a fleet $M = \{1, ..., m\}$ of identical vehicles with capacity Q are stationed, while the remaining node set $V' = \{1, ..., n\}$ is the set of n customers. A non-negative cost –or distance, or travel time– c_{ij} is associated to every edge (i, j). Each customer $i \in V'$ has a demand of q_i units. The optimization problem is to determine which customers are served by each vehicle and what route the vehicle follows to serve those assigned customers, while minimizing the operational costs of the fleet, such

as travel distance, gas consumption, and vehicle depreciation. Traditionally, routes are designed to start and end at the depot, every customer is visited exactly once by exactly one vehicle, and the total demand of any route cannot exceed the available vehicle capacity.

This chapter provides background on the vehicle routing problem. It is organized as follows. Section 2.2 reviews some of the mathematical formulations employed. Each formulation below is provided in complete form to promote readability. Sections 2.3 to 2.5 review the most representative techniques used to solve the VRP. Bodin and Golden (1981) classify these techniques whereas a full complete survey and description is found in Laporte et al. (2000), Toth and Vigo (2002), Cordeau et al. (2002), Cordeau et al. (2005), and Laporte (2007). These techniques include exact algorithms, classical heuristic algorithms (i.e., constructive, saving, improvement, sweep, petal, and matching algorithms), and metaheuristic algorithms. Section 2.6 discusses the dynamic aspects appearing in the vehicle routing problem and presents a classification scheme for the VRP based on the staticism/dynamism of the problem parameters, the availability of information relevant to solve the problem, and the method used to model the unknown information.

2.2 Models for the VRP

Although different authors have implemented various formulations of the vehicle routing problem, this section presents some VRP models based on the work of Baldacci et al. (2004).

2.2.1 A Two-Index Vehicle Flow Formulation

Notation

E	Set of edges
c_{ij}	Nonnegative cost of edge $\{i,j\} \in E$
q_i	Demand of client i
Q	Vehicle capacity
M	Fleet size
V	$\{0,1,\ldots,n\}$ is the set of nodes. Node 0 represents the depot
V'	$V' = V \backslash \{0\}$ is the set of n customers
S	Subset of customers $S \subseteq V'$
$\bar{S}$	Complementary set of nodes $V \backslash S$
$\mathcal{J}$	Set of customers $\mathcal{J} = \{S : S \subseteq V', \lvert S\rvert \geq 2\}$
$r(S)$	Minimum number of vehicles of capacity Q needed to satisfy the demand of customers in S
ξ_{ij}	0, if edge $\{i,j\} \in E$ is not in the solution; 1, if the edge is in the solution; and 2, if a route including the single customer j is selected in the solution

$$\text{Minimize} \sum_{\{i,j\} \in E} c_{ij}\xi_{ij} \tag{2.1}$$

Subject to:

$$\sum_{\substack{j \in V \\ i<j}} \xi_{ij} + \sum_{\substack{j \in V \\ i>j}} \xi_{ji} = 2; \forall i \in V' \tag{2.2}$$

$$\sum_{i \in S} \sum_{\substack{j \in \bar{S} \\ i<j}} \xi_{ij} + \sum_{i \in \bar{S}} \sum_{\substack{j \in S \\ i<j}} \xi_{ij} = 2r(S); \forall S \in \mathcal{J} \tag{2.3}$$

$$\sum_{j \in V'} \xi_{0j} = 2M \tag{2.4}$$

$$\xi_{ij} \in \{0,1\}; \forall \{i,j\} \in E \backslash \{\{0,j\} : j \in V'\} \tag{2.5}$$

$$\xi_{0j} \in \{0,1,2\}; \forall \{0,j\} \in E, j \in V' \tag{2.6}$$

Constraints (2.2) are the degree constraints for each customer. Constraints (2.3) are the subtour elimination constraints which, for any subset S of customers that does not include the depot, impose that $r(S)$ vehicles enter and leave S. Constraints (2.4) state that M vehicles must leave and return to the depot. Constraints (2.5) and (2.6) are the integrality constraints.

2.2.2 A Multicommodity Flow Formulation

Notation

c_{ij}	Nonnegative cost of edge $\{i, j\} \in E$
q_i	Demand of client i
Q	Vehicle capacity
M	Fleet size
V	$\{0, 1, \ldots, n\}$ is the set of nodes. Node 0 represents the depot
V'	$V' = V \backslash \{0\}$ is the set of n customers
ξ_{ij}	1, if arc (i, j) is in the optimal solution. 0 otherwise
y_{ij}^l	Amount of demand destined to customer $l \in V'$ that is transported on arc (i, j)

$$\text{Minimize} \sum_{\substack{i,j \in V \\ i \neq j}} c_{ij} \xi_{ij} \tag{2.7}$$

Subject to:

$$\sum_{i \in V} \xi_{ij} = 1; \forall j \in V' \tag{2.8}$$

$$\sum_{j \in V} \xi_{ij} = 1; \forall i \in V' \tag{2.9}$$

$$\sum_{j \in V'} \xi_{0j} = M \tag{2.10}$$

$$\sum_{j \in V'} \xi_{j0} = M \tag{2.11}$$

$$\sum_{i \in V} y_{ij}^l - \sum_{i \in V} y_{ji}^l = \begin{cases} q_l, j = l & \forall l \in V', \\ 0, j \neq l & \forall j, l \in V' \\ -q_l, j = 0 & \forall l \in V' \end{cases} \tag{2.12}$$

$$y_{ij}^l \leq q_l \xi_{ij}; \forall i, j \in V, i \neq j; \forall l \in V' \tag{2.13}$$

$$\sum_{j \in V'} \sum_{l \in V'} y_{ij}^l \leq Q - q_i; \forall i \in V \tag{2.14}$$

$$y_{ij}^l \geq 0; \forall i, j \in V, i \neq j; \forall l \in V' \tag{2.15}$$

$$\xi_{ij} \in \{0,1\}; \forall i,j \in V, i \neq j \tag{2.16}$$

Constraints (2.8) and (2.9) ensure that each customer is visited exactly once. Constraints (2.10) and (2.11) ensure all M vehicles in the fleet leave from and return to the depot. Constraints (2.12) and (2.13) are the commodity flow constraints to guarantee that each demand is satisfied. Constraints (2.14) ensure that vehicle capacity is not exceeded. Finally, constraints (2.15) and (2.16) are the integrality and binary conditions, respectively.

2.2.3 A Set-Partitioning Formulation

Notation

V	$\{0,1,\ldots,n\}$ is the set of nodes. Node 0 represents the depot
V'	$V' = V \backslash \{0\}$ is the set of n customers
$\mathcal{R}$	Index set of all feasible routes
M	Fleet size
$\widehat{c_j}$	Cost of route $j \in \mathcal{R}$
a_{ij}	Binary coefficient equal to 1 if customer i belongs to route $j \in \mathcal{R}$. 0, otherwise
ζ_j	1, if route $j \in \mathcal{R}$ is in the optimal solution. 0, otherwise

$$\text{Minimize } \sum_{j \in \mathcal{R}} \widehat{c_j} \zeta_j \tag{2.17}$$

Subject to:

$$\sum_{j \in \mathcal{R}} a_{ij} \zeta_j = 1; \forall i \in V' \tag{2.18}$$

$$\sum_{j \in \mathcal{R}} \zeta_j = M \tag{2.19}$$

$$\zeta_j \in \{0,1\}; \forall j \in \mathcal{R} \tag{2.20}$$

Constraints (2.18) ensure that each customer is visited exactly once. Constraints (2.19) ensure all M vehicles are in the solution. Finally, Constraints (2.20) are the binary conditions.

2.2.4 A Two-Commodity Flow Formulation

Notation

V	$\{0, 1, \ldots, n\}$ is the set of nodes. Node 0 represents the depot
E	Set of edges
G	Undirected graph $G = (V, E)$
$\bar{V}$	$\bar{V} = V \cup \{n + 1\}$
V'	$V' = \bar{V} \backslash \{0, n + 1\}$
$\bar{G}$	Extended graph $\bar{G} = (\bar{V}, \bar{E})$
$\bar{S}$	Complementary set of nodes $\bar{V} \backslash S$
M	Fleet size
Q	Vehicle capacity
c_{ij}	Cost of edge $\{i, j\}$
q_i	Demand of client i
$q(V')$	Sum of demands of customers in set V'
ξ_{ij}	1, if edge $\{i, j\} \in \bar{E}$ is in the solution. 0, otherwise
x_{ij}	Load of vehicle associated with edge $\{i, j\}$
x_{ji}	Empty space on the vehicle associated with edge $\{i, j\}$ (i.e., $x_{ji} = Q - x_{ij}$)

$$\text{Minimize} \sum_{\{i,j\} \in \bar{E}} c_{ij} \xi_{ij} \tag{2.21}$$

Subject to:

$$\sum_{j \in \bar{V}} (x_{ji} - x_{ij}) = 2q_i; \forall i \in V' \tag{2.22}$$

$$\sum_{j \in V'} x_{0j} = q(V') \tag{2.23}$$

$$\sum_{j \in V'} x_{j0} = MQ - q(V') \tag{2.24}$$

$$\sum_{j \in V'} x_{(n+1)j} = MQ \tag{2.25}$$

$$x_{ij} + x_{ji} = Q\xi_{ij}; \forall \{i, j\} \in \bar{E} \tag{2.26}$$

$$\sum_{\substack{j \in \bar{V} \\ i<j}} \xi_{ij} + \sum_{\substack{j \in \bar{V} \\ i>j}} \xi_{ji} = 2; \forall i \in V' \tag{2.27}$$

$$x_{ij} \geq 0,\ x_{ji} \geq 0; \forall \{i,j\} \in \bar{E} \tag{2.28}$$

$$\xi_{ij} \in \{0,1\}; \forall \{i,j\} \in \bar{E} \tag{2.29}$$

Constraints (2.22) ensure that the inflow minus the outflow at each customer is equal to $2q_i$. Constraint (2.23) forces the outflow at the depot to equal the total customer demand. Constraint (2.24) defines the residual capacity of the vehicle fleet. Constraint (2.25) ensures the inflow at source $n+1$ is the total capacity of the vehicle fleet. Constraints (2.26) define the edges of a feasible solution. Constraints (2.27) force a customer to be connected to two edges. Constraints (2.28) and (2.29) are integrality and binary conditions, respectively.

2.3 Exact Algorithms for the VRP

2.3.1 Branch and Bound

Christofides and Eilon (1969) present an approach for solving the vehicle dispatching problem. Their solution method is based on a branch-and-bound (B&B) algorithm designed to solve a traveling salesman problem. Their VRP is formulated as a TSP by deleting the depot and replacing it with as many artificial copies of the depot as there are vehicles in the fleet. Traveling from one artificial depot to another is disabled by setting the distance between them to a large cost (i.e., infinity). The number of artificial depots N (i.e., the number of vehicles required in the final solution) has a lower bound determined by the vehicle capacity and the demand of all customers as:

$$N \geq \sum^{n} q_i / C$$

where q_i is the demand of customer i, n is the number of customers, and C is the vehicle capacity.

Obviously, there may exist no feasible solution for this value of N. Therefore, the problem is solved for several values of N, and the best solution among these is chosen as the final solution.

Before branching to a new node, the load capacity and distance limit of the vehicle in question as well as the remaining fleet capacity are evaluated. If any constraints are violated, there is no need to keep exploring the current branch. In order to reduce the search space, the bounds for nodes of the decision tree are determined by computing a minimal spanning tree. The B&B algorithm was tested on two instances with 6 and 13 customers. Another 8 problems, having 21 to 100 customers, required excessive computation time and memory space requirements. The computation time for the solved problems are 1.5 and 15 minutes, respectively, but keep in mind the work was conducted in 1969.

Miller (1995) describes a B&B algorithm where the lower bounds are computed by relaxing the subtour elimination and vehicle capacity constraints to produce a b-matching problem. This algorithm differs from others in that b-matching, instead of spanning trees, forms the kernel of the relaxation. The algorithm is tested on 11 instances taken from the literature involving 7 to 61 customers. The optimal solution for the largest problem is obtained in about 16 minutes whereas for a problem with 51 customers, it took about 4 hours.

2.3.2 Branch-and-cut

Baldacci et al. (2004) describe a branch-and-cut procedure for the VRP based on an integer programming formulation in the form of the two-commodity network flow problem presented in Section 2.2. A lower bound is computed based on the linear relaxation of the formulation, strengthened by a set of flow and capacity inequalities. The algorithm is tested on 19 problems taken from the literature, involving 15 to 134 customers, and 8 randomly generated instances, involving 30 to 100 customers. The algorithm successfully solved problems involving up to 80 customers and an instance involving 135 customers. It took about 2 hours and 30 minutes to optimally solve the problem with 135 customers.

Table 2.1: Representative Exact Algorithms for the VRP.

Algorithm	Year	Description & Remarks
Christofides and Eilon	1969	Branch and Bound
Miller	1995	Branch and Bound
Hadjiconstantinou et al.	1995	Set-partitioning
Baldacci et al.	2004	Branch and Cut

2.3.3 Set-partitioning

Hadjiconstantinou et al. (1995) present a tree-search procedure based on the use of lower bounds that are derived from a combination of two different relaxations of the original problem: q-paths (i.e., chains of customers whose weight is equal to q) and k-shortest paths. The algorithm is evaluated using 25 problem sets (involving 15 to 150 customers), 10 of them from Eilon et al. (1971) and 15 randomly generated with customers distributed uniformly. The algorithm found an optimal solution for problems with up to 50 customers but could not solve problems with 75 or more customers within a 12 hour computational limit. The largest solvable problem, 50 customers, solved in about 2 hours.

2.4 Classical Heuristic Algorithms for the VRP

In operations research, heuristics are generally simple search algorithms designed to find a solution to an optimization problem. Heuristics are a set of rules logically designed to solve an optimization problem based on a specific objective, such as minimizing costs or maximizing profits. In general, the design of heuristics follows the common sense of the designer and his/her perception of the problem. There is no a unique way to design, and of course implement, heuristics; the only limitation is creativity. There are two aspects characterizing heuristics: quality of the solution and computational time. Although heuristics can find good solutions without guaranteeing optimality, these solutions can be found in reasonable computational time. Many heuristics have been invented

to solve the VRP. Some of them present new thoughts and challenges, whereas others are not as creative or simply follow previous techniques. The next sections review some of the classical heuristics used to solve the VRP.

2.4.1 Constructive Algorithms

In their route building heuristic, Dantzig and Ramser (1959) use “stages of aggregation” and pair customers whose combined demand does not exceed some fraction of the vehicle capacity. At every stage, pairs of customers obtained in previous stages are combined so the capacity of a vehicle is not exceeded. To calculate the number of stages of aggregation, they sort the customers based on their demands and determine the maximum number of customers that a single vehicle can serve.

In the initial solution, a vehicle serves exactly one customer so the initial solution contains as many routes as customers. These pairs are called the “basic set”. In the first stage, a series of *rapid corrections* are executed by bringing into the solution non-basic pairs with the smallest inter-pair distances whose combined demand does not exceed the fraction of the truck capacity established in the stage. These rapid corrections are repeated as long as non-basic pairs are available.

The customer pairs obtained in the first stage are combined in the second stage to minimize the distance traveled by all vehicles. A matrix containing the minimum distances between pairs and the depot is created. This “distance matrix” is used in every stage. Subsequent stages repeat the process until the vehicles are near capacity. The selection criterion for pairing customers is focused on filling the vehicles and minimizing the sum of inter-pair distances within a route. Although “rapid corrections” allow reducing the inter-pair distances, once customers are grouped they are not separated in further stages. This myopic approach does not focus on minimizing the total distance of all vehicles.

2.4.2 Savings Algorithms

The calculation of the distance matrix in Dantzig and Ramser's method (1959) can be tedious and computationally demanding. The savings algorithm of Clarke and Wright (1964) does not calculate this matrix, which reduces computational efforts. Clarke and Wright also remove the restriction that only customers whose combined demand does not exceed a fraction of the vehicle capacity can be grouped. Instead, they group any customers whose combined demand does not exceed the capacity of a vehicle. There is both a concurrent and a sequential version of the algorithm. The concurrent version creates routes simultaneously. Initially every customer is served by a separate vehicle. The algorithm repeatedly takes a pair of customers from two different routes and calculates the distance saving of the four possible ways in which the two customers can be linked and the routes split. The two customers with the maximum savings are linked and new feasible routes are produced. This procedure stops when no further savings are attainable. The sequential version creates one route at a time by selecting a seed customer and iteratively linking that customer using the highest savings and the last linked customer. When no more customers can be linked, a new seed customer is selected and the procedure continues until all the customers are served. Although routes are not optimized in the final allocation, the computational results demonstrate better performance than Dantzig and Ramser's (1959) in 17 of 31 different problem sets. Clarke and Wright suggest, however, re-optimizing the routes as independent TSPs to obtain even better solutions. This "savings" algorithm was developed as a greedy approach, but is really an improvement method.

2.4.3 Single-Route Improvement Algorithms

Single-route improvement algorithms try to improve an existing solution by rearranging the order in which customers are served within the routes. In this sense, every route can be thought of as an independent TSP. The most representative procedures are given by Lin and Kernighan (1973), Or (1976), Potvin and Rousseau (1995), and Renaud et al. (1996). These algorithms are iterative procedures that improve the solution by relocating "customers" or "edges" within a route. The

improvement search stops when no further cost reductions can be found.

Or (1976) proposes a "customer" exchange heuristic that tries to improve a route by relocating sequences of one, two, or three adjacent customers within the route. For example, an Or-opt-1 exchange considers each customer in a route and tries to improve the tour by inserting that customer at another location. The classical k-opt is an "edge" exchange heuristic which deletes k edges from a route, temporarily reconnects the remaining portions of the route in all the possible ways and selects the exchange providing the best cost for the route. Lin and Kernighan (1973) discuss a *sequential exchange* method based on a generalization of the k-opt transformation. Their procedure does not use a fixed value for k, but tries to identify the highest k each iteration and, obviously, the edges that can be exchanged to best improve the route cost. If a best improvement is found, edges are exchanged and the process is repeated until no further improvement can be made by the procedure. The algorithm produces optimal solutions for all problems tested including problems in the literature for the TSP and randomly generated instances involving up to 110 cities.

The improvement algorithm of Potvin and Rousseau (1995) transforms a VRP into an equivalent TSP by creating copies of the depot. The idea of creating copies of the depot was previously explored in other studies (Christofides and Eilon, 1969; Potvin et al., 1989). Once the VRP solution is transformed into a TSP, a proposed 2-opt* exchange heuristic is applied where two links are replaced by two new links in such a way that the TSP route is divided into two subroutes without reversing any portion of the routes (as opposed to the 2-opt that can reverse a segment of the route). The new solution is valid only if there is a copy of the depot in both subroutes. The 2-opt* exchange is particularly well suited for problems with time windows because it preserves the orientation of the routes by introducing the last customers in a given route at the end of another route. The algorithm was tested on Solomon's test problems (Solomon, 1987) and randomly generated problems with 100 customers. The results indicate that, even though the Or-opt dominates, the 2-opt* algorithm is fast and effective on problems with tight time windows. A hybrid algorithm that merges 2-opt* and Or-opt is tested and outperforms the classical 3-opt algorithm.

Renaud et al. (1996) describe a 3-phase algorithm that uses a route improvement heuristic for the TSP based on a 4-opt* move. This move is similar to the classical 4-opt, but reduces the number of possible reconnections to 8 (the standard 4-opt produces 48) by inserting the cheapest of two predefined edges. The other three edges are selected from the remaining 8 neighbor tours. The algorithm was tested on 100 randomly generated instances with customers ranging from 50 to 500. The improvement heuristic was compared with other improvement heuristics including 2-opt, 3-opt, and Or-opt. Results indicate that 4-opt* dominates Or-opt and considerably improves over 3-opt in terms of processing time.

2.4.4 Sweep Algorithms

Wren and Holliday (1972) present an algorithm for the VRP which allows routes emanating from several depots subject to a limited number of vehicles at each depot. The algorithm is composed of a constructive method followed by an improvement process that moves customers either within or between routes. The contribution of the proposed algorithm is the constructive procedure that sorts the customers in a novel manner and assigns them sequentially to the nearest feasible route. A feasible route is either an existing route with enough spare capacity or a new route. Customers are first assigned to the nearest depot. The constructive algorithm is informally known as a sweep algorithm because it sweeps a “ray” from each depot in a clockwise direction and determines a bearing for customers assigned to that depot. Customers are then sorted by the bearings regardless of the depots and assigned to the nearest feasible route. The algorithm (constructive plus refining process) was tested on 9 problem sets. In the first 6 sets, ranging in size from 21 to 36 customers and only one depot, the algorithm improved 2 solutions obtained from a version of the Clarke and Wright method and Christofides and Eilon (1969). In the last 3 sets, involving 50, 75 and 100 customers, the algorithm provided solutions with fewer vehicles than the Clarke and Wright method in two cases and a significant reduction in traveled distance in the third case. The authors also provide a case study where the algorithm outperformed a commercially available program to solve VRP with multiple depots that first allocates the customers to depots and then applies the Clarke and Wright

method to each depot independently.

Gillett and Miller (1974) introduce a procedure that is officially called the sweep algorithm and is quite similar to the algorithm presented by Wren and Holliday (1972) but with two differences. The first difference is that Gillett and Miller (1974) use a single depot, so the routes are made up while the "ray" sweeps around. Thus, there is no further need for finding a nearest depot. The second difference is that Gillett and Miller (1974) sweep the "ray" both clockwise and counterclockwise. The best solution is then selected. These two versions are called the *forward* and *backward* sweep algorithm, respectively.

The sweep algorithm of Gillett and Miller (1974) is tested on 12 problem sets where the number of customers ranges from 21 to 250. Of these problem sets, 7 are solved by Christofides and Eilon (1969) and 5 are new problems. In terms of total traveled distance, the Gillett and Miller (1974) algorithm outperforms the Christofides and Eilon (1969) algorithm in 3 cases, matches the solution in 2 cases, and provides a less attractive solution in the other 2 cases. With respect to the algorithm of Wren and Holliday (1972), 4 of the problem sets are solved by both algorithms. From those, Gillett and Miller's outperforms Wren and Holliday's in 3 cases.

2.4.5 Petal Algorithms

The algorithm of Gillett and Miller (1974) produces routes with a petal-like structure. It seems natural to expect that an optimal VRP solution, barring any unusual side constraints, includes routes that do not cross each other. Based on this assumption, Foster and Ryan (1976) explore a subset of feasible VRP petal solutions, referred to as the "petal set", and reduce the feasible region by imposing the constraint that neither deliveries within a "petal" are bypassed nor adjacent routes cross. They solve an over-constrained LP model of the VRP to optimality by producing a set of feasible routes and then separately solving a traveling salesman problem for each route. Foster and Ryan's algorithm is evaluated using 13 problems from various authors (Clarke and Wright, 1964; Gaskell, 1967; Christofides and Eilon, 1969; Gillett and Miller, 1974), involving 21 to 100

customers. The algorithm provides 11 new best solutions and generally outperforms the solutions obtained from the two sweep approaches (Wren and Holliday, 1972; Gillett and Miller, 1974).

The petal algorithm of Ryan et al. (1993) is based on the work of Gillett and Miller (1974) and Foster and Ryan (1976). A VRP optimal solution can include crossing routes if there are restrictions on the structure of feasible routes, such as vehicle capacity, route distance limit, and time restrictions. The distinction with this work is the use of a shortest path technique to produce petals and their associated routes, rather than an LP. The performance of the shortest path method for finding a shortest set of petals and the LP method for finding an equivalent optimal petal solution have been compared by solving the problems defined in Altinkemer and Gavish (1991). The results show that the shortest path method outperforms the LP method by two orders of magnitude, in terms of the CPU time.

2.4.6 Sequential Route-Building Algorithm

All the existing sequential saving algorithms select a customer to insert based on the savings associated with the insertion. However, the customer is inserted into the route under construction following the position of the last inserted customer. Mole and Jameson (1976) describe a savings algorithm that sequentially constructs routes and inserts unserved customers into the route under construction, R_k. The proposed method follows three steps to determine not only the next customer to be inserted, but also where within R_k to place the customer. In the first step, the most advantageous feasible position on R_k for each unserved customer is determined. In the second step, the next unserved customer is identified and inserted into R_k. In the third step, the possible resequencing of customers on R_k is explored via a 2-opt operator. If the capacity of R_k is exhausted, a new route is started ($k = k + 1$). The process is repeated until all customers are in a route. Two parameters, λ and μ, are used in the algorithm to vary the criteria used to choose the best unserved customer to be inserted. A "refine" procedure, which transfers customers from one route to another, is used to improve the routes. The algorithm is tested on 10 instances taken from Christofides and Eilon

(1969) and clearly outperforms Clarke and Wright, but does not provide better solutions than Wren and Holliday.

Christofides et al. (1979) present a sequential procedure that uses two phases. In the first phase, routes are constructed quite similar to Mole and Jameson (1976), but the first unserved customer x_{i_h} inserted into a new route R_h is selected arbitrarily. In Phase 2, routes are constructed in parallel, based on the routes from Phase 1. The algorithm is tested on 14 problem sets, with customers ranging from 50 to 199, taken from Christofides and Eilon (1969) and some structured problems. Results are compared in terms of total traveled distance with those obtained from the implemented versions of various algorithms (Clarke and Wright, 1964; Mole and Jameson, 1976; Gillett and Miller, 1974). The comparison clearly reveals that the presented algorithm outperforms Mole and Jameson by finding better solutions on 12 of the tested problems.

2.4.7 Cluster-First Route-Second Algorithms

Cluster-first, route-second algorithms are two-phase methods that divide the VRP into two subproblems. In phase 1, customers are assigned, or grouped/clustered, to vehicles without considering the order of servicing the assigned demands. In phase 2, customers are rearranged to try and find an optimal or near-optimal solution for each cluster. Different techniques are used to cluster the customer, including the sweep and the petal algorithms. To design efficient routes for each cluster, any TSP heuristic can be utilized including the single-route improvement algorithms cited previously.

Fisher and Jaikumar (1981) present a heuristic that solves an integer program for a generalized assignment problem to optimally assign customers to vehicles. Among the customers, "seed" customers are selected either manually (via the user preferences, expertise, etc.) or heuristically. Customers are then allocated to the selected "seed" customers at a minimum cost. The cost of allocating a customer to a "seed" customer is estimated by the route formed by the two customers through the depot. This makes the objective function of the generalized assignment problem an approximation of the cost of the TSP. A complete solution is obtained by applying a TSP heuristic to

each vehicle (i.e., each cluster). Computational tests are performed on 12 test problems taken from the literature. The problems range in size from 50 to 199 customers, 5 to 19 vehicles, and a single depot. In terms of total traveled distance, the algorithm provided 9 new best solutions to the VRP and equaled 2 of the best existing solutions. In general, the generalized assignment based heuristic outperforms both the Clarke and Wright (1964) and the Gillett and Miller (1974) algorithms.

Bramel and Simchi-Levi (1995) present a location based heuristic that clusters customers by approximating the VRP with a *capacitated concentrator location problem* (CCLP). The approximated solution to the CCLP specifies "seed" customers and the customers allocated to them without violating capacity constraints. They use two ways to determine the cost of allocating a customer to a "seed" customer. In the first implementation, the cost corresponds to the length of the route through the two customers and the depot (similar to Fisher and Jaikumar, 1981). In the second implementation, the cost corresponds to the direct round-trip between the "seed" and the customer. A computational experiment is conducted on 7 standard problem sets taken from Christofides et al. (1979), ranging in size from 50 to 199 customers. In general, the two implementations improved only two of the existing solutions on the test problems. The philosophy of the first implementation is similar to Fisher and Jaikumar (1981), but provides better solutions in only 2 of the 7 cases. However, the second implementation has the advantage of being asymptotically optimal, which means that the deviation from the optimal solution tends towards zero as the number of customers increases.

2.4.8 Route-First Cluster-Second Algorithm

Route-first, cluster-second algorithms are two phase methods that construct a TSP tour during phase 1 that connects all the customers and then divides them into segments in phase 2, subject to the vehicle capacity constraints. Each segment is then serviced by a vehicle. Beasley (1983) considers a route-first cluster-second method for the VRP. Although the author cites similar approaches used in the literature, this work appears to be the first attempt to evaluate this type of algorithm on

standard VRPs. The algorithm randomly generates an initial "giant" tour visiting all the customers and excluding the depot and improves the tour via a 2-opt operator. In order to reduce the fleet size as much as possible, a large positive constant is added to the inter-customer distances. The "giant" tour is then partitioned by means of a shortest path algorithm. The routes in the final partitions are improved using a 3-opt operator. Computational results and a comparison with the savings algorithm of Clarke and Wright (1964) use the 10 problems taken from Eilon et al. (1971). For each problem, 25 "giant" tours are generated and the best solution is kept. The developed algorithm clearly outperforms the algorithm of Clarke and Wright. However, no other existing results on VRP instances are used for benchmarking. In the evolutionary algorithm of Prins (2004), giant tours are created with the ordered sequence of customers and a splitting procedure is utilized to determine the best way to separate the routes in the giant tour.

2.4.9 Matching Algorithm

Altinkemer and Gavish (1991) present a parallel savings based algorithm (PSA) which is an improvement of the saving algorithm of Clarke and Wright (1964). In the algorithm of Clarke and Wright only one pair of routes can be merged whereas multiple pairs of routes can be merged at each iteration of the proposed PSA. The number of pairs of routes merged is determined by solving a weighted matching problem (i.e., finding the largest size set of edges such that each customer is linked to at most one route at the maximum saving possible). As in Clarke and Wright's algorithm, every customer is initially served by a separate vehicle. When two routes merge, the customers in both routes are served by a single route. At each iteration, the exact savings obtained by merging routes p and q, S_{pq}, is calculated for all possible pairs of routes without exceeding the vehicle capacity. New routes are formed by merging the matched routes. The procedure repeats until no more routes can be merged.

One disadvantage of this PSA algorithm is that a TSP is solved at each iteration to calculate the savings S_{pq} for every pair of routes considered for merging. Thus, two additional versions are

proposed to estimate the savings instead of solving a TSP. One of them solves a TSP but only after obtaining the final routes. The three algorithms are tested using 14 problem sets taken from the literature. The problem size ranges from 50 to 200 customers, customer demands are not equal, and the vehicle capacity is the same. The PSA improves 6 of the previously known best solutions for the 14 problem sets. From those 6 improved solutions, 4 are obtained from PSA and the others from the version that solves the TSP after the final routes. As expected, CPU times from PSA are larger than from the two additional versions due to the solution of TSPs at each iteration. CPU times from the two versions are quite similar, which reveals that most of the computing time is spent solving the weighted matching problem.

2.4.10 Multiple-Route Improvement Algorithms

Multiple-route improvement algorithms improve an existing VRP solution by combining and modifying various routes. Most of these algorithms use the operators of Van Breedam (1995) and Kindervater and Savelsbergh (1997) or more complex operators such as the ones used by Thompson and Psaraftis (1993).

Kindervater and Savelsbergh (1997) describe three basic k-exchange operators that relocate customers between two routes: (1) *relocation*, which moves k consecutive customers (usually $k \leq 3$) from one route to another; (2) *exchange*, which allows any two routes to exchange k consecutive customers; and (3) *crossover*, which allows any two routes to exchange k consecutive customers in such a way that the last part of either route becomes the last part of the other. Kindervater and Savelsbergh (1997) report finding no studies that compare these three operators with other algorithms.

Thompson and Psaraftis (1993) investigate a neighborhood search based on *cyclic k-transfers* to solve the multi-vehicle routing problem. The procedure attempts to improve a solution by transferring k demands among a cyclic permutation of routes. They also study a special case, *b-cyclic k-transfers*, which specifies the transfer among b routes. This search procedure is complex because

a TSP is solved for each modified route in the permutation subset to evaluate the transfer cost and the number of k demands that can be transferred is large. The cyclic transfer algorithms are tested using 3 standard VRPs taken from Eilon et al. (1971). These problems have 50, 75, and 100 customers. The best solutions found are compared with solutions in the literature obtained with various algorithms (including Clarke and Wright, 1964; Gillett and Miller, 1974; Fisher and Jaikumar, 1981). The results reveal that the solution quality in terms of total distance is not better than Fisher and Jaikumar (1981) but is quite close to it. The algorithms are also tested using six test sets from Solomon (R1, C1, RC1, R2, C2, RC2) for the VRP with time windows, each set contain from 7-12 instances all with 100 customers. On average, the cyclic transfer algorithms provide a better solution in 4 of the test sets.

Table 2.2 provides a brief summary of the classical heuristic algorithms defined for and tested on VRP-types of problems.

2.5 Metaheuristics for the VRP

Metaheuristics are solution methods that guide a subordinate heuristic algorithm to escape from regions having local optimal solutions and thus perform a more effective search in the solution space. These methods can use different techniques to avoid local optimum, including randomization, population-based procedures, and memory-based techniques. This section overviews different metaheuristic approaches used to solve the VRP, including simulated annealing, tabu search, genetic algorithms, and ant colony optimization.

2.5.1 Simulated Annealing (SA)

This technique was first introduced by Kirkpatrick et al. (1983) as an analogy between the annealing process of solids and the problem of solving combinatorial optimization with the objective of converging to an optimal solution. The analogy provides a useful connection between the behavior

Table 2.2: Representative Classical Heuristic Algorithms for the VRP.

Algorithm	Year	Description & Remarks
Dantzig and Ramser	1959	Constructive algorithm. First approach
Clarke and Wright	1964	Saving. Concurrent & sequential
Wren and Holliday	1972	Sweep Algorithm. Multiple depots
Lin and Kernighan	1973	Single-route improvement
		Sequential k-exchange
Gillett and Miller	1974	Sweep Algorithm. Single depot
Or	1976	Single-route improvement
		Consecutive customers relocation
Foster and Ryan	1976	Petal algorithm. Optimal petal solution
Mole and Jameson	1976	Sequential Route-Building
		Insertion position check
Christofides et al.	1979	Sequential Route-Building
		Sequential & Parallel construction
Fisher and Jaikumar	1981	Cluster-First Route-Second
		Generalized Assignment + TSP
Beasley	1983	Route-First Cluster-Second
Altinkemer and Gavish	1991	Matching Algorithm. Matching clusters
Ryan et al.	1993	Petal algorithm
Thompson and Psaraftis	1993	Multiple-Route Improvement
		b-cyclic k-transfer
Potvin and Rousseau	1995	Single-route improvement. Based on 2-opt*
Bramel and Simchi-Levi	1995	Cluster-First Route-Second
Renaud et al.	1996	Single-route improvement
Kindervater and Savelsbergh	1997	Multiple-Route Improvement

of systems in thermal equilibrium at a finite temperature and a combinatorial optimization, which provides a new method to solve this type of problem.

To overcome local optimum in optimization applications, SA allows hill-climbing (non-improving) moves with a probability that depends on the magnitude of the increase in the cost function and on the search time to date. Osman (1993) developed a SA algorithm where the local search approach is based on a λ-interchange descent mechanism to explore new solutions. They use $\lambda = 1$, so the neighboring solutions can be obtained by exchanging a customer between any pair of routes. The criterion to select the best neighbor solution uses two different strategies: the best-improvement strategy, which examines all candidates in the neighborhood and selects the one with the best solution cost $C(S')$ according to an acceptance criterion, and the first-improvement strategy, which immediately accepts the first candidate satisfying the acceptance criterion. The algorithm uses: 1) a starting and final temperature (T_s and T_f), 2) a decrement rule (i.e., a temperature reduction factor α) to change the temperature T_k after each iteration k, 3) an update rule for temperature reset variables T_r after the system freezes, and 4) a stopping criterion, which is the total number of temperature resets performed since the best solution was found. The best solution found during the search, S_b, is kept instead of the one obtained in the last iteration. The algorithm performs a single iteration at every temperature level.

To evaluate SA performance on 17 test problems, Osman (1993) used the algorithm of Clarke and Wright and two hybrid approaches combining the SA approach with the tabu search (TS). The hybrid approaches use the two selection strategies previously described: best-improvement and first-improvement. The results of the experiment are summarized as:

- SA outperforms the existing heuristics and provides new best solutions.
- Both hybrid approaches with best-improve strategy and first-improve strategy outperformed the SA method in both computational time and solution quality.
- SA reduced the total number of vehicles used with respect to the existing solutions.

The Osman SA method was implemented before the TS approach of Gendreau et al. (1994). However, the latter TS approach provided better solutions in terms of the objective value in three of the problem sets, equal solutions for four problems, and worse solutions for six problems. In other instances, TS provided better objective functions but used a higher number of vehicles. Better results can be obtained with the SA approach by using a post-optimization procedure which was not used by Osman.

Van Breedam (1995) describes the use of SA-based improvement methods for the VRP. Feasible solutions are found by using three different multiple-route improvement heuristics and those embedded in a SA metaheuristic. The descent implementations use *string relocation* to insert a customer(s) from one route into another, *string exchange* to exchange customer(s) between any two routes, and *string mix* to combine those into a single operator. The Van Breedam SA implementation uses a traditional scheme. A maximum number of feasible solutions are generated for every temperature and the best one is kept. To offset the limitations of convergence to local optimum, a non-improving neighbor solution is accepted with some probability. After a fixed number of iterations, the SA algorithm stops. As a particular characteristic, they use a distance limit on potential moves to restrict the neighborhood and thus improve computational times.

The descent algorithms and the same algorithms combined with the SA metaheuristic were tested using the 14 test problems of Christofides et al. (1979) to compare their results. To evaluate the quality of the solutions obtained with the SA-based algorithm, a comparison was carried out with existing TS solutions and the SA solutions of Osman (1993). Results reveal that, as expected, the SA-based versions of the three improvement methods gave better results in comparison with the pure descent variants. The comparison with Osman's SA algorithm indicated that both implementations provide solutions with similar quality and neither one dominates the other. However, the TS implementations clearly outperformed SA on the problem sets in terms of both solution quality and computation time.

2.5.2 Tabu Search (TS)

TS is a strategy that uses local search and flexible memory structures to learn from the search history and overcome local optimum (Glover, 1986). A local search is performed until a local optimum is found. Systematic up-hill moves are used to escape regions of local optimum, explore the search landscape and, hopefully, find a global optimum. Cycling moves are forbidden by the use of a tabu list, or short-term memory, that records the recent history of the search. Promising solutions, or attributes, are reinforced by the use of a recency memory, or intermediate-term memory, that records the number of consecutive iterations that some attributes have been present in the current solution. New solution regions are explored by diversifying the search using a frequency memory, or long-term memory, that records the number of iterations that some attributes have been present in a selected solution during the search and using that information to build new, more diverse solutions.

Taillard (1993) presents a parallel iterative search method for the VRP based on TS and two partitioning methods. The algorithm partitions a full VRP into subproblems defined by sectors and polar regions. Each subproblem is solved independently using TS. Once the subproblems are solved, they are grouped together to construct a full solution to the original problem. This solution is partitioned again and the process repeats. The TS is based on two neighborhoods created by moving a customer from one route to another and by exchanging customers between routes. The search is diversified by penalizing the moves that are frequently performed. The penalty value varies with the frequency of moves and with a weight that is randomly generated at each iteration within a range whose length depends on the move value and the problem size. Similarly, the tabu tenure is determined randomly and is problem size dependent. Although a TSP is approximately solved to determine the cost of a move, routes are exactly solved periodically during the search to produce optimal routes. The algorithm partitions the problem differently when customers are uniformly distributed around the depot and when they are not. The algorithm provides solutions with a quality that is at least as good as the best published values on the 14 problems proposed in Christofides et al. (1979) and improves 5 of them.

Gendreau et al. (1994) propose a TS approach, TABUROUTE, to solve the VRP with capacity and length restrictions. In their approach, the neighborhood is built by removing a customer from one route and reinserting it into another. The reinsertion method differs from a regular insertion in the sense that, only routes containing nearby customers are considered for insertion. The current route is optimized when the customer is reinserted into another route.

The procedure considers the set of all possible customers to reinsert. It randomly selects a subset of customers. Then, for the selected candidates from the subset, all the potential moves are evaluated moving the customer from the current route to another empty route or a route including the closest customers. The insertion cost is then calculated. They use a simple aspiration criterion where a tabu candidate is selected only if its value yields a solution better that the best found so far. A penalty value is assigned based on any excess in the vehicle capacity and the number of times a customer has been removed (this is to diversify the search). The candidate with the lowest value is identified and selected. The current solution is updated with the best candidate unless the following three conditions are true: a) the penalty value is greater than the current, b) the current solution is feasible, and c) the current solution was not improved in the previous iteration by rearranging all the routes independently. If the current solution is not updated, it is obtained just by rearranging the routes. If no improvement has been obtained for a maximum number of iterations, the algorithm stops. To intensify the search, the procedure is executed using the best feasible or infeasible solution found so far.

Their results show that TS is a good alternative to solve VRPs, and typically outperforms the existing heuristics. The implementation of Gendreau et al. (1994) outperforms the best known solutions in 11 of 14 test problems. The authors conclude that the success of the procedure lies in the fact that infeasible solutions are allowed through penalty terms and also the current solution is perturbed periodically, so the risk of getting trapped in a local optimum is reduced.

Rochat and Taillard (1995) present a very interesting and novel probabilistic technique to diversify, intensify, and parallelize a local search for VRPs. This technique uses a local search which

is based on the TS of Taillard (1993), but improves the partitioning procedure and replaces the exact algorithm for optimizing routes with a heuristic approach. The novelty of this technique relies on the method used to exploit the concepts of diversification and intensification. The search is diversified by generating with the local search a set of unique solutions. There is a random component in the local search, so the algorithm produces different solutions each run. A pool of *elite routes* is created from these initial solutions. These *elite routes* are probabilistically extracted and used to generate a partial solution, S. routes belonging to better solutions are more likely to be extracted. Customers not included in S are allocated by solving an independent VRP, which produces a feasible solution S'. Then, solutions S and S' are combined to create a feasible solution to the initial VRP. This solution is then improved using a local search. The best routes of the improved solution are included in the pool. Identical routes are not removed from the pool and, as the pool grows, there exist routes that are not modified during the search. These routes contain strongly determined variables and are more likely to be included in the final, hopefully optimal, solution. As far as the process goes, the best routes are more frequently extracted from the pool and the search progressively changes from a diversification to an intensification approach.

The behavior of the diversification and intensification technique is analyzed in terms of computation time with respect to the TS of Taillard (1993) and Gendreau et al. (1994). The new technique is much faster than the previous TS approaches to produce solutions at certain average levels of quality above the best known solutions, especially on problems with more than 100 customers. The proposed technique accompanied by a postoptimization procedure (based on a set partitioning problem) improves the quality of 4 of the best solutions reported in the literature for the VRP. The TS used in the experiment (a modified version of Taillard, 1993) improves or reaches the quality of about 27 out of 56 best solutions previously published for the VRP with time windows.

Xu and Kelly (1996) develop a TS approach composed of a network flow model, a direct customer swap, and a TSP component to solve the classical VRP. This TS heuristic relaxes the capacity constraints through the use of penalty parameters dynamically changed during the search. A network flow model is used to optimally exchange a number c of customers (which is cyclicly changed

during the search) between routes. Although the exchanges are made to local optimality, the exchange costs may only estimate the actual move costs as the least insertion positions need to be found. This philosophy of optimally solving an approximate model is similar to the generalized assignment heuristic of Fisher and Jaikumar (1981), mentioned previously. The direct customer swap procedure simply exchanges two customers between two routes. The TSP component is employed as an improvement method that uses the 3-opt operator and a TS heuristic for the TSP. The tabu tenure is randomly generated within a fixed interval. The diversification strategy relies on a long-term, frequency-based memory that gives preference to those customers that appear less frequently in a specific route. The search is intensified by restarting the search periodically from an *elite* solution obtained from the repository.

An important characteristic of this TS implementation is that infeasible solutions are allowed via penalty parameters. The search oscillates between feasibility and infeasibility. The network flow moves dominate the search whereas direct customer swaps are executed periodically, or under some specific conditions, to help produce feasible solutions of high quality when infeasible solutions exhibit low capacity violation. If the capacity violation is high, the penalty parameters are modified to drive the search back to feasibility. Computational tests are conducted on 7 benchmark problems. The developed TS approach provides high quality solutions in reasonable times when compared with the best solutions published (Taillard, 1993; Rochat and Taillard, 1995). Compared to the TS of Gendreau et al. (1994), the proposed TS provides slightly better solutions on 3 out of 7 tested problems. Another test conducted determined that the network flow moves seems to provide the highest contribution to the solution quality.

Toth and Vigo (2003) present a variant of the traditional TS approach, called GTS, that uses a candidate list strategy to drastically restrict the search neighborhoods. This approach uses four neighborhoods based on the classical k-exchange (i.e., $k \leq 4$) and has similarities to the TS of Xu and Kelly: 1) infeasible solutions are allowed during the search by using penalty parameters; 2) the penalty values are dynamically updated during the search; and 3) the tabu tenure is randomly generated within a fixed interval. However, the characteristic that differentiates this TS from pre-

vious approaches is the use of *granular neighborhoods* that discard a large number of unpromising candidate moves and allow exploring only a small subset of them, containing the most promising ones. This is accomplished by reducing the original complete graph $G = (V, A)$ to a new sparse graph $G' = (V, A')$ containing *short* arcs whose cost is not greater than the *granularity threshold* value:

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

where β is a positive *sparsification parameter*, z' is the value of a heuristic solution provided by any traditional heuristic, n represents the number of customers, and K corresponds to the fleet size. The sparse graph also contains all arcs incident to the depot, those belonging to the best solution found, and to the current solution. The same graph is periodically rebuilt using an appropriate β value which computationally gives the best performance. Intensification and diversification strategies are adopted by dynamically modifying the structure of the sparse graph. That is, small β values produce an intensified search, whereas large β values diversify the search. Whenever the current best solution is not improved after certain iterations, the β value is increased and a new sparse graph is built.

The GTS algorithm was tested on instances from the literature with up to 500 customers. The quality of the obtained solutions was compared with those obtained by Gendreau et al. (1994) and Xu and Kelly (1996). On problems with less than 200 customers, the solution quality was comparable to or better than the solutions obtained by the others. In terms of computing time, the GTS algorithm was on average five times faster than the other approaches. On problems with more than 200 customers, however, the GTS algorithm was able to improve some of the best existing solutions. Note that Toth and Vigo (2003) provide a detailed summary of the commonly used VRP instances for benchmarking including the number of customers and vehicles, vehicle capacities, route maximum capacities and service times for some instances, the reference to the paper where the instances are first described, the best solutions known, and the paper where the best solution is

reported.

2.5.3 Genetic Algorithms (GAs)

Genetic algorithms are evolutionary algorithms that use a population of potential solutions, called chromosomes, and analogies of genetic crossover and mutation to recombine chromosomes and produce new solutions. The search is guided by evaluating the objective function of all the solutions in the population. Solutions with a better value, or fitness, replace old solutions and "survive" into subsequent generations.

Baker and Ayechew (2003) apply a GA to the VRP. In their approach, given n customers and m vehicles, the chromosome for a solution is a string of size n where each gene represents a customer and has a decimal coded value in the range $[1, m]$ corresponding to the vehicle number to which the customer is assigned. This means that the vehicle routes are not specified explicitly, but once we know which vehicles visit the customers, it is possible to construct the routes.

They use two methods to generate an initial population of structured solutions. The first one is based on the sweep algorithm of Gillett and Miller (1974) and the second one uses the assignment heuristic of Fisher and Jaikumar (1981). The authors performed preliminary tests and realized that using random solutions slowed the convergence of the GA; this method was not used in the final version of the GA. Thus, each structured approach was used to generate half the initial population while the individuals are ensured unique. The population size varies with the problem size. For larger problems, it is 50 whereas for smaller problems the population size is 30.

Each generation parents are selected for reproduction by the binary tournament method. To select each parent, two individuals are chosen at random and the one with the best fitness value is selected. The offsprings are produced from the parents using a standard 2-point crossover procedure in which the two points are selected randomly. Offsprings that duplicate existing members are discarded. In addition to the crossover procedure, mutation was applied to the offspring. In mutation, two genes (or customers) are selected at random and their values are exchanged whenever the two

customers belong to two different vehicles. This procedure is equivalent to swapping two customers from different routes, commonly used to solve the TSP.

To select the members of the population to be replaced, a *ranking replacement* method is used. In this method, the population is divided into four subsets according to the fitness and unfitness value of the offspring and the members. The unfitness value is defined as the excess in capacity and/or distance in the violated constraints. The member with the worst unfitness is selected for replacement from the first non-empty subset. If the offspring does not duplicate another member, it enters and the chosen member is removed from the population. The stopping criterion can be based on the number of generations, number of generations with no improvements, or elapsed time. In either case, no improvements were found in terms of solution quality.

The results did not demonstrate a benefit to using a random initial population. The GA converges slower when a random initial population is generated instead of a structured start. Averaged over 14 problems, the 2-point crossover produced lower total distances for the best population member than the 1-point crossover; the *ranking replacement* method produced lower total distances than the worst fitness/unfitness replacement method. "The best known results for VRP have been obtained with tabu search and simulated annealing", but "it appears that GAs have not yet made a great impact on the VRP" (Baker and Ayechew, 2003).

2.5.4 Ant Colony Optimization (ACO)

Ant colony optimization metaheuristics intend to mimic the foraging behavior of real ants to solve real-life path finding problems, such as the search for food. Ants secrete pheromone along the path they use when traveling from the nest to the place where the food is located. This substance allows the ants to communicate indirectly so other ants can follow the same path. As more ants follow the path, the route becomes more attractive for subsequent ants. ACO is based on the interaction of a colony of "artificial" ants using "artificial" pheromone trails. These trails provide numerical information which is adapted during the algorithm run and used by the artificial ants to probabilistically

construct a solution to the path-finding problem.

Scientists have explored different hard combinatorial optimization problems using this ant behavior analogy. Kawamura et al. (1998), Bullnheimer et al. (1999), and Mazzeo and Loiseau (2004) consider the ACO applied to the VRP. Kawamura et al. (1998) propose a cooperative search algorithm based on pheromone communication for solving the VRP. The algorithm consists of multi-agents that provide a partial solution consisting of a single vehicle route. The initial solution for each search agent is randomly generated. At each iteration, the route of each agent is modified by replacing a customer at random and optimizing the resulting route by means of a classical 2-opt operator. If an improvement is found, the partial solution of the agent is updated. Otherwise, it is accepted with a probability that depends on the pheromone information associated to the replaced customer and the search agent. The procedure is executed a fixed number of iterations. They generate two test problems with 60 and 30 customers and found the optimal solution in both cases. However, no comparisons are made with other existing VRP solution methods.

Bullnheimer et al. (1999) describe how to construct vehicle routes and how to update the pheromone trails in a basic ant system for the VRP as well as their improved ant system algorithm. First, for every ant, the construction of the routes is done by means of a local heuristic function where ants successively choose customers to visit until all customers are visited. Whenever a capacity constraint is violated, the ant returns to the central depot and starts a new route. The information regarding how good was a customer in previous iterations is stored in the pheromone trails τ_{ij} associated with the arc connecting two customers, whereas the information of how good is the next arc to take (i.e., the visibility, denoted by η_{ij}) is used by the local heuristic function. In the local heuristic, the next customer to visit depends on a probability distribution constructed using the trail intensities, the visibility, and two other parameters, α and β, to establish the relative influence of the visibility versus the pheromone trails, respectively. After an artificial ant has constructed a feasible solution, the pheromone trails are updated. Bullnheimer et al. suggest different techniques for trails update, such as using only the contribution of the best ant or using an elite-list of artificial ants.

To improve the performance of the basic ant system, they solve the TSP for the routes generated by the ants and use a candidate list sorted by the increasing distances for the selection of customers in the local search. The performance of the ACO approach was compared to other heuristics in 14 test problems. Results conclude that, although the best known solutions could not be improved on the problems tested, the proposed ant system approach is competitive in terms of processing time and becomes a viable alternative to solve vehicle routing problems.

Mazzeo and Loiseau (2004) investigated different alternatives for each component of the ACO algorithm. For example, the route building can be sequential or parallel. Sequential means that each ant constructs the route for a vehicle until the capacity is reached and then continues with other vehicles until all customers are visited, as implemented by Bullnheimer et al. (1999). In parallel means that each ant constructs the routes for all vehicles at the same time. At each iteration, only one customer is chosen, according to the transition rule. Then, the best route is extended. The transition rule might be *random-proportional* or *pseudo-random-proportional*. In the former case, the next customer is randomly selected based on a probability distribution. In the latter case, the customer is selected based on the same probability distribution and also on the best option. The pheromone update at the end of each iteration varies: use all the solutions, an elite-list of solutions, only information of the best solution of the previous iteration, or locally each time an ant moves from one customer to another. In addition, they use a reduced neighborhood list when the problem is large and an improving heuristic to modify the ant solutions after each iteration.

Mazzeo and Loiseau (2004) experimented to determine the best alternatives: parallel route building, best solution global pheromone update, a reduction of 25% of the candidate list, and randomly located ants in a number lower than the number of customers. The results obtained are similar to those of Bullnheimer et al. (1999) in the sense that ACO does not clearly outperform existing heuristics for the VRP, but is still a promising VRP technique. Table 2.3 provides a summary of metaheuristic algorithms defined for and tested on VRPs.

Table 2.3: Representative Metaheuristic Algorithms for the VRP.

Algorithm	Year	Description
Osman	1993	SA
Taillard	1993	TS
Gendreau et al.	1994	TS
Van Breedam	1995	SA
Rochat and Taillard	1995	TS
Xu and Kelly	1996	TS
Kawamura et al.	1998	ACO
Bullnheimer et al.	1999	ACO
Toth and Vigo	2003	TS
Baker and Ayechew	2003	GA
Mazzeo and Loiseau	2004	ACO

2.6 Static and Dynamic VRPs

In more realistic applications, some parameters in the VRP vary as a function of other parameters. Although any parameter may vary with others, such as weather and traffic conditions, they usually vary as a function of time. In the last decade, there has been increased interest in studying dynamic VRPs as a consequence of the technological revolution and advances in communications. Technology has yielded devices such as GPS and on-board computers that allow companies to continually update information and thus enhance the performance of decision systems. Although there exist various classification schemes for vehicle routing (Bodin and Golden, 1981; Desrochers et al., 1990; Psaraftis, 1995; Carlton, 1995), those schemes focus on problems where information relevant to their solution is not updated in real-time. Terms like *dynamic*, *stochastic*, and *real-time* are commonly used in recent VRP publications, but it is still unclear what these articles mean by stochastic, dynamic, or real-time VRPs. There does not seem to be any unified criteria to classify dynamic and real-time VRPs; authors seem to use the terms interchangeably.

In this section, a classification of VRPs is presented based on three criteria: staticism/dynamism of the parameters of the problem, the knowledge of information relevant to the design of its solution, and the method to model the unknown data (see Figure 2.1). This new classification uses up-to-date definitions to organize the different types of VRPs. See, for instance, the definitions and concepts by Bertsimas and Van Ryzin (1991) and Ghiani et al. (2003). Bertsimas and Van Ryzin define the probabilistic VRP as "inherently static and solved a priori using only probabilistic information". According to Ghiani et al., "A VRP is said to be *static* if its input data (travel times, demands,...) do not depend explicitly on time, otherwise it is *dynamic*. Moreover, a VRP is *deterministic* if all input data are known when designing vehicle routes, otherwise it is *stochastic*."

Currently, some authors agree about the definition of dynamic VRPs. Under this definition, a problem is dynamic if the input data is unknown, or partially known, at the time an initial solution is obtained; the unknown data is revealed as the current solution executes. This means that the information can change after the initial routes have been designed (see for instance Psaraftis, 1995; Ichoua et al., 2000; Ichoua et al., 2007; Larsen et al., 2007). The problem is static if the input data is known before the routes are designed and does not change afterwards. Clearly, this does not align with Bertsimas and Van Ryzin (1991) and Ghiani et al. (2003).

The concept of dynamism is broader. Let $x(t_{0^-})$ be a solution to a combinatorial optimization problem $P(a)$ with set of parameters a. Solution $x(t_{0^-})$ is obtained at time $t = t_{0^-}$ based on the known information, $I(t_{0^-})$. Let t_0 be the instant when the execution of solution $x(t_{0^-})$ begins and $\Delta t_e > 0$ be its execution time. The execution of the solution then ends at $t = t_0 + \Delta t_e$. Clearly, the problem is dynamic if there exists $0 < \Delta t < \Delta t_e$ such that $I(t_{0^-}) \neq I(t_{0^-} + \Delta t)$.

However, there are other instances where the problem is dynamic as well. Suppose some problem parameters are time-dependent, i.e., the problem is now $P(a(t))$. The problem is defined as $P(a(t_{0^-}))$ at $t = t_{0^-}$, while it is defined as $P(a(t_{0^-} + \Delta t))$ at $t = t_{0^-} + \Delta t$. $P(a(t))$ is dynamic if there exists $0 < \Delta t < \Delta t_e$ such that $a(t_{0^-}) \neq a(t_{0^-} + \Delta t)$. This means that, although the input data is known before the initial routes are designed and executed, time is considered in the

problem. For example, in the PVRP customers have fixed, and probably different, daily demands. Even if their demands are known before the routes are designed, days are considered a variable in the problem in order to produce a solution.

In the classification scheme illustrated in Figure 2.1, three factors formally define a problem type. The first factor defines the time-dependency of the problem parameters and/or the variability of the input data. In the context of VRPs, these parameters include travel times, customer demands, and customer presence. A problem is *dynamic* if: 1) some problem parameters are time-dependent, or 2) the input data when the solution is obtained and executed differs from that at the moment its execution ends; otherwise, it is *static*. The first factor is coded using α and β. The α defines whether problem parameters are time dependent, or not, $\overline{\alpha}$. The β defines whether the input data changes over time, or not, $\overline{\beta}$. Under this notation, a problem is: 1) dynamic if the logical expression $(\alpha + \beta)$ is true, or 2) static if it is false, i.e., $\overline{(\alpha + \beta)}$.

The second factor defines the availability, or knowledge, of the information relevant to solve the problem. A problem is *deterministic* if all the information relevant to solve the problem is available, or known. A problem is classified as deterministic if the logical value of parameter π is true. If a problem is not deterministic, its category depends on the third factor.

The third factor defines how the unknown information is modeled. A problem is *stochastic* if unknown information is forecasted or modeled probabilistically. In this case, unknown variables are modeled as random variables and historical data, for example, are used to estimate their values. This modeling method takes advantages of the potential benefits of considering the stochastic aspects of the problem (Ichoua et al., 2000). If no model can be used, the information remains unknown and the problem is classified as *real-time*. In such a case, an initial plan is based on the known information and the solution is updated, or re-optimized, regularly as the unknown variables are revealed during the operation. The third factor is coded with μ. Using π and μ, a problem is classified as: 1) stochastic if the logical expression $\bar{\pi} \cdot \mu$ is true, or 2) real-time if the logical expression $\bar{\pi} \cdot \bar{\mu}$ is true. Note that the method used to model the unknown information does not have anything to do with

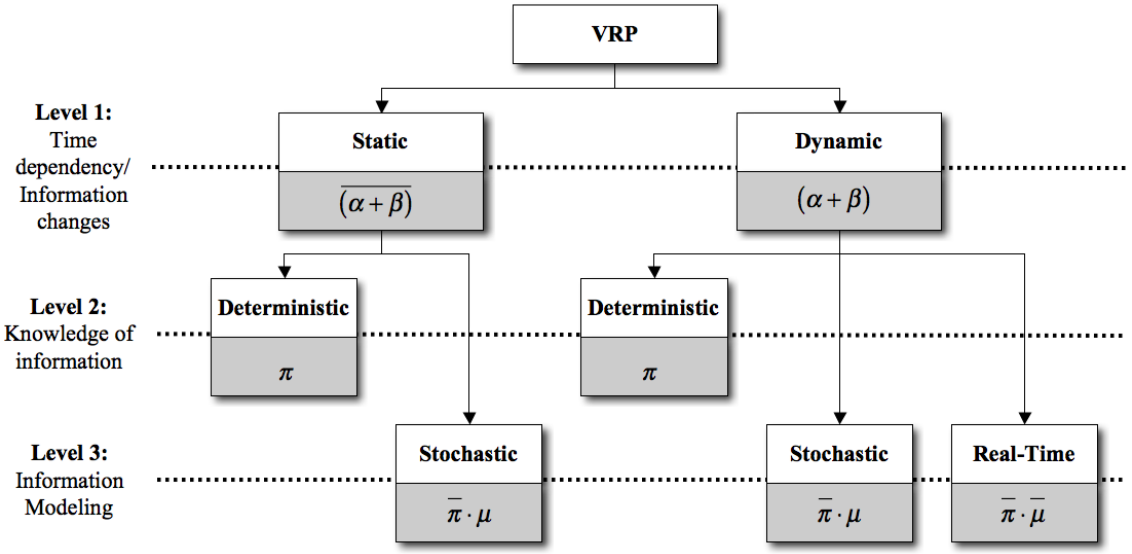

Figure 2.1: Classes of Dynamic VRP.

the solution method. A solution method utilizes the input data to produce a solution regardless of the nature of the data, deterministic or stochastic. Some authors refer to dynamic non-deterministic VRPs as real-time, or *on-line*, problems and do not differentiate the probabilistic estimation of the unknown problem information. This is the case of Ghiani et al. (2003), who state that a plan is not elaborated beforehand, but customer requests are assigned to vehicles in an on-going fashion as new data arrive.

The classification scheme illustrated in Figure 2.1 shows a tree structure with blocks representing the nodes of the tree. Each block has a label and a logical expression in terms of the notation previously described. Labels are used from top to bottom to construct the names of the problem types whereas logical expressions are used to classify the problems. For example, a problem is *static stochastic* if logical expressions $\overline{(\alpha+\beta)}$ and $\bar{\pi}\cdot\mu$ are both true. Similarly, a problem is *dynamic deterministic* if logical expressions $(\alpha+\beta)$ and π are both true. From the figure, we note that a static problem can be either deterministic or stochastic while a dynamic one can be deterministic, stochastic, or real-time. For simplicity, a dynamic real-time problem is simply called real-time. Examples of some well-known routing problems are presented below to provide a guide to classify

routing problems.

2.6.1 Static VRPs

Static Deterministic VRPs

In static deterministic routing problems there are no time-dependent parameters in the problem, the input data does not change over time, and all the data is known. Within this category, we find problems such as the capacited VRP, VRP with multiple depots, VRP with split deliveries, VRP with pick-ups and deliveries, and VRP with backhauls. The static version of the dial-a-ride problem (DARP) described in Cordeau and Laporte (2003) where all requests are known in advance also belongs to this category.

Static Stochastic VRPs

In static stochastic VRPs there are no time-dependent parameters in the problem and the input data does not change over time. Although not all the data is known, unknown data can be modeled using some probability distributions and random variables. These problems are solved in two stages. In the first stage, an *a priori* solution is found which takes into account the possible realizations of the problem. In the second stage, a recourse is applied to the solution found in the first stage according to the actual problem realization. In stochastic programming, two versions of the problem can be considered: 1) chance constrained programming (CCP) where the objective is to minimize the planned route costs subject to a bound in the probability of violating a capacity constraint without considering the costs of recourse, and 2) stochastic programming with recourse (SPR) where the objective is to minimize the cost of the solution found in the first stage plus the expected cost of recourse.

Within static stochastic problems, we find the VRP with stochastic demands, VRP with stochastic customers, VRP with stochastic customers and demands, VRP with stochastic travel times, and VRP with stochastic service times.

VRP With Stochastic Demands (VRPSD) The *Vehicle Routing Problem with Stochastic Demands* is probably the most studied of all the SVRPs. A good summary of contributions and publications can be found in Gendreau et al. (1996). The VRPSD is defined on a graph $G = (V, A)$, where $V = \{v_0, v_1, ..., v_n\}$ represents the set of vertices with known and fixed locations and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ represents the set of arcs. Vertex v_0 corresponds to a central depot where m identical vehicles with capacity Q are located, whereas the other vertices correspond to customers with a nonnegative demand q_i. Each arc (v_i, v_j) has an associated cost c_{ij}, usually representing distances or travel times. The particular characteristic of the VRPSD is that customer demands, q_i, are random variables, ξ_i, usually but not necessarily assumed to be independent. The value of ξ_i may become known upon arriving at the customer location or before leaving the previous customer.

VRP With Stochastic Customers (VRPSC) The *Vehicle Routing Problem with Stochastic Customers* can be defined on the same graph $G = (V, A)$ where vertex v_0 defines a central depot with some fleet of vehicles and the parameter c_{ij} is used to represent the cost of arc (v_i, v_j). The characteristic that differentiates this problem from other SVRPs is that each vertex v_i is present in the graph with probability p_i, whereas the customer demand q_i remains deterministic.

VRP With Stochastic Customers and Demands (VRPSCD) The *Vehicle Routing Problem with Stochastic Customers and Demands* is a combination of the VRPSC and the VRPSD. Absent customers are represented by a set of customers with zero demand and present customers have positive demands known only when the vehicle arrives at the customer location. Obviously, this problem is more difficult than the previous SVRP because both customers and their demands are uncertain when the solution is obtained.

VRP With Stochastic Travel Times (VRPST) The *Vehicle Routing Problem with Stochastic Travel Times* is also defined on graph $G = (V, A)$. However, the cost c_{ij} of arc (v_i, v_j) specifically

represents the travel time between the two vertices and is a random variable. Usually the fleet size is a decision variable in this problem, which is solved to maximize the probability of completing the tours within a given deadline.

VRP With Stochastic Service Time (VRPSST) The *Vehicle Routing Problem with Stochastic Service Time* is also defined on the graph $G = (V, A)$, but additional parameters are used. First, arc (v_i, v_j) has an associated travel time t_{ij}. Second, vehicles in the fleet have an operational time restriction τ. Third, all customer demands q_i are deterministic. And fourth, each vertex v_i $(i > 0)$ has an associated random variable, ξ_i, representing the service time with mean μ_i and and variance σ_i^2.

2.6.2 Dynamic VRPs

Dynamic Deterministic VRPs

In dynamic deterministic problems, there either are time-dependent parameters in the problem or the input data changes over time. However, any time-dependency or data variation is known. Within this category, we find problems such as the VRP with time windows, and periodic VRP. The dynamic version of the dial-a-ride problem described in Madsen et al. (1995), where customers provide time windows on the origin, destination, or both, also belongs to this category.

The time-dependent vehicle routing problem (TDVRP) is another dynamic deterministic problem, where travel times depend on the distance between customers and the time of day. The TDVRP accounts for variations in the travel time due to random events, such as traffic congestion, accidents, and weather conditions, and temporal variations that result from seasonal, weekly, daily, or hourly cycles. The assumption that the travel times are deterministically known and constant is an approximation of actual conditions (Malandraki and Daskin, 1992). Although the TDVRP traditionally assumes that only the travel times vary over time, there are a few recent studies that consider the

Table 2.4: Available literature on TSP with moving-customers.

Author(s)	Year	Description & Remarks
Helvig et al.	1998	Moving-target TSP. Related problems
Helvig et al.	2003	Moving-target TSP
Bourjolly et al.	2006	Moving-target TSP

possibility of customers with dynamic locations. The literature on VRP with moving-customers is very scarce. As far as we are aware, it can be summarized as shown in Table 2.4.

Helvig et al. (1998) and Helvig et al. (2003) introduce a generalization of the TSP with moving customers and propose the first heuristic for this type of problem. This research is motivated by some applications that reveal customers moving, such as a supply ship supplying patrolling boats, or an aircraft intercepting a number of mobile ground units. The moving-target TSP is formulated as follows. Given a set $S = \{s_1, ..., s_n\}$ of *targets*, each s_i moving at constant velocity $\vec{v_i}$ from an initial position p_i, and given a *pursuer* starting at the origin and having maximum speed $v > |\vec{v_i}|$, find the fastest tour starting and ending at the origin, which intercepts all targets.

Bourjolly et al. (2006) show the On-Orbit Servicing problem (OOS) as another line of research where moving-target TSP can be applied. The concept of OOS can be summarized as an orbital "depot" full of consumables and spare parts for spacecraft, and a "servicing platform" (based at this depot) to service a set of client spacecraft and then return to the depot to resupply. Their work is motivated by the problem of maintaining and repairing satellites in orbit, which are continuously in motion.

Dynamic Stochastic VRPs

In dynamic stochastic problems, there either are time-dependent parameters in the problem or the input data changes over time. Although not all the data is known, any unknown time-dependency or data variation is modeled using some probability distributions and random variables. Within this

category, we find emergency response systems, automobile road service, inventory routing problems (IRP), and other problems, such as the dynamic stochastic VRP defined by Bertsimas and Van Ryzin (1991), where demands vary over time, are stochastic, and follow a Poisson process. Weintraub et al. (1999) present an emergency service where repair vehicles are dispatched to service electrical breakdowns in a metropolitan area. Breakdowns can occur anywhere, but knowledge of the probability of breakdowns in the electrical system is used to predict the service requests, generate vehicle routes efficiently, and reduce the response time. In IRPs, a central supplier delivers goods to retailers on a repeated basis. The customer's consumption rate is unknown. If a reactive strategy is adopted, deliveries are made based on the actual inventory levels of retailers at a high cost due to potential stock-outs. However, if the customer's consumption rate is represented by a random variable with known probability distribution, a plan is made based on expected customer's consumption rate reducing the delivery costs of stock-outs. See the IRPs described by Berman and Larson (2001) and Jaillet et al. (2002). In spite of the elaboration of a plan, it can be modified as required. For example, Bell et al. (1983) present an IRP where historical data on customer demands is used to project the inventory level in each customer at any point in time. The problem is titled on-line because a detailed schedule is produced for a short planning horizon, but it can be updated when the actual inventory levels are disclosed.

Real-Time VRPs

In real-time problems, either there are time-dependent parameters in the problem or the input data changes over time, and there is no way to forecast the future. An example of a problem within this category is the classical dial-a-ride problem where customer requests are evaluated as they arrive. A particular characteristic of real-time VRPs is that arriving requests can be accepted, postponed, or rejected. Once a request is accepted, it must be serviced. Another example in this category is the dynamic VRP described by Bent and Van Hentenryck (2004), where requests have associated service times and time windows. Requests are numbered and evaluated in the chronological order of their arrival. The objective is to service as many customers as possible.

Chapter 3

Split Delivery VRP (SDVRP)

3.1 Problem Definition

In the VRP defined in Chapter 2, every customer is visited exactly once by exactly one vehicle, and the total demand of any route cannot exceed the vehicle capacity available. In reality, however, there may be cases where either a customer demand exceeds the vehicle capacity or a savings both in terms of the total distance and the number of vehicles can be obtained by allowing customers to be visited more than once. The split delivery vehicle routing problem (SDVRP) allows the use of multiple vehicles to satisfy demand points and potentially reduce the total cost by splitting deliveries (Dror and Trudeau, 1989). The computational complexity of this problem remains NP-hard (Dror and Trudeau, 1990). In the literature, a variant of the SDVRP, the k-SDVRP, can be found. Archetti et al. (2001) and Archetti et al. (2005) define the k-SDVRP as a special case of the SDVRP where vehicles have a capacity of k units, $k \in \mathbb{Z}^+$. They show that the 2-SDVRP is solvable in polynomial time when some specific conditions on the distances are satisfied, while the problem with $k \geq 3$ is NP-hard, as proved by Dror and Trudeau (1990). They also show that the 2-SDVRP may be reduced to a problem of possibly smaller size, where each customer has unitary demand.

In any case, vehicle capacities are usually greater than two units. As such, large instances of routing problems are commonly solved via heuristic algorithms so that good solutions are found in

reasonable computational time and with reasonable use of computational resources.

3.2 Benefits of SDVRP

At first glance, one may believe that the benefits of allowing split deliveries are small when the customer demands are either considerably small with respect to the vehicle capacity or close to the vehicle capacity. In their experimental study, Dror and Trudeau (1989) showed that if customer demands are low relative to the vehicle capacity and the triangular inequality holds ($c_{ij} \leq c_{ik} + c_{kj}$, for all i, j, and k), the split demand benefits are actually small. In contrast, when the customer demand is larger, at least 10% of the vehicle capacity, the cost of a SDVRP solution is considerably lower than the cost of a VRP solution. Figure 3.1 shows a simple example to illustrate the potential benefits of allowing split deliveries in terms of the number of vehicles and the solution cost. In this figure, there are 12 customers with demand $q_i = 60$ symmetrically located on a circle of radius r centered at the depot, where a fleet of vehicles with capacity $Q = 100$ is located. The optimal VRP solution is illustrated in Figure 3.1(a). It employs 12 vehicles with an average utilization of 60% and has a value of $z = 240r$ distance units. If split deliveries are allowed, all customer demands can be supplied as illustrated in Figure 3.1(b) with only 8 vehicles, an average utilization of 90%, and a solution value of $z = 201.4r$.

Gendreau (2006) extends the scenario depicted in Figure 3.1 and makes a theoretical generalization. He considers a circle centered on the depot with radius M, $n = 2k$ demand points with demand of k units located equidistantly on the circle, and vehicle capacity $Q = 2k - 1$. The optimal VRP solution consists of $2k$ independent tours with a cost of $2nM$. A feasible SDVRP solution consists of k routes visiting two consecutive customers (and leaving a unit of demand at the second customer) plus a last route visiting the the k remaining unsatisfied unit demands. If ϵ defines the distance between two consecutive customers in the circle, each one of the first k routes have a cost of $2M + \epsilon$ and the last route has a cost less than $2M + 2\pi M$. As the number of demand points grows, the cost of the SDVRP solution converges to be 50% of that of the VRP solution.

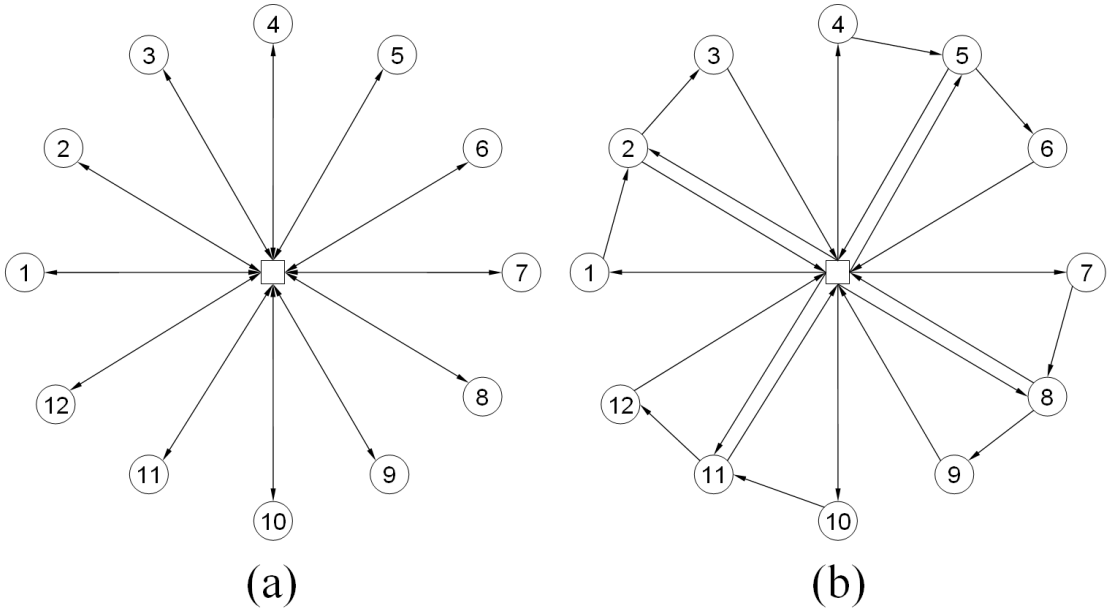

Figure 3.1: Illustration of savings by SDVRP: (a) VRP solution and (b) SDVRP solution.

Archetti et al. (2006) provide a worst-case analysis for the SDVRP and show that the savings in delivery costs obtained by allowing split deliveries is at most 50% and that bound is tight (i.e., there exist an example in which the value of the optimal VRP solution doubles the value of the optimal SDVRP solution). This analysis, however, does not provide any insight into the relation between the customer characteristics and the savings by allowing split deliveries. Archetti et al. (2008) characterize distribution environments and conduct a very thorough study (the most detailed found so far) of the value and benefits of allowing split deliveries. The focus of the study was to determine the practical implications of split deliveries for different customer characteristics, particularly in terms of geographic distributions of customers and demand distributions of customers. The benefits are quantified in: 1) the reduction in the number of routes required to fully supply all customer demands and 2) the reduction in delivery costs.

To quantify the reduction in the number of delivery routes, the ratio $\frac{r(VRP)}{r(SDVRP)}$ is studied, where the numerator and denominator represent the number of routes required in a VRP and a SDVRP solution to fully supply all demands, respectively. A mathematical analysis is used to

prove that the maximum reduction in the number of routes that can be achieved by allowing split deliveries is 50%. Moreover, the analysis confirms that the largest reduction is obtained when the mean customer demand is between 50% and 70% of the vehicle capacity and the demand variances are relatively small.

Regarding the reduction in delivery costs, they empirically study the ratio $\frac{z(VRP)}{z(SDVRP)}$ -where the numerator and denominator represent the VRP and SDVRP solutions, respectively- through the use of the granular TS heuristic for the VRP (Toth and Vigo, 2003) and a TS for the SDVRP (Archetti et al., 2006), which is described later in this chapter. To conduct the experiment, instances are constructed from three instances of Solomon's benchmark data for the VRP. To determine any dependence of the ratio on the geographic distribution of customers, one instance from each problem type (i.e., R101 for random locations, C101 for clustered locations, and RC101 for mixed locations) was used. For each set of customer locations, instances for various combinations of mean demand and demand variances are created. According to the results, there does not appear to be a dependence on the geographic distribution of customers, but there does appear to exist a dependence on the demand variance.

3.3 Various Models for the SDVRP

3.3.1 Formulation of Dror and Trudeau (1990)

Notation

C_{ij}	The distance ("cost") between demand points i and j
d_i	The daily demand at point i
Q_v	Capacity of vehicle v
x_{ij}^v	1 if vehicle v travels directly from point i to j, and $x_{ij}^v = 0$ otherwise
y_{iv}	The fraction of point demand i delivered by vehicle v
NV	The number of vehicles in the fleet
S	Set of all cycles on the set N which include the depot. The point 0 denotes the depot

$$\text{Minimize } z_s = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{v=1}^{NV} C_{ij} x_{ij}^v \tag{3.1}$$

Subject to:

$$\sum_{v=1}^{NV}\sum_{i=0}^{n} x_{ij}^{v} \geq 1; \text{for } j = 0, \ldots, n \tag{3.2}$$

$$\sum_{i=0}^{n} x_{ip}^{v} - \sum_{j=0}^{n} x_{pj}^{v} = 0; \text{for } p = 0, \ldots, n; v = 1, \ldots, NV \tag{3.3}$$

$$\sum_{v=1}^{NV} y_{iv} = 1; \text{for } i = 1, \ldots, n \tag{3.4}$$

$$\sum_{i=1}^{n} d_i y_{iv} \leq Q_v; \text{for } v = 1, \ldots, NV \tag{3.5}$$

$$y_{iv} \leq \sum_{j=0}^{n} x_{ji}^{v}; \text{for } i = 1, \ldots, n; v = 1, \ldots, NV \tag{3.6}$$

$$\mathbf{X} \in S \tag{3.7}$$

$$x_{ij}^{v} \in \{0, 1\}; \text{for } i = 0, \ldots, n; j = 0, \ldots, n; v = 1, \ldots, NV \tag{3.8}$$

$$y_{iv} \geq 0; \text{for } i = 1, \ldots, n; v = 1, \ldots, NV \tag{3.9}$$

Equation (3.1) denotes the objective function represented by the total distance traveled by the fleet of vehicles. Constraints (3.2) ensure that each customer is visited at least once. Constraints (3.3) stipulate that a vehicle visiting a customer has to leave the customer. Constraints (3.4) ensure that all customers are fully supplied. Constraints (3.5) ensure that the total demand of any route cannot exceed the vehicle capacity available. Constraints (3.6) state that a customer is supplied only if it is visited. Constraints (3.7) ensure that the tours start and end at the depot. Constraints (3.8) and (3.9) represent binary and non-negativity conditions for x_{ij}^{v} and y_{iv}, respectively.

3.3.2 Formulation of Frizzell and Giffin (1992)

Notation

d_{ij}	Cost of traveling between customer i and customer j
w_i	Demand of customer i
m_k	Capacity of vehicle k
x_{ijk}	1 if the vehicle k travels directly from customer i to customer j, and $x_{ijk} = 0$ otherwise
f_{ik}	The fraction of demand of customer i delivered by vehicle k
v	The number of vehicles in the fleet
n	The total number of customers
y_i	An arbitrary real number (i.e., usually taken to be the position number of customer i in a TSP tour)

$$\text{Minimize } \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{v} d_{ij}x_{ijk} \tag{3.10}$$

Subject to:

$$\sum_{k=1}^{v}\sum_{i=0}^{n} x_{ijk} \geq 1; \forall j = 0, \ldots, n \tag{3.11}$$

$$\sum_{i=0}^{n} x_{ihk} - \sum_{j=0}^{n} x_{hjk} = 0; \forall h = 0, \ldots, n; k = 1, \ldots, v \tag{3.12}$$

$$\sum_{k=1}^{v} f_{ik} = 1; \forall i = 1, \ldots, n \tag{3.13}$$

$$\sum_{i=1}^{n} w_i f_{ik} \leq m_k; \forall k = 1, \ldots, v \tag{3.14}$$

$$f_{ik} \leq \sum_{j=1}^{n} x_{jik}; \forall i = 1, \ldots, n; k = 1, \ldots, v \tag{3.15}$$

$$y_i - y_j + n x_{ijk} \leq n - 1; \forall 1 \leq i \neq j \leq n; 1 \leq k \leq v \tag{3.16}$$

$$x_{ijk} \in \{0, 1\}; \text{for } i, j = 0, \ldots, n; k = 1, \ldots, v \tag{3.17}$$

$$f_{ik} \geq 0; \text{for } i = 1, \ldots, n; k = 1, \ldots, v \tag{3.18}$$

Equation (3.10) denotes the objective function minimizing the total distance traveled by the fleet of vehicles. Constraints (3.11) ensure that each customer is visited at least once. Constraints (3.12) force a vehicle visiting a customer to leave the customer. Constraints (3.13) ensure that all customers are fully supplied. Constraints (3.14) ensure that the total demand of any route cannot exceed the vehicle capacity available. Constraints (3.15) ensure that a customer is supplied only if it is visited. Constraints (3.16) ensure that the tours start and end at the depot. Constraints (3.17) and (3.18) represent binary and non-negativity conditions for x_{ijk} and f_{ik}, respectively.

The subtour elimination constraints (3.16) of the above formulation differs from the one by Dror and Trudeau (1990), constraints (3.7). As stated by Frizzell and Giffin, these constraints can be replaced by any inequalities which prevent subtours.

3.3.3 Formulation of Dror et al. (1994)

Notation

c_{ij}	Nonnegative distance associated to arc (i, j)
q_i	Nonnegative demand at vertex i
Q_v	Capacity of vehicle v
x_{ijv}	1 if vehicle v travels directly from point i to j, and $x_{ijv} = 0$ otherwise
y_{iv}	Proportion of the ith customer demand delivered by vehicle v
$\bar{m}$	The number of vehicles in the fleet

$$\text{Minimize } \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{v=1}^{\bar{m}} c_{ij}x_{ijv} \tag{3.19}$$

Subject to:

$$\sum_{i=0}^{n} x_{ikv} - \sum_{j=0}^{n} x_{kjv} = 0; \forall k = 0, \ldots, n; v = 1, \ldots, \bar{m} \tag{3.20}$$

$$\sum_{v=1}^{\bar{m}} y_{ij} = 1; \forall i = 1, \ldots, n \tag{3.21}$$

$$\sum_{i=1}^{n} q_i y_{iv} \leq Q_v; \forall v = 1, \ldots, \bar{m} \tag{3.22}$$

$$\sum_{j=0}^{n} x_{ijv} \geq y_{iv} \leq; \forall i = 1, \ldots, n; v = 1, \ldots, \bar{m} \tag{3.23}$$

Subtour elimination and connectivity constraints (3.24)

$$x_{ijv} \in \{0,1\}; \forall i,j = 0,\ldots,n; v = 1,\ldots,\bar{m} \tag{3.25}$$

$$0 \leq y_{iv} \leq 1; \forall i = 1,\ldots,n; v = 1,\ldots,\bar{m} \tag{3.26}$$

Equation (3.19) denotes the objective function represented by the total distance traveled by the fleet of vehicles. Constraints (3.20) are flow conservation conditions. Constraints (3.21) ensure that all customers are fully supplied. Constraints (3.22) ensure that the total demand of any route cannot exceed the vehicle capacity available. Constraints (3.23) force that a customer is not supplied by a vehicle if the vehicle does not visit the customer. Constraints (3.24) are initially relaxed and successively introduced as part of the branch and bound exact algorithm of Dror et al. (1994) described later. Constraints (3.25) and (3.26) represent binary and non-negativity conditions for x_{ijv} and y_{iv}, respectively.

3.3.4 Formulation of Frizzell and Giffin (1995)

This mixed-integer programming formulation is based on Dror and Trudeau's (1990). It includes additional constraints for the time windows considered by Frizzell and Giffin.

Notation

d_{ij}	Cost of traveling between customer i and customer j
w_i	Demand of customer i
m_k	Capacity of vehicle k
x_{ijk}	1 if the vehicle k travels directly from customer i to customer j, and $x_{ijk} = 0$ otherwise
f_{ik}	The fraction of demand of customer i delivered by vehicle k
V	$\{1, \ldots, v\}$ the set of all vehicles in the fleet
N	$\{1, \ldots, n\}$ the set of all customers
D_{ik}	The departure time of vehicle k from customer i
e_i	The beginning of customer i's time window
t_{ij}	The time required to travel from customer i to customer j
l_i	The end of customer i's time window
S	Set of subtour breaking constraints

$$\text{Minimize } \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{v} d_{ij}x_{ijk} \tag{3.27}$$

Subject to:

$$\sum_{k=1}^{v}\sum_{i=0}^{n} x_{ijk} \geq 1; \forall j \in N \tag{3.28}$$

$$\sum_{i=0}^{n} x_{ihk} - \sum_{j=0}^{n} x_{hjk} = 0; \forall h \in N, k \in V \tag{3.29}$$

$$\sum_{k=1}^{v} f_{ik} = 1; \forall i \in N \tag{3.30}$$

$$\sum_{i=1}^{n} w_i f_{ik} \leq m_k; \forall k \in V \tag{3.31}$$

$$x_{ijk} = 1 \Rightarrow D_{ik} + t_{ij} \leq D_{jk}; \forall (i,j) \in N, k \in V \tag{3.32}$$

$$e_i \leq D_{ik} \leq l_i; \forall i \in N, k \in V \tag{3.33}$$

$$f_{ik} \leq \sum_{j=1}^{n} x_{jik}; \forall i \in N, k \in V \tag{3.34}$$

$$x_{ijk} \in S \tag{3.35}$$

$$x_{ijk} \in \{0,1\}; \forall (i,j) \in N, k \in V \tag{3.36}$$

$$f_{ik} \geq 0; \forall i \in N, k \in V \tag{3.37}$$

Equation (3.27) denotes the objective function represented by the total distance traveled by the fleet of vehicles. Constraints (3.28) ensure that each customer is visited at least once. Constraints (3.29) force a vehicle visiting a customer to leave the customer. Constraints (3.30) ensure that each customer is fully supplied. Constraints (3.31) ensure that the total demand of any route cannot exceed the vehicle capacity available. Constraints (3.32) and (3.33) ensure time windows feasibility. Constraints (3.34) ensure that a customer is supplied only if it is visited. Constraints (3.35) ensure that the tours start and end at the depot. Constraints (3.36) and (3.37) represent binary and non-negativity conditions for x_{ijk} and f_{ik}, respectively.

There is a small difference between the above mixed-integer programming formulation and the one of Dror and Trudeau (1990). Frizzell and Giffin (1995) consider the SDVRP with hard time windows (SDVRPTW) where vehicles must arrive at the customers within the time windows $[e_i, l_i]$. If the arrivals occur before e_i, vehicles wait at the customer. In Dror and Trudeau's formulation there is no infeasibility due to arrival times so a customer can be visited anytime. To handle the time windows, Frizzell and Giffin incorporate constraints (3.32) and (3.33). In addition, the parameter t_{ij} is used to set the travel time between two delivery points. This parameter does not explicitly appear in the notation given by the authors, but it is very easy to imply its meaning from the formulation.

3.3.5 Formulation of Belenguer et al. (2000)

This integer programming formulation differs from those previously in the fact that all vehicles are used, as explained below.

Notation

c_{ij}	The distance between clients i and j
d_i	Demand of client i
Q	Vehicle capacity
K	Fleet size
M	A constant "big enough"
V	$\{0, \ldots, n\}$ is the set of vertices. Vertex 0 represents the depot
E	$\{(i,j), i,j \in V, i < j\}$ is the set of edges
S	$S \subseteq V$ is a subset of vertices
$\bar{S}$	Set of vertices in $V \backslash S$
$d(S)$	Sum of the demands of the vertices in S
$\delta(S)$	Set of edges with an endpoint in S and the other in $\bar{S}$
x_{ij}^h	The number of times that vehicle h uses edge (i,j), $\forall (i,j) \in E$
y_{ih}	1 if vehicles h visits client i, and 0 otherwise, $\forall i \in V \backslash \{0\}$
d_i^h	The portion of the demand of client i serviced by vehicle h, $\forall i \in V \backslash \{0\}$

$$\text{Minimize} \sum_{h=1}^{K} \sum_{(i,j) \in E} c_{ij} x_{ij}^h \tag{3.38}$$

Subject to:

$$\sum_{h=1}^{K} y_i^h \geq 1; i = 1, \ldots, n \tag{3.39}$$

$$\sum_{i=1}^{n} y_i^h \geq 1; h = 1, \ldots, K \tag{3.40}$$

$$\sum_{i=1}^{n} x_{0i}^h \geq 2; h = 1, \ldots, K \tag{3.41}$$

$$\sum_{(i,j) \in \delta(i)} x_{ij}^h \geq 2 y_i^h; h = 1, \ldots, K; i = 1, \ldots, n \tag{3.42}$$

$$x_{ij}^h \leq M y_i^h; \forall (i,j) \in E; h = 1, \ldots, K; i = 1, \ldots, n \tag{3.43}$$

$$\sum_{(i,j) \in (S, \bar{S})} x_{ij}^h \geq 2 y_u^h; \forall S \subseteq V \backslash \{0\}; 2 \leq |S| \leq n-1; \forall u \in S; h = 1, \ldots, K \tag{3.44}$$

$$\sum_{h=1}^{K} \sum_{(i,j) \in (S, \bar{S})} x_{ij}^h \geq 2 \lceil \frac{d(S)}{Q} \rceil; \forall S \subseteq V \backslash \{0\}; 2 \leq |S| \leq n-1 \tag{3.45}$$

$$d_i^h \leq d_i y_i^h; h = 1, \ldots, K; i = 1, \ldots, n \tag{3.46}$$

$$\sum_{h=1}^{K} d_i^h = d_i; i = 1, \ldots, n \tag{3.47}$$

$$\sum_{i=1}^{n} d_i^h \leq Q; h = 1, \ldots, K \tag{3.48}$$

$$x_{ij}^h \in \mathbb{Z}^+; h = 1, \ldots, K; \forall (i,j) \in E \tag{3.49}$$

$$y_i^h \in \{0,1\}; h = 1, \ldots, K; i = 1, \ldots, n \tag{3.50}$$

$$d_i^h \in \mathbb{Z}^+; h = 1, \ldots, K; i = 1, \ldots, n \tag{3.51}$$

Equation (3.38) denotes the objective function represented by the total distance traveled by the fleet of vehicles. Constraints (3.39) ensure that each customer is visited at least once. Constraints (3.40) force that each vehicle visits at least one customer. Constraints (3.41) stipulate that all vehicles visit the depot. Constraints (3.42) ensure that tours start and end at the depot. Constraints (3.43) prevent use of an edge if a vehicle does not visit a customer. Constraints (3.44) force the use of an edge if a vehicle does visit a customer. Constraints (3.45) ensure the capacity constraint is not violated and prevent the existence of subtours. Constraints (3.46) stipulate that a vehicle does not supply an unvisited customer. Constraints (3.47) force supplying each customer's demand. Constraints (3.48) ensure the vehicle capacity is not exceeded. Constraints (3.49)-(3.51) represent binary and integer conditions for x_{ij}^h, y_i^h, and d_i^h, respectively.

3.4 Exact Algorithms for the SDVRP

Few algorithms are found in the literature to exactly solve the SDVRP. It is often impractical to look for optimal solutions of routing problems as the needed computational resources can be enormous. As far as we know, there exist only three exact approaches for the SDVRP.

3.4.1 Branch and Bound Algorithm of Dror et al. (1994)

Dror et al. (1994) propose an integer linear programming formulation and describe a B&B algorithm based on new classes of valid inequalities for the SDVRP. The algorithm uses the formulation described previously in Section 3.3.3 and the heuristic approach of Dror and Trudeau (1989) to calculate an upper bound. The heuristic approach is described later in this chapter. A lower bound is computed by relaxing the original formulation, adding new valid constraints, and using the simplex method. Although a full implementation of the B&B algorithm would provide an optimal solution, the relaxed problem is solved only at the root of the B&B tree as the objective of the study is to determine the strength of the inequalities. The gap between the heuristic solution and the lower bound is drastically reduced by the inclusion of various cuts in problems with 10, 15 and 20 customers. A problem with 10 customers is solved to optimality. They point to the low gaps obtained with the B&B algorithm to indicate the quality of their heuristic approach.

3.4.2 Column Generation Algorithm of Sierksma and Tijssen (1998)

Sierksma and Tijssen (1998) use a SDVRP applied to the transportation schedule of helicopters to offshore platforms in the North Sea for crew exchange of people employed on those platforms. The helicopters are based at an airport near Amsterdam. They propose a set-covering formulation for the SDVRP and solve its relaxation using a simplex algorithm and a column generation technique that includes a knapsack problem and several TSPs. The solution is a non-integer optimal schedule. This linear solution is transformed into an integer, not necessarily optimal, solution by means of an

iterative rounding procedure. Finding the exact solution takes a considerable amount of time and is suitable for long-term planning. However, given the size of the problem (51 platforms or customers), a Cluster-and-Route procedure is proposed that obtains an approximate solution much quicker. The heuristic procedure clusters and routes the customers but differs from previous clustering procedures in the fact that the clustering and routing are performed simultaneously. The farthest unserved customer from the depot is selected as a seed and the closest unserved customers are routed with it. When the tour capacity is reached, a new seed customer is selected and the procedure is repeated until all demands are fully supplied.

The exact, rounding, and heuristic procedures are tested on the problem having 51 real platforms with 11 different demand quantities. The results are compared to solutions obtained with modified versions of the sweep algorithm of Gillett and Miller (1974) and the saving algorithm of Clarke and Wright (1964). The gap between the lower non-integer bound and the solutions obtained with both the rounding procedure and the cluster-and-route approach are less than 5%. Although the developed heuristic approach improved over the other two modified algorithms, the two prior algorithms were not intended for SDVRP instances, so it is difficult to make a fair comparison.

3.4.3 Dynamic Programming Formulation of Lee et al. (2006)

Lee et al. (2006) propose an entirely new approach for the multiple-vehicle routing problem with split pick-ups (mVRPSP) based on a deterministic dynamic program model and a shortest path search algorithm. The mVRPSP is equivalent to the SDVRP, but vehicles are to pickup supplies from different suppliers and then go back to the depot. Based on some properties of optimal solutions of the mVRPSP, they reformulate the original dynamic program to find an equivalent model with a finite action and state space without loss of optimality. The reduced model is associated with a directed network, which is then solved as a shortest path problem. The authors claim that their procedure is exact because the A^* algorithm is used to solve the shortest path, which is an algorithm known to find an optimal solution when it is accompanied with a guidance function. The algorithm

is used to solve small instances with 4, 5, and 7 suppliers and the optimal solution is obtained in all cases. The proposed shortest path approach takes significantly shorter time to solve all the instances.

3.5 Bounds for the SDVRP

3.5.1 Cutting-plane Algorithm of Belenguer et al. (2000)

Belenguer et al. (2000) calculate lower bounds to optimal solutions of SDVRP instances based on a linear program formulation LP_j and a cutting-plane algorithm. The cutting-plane algorithm starts from an initial lower bound, LP_0, calculated by solving the initial formulation via a linear programming code (CPLEX 3.0). Valid inequalities are developed and used to determine the feasibility of the solutions obtained by the algorithm. Any violated inequality, if existing, is added to the initial formulation and the process is repeated to calculate a better bound. If no inequality violation is found in the new solution, the cutting-plane algorithm stops and provides a final lower bound, LB. The quality of bound LB is determined by comparing it with the optimal integer SDVRP solution value. This optimal value is obtained at a higher computational cost through the cutting-plane algorithm, by solving at each iteration an integer program by adding the integrality constraint corresponding to each variable. The cutting-plane algorithm is tested on 11 instances from the TSPLIB and a set of 14 randomly generated instances. Optimal solutions are found for instances ranging from 21 to 50 customers.

3.6 Classical Heuristics for the SDVRP

3.6.1 Algorithm of Dror and Trudeau (1989)

Dror and Trudeau (1989) propose a local search to solve the routing problem with split deliveries. This is a two-stage algorithm that first constructs a feasible VRP solution and from this generates a feasible SDVRP solution if split deliveries improve the initial VRP solution. The first stage uses

three subroutines: (i) an initial route generator based on the algorithm of Clarke and Wright (1964); (ii) a node interchange based on a one-node and two-node swap; and (iii) a route improvement based on a 2-opt procedure. The second stage uses: (i) a *2-split* interchange and (ii) a route addition routine. Given a demand point, the *2-split* creates a neighborhood with all the possible alternatives that remove the demand point and insert it into two other routes whose combined spare capacity is sufficient for the demand. At each iteration, the candidate with the highest saving is selected and the search terminates when improvements cease. After this local search, a route addition routine creates new routes to eliminate split deliveries as long as a reduction in the total routing cost is obtained. From this search neighborhood, the candidate solution with the highest saving is selected. The routine stops when no more improvements are found.

To test the potential savings associated with SDVRP, 180 problem sets are generated and solved with the proposed local search and the solutions are compared to VRP solutions. The number of customer varies from 75 to 150, the customer demands range from $0.1Q$ and $0.9Q$, where Q is the vehicle capacity fixed at 160 units. The customer locations are obtained from Eilon et al. (1971) and modified in a systematic/random fashion. A paired one-tailed t-test comparison was conducted, and the results show no significant difference between the SDVRP and VRP solutions for small customer demands in the range $[0.01Q, 0.1Q]$. However, for all other problem sets, SDVRP solutions significantly outperform VRP solutions in terms of travel distance.

3.6.2 Algorithm of Frizzell and Giffin (1992)

Frizzell and Giffin (1992) provide a construction heuristic to solve the SDVRP. They consider grid network distances, lower and upper bounds to limit the ways to split deliveries, and splitting costs in the objective function. Frizzell and Giffin group, or cluster, the customers whose distances to each other are lower than a predefined value. The mechanism to group customers is what they call clustering of adjacent customers, CAC. Any customer which does not meet this condition is clustered in another group. If the combined demand within a cluster exceeds the vehicle capacity,

the algorithm sequentially allocates vehicles to another group until all demands are fully supplied.

Once the clusters are formed, nearest neighbor blocking, NNB, is used to decide the order in which customer demands are assigned to vehicles in order to produce distance savings. This order does not have anything to do with the sequence of customers within a tour. The NNB mechanism takes the two nearest customers (i.e., $d_{ij} = \min[d_{ik}]|k \neq 0, i$) with unassigned demands and, from those, assigns the one which is farthest from the depot (i.e., customer $i|d_{i0} > d_{jk}$) the maximum possible demand in the route under consideration.

Since grid network instead of Euclidean distances are used, it is not possible to compare their results with previous work. Therefore, 1050 problem sets are generated by assigning customer locations randomly on the grid network, the number of customers ranges from 20 to 100, the vehicle capacity is 100, and customer demand is uniformly distributed from 1 to 100. The results show significant savings in 73.1% of the problem sets in terms of travel distance and number of vehicles required with respect to a basic construction heuristic that does not use the NNB mechanism.

3.6.3 Algorithm of Frizzell and Giffin (1995)

Frizzell and Giffin (1995) propose a mixed-integer formulation and develop a construction heuristic and two improvement methods to approximately solve the SDVRP with time windows (SDVRPTW). Their constructive procedure sorts the customers and sequentially assigns them to vehicles until all demands are filled. Customers are sorted according to their distance from the depot and their time windows. Vehicles are allocated to deliver the maximum possible demand to all customers. However, a customer demand can be split when it exceeds the spare capacity of the assigned vehicle. If the current vehicle fleet cannot service all customers, the fleet size is augmented. Improvements are obtained from the initial solution by moving a customer from one route to another or by exchanging any two customers between two routes when a savings in the objective function results.

3.6.4 Algorithm of Mullaseril et al. (1997)

Mullaseril et al. (1997) describe a feed distribution problem encountered on a cattle ranch in Arizona. They study the problem of scheduling a fleet of trucks for feed distribution in a large livestock range. The characteristics of this problem are: 1) different feed types are required in each stage of the growth of cattle; 2) the feed type, volume, and feeding time for each pen may vary from day to day; 3) the fleet is composed of 5 trucks with different capacities; and 4) a pen may need to have feed delivered from more than one route due to inaccuracies in the weight and loading of the trucks. Since each vehicle delivers only one type of feed, the feed delivery problem is decomposed into a different routing problem for each type of feed. Each subproblem is then modeled as a routing problem with split deliveries and time windows (SDVRPTW).

Their solution strategy for this problem adapts the algorithm of Dror and Trudeau (1989). Initially, a feasible VRP solution is generated and improved by an arc interchange adapted to include time windows. Then, split deliveries are introduced via the *k-split* interchange if the total travel distance can be reduced. Since the solution must consider time windows, the candidate list is pruned to those routes respecting the time windows constraint. To mitigate a potential reduction in the number of candidates, the *k-split* operator uses $2 \leq k \leq M$, where M is the number of candidate routes generated usually less than 10. Finally, the route addition improvement approach is used, but a check is done for capacity and time feasibility. The algorithm is run and the solutions, VRP and SDVRP, are compared with the current practice at the ranch. Substantial reductions in the total distance covered are obtained (from 25% to 40%). SDVRP solutions improve VRP solutions, especially when time windows constraints are respected.

3.7 Metaheuristic Algorithms for the SDVRP

3.7.1 Tabu Search of Ho and Haugland (2004)

Ho and Haugland (2004) developed a TS to solve instances of the SDVRPTW. They construct an

initial solution by checking customers in sequence and appending the nearest un-routed customer to the latest routed customer in feasible routes. If the customer demand exceeds the capacity, the current route is deemed full loaded, the demand is split, and a new route is created to supply the remaining demand. Once the route schedule is constructed, the TS commences. Each iteration, the best feasible candidate among four neighborhoods, N_1 to N_4, is selected and the neighboring solution is evaluated for improvements. The neighborhoods examined are: (N_1) relocating a customer between routes; (N_2) eliminating a split delivery between two routes and introducing a new delivery between the same two routes; (N_3) exchanging two customers between two routes; (N_4) performing a 2-opt operation between two routes. If the candidate move is tabu, a best so far aspiration criteria overrides the tabu status. Once the neighboring solutions are chosen, routes with η customers or less are eliminated by inserting those customers into non-empty routes via the same relocate operator used for N_1. This customer relocation is performed once every q iterations. In addition, a customer can be relocated in a least cost position within a route after v consecutive iterations without improvement. The repetitive procedure stops after y consecutive iterations with no improvement. Finally, a post-optimization phase is applied to the best solution.

3.7.2 Tabu Search of Archetti et al. (2006)

Archetti et al. (2006) propose a TS to solve the SDVRP where a customer is removed from a set of routes serving it and inserted into a new route or into an existing route that has spare capacity. The scheme of the procedure employs an initial solution, a TS, and an improvement phase. The TS uses a list, O_i, with all the routes visiting customer i in descending order based on the saving obtained when removing i from the route. A neighborhood is constructed by inserting a customer i into a route r and removing it from a subset $U \subseteq O_i - \{r\}$. The neighbor yielding the best objective function value is selected. Parameter θ, the tabu tenure, is a random number from an interval based on the number of customers and the number of routes in the current solution. If a neighbor solution yields a solution better than the best encountered so far, that solution is always accepted. The search concludes after n_{max} iterations without improving the best solution found so far.

Table 3.1: Existing literature on SDVRP.

Algorithm	Year	Description & Remarks
Dror and Trudeau	1989	SDVRP is proposed; introduce splits and local search
Dror and Trudeau	1990	Properties and complexity
Frizzell and Giffin	1992	Grid network distances; construction heuristic
Dror et al.	1994	Properties; branch and bound
Frizzell and Giffin	1995	Grid network; SDVRPTW; local search; shift and swap
Mullaseril et al.	1997	Application feed distribution; adapted to time windows
Sierksma and Tijssen	1998	Application crew exchange; column generation
Belenguer et al.	2000	Lower bounds
Archetti et al.[2]	2001	SDVRP with small capacities
Ho and Haugland	2004	Time windows; split relocate and tabu search
Archetti et al.	2005	Complexity; vehicles with capacity of k units
Lee et al.	2006	Dynamic programming and shortest path
Archetti et al.	2006	Eliminate splits and tabu search
Archetti et al.[2]	2008	Empirical analysis; benefits of allowing split deliveries
Archetti et al.	2006	Worst-case analysis and potential savings

[2]To appear

The algorithm was compared to the optimal solution in small instances (up to 15 customers) providing the same optimal solution for these problems in less than one second. For larger problems (between 50 and 199 customers), the problem is hard to solve to optimality so the performance of the procedure was evaluated by comparing the results with those of another heuristic (Dror and Trudeau, 1989). The comparison shows that the algorithm by Archetti et al. (2006) almost always provides better solutions on the tested problems.

Table 3.1 provides a brief summary of the solution approaches defined for and tested on SDVRP types of problems.

Chapter 4

VRP With Stochastic Customers (VRPSC)

The study of combinatorial problems with stochastic elements has received increasing attention recently due to the uncertainties inherent in many real world applications. Technological developments, such as onboard computers and communication systems allow updates to problem information and plan changes during the execution of a schedule. In addition, the desire to model problems in a more realistic way and analyze the robustness of optimal solutions of deterministic problems when instances are randomly changed motivate the incorporation of stochastic elements into the problem (Jaillet, 1993). This chapter reviews literature on the VRP with stochastic customers, also known as the probabilistic vehicle routing problem (PVRP). This review begins with the available literature on problem instances where a single vehicle with infinite capacity is used to service the customers. This problem is commonly known as the TSP with stochastic customers, or the probabilistic traveling salesman problem (PTSP). Then, the literature on the relaxation of this problem using multiple vehicles with finite capacity, which is the focus of this research, is reviewed.

4.1 Solution Concepts

Stochastic optimization problems are usually solved using either of two strategies. One strategy is to obtain the optimal solution to every realization of the problem each time a random variable becomes known. This strategy is known as re-optimization. However, such a strategy can become undesirable in certain occasions due to the lack of the required resources or to the high computational cost of finding an optimal solution to every realization, even if the resources are available. Thus, a second strategy looks for an *a priori* solution, which minimizes the total expected cost based on the probability distribution of the stochastic variables.

A stochastic problem is usually modeled in two stages. In the first stage, an *a priori* solution is obtained with the information available at that time. When the values of certain random variables are revealed, a recourse, or corrective action, occurs in a second stage in order to maintain feasibility in the solution. Such a recourse usually generates a variation in the objective function value which should be accounted for during the generation of the *a priori* solution. The problem is modeled as a Chance Constrained Program (CCP) when the *a priori* solution is obtained without considering the costs of the corrective actions. In such a case, if uncertainty affects feasibility and the objective function is deterministic, it may be necessary to respect the constraints with certain probability. In a more general approach, the problem is modeled as a Stochastic Program with Recourse (SPR) when the expected cost of the corrective actions taken in the second stage is considered in the *a priori* solution (Ghiani et al., 2003).

4.2 TSP With Stochastic Customers

Just as the classical VRP is a generalization of the deterministic TSP, the VRPSC is a generalization of the TSP with stochastic customers. The TSPSC first appears in the literature in the doctoral thesis of Jaillet (1985) as a variant of the TSP where the presence of some customers is uncertain. This means that, although the customer demands are known, the presence of a sub-set of k customers

($0 \leq k \leq n$) is uncertain when the tour is designed. While the presence of some customers is known beforehand, other customers are present with some probability. Such probabilities are assumed independent for the k customers.

The PTSP is solved finding an *a priori* tour which includes all the n potential customers, i.e., deterministic and stochastic, in such a way that, given any realization of the problem, the customers are visited in the same order as they appear in the *a priori* tour and absent customers are simply skipped. The design of the *a priori* tour minimizes the total expected traveled distance, which is calculated based on all the problem realizations. The reasons for not re-optimizing the tour in a second stage can vary according to the policies of a company, for example the lack of resources, the cost and effort to re-optimize, or simply the desire of establishing a regular tour (Bertsimas, 1988; Bertsimas, 1989).

Let V be the set of all n potential customers, τ be an *a priori* tour, $S \subseteq V$ be a realization of the problem, $L_\tau(S)$ be the length of tour τ to visit the subset of customers S, $p(S)$ the probability that only customers in S are present. The TSPSC seeks an *a priori* tour τ_p visiting all n potential customers while minimizing:

$$E[L_\tau] = \sum_{S \subseteq V} p(S) L_\tau(S) \tag{4.1}$$

In his doctoral thesis, Jaillet (1985) shows that the *a priori* solution to the deterministic version of the TSP can be arbitrarily bad for the TSPSC (Gendreau et al., 1996). Berman and Simchi-Levi (1988) examine finding the optimal *a priori* tour and location for the TSP with non-homogeneous customers. In contrast to other work where customers are homogeneous (Jaillet, 1985), the probabilities associated with customer presence vary. They find a lower bound on the value of the objective function. This bound is used in a branch-and-bound algorithm to find the optimal *a priori* tour. Given the *a priori* tour, the optimal home location for the service unit is found to minimize the expected length of the tour. The solution to the problem serves as an upper bound for the problem with different traveling salesman tours. Jaillet (1988) derives closed form expressions for comput-

ing efficiently the expected length of any given tour when each customer has the same probability of being present. The problem is represented as a Bernoulli process and it is shown that the optimal TSP tour can be a very poor solution to the corresponding PTSP. However, under conditions specified in the study, an optimum TSP tour can optimally solve the PTSP for any instance and for any probability distribution of uncertain customers.

Bertsimas and Howell (1993) examine the PTSP, derive some results, and consider its relation with the TSP. They find upper and lower bounds and compare various TSP heuristics applied to the PTSP with the re-optimization strategy. Bertsimas and Howell (1993) show that, in general, a TSP solution through n probabilistic points is potentially a very poor solution to the PTSP. The difference between the expected lengths of the optimal PTSP tour and the optimal TSP tour can be very large particularly when all customers have the same probability of being present and that probability is small. Bertsimas and Howell (1993) also show that a PTSP tour in the Euclidean plane can cross itself. This aspect should be taken into account particularly when TSP algorithms are used to solve the probabilistic counterpart. In addition, when the Euclidean metric is used and the customers are uniformly distributed in the unit square, a PTSP heuristic is shown to be very close to the re-optimization strategy.

Jaillet (1993) provides a probabilistic analysis of the PTSP and presents general finite-size bounds and limit theorems for the objective function. He proves the asymptotic convergence for the PTSP with respect to the re-optimization strategy. Laporte et al. (1994) formulate the PTSP as a stochastic linear integer program and solve it with a branch-and-cut approach, which is the first exact algorithm proposed. They solve to optimality problems with up to 50 customers and conclude that problems are more difficult to solve when the number of uncertain customers grows or the probability of being present reduces.

Table 4.1 summarizes the research on PTSP-types of problems.

Table 4.1: Research conducted on the PTSP

Algorithm	Year	Remarks
Jaillet	1985	Ph.D. Thesis
Jezequel	1986	M.S. Thesis
Berman and Simchi-Levi	1988	Non-homogeneous customers
Bertsimas	1988	Bounds; algorithms to compute the expected length
Jaillet	1988	Closed form expressions for the expected length
Bertsimas	1989	Bounds; algorithms to compute the expected length
Bertsimas et al.	1990	Applications, deterministic algorithms
Bertsimas and Howell	1993	TSP vs PTSP, bounds, and TSP heuristics
Jaillet	1993	Probabilistic analysis and asymptotic convergence
Laporte et al.	1994	Stochastic program and first exact approach

4.3 VRP With Stochastic Customers

The VRPSC appears in the literature the first time in the master's thesis of Jezequel (1986). This variant of the classical vehicle routing problem includes customers whose presence is unknown when the routes are designed. The presence of some customers is known beforehand, while a subset of k customers ($0 \leq k \leq n$) are present with probability P_i. Such probabilities are assumed independent for the k customers. The optimization problem finds a fixed set of routes to satisfy all customer demands while minimizing the total expected travel distance, which corresponds to the total expected distance of the fixed set of routes plus the expected value of the extra distance required in case the demand exceeds the vehicle capacity and the vehicle is forced back to the depot.

In the classical VRP, routes can be designed in advance because all the information is available at the time of the routes design. This is not the case in the VRPSC because some of the customers can be absent during the design of the routes. When the *a priori* routes are executed, some customers do not require any service. If this is known before the execution of the current plan, the plan can be modified to avoid visiting customers not requiring service. Bertsimas (1988) shows two strategies

to perform the *a priori* routes, *a* and *b*, depending on the time the presence, or service, of uncertain customers becomes known. When the customer service becomes known at the vehicle arrival, the vehicle visits all the assigned customers in the same order as they appear in the *a priori* tour, but serves only the customers requiring service (strategy *a*). If, on the other hand, the information on demands is known before performing the routes, customers not requiring service are simply skipped (strategy *b*). Bertsimas (1988) finds algorithms to compute the expected length, examines combinatorial properties of the problem and provides bounds for the PVRP and the re-optimization strategy, makes worst-case analysis for some proposed heuristics, and proposes some asymptotically optimal algorithms. Bertsimas demonstrates that the strategy of finding an *a priori* tour is a practical alternative to the re-optimization strategy.

The two strategies *a* and *b* shown by Bertsimas (1988) use fixed routes. These two strategies are called *fixed routes* and *semi-fixed routes* by Waters (1989). The cost of the first strategy does not change with the absence of customers, while the second strategy produces cumulative savings with increasing number of absent customers. A *variable routes* strategy to increase the savings considers all the customers needing service and re-optimize the solution. Figure 4.1 shows a comparison of the three strategies when customer 1 is absent. In part a), customers are visited as in the initial routes, but only the customers requiring service are serviced. In part b), customer 1 is simply skipped from the *a priori* solution, which produces a reduction in the total traveled distance. In part c), the problem is resolved given that customer 1 is absent. When the number of absent customers is small, the strategy of fixed routes might be acceptable. However, when this number gets larger, fixed routes might yield a poor solution and the strategy of re-optimizing could provide valuable savings. Although the third strategy is not always possible, the core question is: how large are the potential savings from re-optimizing and using variable routes, and when is the re-optimization strategy worthwhile?. Waters (1989) answers this question using an empirical analysis on 100 randomly generated problems with a total of 200 customers and randomly removing from 1 to 40 customers from the problem. The savings in distance are calculated with respect to the solution with fixed routes and are found to increase linearly with the number of absent customers when this

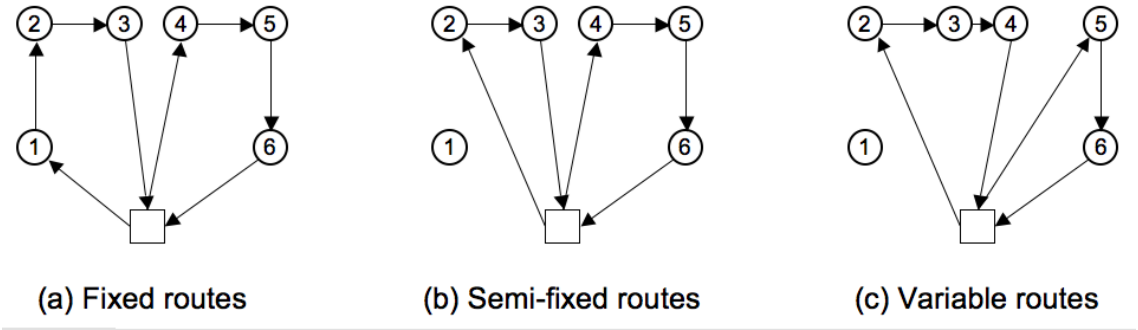

Figure 4.1: Comparison of fixed routes, semi-fixed routes, and variable routes strategies.

number is relatively small. With up to 20% of customers being absent, the results reveal that, on average, skipping customers under the semi-fixed routes strategy reduces the distance by 2.60 units per absent customer, while re-optimizing the routes provides a reduction of 5.65 units per absent customer. There may also be a reduction in the number of vehicles used when the solution is re-optimized. In the case of variable over fixed routes, a mathematical analysis finds an upper bound on the reduction in the number of vehicles and a minimum number of absent customers required to make the re-optimization strategy worthwhile. In the case of variable over semi-fixed routes, the average number of absent customers before the first vehicle can be saved is about 30% and a second vehicle is saved when approximately 37.5% of customers are absent. From a practical perspective, using a real world scenario, the benefits of re-optimizing are too low and this strategy is not worth the effort.

Bertsimas et al. (1990) characterize the asymptotic behavior of the re-optimization and the *a priori* strategies and observe that both have close asymptotic performance. From a computational complexity point of view, Bertsimas et al. (1990) prove that finding the optimal *a priori* solution is NP-hard. When uncertain customers have the same presence probability and this probability is large, a heuristic for the deterministic VRP behaves well for the corresponding probabilistic problem. If, however, the probability is small, the optimal deterministic solution is an arbitrarily bad approximation to the optimal *a priori* solution.

Table 4.2: Research conducted on the probabilistic VRP

Algorithm	Year	Remarks
Jezequel	1986	M.S. Thesis
Bertsimas	1988	Bounds; algorithms to compute the expected length
Waters	1989	Empirical analysis and potential savings
Bertsimas et al.	1990	Applications, deterministic algorithms
Bent and Van Hentenryck	2004	Non-unit demands, pool of solutions

Bent and Van Hentenryck (2004) introduce time windows to the PVRP and propose a multiple scenario approach (MSA) to continuously generate and solve realizations of the problem. These solutions include both known and unknown customer requests. Solutions in the pool are consistent with the current solution and are used by a consensus function to choose a distinguished solution. This distinguished solution is the solution most similar to the pool of solutions and is taken as the new current solution. Since they generate new solutions continuously, a greedy approach is used to generate the new solutions from the pool. The simple greedy approaches work better than sophisticated algorithms –at least in the case with time windows– since more robust approaches tend to produce tight solutions that do not accommodate future requests. The complexity of the problem may vary greatly according to the degree of dynamism, i.e., the ratio *uncertain customers/total customers*. Bent and Van Hentenryck use chronological customer presence (i.e., customers appear any time during the day), and probability distributions to model uncertainty of customer presence, and non-unit customer demands. This differs from previous work where customer presence is known or determined probabilistically before executing the routes and demands are assumed unitary.

Table 4.2 summarizes research conducted on the PVRP.

4.4 Summary

This chapter reviewed the existing literature on the vehicle routing problem with stochastic customers. Although there exist various studies on this problem, it has not been extensively explored. The existing literature shows theoretical analysis, properties of the problem, probabilistic analysis, and description of some heuristic methods to solve the problem. Most of these methods are adapted versions of existing algorithms to solve the deterministic problem. There is still a lot of work to do in the design of search algorithms that would facilitate a better understanding of the problem, gain more knowledge on it, and find new more effective solution techniques.

Part II

Solution Approaches

Chapter 5

An Adaptive Memory Algorithm for the SDVRP[1]

5.1 Introduction

The vehicle routing problem (VRP), or truck dispatching, was first formulated by Dantzig and Ramser (1959) and is a core problem in transportation, logistics, and supply chain management. The VRP involves a fleet of vehicles with fixed characteristics (i.e., speed, capacity, etc.) stationed at a central depot and a set of geographically scattered points (i.e., cities, warehouses, schools, customers, etc.) with fixed demands. Vehicles are used to visit and fully supply the demand of these points. The optimization problem is to determine which customers are visited by each vehicle and what route will the vehicle follow to serve those assigned customers, while minimizing the operational costs of the fleet, such as travel distance, gas consumption, and vehicle depreciation. Routes are designed to start and end at the depot, the demand of every customer is fully supplied by exactly one vehicle, and the total demand met by any route cannot exceed the vehicle capacity.

In reality, however, there may be cases where either a customer demand exceeds the vehicle capacity or a savings in terms of the total distance or the number of vehicles can be obtained by

[1]This chapter is found as Aleman et al. (2007).

serving customers with more than one vehicle. The split delivery vehicle routing problem (SDVRP) relaxes the VRP restraints and allows the use of multiple vehicles to satisfy customer demand points and potentially reduce the total delivery cost by splitting customer deliveries among vehicles (Dror and Trudeau, 1989). The computational complexity of the SDVRP remains NP-hard (Dror and Trudeau, 1990). Archetti et al. (2005) define the k-SDVRP as a special case of the SDVRP where vehicles have a capacity of k units, $k \in \mathbb{Z}^+$. The Archetti et al. (2005) study shows that the 2-SDVRP is solvable in polynomial time when some specific conditions on the distances are satisfied, while the problem with $k \geq 3$ remains NP-hard. Archetti et al. (2005) also show that the 2-SDVRP may be reduced to a problem of possibly smaller size, where each customer has unitary demand.

The SDVRP is defined on an undirected graph $G = (V, E)$ where $V = \{0, 1, ..., n\}$ is the set of $n + 1$ nodes of the graph, and $E = \{(i, j) : i, j \in V,\ i < j\}$ is the set of edges connecting the nodes. Node 0 represents a depot where a fleet $M = \{1, ..., m\}$ of identical vehicles with capacity Q are stationed, while the remaining node set $N = \{1, ..., n\}$ represents the customers. A non-negative cost, usually a function of distance or travel time, c_{ij} is associated with every edge (i, j). Each customer $i \in N$ has a demand of q_i units. The optimization problem is to determine which customers are served by each vehicle and what route will the vehicle follow to serve those assigned customers, while minimizing the operational costs of the fleet, such as travel distance, gas consumption, and vehicle depreciation.

In this chapter a solution method that uses a constructive heuristic approach and a fixed number of vehicles is proposed to construct an initial solution by inserting unassigned customers sequentially into the solution under construction. A sequence is a list of customers in a specific order. When the solution is complete, the sequence of customers is modified based on the characteristics of the constructed solution. Once a new sequence of customers is determined, the constructive heuristic approach is executed again to find another solution. Again, the sequence of customers is modified and the procedure is repeated until no better solutions can be found. The best solution found during this iterative constructive approach is then improved using a variable neighborhood descent (VND) procedure. This is the first time a variable neighborhood search is used to solve the

SDVRP.

This chapter is organized as follows. Section 5.2 provides a review of the existing literature on SDVRP. The algorithms are described in Section 5.3. Numerical experiments comparing the proposed algorithms with other existing methods found in the literature are presented in Section 5.4. Conclusions and future directions are provided in Section 5.5.

5.2 Literature Review

5.2.1 Benefits of SDVRP

At first glance, one may believe that the benefits of allowing split deliveries are small when the customer demands are either considerably small with respect to the vehicle capacity or close to the vehicle capacity. In their experimental study, Dror and Trudeau (1989) showed that if customer demands are low relative to the vehicle capacity and the triangular inequality holds ($c_{ij} \leq c_{ik} + c_{kj}$, for all i, j, and k), the split demand benefits are actually very little. In contrast, when the customer demand is larger, at least 10% of the vehicle capacity, the cost of a SDVRP solution is considerably lower than the cost of a VRP solution. Figure 5.1 shows a simple example to illustrate the potential benefits of allowing split deliveries in terms of the number of vehicles and the solution cost. In this figure, there are 12 customers with same demand $q_i = 60$ symmetrically located on a circle of radius r centered at the depot, where a fleet of vehicles with capacity $Q = 100$ is located. The optimal VRP solution is illustrated in Figure 5.1(a). It employs 12 vehicles with an average utilization of 60% and has a value of $z = 240r$ distance units. If split deliveries are allowed, all customer demands can be supplied as illustrated in Figure 5.1(b) with only 8 vehicles with an average utilization of 90% and a solution value of $z = 201.4r$.

Archetti, Savelsbergh, and Speranza (2006) provide a worst-case analysis for the SDVRP and show that the savings in delivery costs that can be obtained by allowing split deliveries is at most 50% and that this bound is tight (i.e., there exists an example in which the value of the optimal

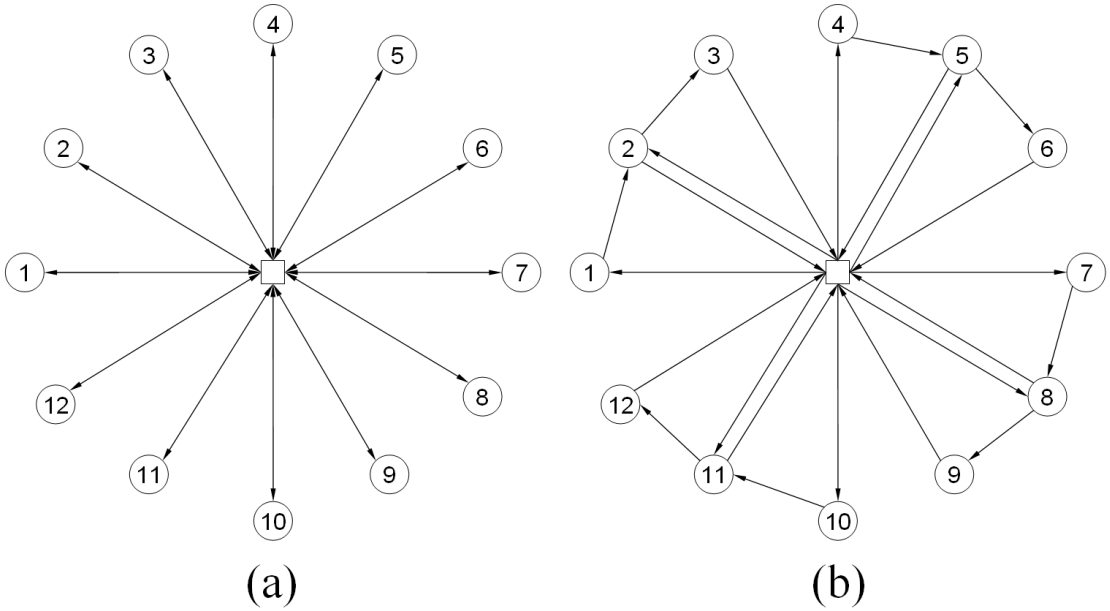

Figure 5.1: Illustration of savings by SDVRP: (a) VRP solution and (b) SDVRP solution.

VRP solution doubles the value of the optimal SDVRP solution). This analysis, however, does not provide insight into the relation between the customer characteristics and the savings attained by allowing split deliveries.

Archetti et al. (2008) characterize distribution environments and conduct a thorough study (the most detailed found so far) of the value and benefits of allowing split deliveries. The focus of their study is to determine the practical implications of split deliveries for different customer characteristics, particularly in terms of the geographic and demand distribution of customers. The benefits are in: 1) the reduction in the number of routes, and thus vehicles, required to fully supply all customer demands and 2) the reduction in delivery costs.

Archetti et al. (2008) use a mathematical analysis to prove that the maximum reduction in the number of routes that can be achieved by allowing split deliveries is 50%. Moreover, their analysis confirms that the largest reduction can be obtained when the mean customer demand is between 50% and 70% of the vehicle capacity and the demand variances are relatively small. However, while there does not appear to be a dependence of delivery cost reductions on the geographic distribution

of customers, there does appear to exist a dependence on the demand variance.

5.2.2 Existing SDVRP Algorithms

Both exact and heuristic algorithms have been used to solve the SDVRP. Although exact algorithms solve instances to guaranteed optimality, they can be unpractical to use in solving large instances due to the computational costs involved. The largest SDVRP instance solved to optimality includes 50 customers (Belenguer et al., 2000).

There are some exact approaches found in the SDVRP literature. Dror et al. (1994) propose an integer linear programming formulation and describe a branch-and-bound algorithm based on new classes of valid inequalities for the SDVRP. Sierksma and Tijssen (1998) apply the SDVRP to building the transportation schedule of helicopters supporting offshore platforms in the North Sea for crew exchange of people employed on those platforms. They propose a set-covering formulation for the SDVRP and solve its relaxation using a simplex algorithm and a column generation technique that includes a Knapsack Problem and several TSPs.

Lee et al. (2006) propose a solution method for the multiple-vehicle routing problem with split pick-ups (mVRPSP) based on a deterministic dynamic program model and a shortest path search algorithm. Based on some properties of optimal solutions of the mVRPSP, they reformulate the original dynamic program to find an equivalent model with a finite action and state space without loss of optimality. The reduced model is associated with a directed network, which is then solved as a shortest path problem. The algorithm is used to solve small instances with 4, 5, and 7 suppliers and the optimal solution is obtained in all cases.

Jin et al. (2007) present an iterative exact method called two-stage approach with valid inequalities (TSVI) to find an optimal solution after a finite number of iterations for SDVRP instances with average customer demands greater than 10% of the vehicle capacity. They divide the problem into a clustering sub-problem and a traveling salesman problem for each vehicle. In a first stage, the clustering sub-problem optimally assigns customer demands to the vehicles without considering

distance costs. In a second stage, a traveling salesman problem is solved via a commercial optimization solver to find the minimal distance traveled by each vehicle. Those distances are added as cuts to the original clustering sub-problem. The process is repeated until no new clusters can be found in the first stage.

Other studies have estimated problem bounds. Belenguer et al. (2000) calculate lower bounds to optimal solutions of SDVRP instances based on a polyhedral study of the problem and a cutting-plane algorithm. The cutting-plane algorithm starts with an initial lower bound, which is calculated by solving the initial problem formulation via a linear programming code. Valid inequalities, or cuts, are developed, added to the formulation, and used to determine the feasibility of the solutions obtained by the algorithm. Any violated inequality is added to the initial formulation and the process is repeated to calculate a better bound. If no inequality violation is found in the new solution, the cutting-plane algorithm stops and provides a final lower bound. Jin et al. (2008) propose a column generation to find lower bounds and an iterative approach to obtain upper bounds for the SDVRP. The approaches are tested on 12 of the 25 instances used by Belenguer et al. (2000) containing large customer demands as the algorithm is not efficient to solve problems with small average customer demands. They suggest solving those instances as capacited vehicle routing problems (CVRPs) rather than SDVRPs. The column generation improves some of the bounds of Belenguer et al. (2000).

Heuristic algorithms are often desirable to solve larger SDVRP instances. Various approaches found in the literature include local, tabu, and scatter search, hybrid approaches, and memetic algorithms. Dror and Trudeau (1989) proposed a local search to solve the routing problem with split deliveries. Theirs is a two-stage algorithm that first constructs a feasible VRP solution and from this generates a feasible SDVRP solution if split deliveries improve the initial VRP solution. Split deliveries are incorporated into the solution by using a *2-split* interchange operator, which creates a neighborhood with all the possible alternatives that remove a demand point from a route and insert it into two other routes whose combined spare capacity is greater than or equal to the demand. A route addition routine may create new routes to try eliminating split deliveries as long as a reduction

in the total routing cost is obtained. Frizzell and Giffin (1992) use a grid network instead of euclidean distances in the SDVRP. They use a constructive approach that clusters adjacent customers and then allocates vehicles to the clusters until the unassigned demand of the cluster is less than the vehicle capacity. For each cluster, a nearest neighbor blocking is used to first assign the demand of the customers farther from the depot. In case the combined demands in the same cluster exceeds the vehicle capacity, the blocking mechanism produces distance savings as the demands to be split are the ones closer to the depot. Bouzaiene-Ayari et al. (1993) suggest an adaptation of the Clarke and Wright algorithm to solve the vehicle routing problem with stochastic demands and split deliveries. This study is apparently the first attempt to solve a stochastic vehicle routing problem with split deliveries.

In a second paper, Frizzell and Giffin (1995) incorporate time windows into the problem (SDVRPTW). The algorithm is similar to the one in Frizzell and Giffin (1992), but now customers are sorted according to the distance from the depot and their time windows. The initial solution is changed by moving a customer to alternate routes or by exchanging any two customers between their assigned routes when a saving in the objective function results. Mullaseril et al. (1997) describe a feed distribution problem encountered on a cattle ranch in Arizona. They study the problem of scheduling a fleet of trucks for the feed distribution in a large livestock range. The solution strategy for this problem is an adaptation of the algorithm of Dror and Trudeau (1989). Since the solution must consider time windows, the candidate list is pruned to those routes respecting the time windows constraint. To mitigate a potential reduction in the number of candidates, the *k-split* interchange operator uses $2 \leq k \leq M$, where M is the number of candidate routes generated, usually less than 10. Finally, a route addition improvement approach is used, but a check is done for capacity and time feasibility.

The tabu search by Ho and Haugland (2004) uses an operator called the relocate split operator. The algorithm starts with the construction of an initial solution by checking customers in a pre-defined sequence and appending the nearest un-routed customer to the latest routed customer. During the tabu search, the best candidate among four neighborhoods is selected at each iteration.

They use standard operators (e.g., customer relocation, customer exchange, and $2 - opt^*$) adapted to the SDVRP context to potentially eliminate split deliveries. They also use the relocate split operator, which uses two routes with a shared delivery and relocates the delivery within the two routes subject to obtaining a reduction in the total distance. Archetti, Hertz, and Speranza (2006) propose a tabu search where a customer is removed from a set of routes serving it and either inserted into a new route or into an existing route that has spare capacity. The tabu search uses a random tabu tenure selected from an interval defined by the number of customers and the number of routes in the current solution. An improvement phase is used after the tabu search in order to eliminate *k-split* cycles. Chen et al. (2007) develop a heuristic that combines a mixed integer program and a record-to-record travel algorithm that starts with an initial SDVRP solution based on the Clarke and Wright algorithm. For each route in the initial solution, a mixed integer program considers the endpoints and the closest neighbors to each endpoint to reallocate the demand of the endpoints and maximize the total savings. An endpoint is reallocated in three ways: 1) no change is made; 2) the endpoint is totally removed from its current route(s) and all of its demand is moved to other route(s); and 3) the endpoint is partially removed from its current route(s) and part of its demand is moved to other route(s). The heuristic is tested on the 49 problems of Archetti, Hertz, and Speranza (2006), 5 random problems of Belenguer et al. (2000) with large customer demands, and 21 new benchmark problems and is shown to clearly outperform the algorithms of Archetti, Hertz, and Speranza (2006).

Other studies covering the SDVRP include the work by Song et al. (2002) who adopt a split delivery scheme to find an allocation of newspaper agents and route vehicles to deliver newspapers while minimizing the delivery costs and reducing the total delay time of the delivery. Various algorithms were used and savings of 15% in the delivery costs and 40% in the delay time were obtained. Nowak (2005) examines a pickup and delivery routing problem with split loads and explores how costs can be reduced by eliminating the constraint that only one vehicle can service a customer. The problem is modeled as a dynamic program and the results show that most benefits with split loads occur when loads are at least half of vehicle capacity. Liu (2005) proposes a two-

stage algorithm with valid inequalities (Jin et al., 2007) and a branch-and-price approach to solve the problem. Wilck and Cavalier (2007) study a modified objective function to consider the impact of the loads in the operational costs and potentially reduce them. Yu et al. (2006) consider an inventory routing problem with split delivery and solve it using lagrangian relaxation and linear programming. Belfiore et al. (2006) implemented a scatter search to solve the problem involving other side constraints including heterogeneous vehicles, time windows, and accessibility constraints applied to a retail market in Brazil. The algorithm was applied to solve real scenarios arising on a daily basis and reduced the operational costs of the fleet compared to current practices in the company. Ambrosino and Sciomachen (2007) deal with a real application for food distribution in an Italian company. They model the problem as a generalization of the asymmetric VRP with split deliveries to determine an efficient distribution plan of fresh/dry and frozen food along the country. The solution algorithm includes a clustering procedure suitably tailored to the conditions of the real problem and a local search to move customers between routes and to split customer demands to improve the solution. Mota et al. (2007) present a scatter search that uses the minimum possible number of vehicles and performs favorably with respect to the tabu searches of Archetti, Hertz, and Speranza (2006) in the tested problems, particularly when the customer demands are below half the vehicle capacity. Tavakkoli-Moghaddam et al. (2007) present a simulated annealing method to solve the problem with heterogeneous vehicles and use a new term in the objective function to maximize the utilization of the vehicle capacity. The algorithm was tested on randomly generated instances only and no benchmark problems were utilized. Boudia et al. (2007) implemented a memetic agorithm with population management that produces high quality solutions and low running times relative to the Splitabu approach of Archetti, Hertz, and Speranza (2006).

5.3 Proposed SDVRP Algorithm

5.3.1 Constructive Heuristic Approach (CA)

An initial SDVRP solution is obtained using a construction procedure that sorts the customers based on the distance from the depot c_{0j} and then creates new routes or modifies existing routes to allocate the customers. A list, L, of customers is created and sorted in descending order based on c_{0j}. Customers are then analyzed in sequence to determine the best way to include them in the solution, either by initiating a new route or by inserting them into existing routes. In the former case, new routes are initialized until a maximum number of routes is reached. This approach utilizes the minimum fleet size required to satisfy the demand constraints. In contrast to the classical VRP where a bin packing problem is solved to find the number of vehicles required to supply all customer demands, any SDVRP instance can be solved using $m = \lceil \sum_{k \in N} q_k / Q \rceil$ vehicles, where $\lceil x \rceil$ is the lowest integer greater than or equal to x. Customers are inserted into existing routes at the cheapest insertion position.

During preliminary experiments, it was found that the insertion method can be improved by using a new mechanism called route angle control (RAC). This mechanism uses the angle formed by customers within routes to help determine the best way to allocate customers in the solution. The polar angle of a customer θ_k relative to the depot is defined as:

$$\theta_k = \arctan \frac{y_k - y_0}{x_k - x_0} \tag{5.1}$$

where (x_k, y_k) represents the location of customer k and customer 0 represents the depot. The angle of a route, θ_r, is defined as the maximum angle formed by the customers in the route, i.e., $\theta_r = \max\{\theta_i - \theta_j; \forall (i,j) \in r\}$. Initially, an existing route has an associated angle formed by the customers visited during the route. When a customer is inserted, this angle may either increase or remain constant if the inserted customer is geographically located between the customers in the

route. If the increased angle exceeds a threshold angle value, a penalty cost is incurred. The penalty cost can be chosen arbitrarily as long as it makes the move prohibitive.

Inspired by the petal algorithm of Foster and Ryan (1976), the problem is partitioned so that routes serve sectors of the region centered at the depot. The region is partitioned as the number of sectors equal to the fleet size, m. Thus, sectors are equally distributed with an angle of value $\theta^* = \frac{2\pi}{m}$. This angle is used as the threshold angle value to penalize insertions that spread routes. In addition to the penalty cost, the RAC considers the route angle that results after inserting a customer in the route. This means that when a customer in L is evaluated for insertion in the solution, it may be located between two existing routes that can serve it without exceeding θ^*. In such a case, the RAC mechanism favors the route closest to the customer for servicing the customer.

Figure 5.2 illustrates the RAC. For this example, assume $m = 8$ so that $\theta^* = 2\pi/8 = \pi/4$. This figure illustrates two cases. In the first case, the insertion of customer j_1 into R_3 would spread the route giving an angle exceeding θ^*; that insertion is penalized to favor the creation of a new route or the insertion into another route, such as R_1. In the second case, customer j_2 can be inserted either into R_1 or R_2 without incurring a penalty because both routes would remain within θ^*. In such a case, j_2 is inserted into the narrower route among R_1 and R_2. In any case, the insertion cost is proportional to the angle of the route after the insertion of the customer. The evaluation of the insertion candidates is described in the algorithm below.

The constructive procedure is summarized as follows.

Notation:

L	List containing all customers $i \in N$
u_i	Unserved demand of customer i
q_i	Demand of customer i
s_r	Spare capacity of route r; $r \in M$
θ_{ir}	Angle of route r after inserting customer i

Step 1 (Sort the customers). Create L and sort the customers to produce $L = \{i_1, i_2, ..., i_n\}$ with $c_{0i_1} > c_{0i_2} > \cdots > c_{0i_n}$.

Step 2 Set i to the first customer in L and $u_i = q_i$

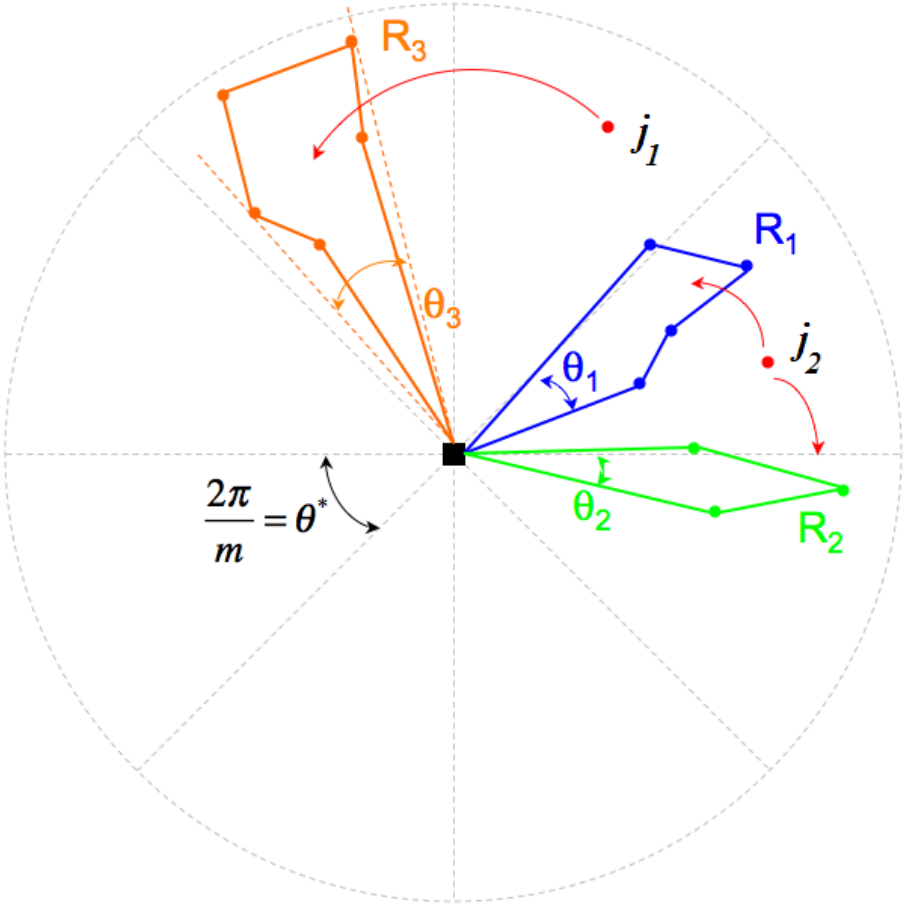

Figure 5.2: Route angle control.

Step 3 (Insertion candidate). Find the cheapest way to insert customer i within existing feasible routes. The cost of inserting customer i into route r, c_{ir}, includes the distance cost and the value obtained by the route angle mechanism. Let i_b and i_a be the preceding and succeeding customers, respectively, in route r after inserting customer i. The insertion cost is then given by:

$$c_{ir} = c_{i_b i} + c_{i i_a} - c_{i_b i_a} + \alpha \times \theta_{ir} + \beta \times \max\{0, \frac{|\theta_{ir} - \theta^*|}{\theta_{ir} - \theta^*}\} \tag{5.2}$$

where α represents a weight for the angle of route r after inserting customer i and β represents a penalty value incurred when the insertion of customer i produces a route angle θ_{ir} exceeding θ^*. The value of β can be any value large enough to favor the insertion of the customer in

other routes. Special care has to be taken in Equation (5.2) to avoid a division by zero when $\theta_{ir} = \theta^*$. The route r yielding the lowest insertion cost is selected as the best insertion candidate.

Step 4 (Insert customer). If the cost of the candidate found in Step 3 is less than the cost of a returning route (that is, $c_{ir} < 2c_{0i}$), insert customer i into the cheapest insertion position of route r. Otherwise, initiate a new route with customer i.

Step 5 (Calculating the quantity). If $s_r \geq u_i$, then $s_r = s_r - u_i$ and $u_i = 0$. Otherwise, $s_r = 0$ and $u_i = u_i - s_r$ (i.e., split delivery occurs).

Step 6 (Optimize route). Using a local search, optimize route r by moving single customers to the cheapest position in the route. If all customers in L are fully supplied, go to Step 8.

Step 7 If $u_i = 0$, go to the next customer i in L and set $u_i = q_i$. Go to Step 3.

Step 8 Stop.

5.3.2 Iterative Constructive Approach (ICA)

A big advantage of constructive procedures is their ease of implementation. However, the initial SDVRP solutions for the constructive approach were found to be less than ideal due to the disadvantages of a constructive procedure. Constructive approaches perform moves with the best immediate benefit while ignoring the effects this can have in later stages of the search. When the solution construction starts, the best moves can be performed. As the search progresses, the number of good alternatives are reduced and the final moves usually have a negative impact on the quality of the final solution.

The iterative approach is based on the presumption that customers inserted in the later stages of the procedure are likely to most deteriorate the solution quality. Thus, those customers will have a higher contribution to the solution value than customers inserted earlier in the solution. The impact of those contributions influence the decisions made when constructing new solutions. As

new solutions are constructed, customers with a history of high contributions are inserted into the solution earlier by changing the structure of L.

Figure 5.3 shows an example of the list L used by the iterative constructive approach to solve a benchmark instance involving 50 customers with vehicle capacity $Q = 160$. For simplicity, the figure only shows the customers assigned to routes R_1, R_4, and 5 of the customers assigned to route R_2. These routes are constructed by inserting sequentially the customers in L as follows. Let L_r be the node list containing the customers forming route R_r following the order in which the customer demands are assigned to the route. Thus, for instance, L_1 provides the order customers are placed into the route while R_1 provides the order those customers are visited. The construction of route R_1 commences with the assignment of the 6 demand units of customer 36 (position 1 in L_1), followed by the 17 demand units of customer 35 (position 2 in L_1), until all demands in L_1 are assigned to R_1. Note that customer 11 in the last position of L_1 is partially supplied by R_1 with only 5 demand units as the vehicle has no more capacity at the moment the demand of customer 11 is assigned to the route. In this particular instance, this demand is $q_{11} = 19$. The remaining 14 units are assigned to route R_2 whose construction is out of the scope of this example. Simultaneously, route R_4 is constructed by assigning the demands of customers 43, 31, 26, 7, 24, 8, 23, 48, 32, and 27. In contrast to R_1, all demands can be fully assigned to R_4 and the vehicle still has 20 units left (see spare Q of R_4 in figure). The process continues following the order in list L until all customer demands are assigned and the solution is complete. In Figure 5.3 L_1 and L_4 are provided explicitly and show R_1 and R_4 graphically.

The customers in L are sorted based on the distance to the depot, so that customers located closer to the depot are inserted into the solution later in the process. Preliminary experiments found that customers near the depot caused route angles to increase since preferred routes lacked capacity to support the customer insertion. Such angle spreading customers need to be inserted earlier so they need to be placed earlier in L. Temporarily removing a customer from a route changes the angle of that route, a value denoted as $\Delta\theta_r$. Let customer i^* have the largest $\Delta\theta_r$, i.e., the customer that most deteriorates the solution. This customer i^* is re-positioned in L to ensure its earlier consideration in

List L of sorted customers

Position in L	1	2	3	4	5	6	7	8	9	…	16	17
Customer	36	40	35	39	43	33	3	20	21	…	28	31

Position in L	18	19	20	21	22	23	24	…	29	30	31	32
Customer	13	29	10	26	7	50	24	…	8	16	23	49

Position in L	33	34	…	40	…	43	44	…	46	…	49	50
Customer	2	22	…	48	…	1	11	…	**32**	…	27	46

List L_1 of sorted customers forming R_1

Position in L_1	1	2	3	4	5	6	7	8	9	10	11	12
Customer	36	35	3	20	21	28	29	16	2	22	1	11
Delivery	6	17	16	28	8	14	6	15	30	8	7	5
Spare Q of R_1	154	137	121	93	85	71	65	50	20	12	5	0

List L_4 of sorted customers forming R_4

Position in L_4	1	2	3	4	5	6	7	8	9	10	11	12
Customer	43	31	26	7	24	8	23	48	**32**	27		
Delivery	11	11	7	19	10	23	16	17	11	15		
Spare Q of R_4	149	138	131	112	102	79	63	46	35	20		

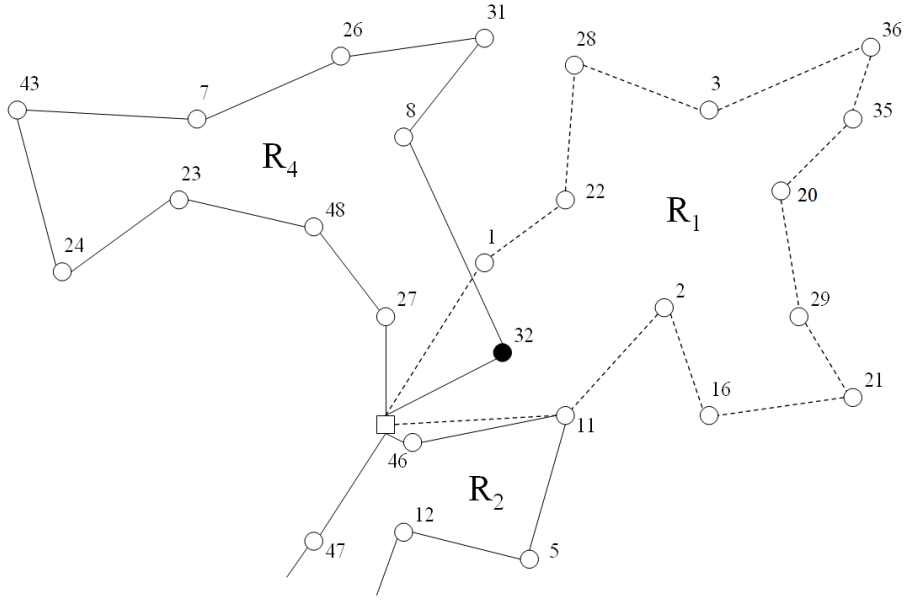

Figure 5.3: Example of partial solution to SDVRP, based on a sequential list of customers, L.

List L_{new} of sorted customers

Position in L	1	2	3	4	5	6	7	8	9	…	16	17
Customer	36	40	35	39	43	33	3	20	21	…	28	31

Position in L	18	19	20	21	22	23	24	…	29	30	31	32
Customer	13	29	10	26	7	50	24	…	8	16	23	49

Position in L	33	34	…	40	…	43	44	45	…	…	49	50
Customer	2	22	…	48	…	**32**	1	11	…	…	27	46

New list L_1 of sorted customers forming R_1

Position in L_1	1	2	3	4	5	6	7	8	9	10	11	12
Customer	36	35	3	20	21	28	29	16	2	22	**32**	1
Delivery	6	17	16	28	8	14	6	15	30	8	11	1
Spare Q of R_1	154	137	121	93	85	71	65	50	20	12	1	0

New list L_4 of sorted customers forming R_4

Position in L_4	1	2	3	4	5	6	7	8	9	10	11	12
Customer	43	31	26	7	24	8	23	48	1	27	46	
Delivery	11	11	7	19	10	23	16	17	6	15	5	
Spare Q of R_4	149	138	131	112	102	79	63	46	40	25	20	

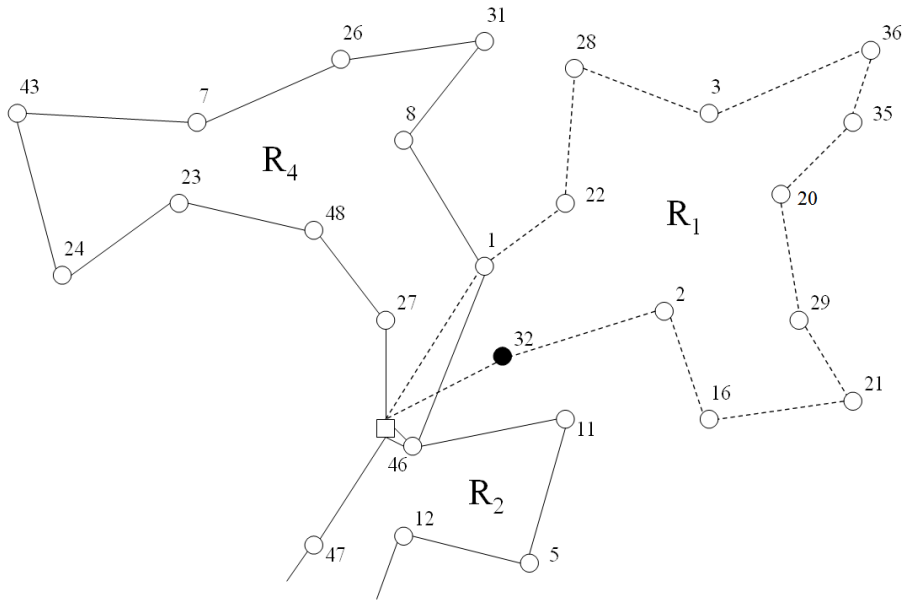

Figure 5.4: Example of iterated solution to SDVRP based on solution information from Figure 5.3 solution.

the constructive approach.

To relocate i^* in L, first the route, r^* closest to i^* is found such that i^* could be inserted at lowest cost if vehicle capacity were sufficient. The node list of this route, L_{r^*}, is then examined to determine where to place i^* so that it can fit within the vehicle capacity. This position is usually earlier in the L_{r^*} sequence of customers. Designate as i_a the customer that immediately succeeds this insertion point. In other words, the customers in L_{r^*} prior to i_a leave enough spare capacity to accommodate the i^* demand. To obtain L_{new} from L, relocate i^* to occur immediately before i_a in L. This new sequential list of customers, L_{new}, is then used to create a new solution to the SDVRP.

In the example illustrated in Figure 5.3, the customer with the highest $\Delta\theta_r$ is shown in bold text in L, $i^* = 32$. Given the design of the routes R_1 and R_4, the insertion of customer 32 into route R_1 seems more reasonable than into route R_4 where it was placed. Based on list L_1, $i_a = 1$ is identified as the last customer in L that spends the capacity of route R_1 necessary to service customer 32. Thus, customer 32 is relocated right before customer 1 in L. This relocation modifies L to produce the list L_{new} shown in Figure 5.4.

List L_{new} in Figure 5.4 is used to execute the constructive heuristic approach again and produce a new solution, as illustrated. The relocation produced the expected insertion of customer 32 into R_1 plus other perturbations to the solution; since R_1 cannot fully service customer 1 (see spare Q of R_1 after this insertion), this delivery is split among R_1 and R_4. The deliveries of R_1 and R_4 to customer 1 are then 1 and 6 units, respectively, as provided by the new L_1 and L_4 in the figure. The relocation of customer 32 in L produced a reduction in the objective function value of the complete solution from $z = 578.83$ in Figure 5.3 to $z = 577.92$ in Figure 5.4. Note that this relocation produced the transfer of customer 32 from R_4 to R_1 (similar to the standard customer shift used in multi-route improvement algorithms) plus the relocation of a split delivery in the solution (similar to the relocate split operator of Ho and Haugland, 2004). This iterative construction of new solutions continues until no improvements to the best solution found in the search are obtained after a predefined number of consecutive iterations.

5.3.3 Variable Neighborhood Descent (VND)

Variable neighborhood search (VNS) is a relatively new meta-heuristic concept based on the principle of systematically changing the neighborhood structure during the search to escape from local optima. This meta-heuristic first appears in the literature in the study of Mladenović and Hansen (1997) where this scheme is shown to outperform other heuristics on the traveling salesman problem. Given N_k ($k = 1, ..., k_{max}$), a finite set of pre-selected neighborhood structures and $N_k(x)$ the set of solutions in the kth neighborhood of x, neighborhoods N_k may be induced from one or more metric functions.

Some variable neighborhood searches have been applied to routing problems. Bräysy (2003) proposes a reactive variable neighborhood search that modifies some parameters and changes the objective function to avoid local optimality. The method is applied successfully to the VRPTW and provided four new best-solutions for the test problems used. Polacek et al. (2004) use variable neighborhood search to solve the multi-depot VRPTW (MDVRPTW). The algorithm outperforms a tabu search, found 10 new best-solutions, and demonstrated superiority on large real-world problems. Kytöjoki et al. (2007) use a variable neighborhood descent to solve large-scale VRPs and accept non-improving solutions by penalizing certain solution features. High quality solutions are found for problems involving up to 20,000 customers.

In Variable Neighborhood Descent (VND), the final solution is a local optimum with respect to all neighborhoods N_k, and thus the chances of finding a global optimum are higher than by using a single neighborhood structure. The proposed VND is defined as follows:

Step 1 Define the set of neighborhood structures to be used. Set k_{max} equal to the number of such structures.

Step 2 Find an initial solution x.

Step 3 Set $k = 1$.

Step 4 Find the first improving neighbor x' of x, $x' \in N_k(x)$.

Step 5 If a neighbor x' was found, set $x = x'$ and go to Step 4. Otherwise, set $k = k + 1$.

Step 6 If $k > k_{max}$ and there were no improvements since $k = 1$, stop.

Step 7 If $k > k_{max}$ and the solution was improved with any $N_k : k > 1$, go to Step 3. Otherwise, go to Step 4.

Neighborhood Structure

Three neighborhoods, N_1, N_2, and N_3, are used in this study and are described below. The first two neighborhoods are based on the well known customer *shift* and customer *swap*, respectively, to move and exchange customers between routes. These operators are adapted from Ho and Haugland (2004) to handle split deliveries. The third neighborhood is based on a new operator, called customer *shift**, that introduces a split delivery when a customer shift is infeasible due to a lack of vehicle capacity in the destination route.

The operators are illustrated in Figures 5.5 to 5.7. In these figures, the depot is graphically represented by a white square, routes R_1 and R_2 represent any two existing routes in the current solution, solid arrows represent the sequence of customers within R_1 and dashed arrows represent the sequence of customers within R_2. Customers are represented by a circle with shading used to differentiate the customers; customers not involved in the transformation and remaining in their original routes are white, customers originally within R_1 and moved to R_2 are gray, and customers originally within R_2 and moved to R_1 are black. Finally, and for the purpose of the operator's description, the spare capacity of route i is denoted by s_i, the demand of customer j is denoted q_j, and the delivery made by route i to customer j is denoted by y_{ij}.

- The customer *shift* is illustrated in Figure 5.5. This operator moves customer $j_1 \in R_1$ to the cheapest position in R_2. If j_1 is split among R_1 and R_2, it is removed from R_1 and the quantity y_{2j_1} is increased to q_{j_1}. This move is feasible when $s_2 \geq y_{1j_1}$.

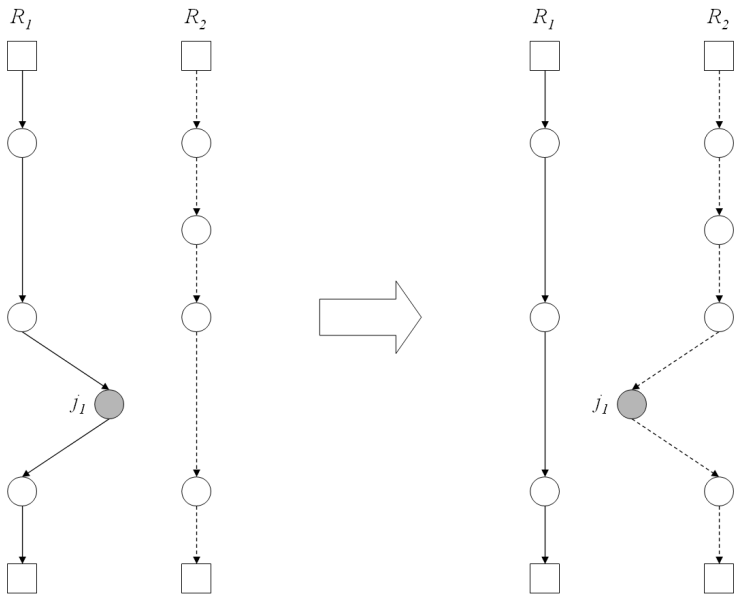

Figure 5.5: Illustration of the shift operator.

- The customer *swap* is illustrated in Figure 5.6. This operator exchanges two customers by removing $j_1 \in R_1$ and $j_2 \in R_2$ and inserting them into the cheapest positions in R_2 and R_1, respectively. If j_2 is split among R_1 and R_2, it is removed from R_2 and the quantity y_{1j_2} is increased to q_{j_2}. This move is feasible when $s_2 + y_{2j_2} \geq q_{j_1}$ and $s_1 + q_{j_1} \geq y_{2j_2}$.

- The customer *shift** is illustrated in Figure 5.7. This operator is a variant of the standard customer *shift* and moves customer $j_1 \in R_1$ to the cheapest position in R_2 and inserts a partial delivery of customer $j_2 \in R_2$ into the cheapest position in R_1. This move is feasible when $s_2 < q_{j_1}$ and $q_{j_1} - s_2 < q_{j_2}$. In other words, the move is feasible when R_2 does not have enough capacity to service j_1 and the demand q_{j_2} of customer $j_2 \in R_2$ is large enough to cover the lack of capacity in R_2. As this transformation allows both R_1 and R_2 to service j_2, the quantities y_{1j_2} and y_{2j_2} are possibly adjusted to avoid any route exhausting the entire capacity. This adjustment helps increase the number of feasible candidates that may be found later in the search.

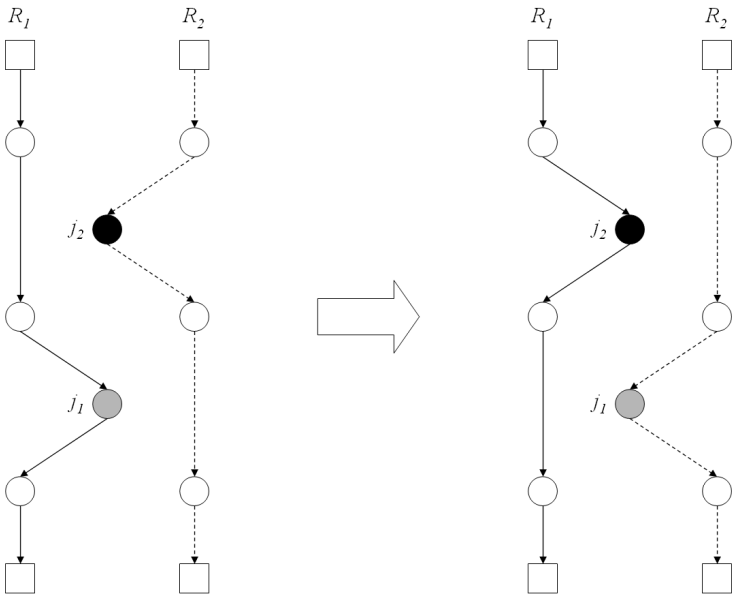

Figure 5.6: Illustration of the swap operator.

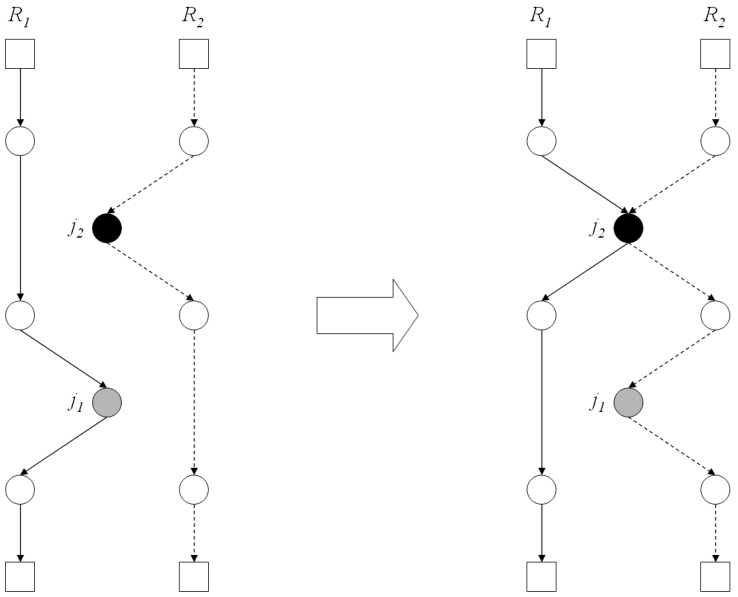

Figure 5.7: Illustration of the *shift** operator.

5.4 Experimental Results

The proposed algorithms were tested on existing problem sets available in the literature: Archetti, Hertz, and Speranza (2006), Belenguer et al. (2000), Chen et al. (2007), and Jin et al. (2007). These problems total 96. The instances from Archetti, Hertz, and Speranza (2006) include the problems with both original and randomly generated customer demands. These instances involve 50, 75, 100, 120, 150, and 199 customers and their demands are randomly generated as a function of the vehicle capacity to test the performance of the algorithms of Archetti, Hertz, and Speranza when the customer demands get large, as in Dror and Trudeau (1989). Instances 1-5 have customers uniformly located around the depot, whereas instances 6 and 7 have clustered customers. In contrast to problem 7, in problem 6 the depot is not centered with respect to the customer locations. The description of these instances can be found in Christofides and Eilon (1969) and Christofides et al. (1979). It is important to mention that the instances with random demands generated by Archetti, Hertz, and Speranza are unavailable. Instances from Boudia et al. (2007) are the same instances tested and generated by Mota et al. (2007) with the generator of Archetti, Hertz, and Speranza. When required, special notes are used at the bottom of the tables to differentiate the instances used by the authors in their corresponding publications. The instances used by Belenguer et al. (2000) include 11 problems taken from TSPLIB and another 14 problems created by randomly generating the customer demands as a function of the vehicle capacity. These instances involve 21 to 100 customers. Chen et al. (2007) recently generated a set of 21 problems involving 8 to 288 customers having a geometric symmetry, a star shape, with the customers located in concentric circles around the depot. Jin et al. (2007) used one TSPLIB instance involving 21 customers and generated four instances with 18, 21, and 22 customers.

The notation `p-aaa-nnn` is used to name each instance, where `p` is an alphabetical character to identify the publication where the problem is given (see Table 5.1), `aaa` is a string of variable length corresponding to the name of the instance adopted on the publication, and the third naming field is a three-digit integer denoting the number of customers. For instance, a-01-050 corresponds

Table 5.1: Test problems naming scheme.

Value First Naming Field p	Publication
a	Archetti, Hertz, and Speranza (2006)
b	Belenguer et al. (2000)
c	Chen et al. (2007)
j	Jin et al. (2007)

to problem 1 from Archetti, Hertz, and Speranza (2006) with 50 customers. The algorithms were implemented in C# and the experiments were carried out on a PC Pentium 4, 2.8GHz processor, and 512MB of RAM. During the experiments, parameters α and β in Equation (5.2) were set to 200 and 1000000, respectively. These values were empirically found to work reasonably well for the tested problems.

In this chapter, a constructive approach with a RAC mechanism is proposed to best allocate customers to routes. To test the performance of RAC, initial solutions are constructed with and without this mechanism and the savings obtained in the total traveled distance when the RAC is actually used are calculated. Except for one case, i.e., problem b-eil33-032, where the depot is not geographically centered with respect to the customer locations, the RAC obtains better solutions. On average, RAC savings are 32.84% on the 96 tested problems. However, these savings were found to vary with two problem characteristics: the geographical distribution of customers and the customer demand range. The RAC does well on problems with a centered depot, less well on problems where the depot is not centered. The tested problems are classified as GC when the depot is geographically centered with respect to the customers (e.g., problems a-01-050 and a-07-100), non-GC (NGC) when the depot is not geographically centered (e.g., problems a-06-120 and b-eil33-032), random (R) when the customers are randomly scattered (e.g., problems a-01-050 and b-eil33-032), or clustered (C) when customers form clusters (e.g., problems a-06-120 and a-07-100). Table 5.2 shows the percentage savings obtained when the RAC is used within the constructive approach for each problem type. Note that RAC is more effective when the depot is centered, i.e.,

Table 5.2: Impact of problem type on the benefits of the route angle control.

Problem Type	Tested problems	Savings due to RAC		
		Min	Avg	Max
GC-R	77	10.76	35.16	46.94
GC-C	11	27.85	37.33	46.92
NGC-R	1	0.00	12.83	26.19
NGC-C	7	8.49	13.97	17.86

Table 5.3: Impact of demand range on the benefits of the route angle control.

Demand range	Tested problems	Savings due to RAC		
		Min	Avg	Max
$D1$	10	8.49	21.83	37.66
$D2$	10	17.15	41.22	46.85
$D3$	10	17.86	39.56	45.38
$D4$	9	17.49	34.79	39.28
$D5$	9	14.75	35.90	41.72
$D6$	8	13.07	31.20	35.28

problem types GC-R and GC-C. Problems with clustered or scattered customers do not seem to affect the performance of the proposed angle control measure.

Table 5.3 shows the percentage savings due to the RAC for different demand ranges, $D1$ to $D6$. These ranges are used by Belenguer et al. (2000) and Archetti, Hertz, and Speranza (2006) with $D1 = [0.01 - 0.10]$ and $D6 = [0.70 - 0.90]$. The results in the table reveal that the RAC produces solutions of considerable less quality to problems with low customer demands in the range $D1$. The reason for this is that fewer vehicles are required to satisfy the capacity constraints so the threshold angle $\theta^* = \frac{2\pi}{m}$ gets larger. As a consequence, the term $\beta \times \max\{0, \frac{|\theta_{ir} - \theta^*|}{\theta_{ir} - \theta^*}\}$ in Equation (5.2) becomes zero for most insertion candidates so no penalties are applied to insertions into far routes.

Tables 5.4 to 5.6 show the solution values obtained with the proposed constructive approach (CA), iterative constructive approach (ICA), and variable neighborhood descent along with the per-

centage improvement with respect to the CA solution value and the running times. The VND was tested using the solutions obtained with the CA and ICA approaches as the initial solutions. However, using the ICA as the initial solution (ICA+VND in the table) produces better results in 78% of the tested problems. As seen in Tables 5.4 and 5.5, although the ICA approach outperforms CA in 35 of the 63 problems, the improvements occur mainly in problems with small customer demands. As demands get larger, ICA does not improve the solution found with CA. The ICA uses a very conservative scheme to improve the search based on the information of constructed solutions. Thus, it is difficult to find solutions of lower value by simply re-positioning a single customer in the list L when the customer demands are large. This fact is reinforced by the results on the Chen et al. (2007) problem set presented in Table 5.6. In this set, customer demands are 60 and 90, the vehicle capacity is 100, and the ratio of the total demand and total capacity is 1 for all problems, which means that all vehicles are fully loaded in the final solution. ICA was able to slightly improve CA in only problem c-SD15-144. In contrast, the ICA+VND approach has a stronger neighborhood structure and a more aggressive search process so the improvements over CA are noticeably higher.

The results on the instances from Archetti, Hertz, and Speranza are compared with those obtained by the best among the three tabu searches of Archetti, Hertz, and Speranza (2006) (Splitabu-DT), the scatter search (SS) of Mota et al. (2007), the hybrid approach (EMIP+VRTR) of Chen et al. (2007), and the memetic algorithm with population management (MA|PM) of Boudia et al. (2007). Since the tabu search has random elements, Archetti, Hertz, and Speranza ran each problem five times. Thus, their average values are provided in the experimental results. Similarly, values reproduced from Chen et al. (2007) correspond to median solution values from 30 different instances for each problem. The computational results of the best among the proposed approaches, ICA+VND, are presented in Tables 5.7 and 5.8. Solution values z and percentage improvements IMP of the other approaches over the objective function value of the ICA+VND solution are shown in Table 5.7. Improvements in bold font denote the cases where the ICA+VND solution has a better value. EMIP+VRTR was not tested on problems a-03-100 so these values are omitted. The results are grouped according to the actual instances used in the experiments. The instances used

Table 5.4: Computational results on problems of Archetti et al. (2006).

		CA		ICA			ICA+VND		
Problem	Demand	z	CPU	z	IMP	CPU	z	IMP	CPU
a-01-050		578.83	0.08	568.67	1.76	2.69	540.82	6.57	10.89
a-02-075		899.11	0.06	889.05	1.12	3.25	880.28	2.09	9.81
a-03-100		873.46	0.16	863.18	1.18	9.34	854.13	2.21	43.50
a-04-150		1121.33	0.33	1108.97	1.10	20.77	1088.91	2.89	129.23
a-05-199		1412.18	0.55	1412.18	-	26.66	1390.55	1.53	534.83
a-06-120		1257.48	0.41	1257.48	-	20.27	1223.28	2.72	257.30
a-07-100		827.59	0.11	826.03	0.19	6.20	824.82	0.33	21.02
a-01-050	[0.01-0.10]	477.66	0.09	477.66	-	4.42	473.22	0.93	4.52
a-02-075	[0.01-0.10]	638.10	0.19	628.30	1.53	11.14	617.65	3.20	51.28
a-03-100	[0.01-0.10]	864.06	0.42	845.95	2.10	27.34	789.16	8.67	415.47
a-04-150	[0.01-0.10]	907.36	0.56	902.48	0.54	45.80	893.49	1.53	666.20
a-05-199	[0.01-0.10]	1163.38	1.06	1126.78	3.15	181.28	1079.04	7.25	3750.44
a-06-120	[0.01-0.10]	1175.43	0.53	1169.57	0.50	28.66	1101.14	6.32	341.59
a-07-100	[0.01-0.10]	706.74	0.48	687.12	2.78	28.72	673.54	4.70	222.42
a-01-050	[0.10-0.30]	787.03	0.02	777.75	1.18	1.09	777.75	1.18	1.59
a-02-075	[0.10-0.30]	1157.90	0.05	1152.97	0.43	2.36	1099.47	5.05	13.19
a-03-100	[0.10-0.30]	1513.33	0.08	1512.37	0.06	3.97	1452.52	4.02	34.09
a-04-150	[0.10-0.30]	2124.89	0.19	2124.89	-	9.30	1978.01	6.91	164.19
a-05-199	[0.10-0.30]	2584.94	0.33	2584.94	-	16.45	2502.54	3.19	248.83
a-06-120	[0.10-0.30]	2996.54	0.13	2979.88	0.56	6.28	2806.92	6.33	54.25
a-07-100	[0.10-0.30]	1555.18	0.09	1490.76	4.14	4.56	1428.27	8.16	22.56
a-01-050	[0.10-0.50]	1098.88	0.03	1098.88	-	1.06	1045.93	4.82	2.81
a-02-075	[0.10-0.50]	1574.85	0.05	1529.71	2.87	2.77	1503.02	4.56	11.25
a-03-100	[0.10-0.50]	2029.21	0.09	2015.64	0.67	4.41	1957.55	3.53	25.16
a-04-150	[0.10-0.50]	2774.54	0.17	2774.54	-	9.06	2685.33	3.22	111.66
a-05-199	[0.10-0.50]	3615.66	0.33	3615.66	-	16.20	3450.84	4.56	339.36
a-06-120	[0.10-0.50]	4212.58	0.13	4212.58	-	6.31	4085.36	3.02	40.53
a-07-100	[0.10-0.50]	2108.74	0.08	2078.99	1.41	4.69	2046.15	2.97	11.92

Continued on next page

Table 5.4: Computational results on problems of Archetti et al. (2006) (*Continued*).

		CA		ICA			ICA+VND		
Problem	Demand	z	CPU	z	IMP	CPU	z	IMP	CPU
a-01-050	[0.10-0.90]	1604.25	0.03	1602.49	0.11	1.38	1547.32	3.55	2.83
a-02-075	[0.10-0.90]	2367.26	0.06	2348.75	0.78	3.36	2212.93	6.52	10.80
a-03-100	[0.10-0.90]	3026.61	0.09	3018.59	0.27	5.08	2925.13	3.35	19.00
a-04-150	[0.10-0.90]	4395.14	0.22	4395.14	-	10.89	4192.50	4.61	141.27
a-05-199	[0.10-0.90]	5559.98	0.38	5545.57	0.26	28.06	5192.06	6.62	662.77
a-06-120	[0.10-0.90]	6541.28	0.16	6541.28	-	7.22	6483.06	0.89	42.00
a-07-100	[0.10-0.90]	3295.74	0.11	3269.30	0.80	4.88	3178.28	3.56	12.89
a-01-050	[0.30-0.70]	1605.45	0.03	1591.06	0.90	1.55	1557.52	2.99	2.20
a-02-075	[0.30-0.70]	2348.60	0.06	2348.60	-	2.97	2241.59	4.56	11.28
a-03-100	[0.30-0.70]	3068.10	0.11	3057.85	0.33	4.78	2945.19	4.01	15.14
a-04-150	[0.30-0.70]	4395.14	0.22	4395.14	-	10.89	4192.50	4.61	143.05
a-05-199	[0.30-0.70]	5680.50	0.45	5680.50	-	18.92	5366.06	5.54	349.97
a-06-120	[0.30-0.70]	6826.12	0.14	6814.97	0.16	7.31	6591.40	3.44	59.20
a-07-100	[0.30-0.70]	3402.47	0.09	3348.16	1.60	5.03	3318.08	2.48	13.69
a-01-050	[0.70-0.90]	2246.82	0.03	2246.82	-	1.63	2215.34	1.40	2.59
a-02-075	[0.70-0.90]	3400.01	0.09	3400.01	-	3.92	3341.26	1.73	10.25
a-03-100	[0.70-0.90]	4526.40	0.13	4526.40	-	6.61	4455.14	1.57	14.31
a-04-150	[0.70-0.90]	6665.56	0.31	6665.56	-	15.34	6513.36	2.28	93.78
a-05-199	[0.70-0.90]	8692.00	0.56	8662.98	0.33	32.63	8368.35	3.72	460.89
a-06-120	[0.70-0.90]	10585.01	0.20	10585.01	-	9.77	10302.16	2.67	59.28
a-07-100	[0.70-0.90]	5196.44	0.13	5196.44	-	6.75	5058.76	2.65	20.70

z denotes objective function value obtained.

IMP denotes percentage objective function improvement over CA.

CPU denotes running time in seconds on a P4, 2.8GHz, 512MB.

Table 5.5: Computational results on the random problems of Belenguer et al. (2000).

		CA		ICA			ICA+VND		
Problem	Demand	z	CPU	z	IMP	CPU	z	IMP	CPU
b-S51D1-050	[0.01-0.10]	477.66	0.09	477.66	-	4.59	473.22	0.93	4.53
b-S51D2-050	[0.10-0.30]	759.56	0.03	745.46	1.86	1.27	732.38	3.58	4.05
b-S51D3-050	[0.10-0.50]	1034.90	0.02	1034.90	-	0.98	1001.22	3.25	2.50
b-S51D4-050	[0.10-0.90]	1740.38	0.03	1740.38	-	1.38	1708.00	1.86	2.89
b-S51D5-050	[0.30-0.70]	1421.74	0.03	1421.74	-	1.16	1404.54	1.21	1.80
b-S51D6-050	[0.70-0.90]	2266.58	0.03	2266.58	-	1.73	2230.06	1.61	2.27
b-S76D1-075	[0.01-0.10]	642.18	0.22	626.72	2.41	21.91	610.23	4.98	63.55
b-S76D2-075	[0.10-0.30]	1199.42	0.06	1196.42	0.25	2.48	1169.80	2.47	7.73
b-S76D3-075	[0.10-0.50]	1584.35	0.05	1584.35	-	2.25	1490.08	5.95	12.23
b-S76D4-075	[0.10-0.90]	2326.64	0.06	2326.64	-	2.73	2220.87	4.55	6.91
b-S101D1-100	[0.01-0.10]	854.05	0.41	831.64	2.62	47.45	765.48	10.37	210.36
b-S101D2-100	[0.10-0.30]	1510.85	0.09	1510.85	-	4.36	1444.96	4.36	26.20
b-S101D3-100	[0.10-0.50]	2167.71	0.08	2144.46	1.07	4.23	1990.28	8.19	27.84
b-S101D5-100	[0.30-0.70]	3062.17	0.09	3046.95	0.50	6.27	2999.31	2.05	18.36

z denotes objective function value obtained.

IMP denotes percentage objective function improvement over CA.

CPU denotes running time in seconds on a P4, 2.8GHz, 512MB.

Table 5.6: Computational results for problems of Chen et al. (2007).

	CA		ICA			ICA + VND		
Problem	z	CPU	z	IMP	CPU	z	IMP	CPU
c-SD01-008	25478.71	0.02	25478.71	-	0.06	22828.43	10.40	0.06
c-SD02-016	73478.71	0.00	73478.71	-	0.19	70828.43	3.61	0.22
c-SD03-016	43058.22	0.02	43058.22	-	0.16	43058.22	-	0.17
c-SD04-024	70448.03	0.00	70448.03	-	0.36	63583.51	9.74	0.55
c-SD05-032	139056.83	0.02	139056.83	-	0.61	139056.83	-	0.69
c-SD06-032	85288.45	0.03	85288.45	-	0.81	83124.14	2.54	0.94
c-SD07-040	364000.00	0.02	364000.00	-	0.94	364000.00	-	1.03
c-SD08-048	509478.71	0.03	509478.71	-	1.36	506828.43	0.52	1.75
c-SD09-048	213794.48	0.03	213794.48	-	1.45	207102.79	3.13	2.91
c-SD10-064	277291.42	0.06	277291.42	-	2.52	274783.08	0.90	3.58
c-SD11-080	1328000.01	0.08	1328000.01	-	3.56	1328000.01	-	3.97
c-SD12-080	727997.00	0.06	727997.00	-	3.59	727997.00	-	4.00
c-SD13-096	1011057.51	0.09	1011057.51	-	5.25	1011057.51	-	5.80
c-SD14-120	1092000.85	0.16	1092000.85	-	7.84	1089349.80	0.24	15.49
c-SD15-144	1522449.07	0.23	1522342.27	0.01	11.75	1516827.58	0.37	18.33
c-SD16-144	375542.10	0.25	375542.10	-	11.77	363526.95	3.20	39.71
c-SD17-160	2655992.75	0.28	2655992.75	-	14.16	2655992.75	0.00	17.42
c-SD18-160	1455999.62	0.28	1455999.62	-	13.88	1444059.28	0.82	40.38
c-SD19-192	2021283.59	0.39	2021283.59	-	20.39	2019119.29	0.11	27.64
c-SD20-240	3983999.63	0.63	3983999.63	-	32.25	3981348.58	0.07	63.18
c-SD21-288	1244552.35	1.05	1244552.35	-	52.90	1179960.15	5.19	738.49

z denotes objective function value obtained.

IMP denotes percentage objective function improvement over CA.

CPU denotes running time in seconds on a P4, 2.8GHz, 512MB.

in this study have the same customer demands used to test SS and MA|PM. However, there is no evidence to support the equality of the customer demands with the other instances (Archetti, Hertz, and Speranza, 2006; Chen, Golden, and Wasil, 2007).

In terms of solution values z in Table 5.7, ICA+VND is comparable with SS and MA|PM. On average, the ICA+VND solutions are within 1.77% and 4.34%, respectively. MA|PM is not improved in any case, but SS is improved in 10 problems with large demands (see the three largest demand ranges in the table). There is a tendency of ICA+VND to perform better as the customer demands get larger. While the other approaches tend to outperform ICA+VND, across the board the ICA+VND is competitive with those approaches. Column m shows the minimum possible fleet size to satisfy all customer demands, which is also the number of vehicles in the final ICA+VND, SS, and MA|PM solutions. Column m' shows the number of vehicles in the final feasible solutions of the tabu search. Chen et al. (2007) do not report the fleet size for EMIP+VRTR in their computational results. A very important consideration is that ICA+VND usually utilizes less vehicles than the tabu search (see bold numbers in column m). Using more vehicles may reduce objective function values thereby somewhat obscuring solution comparisons. Table 5.8 summarizes the reported running times for the existing algorithms and the running times for the ICA+VND approach. The ICA+VND approach only requires a single run, and per Table 5.8 obtains those solutions quicker than the other approaches in most cases.

Computational results on the TSPLIB instances solved by Belenguer et al. (2000) are presented in Table 5.9. This table compares the ICA+VND solution values z with the bounds obtained by Belenguer et al. using a heuristic method and a cutting plane algorithm, the bounds found by Liu (2005) using a branch-and-price approach (B&P), and the bounds obtained by Jin et al. (2008) with a column generation approach (omitted values in the table are not published). ICA+VND solution values are also compared with those found by Boudia et al. (2007) with MA|PM. ICA+VND is competitive with the other approaches and clearly dominates B&P on these instances. To compare with Belenguer et al. and MA|PM, euclidean inter-node distances are also truncated to the nearest integer. In these instances, ICA+VND solutions are within 5.78% above the lower bounds of

Table 5.7: Computational results of ICA+VND on instances of Archetti et al. (2006).

			ICA+VND[(a)]	SS[(a)]		MA|PM[(a)]			Splitabu-DT[(b)]		EMIP+VRTR[(c)]	
Problem	Demand	m	z	z	IMP	z	IMP	m'	z	IMP	z	IMP
a-01-050		5	540.82	531.02	1.81	524.61	3.00	5	533.55	1.34	524.61	3.00
a-02-075		10	880.28	839.75	4.60	823.89	6.41	10	849.54	3.49	840.18	4.56
a-03-100		8	854.13	835.82	2.14	829.44	2.89	8	835.62	2.17	-	-
a-04-150		12	1088.91	1056.92	2.94	1042.37	4.27	12	1069.84	1.75	1041.99	4.31
a-05-199		16	1390.55	1340.44	3.60	1311.59	5.68	16	1342.85	3.43	1307.40	5.98
a-06-120		7	1223.28	1042.97	14.74	1041.20	14.88	7	1056.01	13.67	1043.18	14.72
a-07-100		10	824.82	820.92	0.47	819.56	0.64		825.32	**-0.06**	819.56	0.64
a-01-050	[0.01-0.10]	3	473.22	460.79	2.63	460.79	2.63		463.76	2.00	457.21	3.38
a-02-075	[0.01-0.10]	4	617.65	602.67	2.43	600.06	2.85		605.24	2.01	598.25	3.14
a-03-100	[0.01-0.10]	5	789.16	729.67	7.54	726.81	7.90		752.20	4.68	-	-
a-04-150	[0.01-0.10]	8	893.49	883.05	1.17	875.61	2.00		890.95	0.28	875.16	2.05
a-05-199	[0.01-0.10]	10	1079.04	1039.51	3.66	1018.71	5.59		1056.27	2.11	1040.20	3.60
a-06-120	[0.01-0.10]	6	1101.14	979.57	11.04	976.57	11.31		1084.70	1.49	985.17	10.53
a-07-100	[0.01-0.10]	5	673.54	633.80	5.90	649.73	3.53		648.74	3.68	651.44	3.28

Continued on next page

Table 5.7: Computational results of ICA+VND on instances of Archetti et al. (2006) (*Continued*).

			ICA+VND[(a)]	SS[(a)]		MA|PM[(a)]			Splitabu-DT[(b)]		EMIP+VRTR[(c)]	
Problem	Demand	m	z	z	IMP	z	IMP	m'	z	IMP	z	IMP
a-01-050	[0.10-0.30]	**10**	777.75	769.60	1.05	751.41	3.39	11	761.40	2.10	723.57	6.97
a-02-075	[0.10-0.30]	**15**	1099.47	1074.01	2.32	1074.46	2.27	16	1095.32	0.38	1081.10	1.67
a-03-100	[0.10-0.30]	**20**	1452.52	1416.48	2.48	1392.85	4.11	22	1424.81	1.91	-	-
a-04-150	[0.10-0.30]	**29**	1978.01	1974.70	0.17	1878.71	5.02	32	1918.25	3.02	1844.96	6.73
a-05-199	[0.10-0.30]	**38**	2502.54	2435.08	2.70	2340.14	6.49	41	2384.15	4.73	2258.66	9.75
a-06-120	[0.10-0.30]	**23**	2806.92	2783.10	0.85	2720.38	3.08	26	2918.71	**-3.98**	2568.90	8.48
a-07-100	[0.10-0.30]	20	1428.27	1423.49	0.34	1417.28	0.77		1462.01	**-2.36**	1414.33	0.98
a-01-050	[0.10-0.50]	**15**	1045.93	1025.91	1.91	988.31	5.51	16	1008.67	3.56	943.86	9.76
a-02-075	[0.10-0.50]	**22**	1503.02	1484.62	1.22	1413.80	5.94	24	1443.62	3.95	1393.53	7.28
a-03-100	[0.10-0.50]	**29**	1957.55	1926.15	1.60	1845.30	5.73	33	1894.72	3.21	-	-
a-04-150	[0.10-0.50]	**43**	2685.33	2649.97	1.32	2561.65	4.61	49	2632.71	1.96	2532.93	5.68
a-05-199	[0.10-0.50]	**56**	3450.84	3310.71	4.06	3191.25	7.52	63	3284.47	4.82	3202.57	7.19
a-06-120	[0.10-0.50]	**34**	4085.36	3996.29	2.18	3934.39	3.70	40	4206.12	**-2.96**	3687.06	9.75
a-07-100	[0.10-0.50]	29	2046.15	2022.30	1.17	1994.59	2.52		2029.99	0.79	1973.34	3.56

Continued on next page

Table 5.7: Computational results of ICA+VND on instances of Archetti et al. (2006) (*Continued*).

			ICA+VND[(a)]	SS[(a)]		MA|PM[(a)]			Splitabu-DT[(b)]		EMIP+VRTR[(c)]	
Problem	Demand	m	z	z	IMP	z	IMP	m'	z	IMP	z	IMP
a-01-050	[0.10-0.90]	**25**	1547.32	1580.77	**-2.16**	1467.06	5.19	26	1469.92	5.00	1408.34	8.98
a-02-075	[0.10-0.90]	**37**	2212.93	2233.08	**-0.91**	2102.58	4.99	41	2124.43	4.00	2056.54	7.07
a-03-100	[0.10-0.90]	**48**	2925.13	2932.34	**-0.25**	2780.95	4.93	56	2794.08	4.48	-	-
a-04-150	[0.10-0.90]	**73**	4192.50	4185.68	0.16	4045.87	3.50	84	3909.72	6.74	3945.38	5.89
a-05-199	[0.10-0.90]	**93**	5192.06	5085.64	2.05	4941.22	4.83	107	4853.83	6.51	5094.61	1.88
a-06-120	[0.10-0.90]	**56**	6483.06	6361.46	1.88	6318.37	2.54	67	6583.97	**-1.56**	6079.14	6.23
a-07-100	[0.10-0.90]	48	3178.28	3187.44	**-0.29**	3113.72	2.03		3101.53	2.41	3162.22	0.51
a-01-050	[0.30-0.70]	**25**	1557.52	1568.04	**-0.68**	1477.01	5.17	26	1496.90	3.89	1408.68	9.56
a-02-075	[0.30-0.70]	**37**	2241.59	2228.90	0.57	2132.16	4.88	39	2160.51	3.62	2112.61	5.75
a-03-100	[0.30-0.70]	**49**	2945.19	2986.33	**-1.40**	2858.87	2.93	53	2870.50	2.54	-	-
a-04-150	[0.30-0.70]	**73**	4192.50	4185.68	0.16	4045.87	3.50	80	4039.70	3.64	4011.74	4.31
a-05-199	[0.30-0.70]	**96**	5366.06	5265.01	1.88	5155.36	3.93	103	5102.84	4.91	5088.08	5.18
a-06-120	[0.30-0.70]	**58**	6591.40	6481.09	1.67	6424.71	2.53	65	6639.55	**-0.73**	6123.96	7.09
a-07-100	[0.30-0.70]	49	3318.08	3248.76	2.09	3155.69	4.89		3038.02	8.44	3134.56	5.53
a-01-050	[0.70-0.90]	**40**	2215.34	2312.48	**-4.38**	2154.35	2.75	42	2165.21	2.26	2056.01	7.19
a-02-075	[0.70-0.90]	**60**	3341.26	3387.86	**-1.39**	3200.35	4.22	61	3180.64	4.81	3067.19	8.20
a-03-100	[0.70-0.90]	**80**	4455.14	4580.98	**-2.82**	4312.95	3.19	82	4302.31	3.43	-	-
a-04-150	[0.70-0.90]	**119**	6513.36	6479.46	0.52	6267.48	3.78	123	6196.36	4.87	5950.35	8.64
a-05-199	[0.70-0.90]	**158**	8368.35	8323.72	0.53	8081.58	3.43	162	7944.63	5.06	7207.04	13.88
a-06-120	[0.70-0.90]	**95**	10302.16	10158.32	1.40	10063.47	2.32	99	10304.08	**-0.02**	8941.79	13.20
a-07-100	[0.70-0.90]	80	5058.76	5065.26	**-0.13**	4919.48	2.75		4867.79	3.78	4779.13	5.53

z denotes objective function value obtained. IMP denotes percentage objective function reduction over ICA+VND.

m denotes number of vehicles in final ICA+VND, SS, and MA|PM final solutions. m' denotes number of vehicles in final Splitabu-DT solutions.

[(a)] Tested instances were generated by Mota et al. (2007).

[(b)] Tested instances were generated by Archetti et al. (2006). ICA+VND run using similar problem generator.

[(c)] Tested instances were generated by Chen et al. (2007).

Table 5.8: Running time in seconds for existing approaches and the ICA+VND approach on instances of Archetti et al. (2006).

Problem	Demand	ICA+VND[a]	SS[b]	MA|PM[c]	Splitabu-DT[d]	EMIP+VTR[e]
a-01-050		10.89	24.80	8.53	13.20	1.80
a-02-075		9.81	61.66	35.72	35.80	4.00
a-03-100		43.50	108.80	34.59	57.60	-
a-04-150		129.23	261.28	103.69	389.00	10.00
a-05-199		534.83	352.31	353.84	386.40	18.10
a-06-120		257.30	131.34	50.92	38.40	5.60
a-07-100		21.02	108.41	42.89	49.00	3.70
a-01-050	[0.01-0.10]	4.52	26.86	12.38	4.80	1.90
a-02-075	[0.01-0.10]	51.28	68.80	18.75	13.00	25.80
a-03-100	[0.01-0.10]	415.47	125.06	37.12	31.20	-
a-04-150	[0.01-0.10]	666.20	352.09	100.27	172.80	107.80
a-05-199	[0.01-0.10]	3750.44	963.84	356.22	525.80	413.40
a-06-120	[0.01-0.10]	341.59	163.28	72.98	42.40	36.40
a-07-100	[0.01-0.10]	222.42	80.56	34.97	57.80	53.90
a-01-050	[0.10-0.30]	1.59	26.31	10.22	21.80	3.40
a-02-075	[0.10-0.30]	13.19	86.02	34.14	45.40	57.00
a-03-100	[0.10-0.30]	34.09	98.00	78.06	95.80	-
a-04-150	[0.10-0.30]	164.19	10.06	147.89	393.20	308.00
a-05-199	[0.10-0.30]	248.83	19.11	347.14	754.80	618.50
a-06-120	[0.10-0.30]	54.25	11.33	144.19	142.60	136.40
a-07-100	[0.10-0.30]	22.56	151.25	43.27	146.00	126.50
a-01-050	[0.10-0.50]	2.81	3.84	12.49	28.20	14.70
a-02-075	[0.10-0.50]	11.25	6.09	37.38	123.20	214.00
a-03-100	[0.10-0.50]	25.16	7.55	28.39	136.20	-
a-04-150	[0.10-0.50]	111.66	16.17	224.89	739.20	630.50
a-05-199	[0.10-0.50]	339.36	20.64	436.20	2668.00	1775.70
a-06-120	[0.10-0.50]	40.53	63.80	163.14	268.00	220.70
a-07-100	[0.10-0.50]	11.92	41.23	51.31	292.80	287.60

Continued on next page

Table 5.8: Running time in seconds for existing approaches and the ICA+VND approach on instances of Archetti et al. (2006) (*Continued*).

Problem	Demand	ICA+VND[(a)]	SS[(b)]	MA|PM[(c)]	Splitabu-DT[(d)]	EMIP+VTR[(e)]
a-01-050	[0.10-0.90]	2.83	3.91	21.42	60.80	55.40
a-02-075	[0.10-0.90]	10.80	6.64	46.11	193.40	401.10
a-03-100	[0.10-0.90]	19.00	9.16	84.38	648.60	-
a-04-150	[0.10-0.90]	141.27	25.03	244.91	2278.00	2220.00
a-05-199	[0.10-0.90]	662.77	71.09	725.69	3297.20	3038.10
a-06-120	[0.10-0.90]	42.00	15.86	196.14	877.80	722.80
a-07-100	[0.10-0.90]	12.89	9.08	52.13	259.60	251.20
a-01-050	[0.30-0.70]	2.20	4.25	24.53	48.60	47.90
a-02-075	[0.30-0.70]	11.28	7.14	51.78	128.60	509.60
a-03-100	[0.30-0.70]	15.14	10.36	100.16	810.20	-
a-04-150	[0.30-0.70]	143.05	19.38	244.86	3008.00	3028.30
a-05-199	[0.30-0.70]	349.97	120.28	749.94	3565.60	3035.70
a-06-120	[0.30-0.70]	59.20	17.16	271.39	658.60	605.40
a-07-100	[0.30-0.70]	13.69	9.73	91.31	777.80	716.50
a-01-050	[0.70-0.90]	2.59	4.13	22.91	106.40	135.40
a-02-075	[0.70-0.90]	10.25	7.66	27.48	869.20	811.00
a-03-100	[0.70-0.90]	14.31	12.06	55.75	1398.40	-
a-04-150	[0.70-0.90]	93.78	131.91	401.62	10223.20	10038.80
a-05-199	[0.70-0.90]	460.89	165.28	571.70	21849.20	12542.30
a-06-120	[0.70-0.90]	59.28	20.17	298.08	1825.60	725.40
a-07-100	[0.70-0.90]	20.70	9.19	180.11	1004.40	1024.30

[(a)]P4, 512MB, 2.8 GHz; [(b)]P4, 1.0GB, 2.4GHz; [(c)]3GHz; [(d)]P4, 256MB, 2.4GHz); [(e)]P4, 512MB, 1.7GHz.

Table 5.9: Computational results of ICA+VND on some TSPLIB VRP instances.

	ICA+VND		Belenguer et al. (2000)			MA|PM		
Problem	$z^{(a)}$	CPU$^{(b)}$	UB$^{(a)}$	LB	%ALB	$z^{(a)}$	CPU$^{(c)}$	IMP
b-eil22-021	375	0.70	375	375	0.00	375	4.11	0.00
b-eil23-022	570	0.59	569	569	0.18	569	5.47	0.18
b-eil30-029	520	2.22	510	508	2.31	503	5.70	3.27
b-eil33-032	869	1.86	835	833	4.14	835	5.19	3.91
b-eil51-050	538	10.89	521	511.57	4.91	521	7.28	3.16
b-eilA76-075	875	9.81	832	782.7	10.55	828	35.94	5.37
b-eilB76-075	1055	16.42	1023	937.47	11.14	1019	13.09	3.41
b-eilC76-075	751	26.25	735	706.01	5.99	738	14.75	1.73
b-eilD76-075	714	29.92	683	659.43	7.64	682	23.12	4.48
b-eilA101-100	842	43.50	817	793.48	5.76	818	25.25	2.85
b-eilB101-100	1129	31.13	1077	1005.85	10.91	1082	21.81	4.16

Continued on next page

Belenguer et al. and 2.96% above MA|PM, on average. Although the percentages above the lower bounds of B&P are higher, note that these percentages change with the bounds and also with the strategy to calculate the inter-node distances. However, the upper bounds of B&P are improved in all but one problem, b-eil30-029 (see bold type in columns UB).

The computational results of ICA+VND on the random problems of Belenguer et al. (2000) are presented in Table 5.10. In this table the results are also compared with those of Chen et al. (2007), who used the EMIP+VRTR to solve only the problems with large average demands (omitted values in the table are not published). In these instances ICA+VND is able to improve the upper bounds of Belenguer et al. in problems with large demands with 50 and 75 customers and find the same solution value on two other problems in demand ranges $D2$ and $D3$. With respect to MA|PM, ICA+VND solutions are within 4.04% on average in this problem set. In terms of running time, ICA+VND is highly competitive and finds those solutions quicker than the other approaches in almost all cases.

Table 5.9: Computational results of ICA+VND on some TSPLIB VRP instances (*Continued*).

	ICA+VND		B&P			Jin et al. (2008)			
Problem	z	CPU[b]	UB	LB	%ALB	UB	LB	CPU[d]	%ALB
b-eil22-021	375.28	0.70	**376.00**	373.60	0.45	-	-	-	-
b-eil23-022	569.75	0.59	**608.00**	564.30	0.96	-	-	-	-
b-eil30-029	521.48	2.22	515.30	507.20	2.74	-	-	-	-
b-eil33-032	870.35	1.86	**873.40**	830.20	4.61	-	-	-	-
b-eil51-050	540.82	10.89	**558.50**	507.60	6.14	-	-	-	-
b-eilA76-075	880.28	9.81	**900.70**	800.30	9.09	-	-	-	-
b-eilB76-075	1059.57	16.42	**1163.10**	965.70	8.86	**1063.75**	981.14	45084.00	7.40
b-eilC76-075	758.49	26.25	**809.30**	711.20	6.23	-	-	-	-
b-eilD76-075	719.41	29.92	**768.80**	652.30	9.33	-	-	-	-
b-eilA101-100	854.13	43.50	**910.20**	797.50	6.63	-	-	-	-
b-eilB101-100	1142.02	31.13	**1174.10**	1013.90	11.22	-	-	-	-

z denotes objective function value obtained.

CPU denotes running time in seconds.

IMP denotes percentage objective function reduction over ICA+VND.

%ALB denotes percent ICA+VND above lower bound.

MA|PM uses one more vehicle than the minimum fleet size on instance b-eil30-029.

[a] Objective function value obtained with euclidean distances truncated to the nearest integer.

[b]P4, 512MB, 2.8GHz; [c]PC 3.0GHz; [d]P4, 2GB, 2.8GHz.

Table 5.10: Computational results of ICA+VND on the random problems of Belenguer et al. (2000).

	ICA+VND		Belenguer et al. (2000)			MA|PM		
Problem	$z^{(a)}$	CPU$^{(b)}$	UB$^{(a)}$	LB	%ALB	$z^{(a)}$	CPU$^{(c)}$	IMP
b-S51D1-050	469	4.53	458	454	3.20	458	8.77	2.35
b-S51D2-050	726	4.05	726	676.63	6.80	707	7.44	2.62
b-S51D3-050	994	2.50	972	905.22	8.93	945	7.84	4.93
b-S51D4-050	1700	2.89	1677	1520.67	10.55	1578	11.98	7.18
b-S51D5-050	1399	1.80	**1440**	1272.86	9.02	1351	16.72	3.43
b-S51D6-050	2221	2.27	**2327**	2113.03	4.86	2182	9.92	1.76
b-S76D1-075	603	63.55	594	584.87	3.01	592	15.23	1.82
b-S76D2-075	1165	7.73	1147	1020.32	12.42	1089	30.5	6.52
b-S76D3-075	1485	12.23	1474	1346.29	9.34	1427	12.89	3.91
b-S76D4-075	2205	6.91	**2257**	2011.64	8.77	2117	8.76	3.99
b-S101D1-100	757	210.36	716	700.56	7.46	717	49.75	5.28
b-S101D2-100	1431	26.20	1393	1270.97	11.18	1372	31.72	4.12
b-S101D3-100	1975	27.84	1975	1739.66	11.92	1891	33.98	4.25
b-S101D5-100	2985	18.36	2915	2630.43	11.88	2854	18.66	4.39

Continued on next page

Table 5.10: Computational results of ICA+VND on the random problems of Belenguer et al. (2000) (*Continued*).

	ICA+VND		B&P			EMIP+VRTR			Jin et al. (2008)			
Problem	z	CPU[(b)]	UB	LB	%ALB	z	CPU[(d)]	IMP	UB	LB	CPU[(e)]	%ALB
b-S51D1-050	473.22	4.53	**513.90**	449.90	4.93	-	-	-	-	-	-	-
b-S51D2-050	732.38	4.05	**1296.50**	556.70	23.99	-	-	-	723.37	694.98	5978	5.11
b-S51D3-050	1001.22	2.50	986.00	956.00	4.52	-	-	-	968.85	922.72	607	7.84
b-S51D4-050	1708.00	2.89	1654.00	1623.00	4.98	1586.50	201.74	7.11	1657.61	1505.35	260	11.86
b-S51D5-050	1404.54	1.80	**1434.00**	1416.00	-0.82	1355.50	201.62	3.49	**1439.92**	1297.46	46	7.62
b-S51D6-050	2230.06	2.27	**2316.00**	2270.00	-1.79	2197.80	301.90	1.45	**2300.21**	2108.59	243	5.45
b-S76D1-075	610.23	63.55	-	-	-	-	-	-	-	-	-	-
b-S76D2-075	1169.80	7.73	-	-	-	-	-	-	**1185.72**	1066.17	12806	8.86
b-S76D3-075	1490.08	12.23	-	-	-	-	-	-	**1504.94**	1397.43	2030	6.22
b-S76D4-075	2220.87	6.91	2205.00	2178.00	1.93	2136.40	601.92	3.80	2219.07	2019.91	1813	9.05
b-S101D1-100	765.48	210.36	-	-	-	-	-	-	-	-	-	-
b-S101D2-100	1444.96	26.20	-	-	-	-	-	-	**1474.51**	1349.77	47658	6.59
b-S101D3-100	1990.28	27.84	-	-	-	-	-	-	**2012.86**	1837.33	7959	7.69
b-S101D5-100	2999.31	18.36	-	-	-	2846.20	645.99	5.10	2954.96	2725.5	847	9.13

z denotes objective function value obtained. CPU denotes running time in seconds.

IMP denotes percentage objective function reduction over ICA+VND. %ALB denotes percent ICA+VND above lower bound.

MA|PM uses one more vehicle than the minimum fleet size on instance b-eil30-029.

[(a)] Objective function value obtained with euclidean distances truncated to the nearest integer. [(b)]P4, 512MB, 2.8GHz. [(c)]PC 3.0GHz

[(d)]P4, 512MB, 1.7GHz. [(e)]P4, 2GB, 2.8GHz.

Table 5.11 shows the computational results of ICA+VND on the problem set recently generated by Chen et al. (2007). Each algorithm is presented with the objective function value z and the running time in seconds. The last column, m, in the table contains the number of vehicles in the ICA+VND final solutions. Bold type indicates the best known solution value z to each problem. These problems were generated to have a geometric symmetry, a star shape, with the customers located in concentric circles around the depot (Chen et al., 2007). As one might expect the heuristics, each of which rely upon the proposed route angle computation, do extremely well on these problems. With ICA+VND, 16 of the 21 problems are improved. By examining Table 5.6 it is seen that CA improved upon 12 of the 21 problems. It is not possible to compare the number of vehicles in the final feasible solutions as this information is not published for EMIP+VRTR. In Figure 5.8 the ICA+VND final solution to problem c-SD10-64 is shown. This problem is also illustrated in Chen et al. (2007). The ICA+VND solution has a lower value and uses one less vehicle.

In Table 5.12, solution values z and running times obtained with ICA+VND on instances of Jin et al. (2007) are presented. In this table, results are compared with the optimal solutions found by Jin et al. (2007) with their TSVI approach. ICA+VND found the optimal solution in problems j-eil22-021 and j-J2-021, whereas a small deviation from optimality was obtained in problem j-J1-018. The obtained solutions are within 4.18% of the optimal value, on average.

5.5 Conclusions and Future Directions

This chapter provided a background on the SDVRP and approaches to solve the problem. Three local heuristic search algorithms are presented to solve the SDVRP with the minimum fleet size, examine their performance on available benchmark test problems, and offer insight into heuristic performance. These algorithms are then compared to available algorithms based on a thorough empirical study. The first algorithm is a constructive approach that uses a new route angle control mechanism to quickly find high quality solutions on seven benchmark problems. This approach provides solutions within 9% of the best known solutions on a set of previously employed benchmark

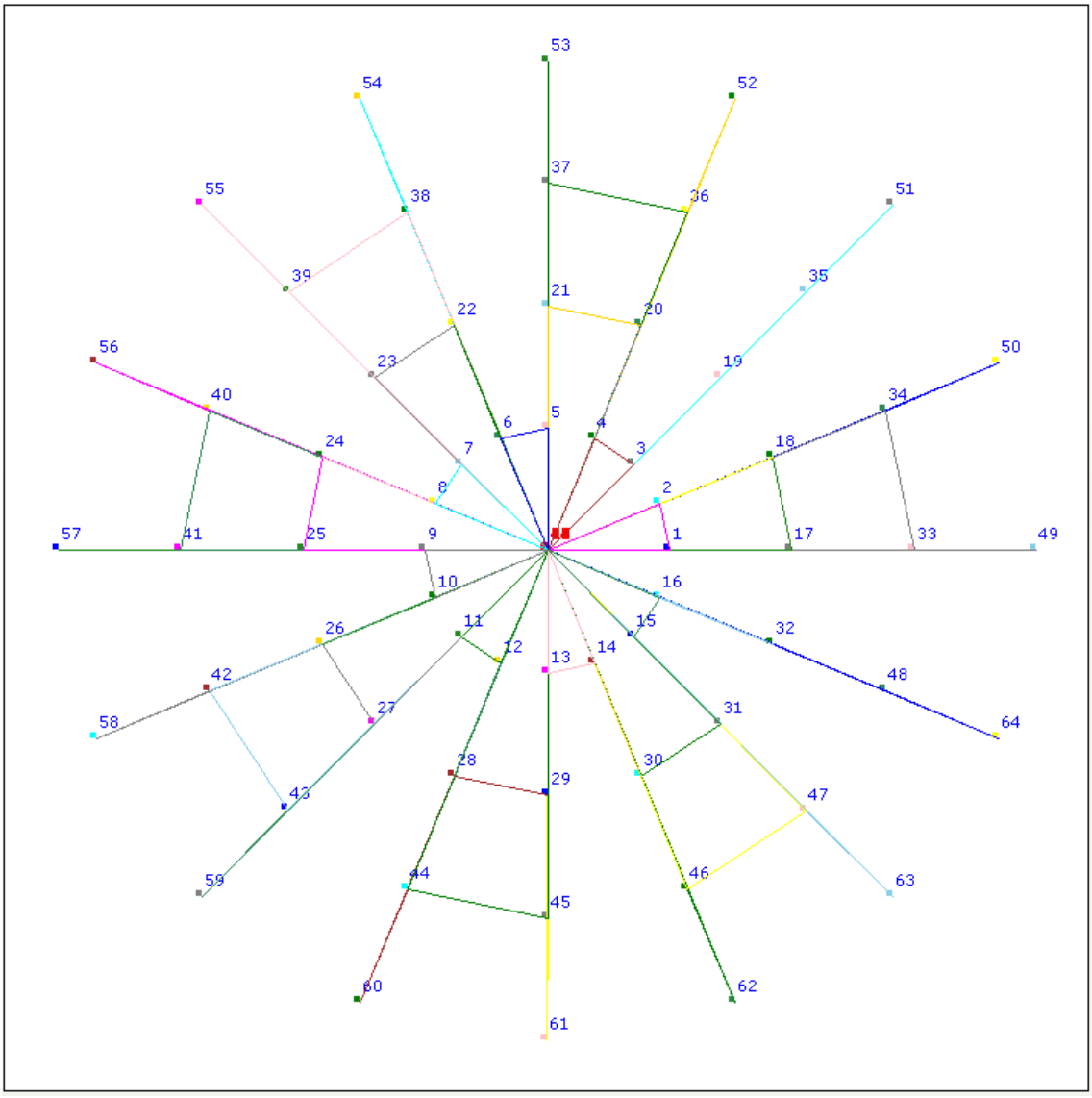

Figure 5.8: ICA + VND solution to problem cheSD10-64. Total distance is 2,747.83 with 48 vehicles.

Table 5.11: Comparison of ICA+VND on Chen et al. (2007) problem set and approach.

	EMIP+VRTR		ICA+VND		
Problem	z	CPU[(b)]	z[(a)]	CPU[(c)]	m
c-SD01-008	**228.28**	0.7	**228.28**	0.06	6
c-SD02-016	714.40	54.4	**708.28**	0.22	12
c-SD03-016	430.61	67.3	**430.58**	0.17	12
c-SD04-024	**631.06**	400	635.84	0.55	18
c-SD05-032	1408.12	402.7	**1390.57**	0.69	24
c-SD06-032	**831.21**	408.3	831.24	0.94	24
c-SD07-040	3714.40	403.2	**3640.00**	1.03	30
c-SD08-048	5200.00	404.1	**5068.28**	1.75	36
c-SD09-048	**2059.84**	404.3	2071.03	2.91	36
c-SD10-064	2749.11	400	**2747.83**	3.58	48
c-SD11-080	13612.12	400.1	**13280.00**	3.97	60
c-SD12-080	7399.06	408.3	**7279.97**	4.00	60
c-SD13-096	10367.06	404.5	**10110.58**	5.80	72
c-SD14-120	11023.00	5021.7	**10893.50**	15.49	90
c-SD15-144	15271.77	5042.3	**15168.28**	18.33	108
c-SD16-144	**3449.05**	5014.7	3635.27	39.71	108
c-SD17-160	26665.76	5023.6	**26559.93**	17.42	120
c-SD18-160	14546.58	5028.6	**14440.59**	40.38	120
c-SD19-192	20559.21	5034.2	**20191.19**	27.64	144
c-SD20-240	40408.22	5053	**39813.49**	63.18	180
c-SD21-288	**11491.67**	5051	11799.60	738.49	216

z denotes objective function value obtained: [(a)] values divided by 100 for comparison purposes (see Table 5.6).

CPU denotes running time in seconds: [(b)]P4, 512MB, 1.7GHz; [(c)]P4, 512MB, 2.8GHz.

m denotes the number of vehicles in the ICA+VND final solution. EMIP+VRTR vehicles not published.

Table 5.12: Computational results of ICA+VND versus optimality on instances of Jin et al. (2007).

	TSVI		ICA+VND		% above
Problem	z^*	CPU	z	CPU	optimality
j-eil22-021	375.28	17 h	375.28	0.7 s	**0.00**
j-J1-018	127.39	13 h	127.76	0.3 s	0.29
j-J2-021	388.44	84 h	388.44	0.5 s	**0.00**
j-J3-022	367.93	17 h	409.19	0.5 s	11.21
j-J4-022	372.2	13 h	407.16	0.5 s	9.39

z^* denotes optimal objective function value.

z denotes objective function value obtained.

CPU denotes running time.

problems. The second algorithm is an iterative approach that executes the constructive approach repeatedly. This algorithm uses the knowledge from past solutions to influence future decisions in the constructive approach and can provide good feasible solutions at a relatively low computational time, again when applied to the set of previously employed benchmark problems. The third algorithm is a variable neighborhood descent that produces the best solutions. On average, ICA+VND found solutions whose values are within 4.18% of optimality on existing benchmark problems, while the optimal solutions were obtained within a second for two problems involving 21 customers. When tested on the newest benchmark problems available for SDVRP research each of the proposed approaches improved significantly upon existing solutions. Overall, the new algorithms were shown to be competitive on general forms of SDVRP instances and a particular good choice for special classes of SDVRP instances.

The proposed route angle control mechanism is easy to implement and looks useful to solve the SDVRP, specially in problems with large customer demands. In the future, other methods can be explored to estimate the threshold angle used by this mechanism and be able to perform better with the constructive approach, especially in problems where the depot is not centered with respect to the customer locations. Although the proposed constructive approach tends to produce non-crossing

routes, optimal solutions can have crossing as well as inner routes so the CA can be guided to design such routes. In addition, more aggressive strategies can be investigated to modify the sequence of customers L used by the constructive approach and force a better exploration of the search space. These strategies have been tested to produce solutions with common attributes and similarities with the best known solutions and can potentially be used for recombination operators and produce high quality solutions.

Chapter 6

A Ring-Based Diversification Scheme for Routing Problems[1]

6.1 Introduction

The vehicle routing problem (VRP) was introduced almost 50 years ago and is still under active investigation by practitioners and researchers. In its classical version, the problem is to effectively design routes that a fleet of homogeneous vehicles will follow to supply the demand of geographically scattered customers without exceeding the vehicle capacity and considering that customers can be visited by exactly one vehicle. Traditionally, solution techniques for the VRP have been classified as exact approaches, classical heuristic algorithms (i.e., constructive, saving, improvement, sweep, petal, and matching algorithms), and metaheuristic algorithms (i.e., tabu search, genetic algorithms, simulated annealing, etc.). See Bodin and Golden (1981), Laporte et al. (2000), Toth and Vigo (2002), Cordeau et al. (2002), Cordeau et al. (2005), and Laporte (2007) for a full complete survey and description of these techniques. The split delivery vehicle routing problem (SDVRP) is a variant of the VRP where individual customer demands can be supplied by multiple vehicles. In contrast to the VRP, there is a limited number of heuristic solution techniques to solve the SDVRP

[1]This chapter is found as Aleman et al. (2008).

including local (Dror and Trudeau, 1989) and tabu search (Archetti, Hertz, and Speranza, 2006). A scatter search method (Mota et al., 2007), a hybrid approach (Chen et al., 2007), a memetic algorithm (Boudia et al., 2007), and a column generation approach (Jin et al., 2008) were recently developed to effectively solve benchmark problems.

Most of the existing techniques to solve the SDVRP perform an aggressive search in certain regions of the solution space, but do not employ diversification strategies to make a better exploration of the space. Diversification methods are usually used within heuristic search methods to increase the effectiveness of the search procedure particularly on hard problems. The exploration of different regions of the solution space helps to overcome any local optimum and increase the chance of finding a global optimal solution. When the search appears to have stagnated, it is useful to examine ways to move the search process into other areas of the search space that may not have been explored. If some predefined criteria are met, the algorithm moves to a "diversified" solution whose attributes differ from those of the already evaluated solutions. Resuming the search from a diversified solution is intended to explore new regions of the search space. Although the exploration of different regions in the solution space can help to find better solutions, the cost in processing time may be high and hence it is sometimes unattractive to diversify the search. This chapter examines this search process tradeoff.

This chapter presents a new diversification scheme for routing problems applied to the SDVRP. This scheme is based on a geographical division of the problem by means of concentric rings centered at the depot that temporarily exclude a subset of customers. A partial solution to the original problem is created and the excluded customers are then incorporated into the solution by means of a constructive approach until a complete solution is obtained. Different ring settings produce varied partitions and thus different solutions to the original problem are obtained. The search is restarted from those solutions and improved via a variable neighborhood descent. The diversification scheme created is used with the constructive approach (CA) and iterative constructive approach (ICA) with route angle control and the variable neighborhood descent (VND) described in Aleman et al. (2007) to obtain SDVRP solutions. The remainder of this chapter is organized as follows.

Section 6.2 provides an up-to-date literature review of the SDVRP and diversification methods applied to VRPs and SDVRPs. The proposed diversification method is described in Section 6.3 and the solution approach is given in Section 6.3.2. Computational results are presented in Section 6.4 with conclusions presented in Section 6.5.

6.2 Background

In this section, an up-to-date literature review of the SDVRP, a review of the CA, ICA and VND approaches of Aleman et al. (2007), and representative diversification methods applied to VRPs and SDVRPs is presented. The review of diversification strategies is limited to how the solutions are generated and does not cover how they evolve during any subsequent search. The studies discussed address the classical VRP and the literature available for the generation of multiple solutions in the context of split deliveries. The number of studies on SDVRPs is limited and there are a limited number of solution methods for this combinatorial problem. These solution methods include some exact approaches for small-sized problems and local search, tabu search, and hybrid methods for larger problems. As discussed below, a couple of publications present population-based solution methodologies including scatter search and memetic algorithms. Simulated annealing has been recently used to find solutions to the capacited VRP with heterogeneous fixed fleet and split services, but no computational results are reported on benchmark problem instances.

6.2.1 Solving the SDVRP

The SDVRP is a relaxation of the classical VRP. SDVRP was first introduced by Dror and Trudeau (1989) and Dror and Trudeau (1990) as a variant of the classical VRP where the demand of a customer can be supplied by one or more vehicles. In the VRP vehicles with the same capacity depart from a central depot and follow designated routes to visit and fully supply the demands of geographically scattered customers. The combined demand of the customers visited by each vehicle cannot exceed the vehicle's capacity. After supplying the customer demands, all vehicles return to

the central depot. The goal is to effectively design the vehicle routes to minimize the total traveled distance.

Mathematically, the SDVRP is defined on an undirected, fully connected graph $G = (V, E)$ where $V = \{0, 1, ..., n\}$ is the set of $n + 1$ nodes of the graph, and $E = \{(i, j) : i, j \in V,\ i < j\}$ is the set of edges connecting the nodes. Node 0 represents a depot where a fleet $M = \{1, ..., m\}$ of identical vehicles with capacity Q are stationed, while the remaining node set $N = \{1, ..., n\}$ represents the customers. A non-negative cost, usually the inter-node distance, c_{ij}, is associated with every edge $(i, j) \in E$. Each customer $i \in N$ has a demand of q_i units and is located at a point (x_i, y_j) in the two-dimensional space with respect to the depot location, (x_0, y_0). The SDVRP potentially allows reducing the operational cost of the fleet, especially when the average customer demand exceeds 10% of the vehicle capacity (as stated by Dror and Trudeau, 1989). In their worst-case analysis of the SDVRP, Archetti, Savelsbergh, and Speranza (2006) show that the reduction in delivery costs that can be obtained by allowing split deliveries is at most 50%, and this reduction bound is tight. Archetti et al. (2008) suggest that the benefits are mainly due to the reduction in the number of vehicles required to supply the customer demands. Their mathematical analysis proved that the maximum reduction in the number of vehicles is 50% and the largest reduction occurs when the mean customer demand is between 50% and 70% of the vehicle capacity and the demand variances are relatively small.

Dror and Trudeau (1989) propose a local search which uses an initial VRP solution and then uses a k-split interchange and route addition operators to introduce split deliveries if reductions in the objective function value are possible with the split delivery. Dror and Trudeau (1990) present some properties and valid inequalities for the SDVRP. Frizzell and Giffin (1992) use grid network distances in the problem and present a constructive approach to cluster the customers and a blocking mechanism to assign the demand of clustered customers to available vehicles. In an apparent first attempt to incorporate uncertainty into the SDVRP, Bouzaiene-Ayari et al. (1993) adapt the Clarke and Wright algorithm to solve the problem with stochastic demands. Dror et al. (1994) describe a branch-and-bound approach using valid inequalities and exactly solve instances with up to 20 cus-

tomers. In a second paper, Frizzell and Giffin (1995) introduce time windows into the problem and use these time windows as a criteria in the constructive approach to assign the customer demands. Mullaseril et al. (1997) adapt the local search of Dror and Trudeau (1989) to model a feed distribution problem on a cattle ranch as a SDVRP with time windows. Sierksma and Tijssen (1998) formulate a set-covering problem and a column generation method to schedule helicopters in the North Sea for crew exchange purposes.

Belenguer et al. (2000) study the SDVRP and estimate lower bounds using a cutting plane algorithm. They generate 14 random instances each having the same distribution of customers but with different ranges of customer demands. Song et al. (2002) model a distribution problem in Korea as a SDVRP to route vehicles and deliver newspapers from a central facility to different distribution centers at different times. Ho and Haugland (2004) present a tabu search to solve the SDVRP with time windows, adapt existing multiple-routes operators to the context of split deliveries, and introduce the relocate split operator which changes the customer being split among two routes. Nowak (2005) study the pickup and delivery routing problem with split loads and present a heuristic approach to solve a real problem. Liu (2005) present a two stage algorithm and a branch-and-price approach to solve some of the problems previously solved by Belenguer et al. (2000). Archetti, Hertz, and Speranza (2006) describe a tabu search approach to solve the SDVRP and solve 7 benchmark and 42 newly generated test problem instances using random customer demands. Their results have been used in recent studies for empirical comparison of algorithm performance. Lee et al. (2006) present a shortest path approach to exactly solve the SDVRP with up to 7 customers.

Belfiore et al. (2006) study the implementation of a scatter search algorithm in an actual problem to supply 519 customers in 12 states in Brazil. The problem involves heterogeneous vehicles, time windows, accessibility constraints, and split deliveries. Yu et al. (2006) propose an approximate linear model with subtour elimination constraints, lagrangian relaxation, and a heuristic method to solve the inventory routing problem with split deliveries. Jin et al. (2007) propose a cutting plane algorithm to optimally solve the SDVRP dividing the original problem into clus-

tering and traveling salesman subproblems. The distances of the traveling salesman subproblems are repeatedly added as bounds to the clustering subproblems to find better solutions. Ambrosino and Sciomachen (2007) model an actual problem in Italy to plan the country-wide distribution of fresh/dry and frozen food. Mota et al. (2007) present a scatter search procedure that uses the minimum fleet size and produces good results on instances previously solved in the literature, particularly on problems with average customer demands less than half of the vehicle capacity. Wilck and Cavalier (2007) consider an objective function involving total distance traveled and vehicle loads. They use a constructive approach to find good solutions to small sized problems. Chen et al. (2007) developed a hybrid approach combining a mixed integer program and a record-to-record travel algorithm that produces high-quality solutions compared to the existing literature. Tavakkoli-Moghaddam et al. (2007) present a mixed-integer linear and a simulated annealing method for solving the SDVRP with heterogeneous vehicles. Boudia et al. (2007) implemented a memetic algorithm with population management and produced high quality solutions on the problems of Archetti, Hertz, and Speranza (2006) and Belenguer et al. (2000). Jin et al. (2008) present a column generation approach to estimate bounds for the SDVRP with large customer demands. The algorithm improves some of the bounds found by Belenguer et al. (2000).

6.2.2 The CA, ICA and VND Solution Approaches

Constructive Approach (CA)

The parallel constructive approach (CA) of Aleman et al. (2007) uses an ordered list L of customers based on the distances from the depot and then inserts them into the solution under construction to initiate new routes or modify existing ones. The farthest customer from the depot is assigned the first position in L whereas the closest customer to the depot is assigned the last position. Once L is designed, customers are sequentially inserted into the routes until all customer demands are satisfied. A customer demand can be split when that demand exceeds the capacity left on the selected vehicle. In this case, any remaining demand is assigned to either an empty vehicle or the best vehicle

available. The characteristic that differentiates the CA from existing constructive approaches is that it uses a novel route angle control mechanism (RAC) to avoid the design of spatially spread routes. The intuition of RAC is to avoid overlapping routes among the vehicles. The RAC mechanism utilizes the angle of a route, defined by the customers assigned to the route, to penalize insertions into far routes and favor closer routes. The polar angle of customer i relative to the depot, denoted θ_i, is defined as:

$$\theta_i = \arctan \frac{y_i - y_0}{x_i - x_0} \tag{6.1}$$

where (x_i, y_i) represents the location of customer i and customer 0 represents the depot. The angle of route R is then defined as $\theta_R = \max\{\theta_i - \theta_j; \forall i, j \in N \cap R\}$. The CA can obtain solutions of good quality at a very low computational effort using the minimum number of vehicles $m = \lceil \sum_{i \in N} q_i / Q \rceil$, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

Iterative Constructive Approach (ICA)

A limitation of the CA is that the customers closest to the depot inserted later in routes due to their position in L, tend to deteriorate the quality of the final solution as there are a limited number of route alternatives available at the moment of their insertion. The Aleman et al. (2007) ICA is an iterative approach that applies adaptive memory and dynamic modification of the list L. The resulting ICA executes the CA iteratively but modifies L each iteration. Customers that cause the widest spread of routes are assigned an earlier position in L to ensure their insertion into more adequate routes. This is done as follows. First, the customer i^* producing the widest route is identified. Second, the closest route r^* to i^* in the current solution is selected. Third, the customer i_a spending the last resources of r^* needed to fully supply customer i^* is determined. Finally, customer i^* is relocated in L so that it will be inserted into the solution right before i_a. This guarantees a full service and a less expensive delivery for i^*.

Variable Neighborhood Descent (VND)

The VND presented in Aleman et al. (2007) looks to improve the SDVRP solutions found with ICA. The VND uses three neighborhoods. The first and second neighborhoods are based on the standard customer *shift* and customer *swap* to move customers among routes. In the case of the customer *shift*, if the customer to be shifted is currently using split delivery, it is simply removed from the origin route and its quantity is increased in the destination route (the split is eliminated). In the case of the customer *swap*, if a customer to be swapped is currently using split delivery, it is also removed from the origin route and its quantity is increased in the destination route. The other customer is then moved from one route to the other. The third neighborhood is a new operator that moves a customer among routes when the destination route does not have enough capacity to cover the demand of the moved customer. In order to make the move feasible, the load of the destination route is released by reducing its delivery to any other customer in the route with enough demand (i.e., larger than the demand of the moved customer). The unserved demand is then served by the original route. At the end, the customer shift is feasible and a new split is introduced by sharing a customer. A detailed illustration of these operators is found in Aleman et al. (2007) along with an empirical evaluation of CA, ICA, and VND performance on all available problem sets.

6.2.3 Solution Diversification Techniques

Despite using a variety of local search and local improvement methods, search heuristics can still have problems finding really good solutions to hard problems. A diversification scheme aims to move the search process into new, hopefully unvisited, regions of the solution space. Once in those new regions, the search process resumes.

One of the first studies for generating diversified solutions for routing problems is by Rochat and Taillard (1995) who partition large problems into independent subproblems, each defined by sectors and regions centered at the depot, and then optimize each subproblem independently. Their diversified solutions are generated with a local search by considering various initial partitions of the

problem. Tarantilis and Kiranoudis (2002) present a population-based heuristic called *BoneRoute* that extracts sequences of nodes, or bones, from the pool of solutions to compose partial solutions. They complete these partial solutions with a constructive approach. The diversified solutions forming the initial pool are generated with the savings algorithm of Paessens (1988). Berger and Barkaoui (2003) propose a hybrid genetic algorithm to evolve two populations using selection, recombination, mutation, and migration operators. The generation of initial solutions is based on a random construction of feasible solutions. A solution is rapidly constructed through a sequential insertion heuristic which inserts customers into randomly chosen positions within routes. Customer insertion order is randomly modified to ensure unbiased solution generation.

Reimann et al. (2004) present the D-Ants algorithm that uses the *SavingsAnts* system of Doerner et al. (2002) as the mechanism to generate a pool of solutions. In the *SavingsAnts* approach, solutions are generated using attractiveness values balancing the savings values of the classical Clarke and Wright algorithm and the pheromone information from previous iterations. The D-Ants approach is effective solving small and large scale benchmark instances as well as real world sized problems.

In his evolutionary algorithm, Prins (2004) proposes a population of solutions initialized using three heuristic methods (Clarke and Wright, 1964; Mole and Jameson, 1976; Gillett and Miller, 1974) and utilizing random permutations of customers to produce a complete population. Chromosomes represent solutions in the form of giant tours formed with the ordered sequence of routes. In genetic algorithms for solving routing problems, each bit in the chromosome usually represents a customer and multiple copies of the depot are used to separate the routes. Instead of using copies of the depot, Prins utilizes an optimal splitting procedure to determine the best way to separate the routes in the chromosome. The routes and the fitness value of each solution are determined by solving a min-cost path problem on an auxiliary graph.

Tarantilis (2005) employes the method of Glover (1998) to generate a collection of diversified solutions and initiate an adaptive memory solution procedure. This methodology systemati-

cally generates different permutations, or sequences, of customers and then successively assigns customers to routes to produce a VRP solution using a generalized assignment process. These diversified solutions are then improved with a tabu search and combined using elite parts of the routes to produce new solutions that update the adaptive memory components.

Only a few studies have used an initial set of solutions as a means to solve the SDVRP. Belfiore et al. (2006) study the implementation of a scatter search for a routing problem with split deliveries, heterogeneous vehicles, time windows, and accessibility constraints applied to a retail market in Brazil. Their's is the first attempt to solve split delivery problems with those particular side constraints. The initial solutions for the scatter search are generated using the constructive heuristic of Dullaert et al. (2002) for the problem with heterogeneous vehicles and time windows. Random elements are used to diversify the solutions.

Mota et al. (2007) propose a scatter search that generates a population of feasible solutions based on a giant TSP tour visiting all customers. The construction of a solution commences by selecting a starting customer in the giant tour and sequentially cutting that tour into individual routes where the demand of the first and last customer of each route is split when that demand does not fit the first vehicle serving it. The selection of nonconsecutive starting customers helps obtain different solutions. Mota et al. (2007) also adapt the algorithm of Clarke and Wright for split deliveries to find another fraction of the population of solutions. This adaptation of the Clarke and Wright algorithm does not guarantee feasibility but produces diversified solutions by statistically prohibiting half of the savings used in the construction of previous solutions. Boudia et al. (2007) solve the SDVRP using a memetic algorithm with population management and create the initial population both heuristically and randomly; two solutions are constructed heuristically by the algorithms of Clarke and Wright (1964) and Gillett and Miller (1974) whereas the rest of the population is generated with a random permutation of the customers. This method to generate the initial population is similar to that of Prins (2004), the only difference is the number of solutions created heuristically. The memetic algorithm produces new best SDVRP solutions for benchmark problems of Christofides and Eilon (1969) and Christofides et al. (1979) involving 75 and 120 customers with original de-

mands and improves the state of the art algorithms (Archetti, Hertz, and Speranza, 2006; Chen, Golden, and Wasil, 2007) in some of the other tested problems.

6.3 An Aggressive Diversification-Based Search Algorithm

6.3.1 A New Diversification Method

CA quickly finds SDVRP solutions whose objective function value is usually within 10% of the best existing solutions. However, those solutions display a common pattern; customers closer to the depot considerably deteriorate the quality of the overall solution. To mitigate this problem, the ICA changes the sequence of customers in L to improve upon CA solutions. The VND further improves the search process using a set of local search moves. Although each method is effective and can avoid getting trapped in a local optimum region, these procedures do not guarantee a thorough exploration of the solution space. Diversification schemes are explicitly designed to improve solution space exploration.

A diversification scheme is proposed and tested based on a geographical division of the customers using rings, or spatial bands, centered at the depot. The geographic space of the problem is marked with rings of varying circumferences used to group the customers. The original problem is partially solved with the customers located inside certain rings. These selected customers are assigned to routes using the CA. Subsequently, the remaining customers, belonging to the other rings, are inserted into the partial solution to yield a complete solution to the original problem.

A ring is defined by an inner and outer radius, r_{in} and r_{out}, measured outward from the depot. The customers inside a ring are those whose distance from the depot is greater than r_{in} and less than or equal to r_{out}, $r_{in} < c_{0j} \leq r_{out}$. Although any number of rings can be used, the proposed scheme uses three non-overlapping rings encompassing all customers. Figure 6.1 illustrates the geographical division used in the scheme applied to a problem involving 50 customers. In the figure, there are three rings of radius r, R, and R_{max}, respectively. These define the rings A, B,

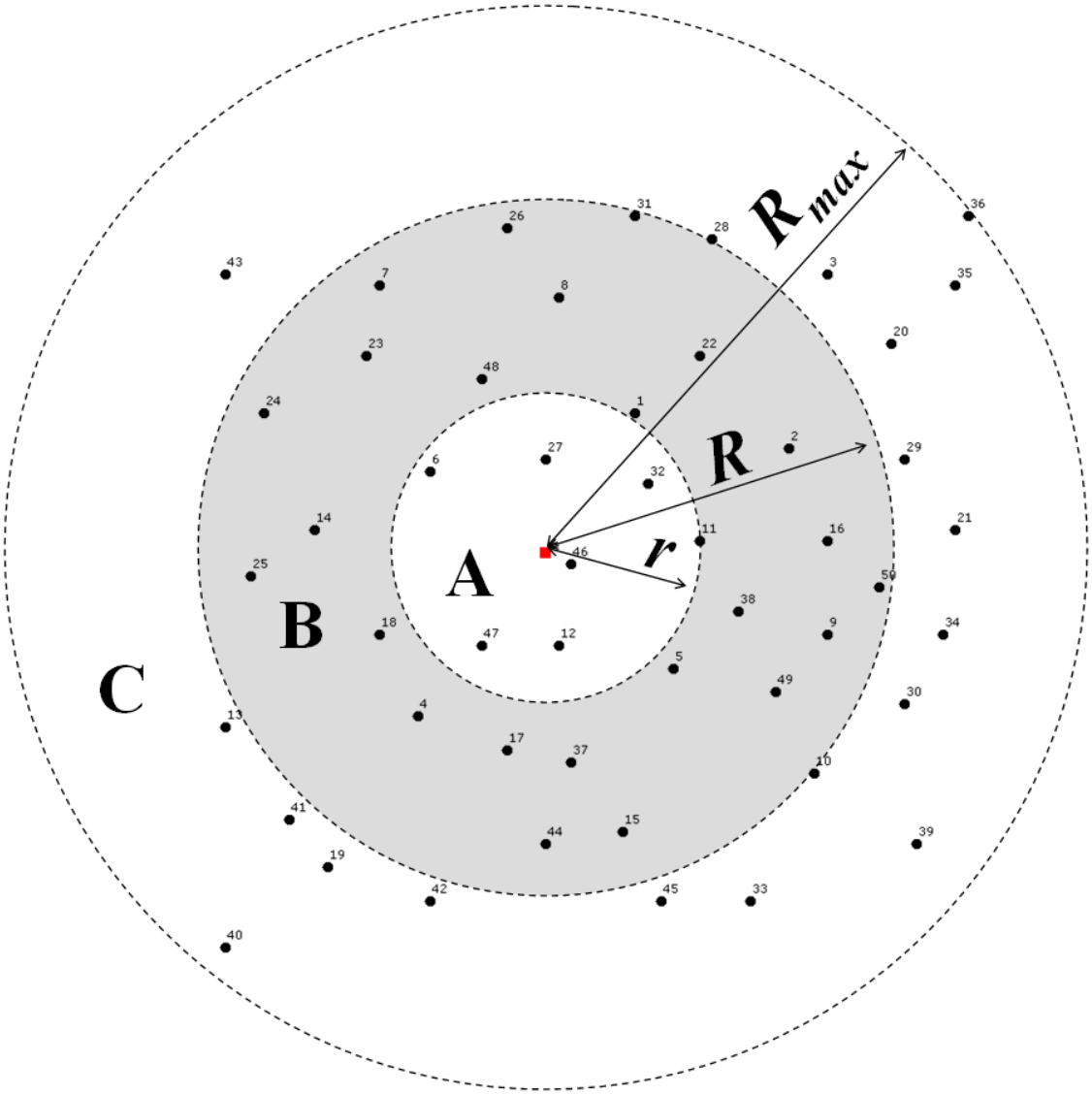

Figure 6.1: Geographical depiction of ring-based partition of 50 customers problem from Archetti et al. (2006).

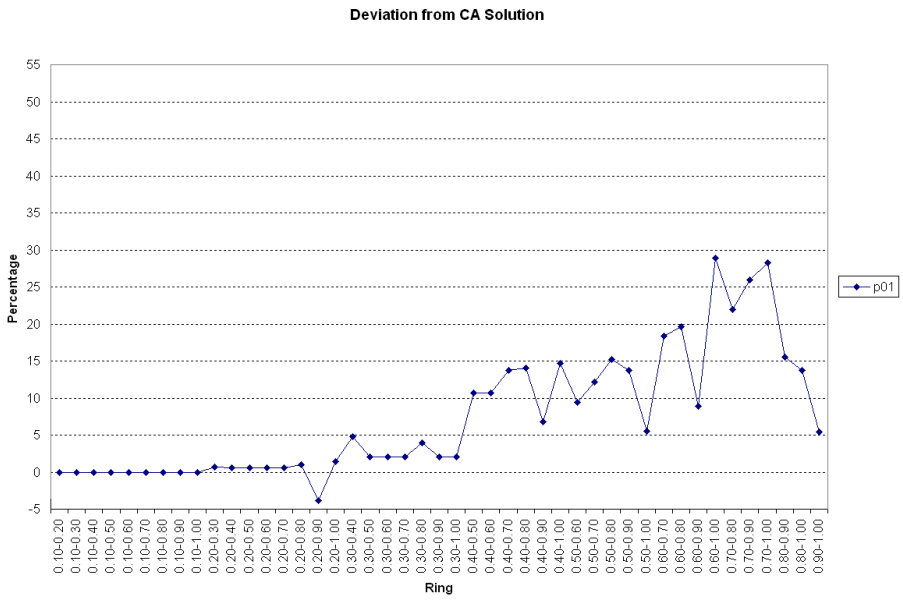

Figure 6.2: Diversified solutions for problem 1 of Archetti et al. (2006) involving 50 customers with the original demands.

and C. For example, ring A is defined by $r_{in} = 0$ and $r_{out} = r$, ring B is defined by $r_{in} = r$ and $r_{out} = R$, and ring C is defined by $r_{in} = R$ and $r_{out} = R_{max}$

The original problem includes the n customers in the set $N = \{1, ..., n\}$. Rings A, B, and C partition N into three subsets N_A, N_B, and N_C such that: 1) $N_A \cap N_B = N_A \cap N_C = N_B \cap N_C = \emptyset$ and 2) $N = N_A \cup N_B \cup N_C$. A partial solution is then obtained using the CA with the customers in N_A and N_C while the customers in N_B are temporarily excluded. The complete solution is then obtained when the customers in N_B are inserted into the partial solution also using the CA. Varying values of r and R varies the size of B, and subsequently N_B, the exclusion set in the method, yielding a variety of solutions. This approach yields a much more aggressive diversification strategy than the one obtained just using local improvement methods such as found with ICA.

The range of diversified solutions based on varied sizes of B is next examined. Figure 6.2 shows the solutions to problem 1 and Figure 6.3 shows solutions to problem 6 from Archetti, Hertz, and Speranza (2006) obtained with the CA and the diversification scheme using different settings for ring B. In each figure, solution values are shown as a percentage deviation from the objective

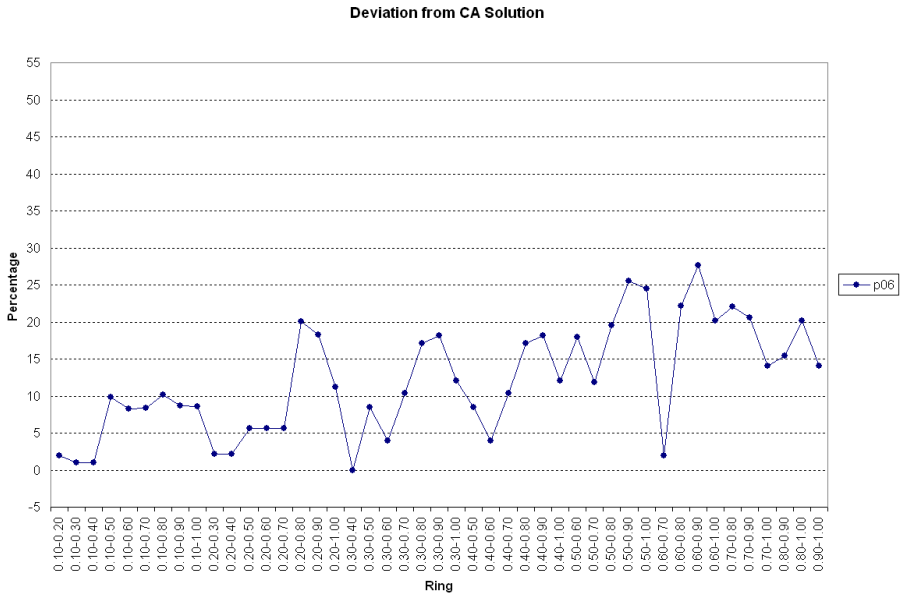

Figure 6.3: Diversified solutions for problem 6 of Archetti et al. (2006) involving 120 customers with the original demands.

function value obtained using the basic CA. A negative deviation indicates that the diversification directly provides a better solution whereas a positive deviation indicates the diversified solution is not as good. The width and location of ring B varies as a function of R_{max}. The inner radius r varies in the range $[0.1R_{max}, 0.9R_{max}]$ in steps of $0.1R_{max}$ whereas the outer radius R varies in the range $[0.2R_{max}, 1.0R_{max}]$ also in steps of $0.1R_{max}$. The ring settings r and R are shown in each figure in the form $r - R$ on the horizontal axis. Note that different settings for the inner and outer radius can be used to generate a larger number of diversified solutions. This diversification scheme can be applied to any routing problem. Also, note that since this scheme generates diverse solutions, the initial solution quality is not a primary concern; local improvement is ultimately applied to the diverse solutions in the computational procedure.

Figures 6.4 to 6.7 illustrate four solutions taken from the generated set shown in Figure 6.2. Figure 6.4 illustrates the solution found with the basic CA while Figure 6.5 illustrates the solution in the diversified set which is the most different from the basic CA solution shown in Figure 6.4. In this case, the number of edges appearing in one solution but not in the other are counted to measure

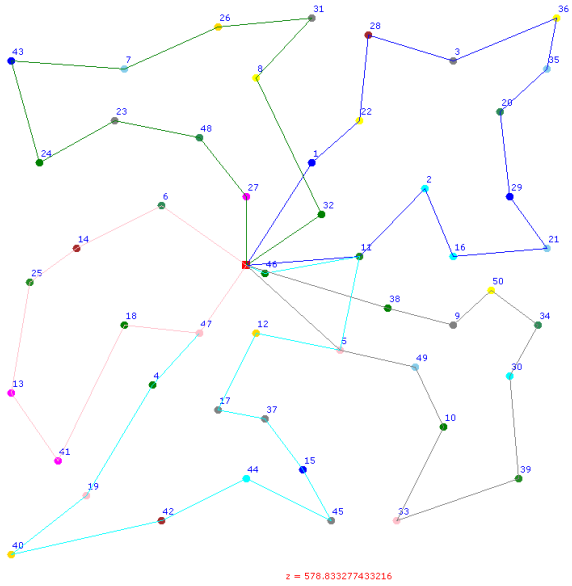

Figure 6.4: Illustration of the basic CA solution for problem 1 of Archetti et al. (2006) involving 50 customers with the original demands ($z = 578.83$).

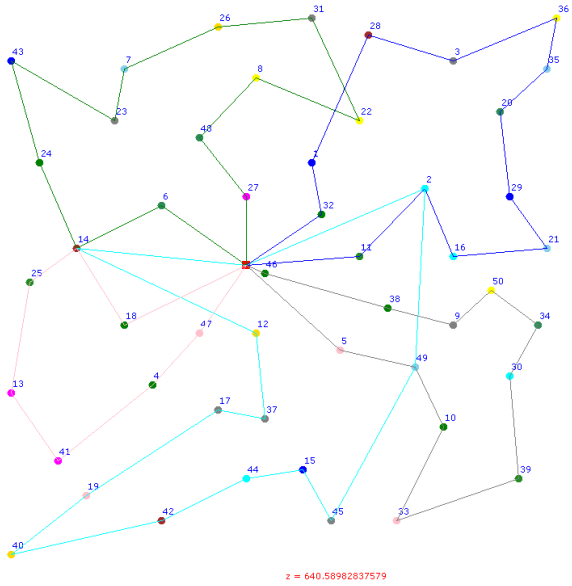

Figure 6.5: Illustration of a solution taken from the set of solutions for problem 1 of Archetti et al. (2006) that most differs from the basic CA solution ($z = 640.58$ and ring settings $0.40 - 0.50$).

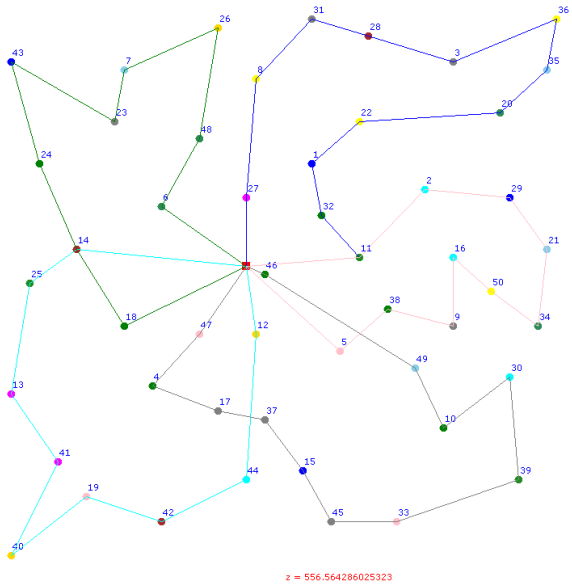

Figure 6.6: Illustration of a solution taken from the set of solutions for problem 1 of Archetti et al. (2006) with the lowest objective function value ($z = 556.56$ and ring settings $0.20 - 0.90$).

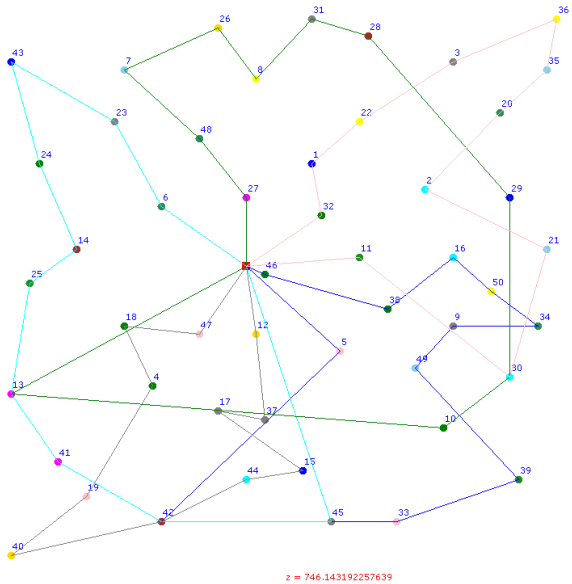

Figure 6.7: Illustration of a solution taken from the set of solutions for problem 1 of Archetti et al. (2006) with the highest objective function value ($z = 746.14$ and ring settings $0.60 - 1.00$).

the difference between two solutions. Figure 6.6 illustrates the solution in the diversified set with the lowest objective function value z, and finally Figure 6.7 illustrates the solution in the diversified set with the highest objective function value z. These figures reinforce with confidence that the ring-based diversification process does in fact generate a variety of solutions.

6.3.2 The ICA+VND With Diversification (iVNDiv) Solution Approach

The proposed solution approach couples the algorithms presented in Aleman et al. (2007) with the new diversification methodology. The idea is to solve the problem using the iterative constructive approach and variable neighborhood descent of Aleman et al. (2007) and then restart the search from different points in the solution space when the diversification phase commences. The result is a multi-start algorithm for the SDVRP.

The details of iVNDiv are given in Algorithms 1 and 2. The set of solutions is generated using Algorithm 1. This algorithm utilizes various ring settings to partition the original problem in a variety of ways and produce different solutions. The algorithm constructs solutions with the constructive approach (CA) of Aleman et al. (2007). Once a complete solution is generated, it is added to the full set of solutions. Before adding solutions, their objective function values are verified to guarantee unique elements in the set of diverse solutions.

With the full set of diversified solutions, the ICA and VND of Aleman et al. (2007) are used to improve the solutions in the diversified set. The number of solutions from the set used as starting points in the solution space varies. The more solutions used, the higher the computational cost. A maximum of 5 starting solutions are used in the proposed iVNDiv to balance the quality of the solution and their running times. These 5 solutions are the best solutions in the set of diversified solutions. The iVNDiv is presented in Algorithm 2.

Algorithm 1: - Generation of set S of solutions

Set $S = \emptyset$.

Let $N = \{1, ..., n\}$ be the set of customers in the original problem.

Let $N_A \subset N$ be the subset of customers in accordance with $c_{0i} \leq r$ for $i \in N$.

Let $N_B \subset N$ be the subset of customers in accordance with $r < c_{0i} \leq R$.

Let $N_C \subset N$ be the subset of customers in accordance with $R < c_{0i} \leq R_{max}$ for $i \in N$.

$width = 0.10$

for $r = 0$; $r < 1.00$; $r = r + width$ **do**

  for $R = r + width$; $R \leq 1.00$; $R = R + width$ **do**

    Design a list L_1 with all customers $i \in N_A \cup N_C$.

    Order the customers in L_1 by nonincreasing distance from the depot.

    With L_1, use the CA to find a partial solution s_1 to the problem.

    Define a list L_2 with all customers $i \in N_B$.

    Order the customers in L_2 by nonincreasing distance from the depot.

    Using the CA, insert the customers in L_2 sequentially into s_1 to produce a complete solution s_2.

    Set $S = S \cup \{s_2\}$.

  end for

end for

return S

Algorithm 2: - ICA+VND With Diversification (iVNDiv)

Execute Algorithm 1 to generate set S of solutions.

Set x_b as the best solution with solution value $f(x_b) = \infty$

$jumpCounter = 0$.

$maxJumps = 5$.

while $jumpCounter < maxJumps$ **do**

  Select the solution x from set S with the lowest objective function value and remove it from S.

  Execute the ICA+VND approach of Aleman et al. (2007) to improve x and obtain x'.

  if $f(x') < f(x_b)$ **then**

    $x_b = x'$

  end if

  $jumpCounter = jumpCounter + 1$.

end while

return x_b

6.4 Computational Results

The iVNDiv was implemented in C# and experiments were carried out using a Pentium 4, 2.8 GHz, 512MB of RAM. The iVNDiv algorithm was tested on problem sets available in the literature including Archetti, Hertz, and Speranza (2006), Belenguer et al. (2000), Chen et al. (2007), and Jin et al. (2008). These sets have been solved with existing approaches which are compared to iVNDiv in this empirical analysis. The tested instances are identified using the form p-aaa-nnn. The first field, p, is an alphabetical character to identify the publication where the problem is identified. The second field, aaa, is a string of variable length corresponding to the name of the instance adopted in the publication, whereas the third field is a three-digit integer denoting the number of customers excluding the depot. The first field, p, takes the following values:

- a Archetti, Hertz, and Speranza (2006)
- b Belenguer et al. (2000)
- c Chen et al. (2007)
- j Jin et al. (2007)

The instances used by Archetti, Hertz, and Speranza (2006) are the same problems 1-5 and 11-12 given in Christofides and Eilon (1969) and Christofides et al. (1979) involving 50 to 199 customers in addition to the depot. In problems 1-5, customers are randomly distributed in the plane, while they are clustered in problems 11-12. From those 7 problems, Archetti, Hertz, and Speranza (2006) generated 42 more by randomly modifying the customer demands at different intervals. These random problems are unavailable. However, Mota et al. (2007) used the same algorithm of Archetti, Hertz, and Speranza (2006) to generate their problems. The problems used in this study were obtained from Boudia et al. (2007). In this analysis, the problems with random customer demands have the exact same demand values as in Mota et al. (2007) and Boudia et al. (2007). Belenguer et al. (2000) used a total of 25 problems: 11 TSPLIB problems involving

21 to 100 customers and 14 randomly generated problems from the TSPLIB (eil51, eil76, and eil101). The same vehicle capacity $Q = 160$ is used in each problem and the customer demands are randomly generated within 6 intervals expressed as a funtion of Q, as in Dror and Trudeau (1989) and Archetti, Hertz, and Speranza (2006). Chen et al. (2007) recently generated a new set of 21 problems involving 8 to 288 customers. Each problem has a geometric symmetry with customers located in concentric circles around the depot. Jin et al. (2007) used a TSPLIB instance with 21 customers and generated 4 problems involving 18 to 22 customers.

Computational results for the Archetti et al. instances are presented in Tables 6.1 to 6.3. The solution values from the existing algorithms are reproduced from the corresponding references. The existing algorithms are the variable neighborhood descent (ICA+VND) of Aleman et al. (2007), the scatter search (SS) of Mota et al. (2007), the memetic algorithm with population management (MA|PM) of Boudia et al. (2007), the three tabu searches (Splitabu, Splitabu-DT, and Fast-Splitabu) of Archetti, Hertz, and Speranza (2006), and the hybrid algorithm (EMIP+VRTR) of Chen et al. (2007). In Table 6.1, columns with header m contain the number of vehicles in the final iVNDiv, ICA+VND, SS, and MA|PM solutions, which is always the minimum possible, whereas columns with header m' contain the number of vehicles in the final solutions found with the tabu searches of Archetti, Hertz, and Speranza (2006). Results in Table 6.1 show that the ICA+NVD algorithm is clearly dominated by its counterpart with the proposed diversification scheme, i.e. iVNDiv. Compared to SS, iVNDiv provides better solutions especially in problems with large customer demands in the ranges [0.10-0.90], [0.30-0.70], and [0.70-0.90], where the largest cost reduction can occur, as shown in Dror and Trudeau (1989) and Archetti et al. (2008). Although iVNDiv finds solutions of similar quality and performs better in one problem, the MA|PM clearly dominates iVNDiv in this problem set.

The comparison with the tabu searches and EMIP+VRTR on the problems with random customer demands is not straightforward. First, there is a potential discrepancy regarding the actual customer demand values used by Archetti, Hertz, and Speranza (2006). Second, the values reproduced from Chen et al. (2007) correspond to the median values from 30 solution instances for

each random problem. Across the board, there is no apparent dominance of iVNDiv over the tabu searches, but the number of vehicles in the final iVNDiv solutions is generally lower than in the tabu search solutions. In some cases, the tabu searches use up to 14 more vehicles than iVNDiv (see for example problem a-05-199 with demands in [0.10-0.90]), which can lead to solutions with lower objective function values but possibly higher operational costs in actual problems. Chen et al. (2007) do not provide the number of vehicles used in their final EMIP+VRTR solutions, but the fleet size is apparently a decision variable. iVNDiv is able to improve EMIP+VRTR in only 1 problem.

Table 6.2 shows the best known SDVRP solutions available in the literature for the instances of Archetti et al. and a comparison with the iVNDiv solutions. Note that EMIP+VRTR values are median values from 30 instances. The best known solutions are presented with their solution values z, the number m of vehicles, and the publication where they are reproduced from. Problem a-01-050 with original demands is optimally solved by Belenguer et al. (2000) and its solution value was calculated using integer inter-node distances. The tables also provide the percentage improvement of the iVNDiv solutions over the best ones. Out of the 49 problems, 26 best solutions have been found with the EMIP+VRTR hybrid approach of Chen et al. (2007), 10 with the MA|PM memetic algorithm, 10 with the tabu searches, and 2 with the scatter search. In this table, it is also important to recall the effect of the number of vehicles in the final solutions and that iVNDiv, SS, and MA|PM utilize the smallest fleet possible. The iVNDiv improves the best known solution to problem a-02-075 in the range [0.01-0.10] and generally uses less vehicles than other approaches.

The running times in seconds are provided in Table 6.3. The characteristics of the machines where the different approaches were run are listed at the bottom of the table. The impact of the diversification scheme on the running time is noticed by comparing the results for iVNDiv and ICA+VND in the table. However, the diversification procedure does not deteriorate the running time considerably as the average customer demands get larger, which is not the case for the tabu searches and EMIP+VRTR. With the tabu searches, one reason for the increase in the computational effort may be the neighborhood structure used in the search. For each customer, Archetti, Hertz, and Speranza (2006) evaluate removals from the visiting vehicles and/or insertions into other vehicles

Table 6.1: Computational results of iVNDiv on instances of Archetti et al. (2006).

			iVNDiv[a]	ICA+VND[a]		SS[a]		MA|PM[a]			Splitabu[b]		Splitabu-DT[b]		Fast-Splitabu[b]		EMIP+VRTR[c]	
Problem	Demand	m	z	z	IMP	z	IMP	z	IMP	m'	z	IMP	z	IMP	z	IMP	z	IMP
a-01-050		5	524.61	540.82	**-3.09**	531.02	**-1.22**	524.61	0.00	5	530.06	**-1.04**	533.55	**-1.70**	533.55	**-1.70**	524.61	0.00
a-02-075		10	851.24	880.28	**-3.41**	839.75	1.35	823.89	3.21	10	851.67	**-0.05**	849.54	0.20	849.54	0.20	840.18	1.30
a-03-100		8	852.74	854.13	**-0.16**	835.82	1.98	829.44	2.73	8	846.18	0.77	835.62	2.01	835.74	1.99	-	-
a-04-150		12	1074.11	1088.91	**-1.38**	1056.92	1.60	1042.37	2.95	12	1062.20	1.11	1069.84	0.40	1088.31	**-1.32**	1041.99	2.99
a-05-199		16	1368.67	1390.55	**-1.60**	1340.44	2.06	1311.59	4.17	16	1367.82	0.06	1342.85	1.89	1346.39	1.63	1307.40	4.48
a-06-120		7	1201.83	1223.28	**-1.79**	1042.97	13.22	1041.20	13.37	7	1084.73	9.74	1056.01	12.13	1056.01	12.13	1043.18	13.20
a-07-100		10	824.78	824.82	0.00	820.92	0.47	819.56	0.63		822.60	0.26	825.32	**-0.07**	825.32	**-0.07**	819.56	0.63
a-01-050	[0.01-0.10]	3	471.92	473.22	**-0.28**	460.79	2.36	460.79	2.36		463.85	1.71	463.76	1.73	463.76	1.73	457.21	3.12
a-02-075	[0.01-0.10]	4	597.46	617.65	**-3.38**	602.67	**-0.87**	600.06	**-0.44**		607.66	**-1.71**	605.24	**-1.30**	605.24	**-1.30**	598.25	**-0.13**
a-03-100	[0.01-0.10]	5	745.35	789.16	**-5.88**	729.67	2.10	726.81	2.49		772.79	**-3.68**	752.20	**-0.92**	752.20	**-0.92**	-	-
a-04-150	[0.01-0.10]	8	891.98	893.49	**-0.17**	883.05	1.00	875.61	1.83		894.98	**-0.34**	890.95	0.11	893.66	**-0.19**	875.16	1.89
a-05-199	[0.01-0.10]	10	1073.55	1079.04	**-0.51**	1039.51	3.17	1018.71	5.11		1073.60	0.00	1056.27	1.61	1068.70	0.45	1040.20	3.11
a-06-120	[0.01-0.10]	6	1087.80	1101.14	**-1.23**	979.57	9.95	976.57	10.23		1095.75	**-0.73**	1084.70	0.29	1084.70	0.29	985.17	9.43
a-07-100	[0.01-0.10]	5	673.54	673.54	0.00	633.80	5.90	649.73	3.53		662.80	1.59	648.74	3.68	653.70	2.95	651.44	3.28

Continued on next page

Table 6.1: Computational results of iVNDiv on instances of Archetti et al. (2006) (*Continued*).

			iVNDiv[(a)]	ICA+VND[(a)]		SS[(a)]		MA|PM[(a)]			Splitabu[(b)]		Splitabu-DT[(b)]		Fast-Splitabu[(b)]		EMIP+VRTR[(c)]	
Problem	Demand	m	z	z	IMP	z	IMP	z	IMP	m'	z	IMP	z	IMP	z	IMP	z	IMP
a-01-050	[0.10-0.30]	**10**	766.19	777.75	**-1.51**	769.60	**-0.45**	751.41	1.93	11	764.40	0.23	761.40	0.62	761.40	0.62	723.57	5.56
a-02-075	[0.10-0.30]	**15**	1099.47	1099.47	0.00	1074.01	2.32	1074.46	2.27	16	1099.03	0.04	1095.32	0.38	1095.32	0.38	1081.10	1.67
a-03-100	[0.10-0.30]	**20**	1425.90	1452.52	**-1.87**	1416.48	0.66	1392.85	2.32	22	1428.87	**-0.21**	1424.81	0.08	1426.18	**-0.02**	-	-
a-04-150	[0.10-0.30]	**29**	1978.01	1978.01	0.00	1974.70	0.17	1878.71	5.02	32	1940.67	1.89	1918.25	3.02	1924.67	2.70	1844.96	6.73
a-05-199	[0.10-0.30]	**38**	2464.65	2502.54	**-1.54**	2435.08	1.20	2340.14	5.05	41	2419.98	1.81	2384.15	3.27	2393.82	2.87	2258.66	8.36
a-06-120	[0.10-0.30]	**23**	2806.92	2806.92	0.00	2783.10	0.85	2720.38	3.08	26	2900.99	**-3.35**	2918.71	**-3.98**	2921.30	**-4.08**	2568.90	8.48
a-07-100	[0.10-0.30]	20	1428.27	1428.27	0.00	1423.49	0.34	1417.28	0.77		1470.96	**-2.99**	1462.01	**-2.36**	1467.32	**-2.73**	1414.33	0.98
a-01-050	[0.10-0.50]	**15**	1039.89	1045.93	**-0.58**	1025.91	1.34	988.31	4.96	16	1007.68	3.10	1008.67	3.00	1008.67	3.00	943.86	9.23
a-02-075	[0.10-0.50]	**22**	1478.67	1503.02	**-1.65**	1484.62	**-0.40**	1413.80	4.39	24	1450.11	1.93	1443.62	2.37	1449.74	1.96	1393.53	5.76
a-03-100	[0.10-0.50]	**29**	1956.13	1957.55	**-0.07**	1926.15	1.53	1845.30	5.67	33	1887.83	3.49	1894.72	3.14	1897.12	3.02	-	-
a-04-150	[0.10-0.50]	**43**	2671.62	2685.33	**-0.51**	2649.97	0.81	2561.65	4.12	49	2634.09	1.40	2632.71	1.46	2650.61	0.79	2532.93	5.19
a-05-199	[0.10-0.50]	**56**	3411.38	3450.84	**-1.16**	3310.71	2.95	3191.25	6.45	63	3298.19	3.32	3284.47	3.72	3314.00	2.85	3202.57	6.12
a-06-120	[0.10-0.50]	**34**	4026.53	4085.36	**-1.46**	3996.29	0.75	3934.39	2.29	40	4166.78	**-3.48**	4206.12	**-4.46**	4261.66	**-5.84**	3687.06	8.43
a-07-100	[0.10-0.50]	29	2007.11	2046.15	**-1.95**	2022.30	**-0.76**	1994.59	0.62		2030.04	**-1.14**	2029.99	**-1.14**	2053.86	**-2.33**	1973.34	1.68

Continued on next page

Table 6.1: Computational results of iVNDiv on instances of Archetti et al. (2006) (*Continued*).

			iVNDiv[a]	ICA+VND[a]		SS[a]		MA|PM[a]			Splitabu[b]		Splitabu-DT[b]		Fast-Splitabu[b]		EMIP+VRTR[c]	
Problem	Demand	m	z	z	IMP	z	IMP	z	IMP	m'	z	IMP	z	IMP	z	IMP	z	IMP
a-01-050	[0.10-0.90]	**25**	1522.43	1547.32	**-1.64**	1580.77	**-3.83**	1467.06	3.64	26	1493.92	1.87	1469.92	3.45	1470.39	3.42	1408.34	7.49
a-02-075	[0.10-0.90]	**37**	2200.51	2212.93	**-0.56**	2233.08	**-1.48**	2102.58	4.45	41	2121.28	3.60	2124.43	3.46	2142.32	2.64	2056.54	6.54
a-03-100	[0.10-0.90]	**48**	2865.86	2925.13	**-2.07**	2932.34	**-2.32**	2780.95	2.96	56	2826.61	1.37	2794.08	2.50	2837.78	0.98	-	-
a-04-150	[0.10-0.90]	**73**	4165.18	4192.50	**-0.66**	4185.68	**-0.49**	4045.87	2.86	84	4006.28	3.82	3909.72	6.13	3987.62	4.26	3945.38	5.28
a-05-199	[0.10-0.90]	**93**	5184.57	5192.06	**-0.14**	5085.64	1.91	4941.22	4.69	107	5039.65	2.80	4853.83	6.38	5044.72	2.70	5094.61	1.74
a-06-120	[0.10-0.90]	**56**	6364.87	6483.06	**-1.86**	6361.46	0.05	6318.37	0.73	67	6560.33	**-3.07**	6583.97	**-3.44**	6730.80	**-5.75**	6079.14	4.49
a-07-100	[0.10-0.90]	48	3156.31	3178.28	**-0.70**	3187.44	**-0.99**	3113.72	1.35		3158.09	**-0.06**	3101.53	1.74	3109.04	1.50	3162.22	**-0.19**
a-01-050	[0.30-0.70]	**25**	1540.39	1557.52	**-1.11**	1568.04	**-1.80**	1477.01	4.11	26	1509.31	2.02	1496.90	2.82	1496.95	2.82	1408.68	8.55
a-02-075	[0.30-0.70]	**37**	2238.98	2241.59	**-0.12**	2228.90	0.45	2132.16	4.77	39	2175.99	2.81	2160.51	3.51	2164.22	3.34	2112.61	5.64
a-03-100	[0.30-0.70]	**49**	2941.64	2945.19	**-0.12**	2986.33	**-1.52**	2858.87	2.81	53	2914.79	0.91	2870.50	2.42	2921.95	0.67	-	-
a-04-150	[0.30-0.70]	**73**	4165.18	4192.50	**-0.66**	4185.68	**-0.49**	4045.87	2.86	80	4146.75	0.44	4039.70	3.01	4139.06	0.63	4011.74	3.68
a-05-199	[0.30-0.70]	**96**	5363.65	5366.06	**-0.04**	5265.01	1.84	5155.36	3.88	103	5368.92	**-0.10**	5102.84	4.86	5312.67	0.95	5088.08	5.14
a-06-120	[0.30-0.70]	**58**	6545.50	6591.40	**-0.70**	6481.09	0.98	6424.71	1.85	65	6866.34	**-4.90**	6639.55	**-1.44**	6829.05	**-4.33**	6123.96	6.44
a-07-100	[0.30-0.70]	49	3225.63	3318.08	**-2.87**	3248.76	**-0.72**	3155.69	2.17		3222.03	0.11	3038.02	5.82	3187.92	1.17	3134.56	2.82
a-01-050	[0.70-0.90]	**40**	2215.34	2215.34	0.00	2312.48	**-4.38**	2154.35	2.75	42	2176.39	1.76	2165.21	2.26	2167.93	2.14	2056.01	7.19
a-02-075	[0.70-0.90]	**60**	3304.24	3341.26	**-1.12**	3387.86	**-2.53**	3200.35	3.14	61	3229.46	2.26	3180.64	3.74	3241.77	1.89	3067.19	7.17
a-03-100	[0.70-0.90]	**80**	4429.21	4455.14	**-0.59**	4580.98	**-3.43**	4312.95	2.62	82	4368.77	1.36	4302.31	2.86	4418.83	0.23	-	-
a-04-150	[0.70-0.90]	**119**	6482.11	6513.36	**-0.48**	6479.46	0.04	6267.48	3.31	123	6354.21	1.97	6196.36	4.41	6513.36	**-0.48**	5950.35	8.20
a-05-199	[0.70-0.90]	**158**	8329.55	8368.35	**-0.47**	8323.72	0.07	8081.58	2.98	162	8343.95	**-0.17**	7944.63	4.62	8586.55	**-3.09**	7207.04	13.48
a-06-120	[0.70-0.90]	**95**	10302.16	10302.16	0.00	10158.32	1.40	10063.47	2.32	99	10550.57	**-2.41**	10304.08	**-0.02**	11007.73	**-6.85**	8941.79	13.20
a-07-100	[0.70-0.90]	80	5028.78	5058.76	**-0.60**	5065.26	**-0.73**	4919.48	2.17		4960.75	1.35	4867.79	3.20	4965.94	1.25	4779.13	4.96

z denotes objective function value obtained. IMP denotes percentage objective function reduction over iVNDiv. m denotes number of vehicles in final iVNDiv, ICA+VND SS, and MA|PM final solutions. m' denotes number of vehicles in final Splitabu, Splitabu-DT, and Fast-Splitabu solutions. [a] Tested instances were generated by Mota et al. (2007).

[b] Tested instances were generated by Archetti et al. (2006). iVNDiv run using similar problem generator. [c] Tested instances were generated by Chen et al. (2007).

Table 6.2: Comparison of iVNDiv to Best Known Solutions for instances of Archetti et al. (2006).

		Best Known			iVNDiv		% above Best	
Problem	Demand	z	m	Source	z	m	z	m
a-01-050		521.00	5	Belenguer et al. (2000)	524.61	5	0.69	0.00
a-02-075		823.89	10	Boudia et al. (2007)	851.24	10	3.32	0.00
a-03-100		829.44	8	Boudia et al. (2007)	852.74	8	2.81	0.00
a-04-150		1041.99		Chen et al. (2007)	1074.11	12	3.08	
a-05-199		1307.40		Chen et al. (2007)	1368.67	16	4.69	
a-06-120		1041.20	7	Boudia et al. (2007)	1201.83	7	15.43	0.00
a-07-100		819.56	10	Boudia et al. (2007)	824.78	10	0.64	0.00
a-01-050	[0.01-0.10]	457.21		Chen et al. (2007)	471.92	3	3.22	
a-02-075	[0.01-0.10]	598.25		Chen et al. (2007)	597.46	4	**-0.13**	
a-03-100	[0.01-0.10]	726.81	5	Boudia et al. (2007)	745.35	5	2.55	0.00
a-04-150	[0.01-0.10]	875.16		Chen et al. (2007)	891.98	8	1.92	
a-05-199	[0.01-0.10]	1018.71	10	Boudia et al. (2007)	1073.55	10	5.38	0.00
a-06-120	[0.01-0.10]	976.57	6	Boudia et al. (2007)	1087.80	6	11.39	0.00
a-07-100	[0.01-0.10]	633.80	5	Mota et al. (2007)	673.54	5	6.27	0.00
a-01-050	[0.10-0.30]	723.57		Chen et al. (2007)	766.19	10	5.89	
a-02-075	[0.10-0.30]	1074.01	15	Mota et al. (2007)	1099.47	15	2.37	0.00
a-03-100	[0.10-0.30]	1392.85	20	Boudia et al. (2007)	1425.90	20	2.37	0.00
a-04-150	[0.10-0.30]	1844.96		Chen et al. (2007)	1978.01	29	7.21	
a-05-199	[0.10-0.30]	2258.66		Chen et al. (2007)	2464.65	38	9.12	
a-06-120	[0.10-0.30]	2568.90		Chen et al. (2007)	2806.92	23	9.27	
a-07-100	[0.10-0.30]	1414.33		Chen et al. (2007)	1428.27	20	0.99	
a-01-050	[0.10-0.50]	943.86		Chen et al. (2007)	1039.89	15	10.17	
a-02-075	[0.10-0.50]	1393.53		Chen et al. (2007)	1478.67	22	6.11	
a-03-100	[0.10-0.50]	1845.30	29	Boudia et al. (2007)	1956.13	29	6.01	0.00
a-04-150	[0.10-0.50]	2532.93		Chen et al. (2007)	2671.62	43	5.48	
a-05-199	[0.10-0.50]	3191.25	56	Boudia et al. (2007)	3411.38	56	6.90	0.00
a-06-120	[0.10-0.50]	3687.06		Chen et al. (2007)	4026.53	34	9.21	
a-07-100	[0.10-0.50]	1973.34		Chen et al. (2007)	2007.11	29	1.71	

Continued on next page

Table 6.2: Comparison of iVNDiv to Best Known Solutions for instances of Archetti et al. (2006) (*Continued*).

		Best Known			iVNDiv		% above Best	
Problem	Demand	z	m	Source	z	m	z	m
a-01-050	[0.10-0.90]	1408.34		Chen et al. (2007)	1522.43	25	8.10	
a-02-075	[0.10-0.90]	2056.54		Chen et al. (2007)	2200.51	37	7.00	
a-03-100	[0.10-0.90]	2746.75	56	Archetti et al. (2006)	2865.86	48	4.34	-14.29
a-04-150	[0.10-0.90]	3849.73	84	Archetti et al. (2006)	4165.18	73	8.19	-13.10
a-05-199	[0.10-0.90]	4737.47	107	Archetti et al. (2006)	5184.57	93	9.44	-13.08
a-06-120	[0.10-0.90]	6079.14		Chen et al. (2007)	6364.87	56	4.70	
a-07-100	[0.10-0.90]	3010.50		Archetti et al. (2006)	3156.31	48	4.84	
a-01-050	[0.30-0.70]	1408.68		Chen et al. (2007)	1540.39	25	9.35	
a-02-075	[0.30-0.70]	2112.61		Chen et al. (2007)	2238.98	37	5.98	
a-03-100	[0.30-0.70]	2764.25	53	Archetti et al. (2006)	2941.64	49	6.42	-7.55
a-04-150	[0.30-0.70]	3967.11	80	Archetti et al. (2006)	4165.18	73	4.99	-8.75
a-05-199	[0.30-0.70]	5001.45	103	Archetti et al. (2006)	5363.65	96	7.24	-6.80
a-06-120	[0.30-0.70]	6123.96		Chen et al. (2007)	6545.50	58	6.88	
a-07-100	[0.30-0.70]	2882.12		Archetti et al. (2006)	3225.63	49	11.92	
a-01-050	[0.70-0.90]	2056.01		Chen et al. (2007)	2215.34	40	7.75	
a-02-075	[0.70-0.90]	3067.19		Chen et al. (2007)	3304.24	60	7.73	
a-03-100	[0.70-0.90]	4278.83	82	Archetti et al. (2006)	4429.21	80	3.51	-2.44
a-04-150	[0.70-0.90]	5950.35		Chen et al. (2007)	6482.11	119	8.94	
a-05-199	[0.70-0.90]	7207.04		Chen et al. (2007)	8329.55	158	15.58	
a-06-120	[0.70-0.90]	8941.79		Chen et al. (2007)	10302.16	95	15.21	
a-07-100	[0.70-0.90]	4773.59		Archetti et al. (2006)	5028.78	80	5.35	

z denotes objective function value obtained.

m denotes number of vehicles in final solution.

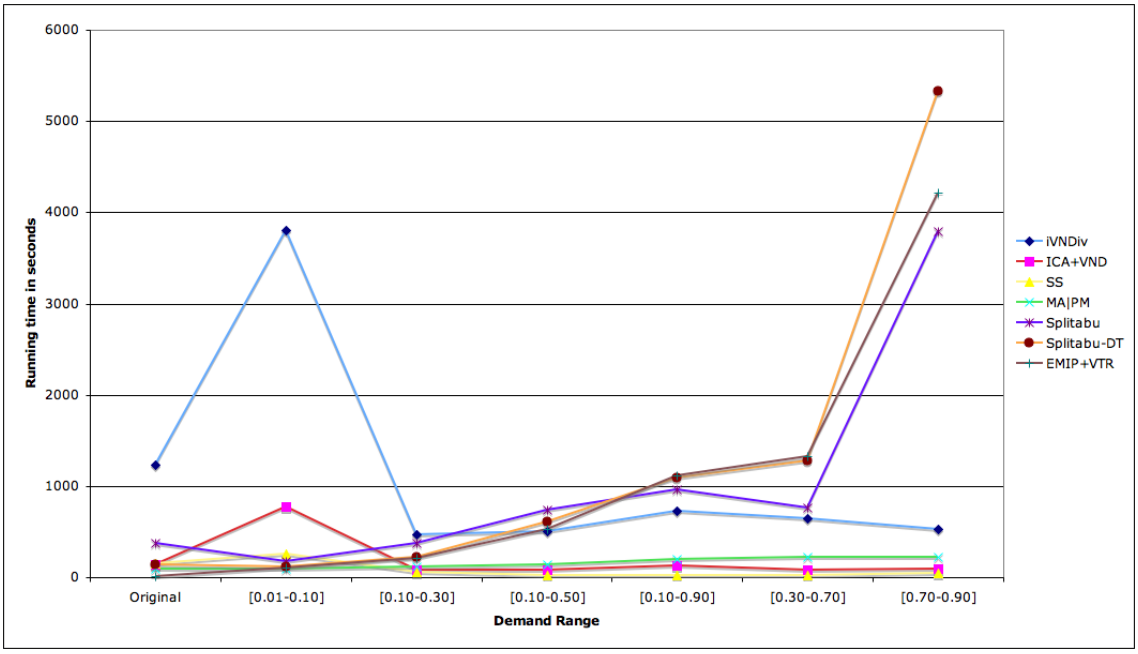

Figure 6.8: Average running times versus demand range for instances of Archetti et al. (2006).

to find a better solution. This operator may be particularly expensive when the number of vehicles is large, as in problems with larger customer demands. In the case of EMIP+VRTR, the number of endpoints increases with the number of routes as one or two endpoints and the closest neighbors to each endpoint are considered for each route. For larger average customer demands, the resulting mixed integer program can be considerable in size and more difficult to solve. Figure 6.8 shows the average running times of the existing algorithms on the instances of Archetti, Hertz, and Speranza (2006) grouped by demand range. Note how the time increases with the customer demand for the tabu searches and EMIP+VRTR. Because both the constructive approach and VND of Aleman et al. (2007) evaluate the cheapest insertion position, a larger number of customers per route increases the complexity of iVNDiv. The large average running time of iVNDiv on problems in the range [0.01-0.10] is caused by a large number of stops per route. Running times for iVNDiv were found related to the ratio n/m (see Table 6.4), which is an estimation of the expected stops per route. This dependency is illustrated in Figure 6.9.

Table 6.3: Running times of iVNDiv on instances of Archetti et al. (2006).

Problem	Demand	iVNDiv[a]	ICA+VND[a]	SS[b]	MA|PM[c]	Splitabu[d]	Splitabu-DT[d]	EMIP+VTR[e]
a-01-050		54.9063	10.89	24.80	8.53	17.00	13.20	1.80
a-02-075		83.2813	9.81	61.66	35.72	63.60	35.80	4.00
a-03-100		319.3281	43.50	108.80	34.59	59.60	57.60	-
a-04-150		1361.1563	129.23	261.28	103.69	439.60	389.00	10.00
a-05-199		3284.6406	534.83	352.31	353.84	1900.40	386.40	18.10
a-06-120		3414.4063	257.30	131.34	50.92	40.00	38.40	5.60
a-07-100		126.0781	21.02	108.41	42.89	86.40	49.00	3.70
a-01-050	[0.01-0.10]	33.7031	4.52	26.86	12.38	9.00	4.80	1.90
a-02-075	[0.01-0.10]	303.7656	51.28	68.80	18.75	42.40	13.00	25.80
a-03-100	[0.01-0.10]	2194.2344	415.47	125.06	37.12	58.60	31.20	-
a-04-150	[0.01-0.10]	3461.4375	666.20	352.09	100.27	258.00	172.80	107.80
a-05-199	[0.01-0.10]	15505.2183	3750.44	963.84	356.22	753.80	525.80	413.40
a-06-120	[0.01-0.10]	3952.6719	341.59	163.28	72.98	60.60	42.40	36.40
a-07-100	[0.01-0.10]	1207.4219	222.42	80.56	34.97	71.20	57.80	53.90

Continued on next page

Table 6.3: Running times of iVNDiv on instances of Archetti et al. (2006) (*Continued*).

Problem	Demand	iVNDiv[(a)]	ICA+VND[(a)]	SS[(b)]	MA|PM[(c)]	Splitabu[(d)]	Splitabu-DT[(d)]	EMIP+VTR[(e)]
a-01-050	[0.10-0.30]	19.7656	1.59	26.31	10.22	27.20	21.80	3.40
a-02-075	[0.10-0.30]	73.0469	13.19	86.02	34.14	78.00	45.40	57.00
a-03-100	[0.10-0.30]	190.5313	34.09	98.00	78.06	121.60	95.80	-
a-04-150	[0.10-0.30]	878.5469	164.19	10.06	147.89	544.80	393.20	308.00
a-05-199	[0.10-0.30]	1457.1563	248.83	19.11	347.14	1224.40	754.80	618.50
a-06-120	[0.10-0.30]	558.5625	54.25	11.33	144.19	516.00	142.60	136.40
a-07-100	[0.10-0.30]	123	22.56	151.25	43.27	85.20	146.00	126.50
a-01-050	[0.10-0.50]	18.1563	2.81	3.84	12.49	55.60	28.20	14.70
a-02-075	[0.10-0.50]	67.7969	11.25	6.09	37.38	71.00	123.20	214.00
a-03-100	[0.10-0.50]	154.4688	25.16	7.55	28.39	205.60	136.20	-
a-04-150	[0.10-0.50]	625.8281	111.66	16.17	224.89	563.80	739.20	630.50
a-05-199	[0.10-0.50]	2173.8438	339.36	20.64	436.20	3810.60	2668.00	1775.70
a-06-120	[0.10-0.50]	358.5625	40.53	63.80	163.14	259.00	268.00	220.70
a-07-100	[0.10-0.50]	107.4688	11.92	41.23	51.31	188.20	292.80	287.60

Continued on next page

Table 6.3: Running times of iVNDiv on instances of Archetti et al. (2006) (*Continued*).

Problem	Demand	iVNDiv[a]	ICA+VND[a]	SS[b]	MA|PM[c]	Splitabu[d]	Splitabu-DT[d]	EMIP+VTR[e]
a-01-050	[0.10-0.90]	16.3594	2.83	3.91	21.42	34.00	60.80	55.40
a-02-075	[0.10-0.90]	71.1094	10.80	6.64	46.11	311.20	193.40	401.10
a-03-100	[0.10-0.90]	126.5156	19.00	9.16	84.38	412.20	648.60	-
a-04-150	[0.10-0.90]	671.3594	141.27	25.03	244.91	1822.40	2278.00	2220.00
a-05-199	[0.10-0.90]	3650.5938	662.77	71.09	725.69	2598.40	3297.20	3038.10
a-06-120	[0.10-0.90]	458.9063	42.00	15.86	196.14	1037.00	877.80	722.80
a-07-100	[0.10-0.90]	96.9844	12.89	9.08	52.13	523.40	259.60	251.20
a-01-050	[0.30-0.70]	15.3281	2.20	4.25	24.53	51.80	48.60	47.90
a-02-075	[0.30-0.70]	80.2969	11.28	7.14	51.78	184.40	128.60	509.60
a-03-100	[0.30-0.70]	103.9375	15.14	10.36	100.16	453.80	810.20	-
a-04-150	[0.30-0.70]	675.3906	143.05	19.38	244.86	1512.40	3008.00	3028.30
a-05-199	[0.30-0.70]	3026.2188	349.97	120.28	749.94	2279.40	3565.60	3035.70
a-06-120	[0.30-0.70]	469.1719	59.20	17.16	271.39	476.60	658.60	605.40
a-07-100	[0.30-0.70]	110.0469	13.69	9.73	91.31	411.00	777.80	716.50
a-01-050	[0.70-0.90]	18.7031	2.59	4.13	22.91	159.80	106.40	135.40
a-02-075	[0.70-0.90]	58.0469	10.25	7.66	27.48	436.60	869.20	811.00
a-03-100	[0.70-0.90]	94.9844	14.31	12.06	55.75	1891.40	1398.40	-
a-04-150	[0.70-0.90]	584.8438	93.78	131.91	401.62	8782.80	10223.20	10038.80
a-05-199	[0.70-0.90]	2124.6563	460.89	165.28	571.70	11346.80	21849.20	12542.30
a-06-120	[0.70-0.90]	636.7188	59.28	20.17	298.08	2032.60	1825.60	725.40
a-07-100	[0.70-0.90]	178.1875	20.70	9.19	180.11	1865.00	1004.40	1024.30

[a]P4, 512MB, 2.8 GHz; [b]P4, 1.0GB, 2.4 GHz; [c]PC 3.0 GHz; [d]P4, 256MB, 2.4 GHz; [e]P4, 512MB, 1.7 GHz.

Table 6.4: Expected stops per route for problems of Archetti et al. (2006).

Demand Range	Stops per route (n/m)	Running time in seconds
[0.70-0.90]	1.25	528.02
[0.30-0.70]	2.04	640.06
[0.10-0.90]	2.08	727.40
[0.10-0.50]	3.46	500.88
[0.10-0.30]	5.09	471.52
Original	11.73	1234.83
[0.01-0.10]	19.15	3808.35

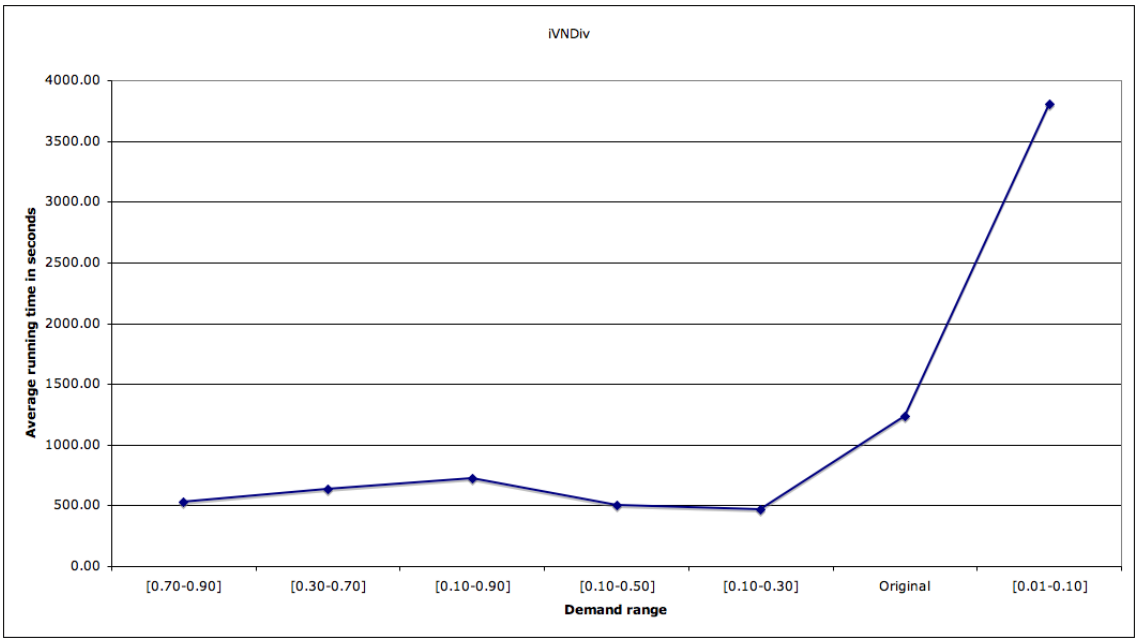

Figure 6.9: Average running times of iVNDiv versus demand range for instances of Archetti et al. (2006).

Table 6.5: Computational results of iVNDiv on some TSPLIB instances.

	iVNDiv		Belenguer et al. (2000)			MA|PM		
Problem	$z^{(a)}$	CPU$^{(b)}$	UB$^{(a)}$	LB	%ALB	$z^{(a)}$	CPU$^{(c)}$	IMP
b-eil22-021	375	4.19	375	375	0.00	375	4.11	0.00
b-eil23-022	570	3.42	569	569	0.18	569	5.47	0.18
b-eil30-029	510	14.47	510	508	0.39	503	5.70	1.37
b-eil33-032	851	14.03	835	833	2.12	835	5.19	1.88
b-eil51-050	521	54.91	521	511.57	1.81	521	7.28	0.00
b-eilA76-075	847	83.28	832	782.7	7.59	828	35.94	2.24
b-eilB76-075	1055	79.00	1023	937.47	11.14	1019	13.09	3.41
b-eilC76-075	746	148.20	735	706.01	5.36	738	14.75	1.07
b-eilD76-075	695	140.83	683	659.43	5.12	682	23.12	1.87
b-eilA101-100	843	319.33	817	793.48	5.87	818	25.25	2.97
b-eilB101-100	1122	185.84	1077	1005.85	10.35	1082	21.81	3.57

Continued on next page

Tables 6.5 and 6.6 show the computational results on the 25 problems given in Belenguer et al. (2000). The iVNDiv solution values are compared with the bounds found by Belenguer et al. with a cutting plane and a heuristic approach, the solution values found with MA|PM, the bounds obtained with the branch-and-price approach (B&P) of Liu (2005), the solution values found with EMIP+VRTR, and the bounds produced by the column generation approach of Jin et al. (2008). Unpublished values are omitted from the tables. Bold type indicate iVNDiv providing a better feasible solution. Note that upper bounds and solution values obtained by Belenguer et al. and MA|PM are calculated by rounding inter-node distance values to the nearest integer.

Table 6.7 shows the computational results on the new 21 problems generated by Chen et al. (2007). The table contains solution values z, running times in seconds, and the percentage improvements of iVNDiv over the EMIP+VRTR hybrid approach. Bold text is used to indicate the new best solutions found so far for this new problem set. According to the literature, this is the first time this problem set is used after Chen et al. (2007) and Aleman et al. (2007). Out of the 21 prob-

Table 6.5: Computational results of iVNDiv on some TSPLIB instances (*Continued*).

	iVNDiv		B&P			Jin et al. (2008)			
Problem	z	CPU[(b)]	UB	LB	%ALB	UB	LB	CPU[(d)]	%ALB
b-eil22-021	375.28	4.19	**376.00**	373.60	0.45	-	-	-	-
b-eil23-022	569.75	3.42	**608.00**	564.30	0.96	-	-	-	-
b-eil30-029	512.72	14.47	**515.30**	507.20	1.08	-	-	-	-
b-eil33-032	853.10	14.03	**873.40**	830.20	2.68	-	-	-	-
b-eil51-050	524.61	54.91	**558.50**	507.60	3.24	-	-	-	-
b-eilA76-075	851.24	83.28	**900.70**	800.30	5.98	-	-	-	-
b-eilB76-075	1059.57	79.00	**1163.10**	965.70	8.86	**1063.75**	981.14	45084.00	7.40
b-eilC76-075	753.29	148.20	**809.30**	711.20	5.59	-	-	-	-
b-eilD76-075	699.35	140.83	**768.80**	652.30	6.73	-	-	-	-
b-eilA101-100	852.74	319.33	**910.20**	797.50	6.48	-	-	-	-
b-eilB101-100	1139.27	185.84	**1174.10**	1013.90	11.00	-	-	-	-

z denotes objective function value obtained. CPU denotes running time in seconds.

IMP denotes percentage objective function reduction over iVNDiv.

%ALB denotes percent iVNDiv above lower bound.

MA|PM uses one more vehicle than the minimum fleet size on instance b-eil30-029.

[(a)] Objective function value obtained with euclidean distances truncated to the nearest integer.

[(b)]P4, 512MB, 2.8 GHz; [(c)]PC 3.0 GHz; [(d)]P4, 2GB, 2.8 GHz.

Table 6.6: Computational results of iVNDiv on the random problems of Belenguer et al. (2000).

	iVNDiv		Belenguer et al. (2000)			MA|PM		
Problem	$z^{(a)}$	CPU$^{(b)}$	UB$^{(a)}$	LB	%ALB	$z^{(a)}$	CPU$^{(c)}$	IMP
b-S51D1-050	466	40.53	458	454	2.58	458	8.77	1.72
b-S51D2-050	725	28.34	**726**	676.63	6.67	707	7.44	2.48
b-S51D3-050	994	14.70	972	905.22	8.93	945	7.84	4.93
b-S51D4-050	1672	16.53	**1677**	1520.67	9.05	1578	11.98	5.62
b-S51D5-050	1385	13.94	**1440**	1272.86	8.10	1351	16.72	2.45
b-S51D6-050	2211	16.83	**2327**	2113.03	4.43	2182	9.92	1.31
b-S76D1-075	600	476.27	594	584.87	2.52	592	15.23	1.33
b-S76D2-075	1138	46.94	**1147**	1020.32	10.34	1089	30.5	4.31
b-S76D3-075	1485	53.34	1474	1346.29	9.34	1427	12.89	3.91
b-S76D4-075	2160	51.84	**2257**	2011.64	6.87	2117	8.76	1.99
b-S101D1-100	740	2125.58	716	700.56	5.33	717	49.75	3.11
b-S101D2-100	1426	217.91	1393	†1270.97	10.87	1372	31.72	3.79
b-S101D3-100	1974	146.61	**1975**	†1739.66	11.87	1891	33.98	4.20
b-S101D5-100	2970	104.05	2915	†2630.43	11.43	2854	18.66	3.91

Continued on next page

Table 6.6: Computational results of iVNDiv on the random problems of Belenguer et al. (2000) (*Continued*).

	iVNDiv		B&P			EMIP+VRTR			Jin et al. (2008)			
Problem	z	CPU[(b)]	UB	LB	%ALB	z	CPU[(d)]	IMP	UB	LB	CPU[(e)]	%ALB
b-S51D1-050	471.92	40.53	**513.90**	449.90	4.67	-	-	-	-	-	-	-
b-S51D2-050	731.01	28.34	**1296.50**	556.70	23.85	-	-	-	723.37	694.98	5978	4.93
b-S51D3-050	1001.22	14.70	986.00	956.00	4.52	-	-	-	968.85	922.72	607	7.84
b-S51D4-050	1680.66	16.53	1654.00	1623.00	3.43	1586.50	201.74	5.60	1657.61	1505.35	260	10.43
b-S51D5-050	1389.40	13.94	**1434.00**	1416.00	-1.91	1355.50	201.62	2.44	**1439.92**	1297.46	46	6.62
b-S51D6-050	2218.23	16.83	**2316.00**	2270.00	-2.33	2197.80	301.90	0.92	**2300.21**	2108.59	243	4.94
b-S76D1-075	606.47	476.27	-	-	-	-	-	-	-	-	-	-
b-S76D2-075	1143.36	46.94	-	-	-	-	-	-	**1185.72**	1066.17	12806	6.75
b-S76D3-075	1490.08	53.34	-	-	-	-	-	-	**1504.94**	1397.43	2030	6.22
b-S76D4-075	2173.61	51.84	**2205.00**	2178.00	-0.20	2136.40	601.92	1.71	**2219.07**	2019.91	1813	7.07
b-S101D1-100	749.19	2125.58	-	-	-	-	-	-	-	-	-	-
b-S101D2-100	1443.44	217.91	-	-	-	-	-	-	**1474.51**	1349.77	47658	6.49
b-S101D3-100	1988.78	146.61	-	-	-	-	-	-	**2012.86**	1837.33	7959	7.62
b-S101D5-100	2984.48	104.05	-	-	-	2846.20	645.99	4.63	2954.96	2725.5	847	8.68

†Earlier termination of the algorithm due to memory overflow (Belenguer et al., 2000). z denotes objective function value obtained.

CPU denotes running time in seconds. IMP denotes percentage objective function reduction over iVNDiv.

%ALB denotes percent iVNDiv above lower bound. [(a)] Obtained with euclidean distances truncated to the nearest integer.

[(b)]P4, 512MB, 2.8 GHz; [(c)]PC 3.0 GHz; [(d)]P4, 512MB, 1.7 GHz; [(e)]P4, 2GB, 2.8 GHz.

lems, the iVNDiv improves the solution values in 15 cases and equals the EMIP+VRTR solution in the problem with 8 customers. The iVNDiv is computationally faster than EMIP+VRTR in all the cases. ICA+VND performs similar to iVNDiv in terms of solution values, although iVNDiv performs better in 7 cases. It is difficult to see how the running time of EMIP+VRTR is affected by the problem size as it uses the maximum amount of computing time to solve the endpoint mixed integer program for problems with at least 24 customers (Chen et al., 2007).

Finally, Table 6.8 shows the solution values and running times in seconds of iVNDiv with respect to the optimal solutions found with the exact approach of Jin et al. (2007). iVNDiv generates high quality solutions that are 1.19% above optimality on average for the tested problems. Table 6.8 reveals how difficult it is to exactly solve small sized problems (up to 21 customers) while iVNDiv finds the optimal solution in three of the tested problems and is quite close to optimality in another problem at a low computational effort.

6.5 Conclusions

Diversification of a local search process can be computationally expensive but is of benefit on harder optimization problems. This research presents a new diversification method for routing problems based on a novel use of spatially varied concentric rings around the routing depot. A set of diversified solutions are used to restart the VND search process of Aleman et al. (2007). A comprehensive empirical test of this new diversification method was conducted and the reported results show the utility of this new diversification scheme. The proposed diversification strategy can be used to solve any variant of the vehicles routing problem as long as the constructive approach considers the corresponding side constraints. Although the proposed diversification scheme is based on a geographical division of the problem by means of concentric rings centered at the depot, this geographical division can be modified. For example, instead of excluding all the customers in a complete ring, it may be divided into sectors to exclude only the customers in those regions of the ring.

There are a couple future avenues of research. For instance, an aggressive diversification

Table 6.7: Computational results of iVNDiv on instances of Chen et al. (2007).

	EMIP + VRTR		ICA + VND		iVNDiv		
Problem	z	CPU[(a)]	z	CPU[(b)]	z	CPU[(b)]	m
c-SD01-008	**228.28**	0.7	228.28	0.06	228.28	0.19	6
c-SD02-016	714.40	54.4	**708.28**	0.22	**708.28**	1.48	12
c-SD03-016	430.61	67.3	**430.58**	0.17	**430.58**	0.58	12
c-SD04-024	**631.06**	400	635.84	0.55	635.84	2.31	18
c-SD05-032	1408.12	402.7	**1390.57**	0.69	**1390.57**	5.55	24
c-SD06-032	**831.21**	408.3	831.24	0.94	831.24	2.95	24
c-SD07-040	3714.40	403.2	**3640.00**	1.03	**3640.00**	8.13	30
c-SD08-048	5200.00	404.1	**5068.28**	1.75	**5068.28**	11.91	36
c-SD09-048	**2059.84**	404.3	2071.03	2.91	2071.03	19.73	36
c-SD10-064	2749.11	400	**2747.83**	3.58	**2742.84**	33.27	48
c-SD11-080	13612.12	400.1	**13280.00**	3.97	**13280.00**	35.16	60
c-SD12-080	7399.06	408.3	7279.97	4.00	**7265.70**	43.13	60
c-SD13-096	10367.06	404.5	**10110.58**	5.80	**10110.58**	50.97	72
c-SD14-120	11023.00	5021.7	10893.50	15.49	**10829.25**	141.77	90
c-SD15-144	15271.77	5042.3	**15168.28**	18.33	**15168.28**	191.66	108
c-SD16-144	**3449.05**	5014.7	3635.27	39.71	3580.07	2120.14	108
c-SD17-160	26665.76	5023.6	26559.93	17.42	**26556.13**	179.61	120
c-SD18-160	14546.58	5028.6	14440.59	40.38	**14372.80**	366.14	120
c-SD19-192	20559.21	5034.2	20191.19	27.64	**20188.62**	330.06	144
c-SD20-240	40408.22	5053	39813.49	63.18	**39803.13**	633.33	180
c-SD21-288	**11491.67**	5051	11799.60	738.49	11682.09	9387.55	216

z denotes objective function value obtained.

CPU denotes running time in seconds.

m denotes the number of vehicles in the final iVNDiv and ICA+VND solutions.

[(a)]P4, 512MB, 1.7 GHz.

[(b)]P4, 512MB, 2.8 GHz.

Table 6.8: Computational results of iVNDiv versus optimality on instances of Jin et al. (2007).

	TSVI		iVNDiv		% above
Problem	z^*	CPU[(a)]	z	CPU[(b)]	optimality
j-Eil22-021	375.28	17 h	375.28	4.19	**0.00**
j-J1-018	127.39	13 h	127.49	1.73	0.08
j-J2-021	388.44	84 h	388.44	4.59	**0.00**
j-J3-022	367.93	17 h	389.54	3.73	5.87
j-J4-022	372.2	13 h	372.2	5.64	**0.00**

z^* denotes optimal objective function value.

z denotes objective function value obtained.

CPU denotes running time: [(a)]—; [(b)]P4, 512MB, 2.8 GHz.

scheme is employed focusing on the best solutions. Future studies might consider examining the worse solutions as a means of potentially maximizing the distance between a current solution and a new search area. Another avenue would be to use the ring-based diversification method as a vocabulary building mechanism to construct either high quality solutions, or to diversify solutions whose components are selected based on low frequency of use. These avenues are currently under investigation; the vocabulary building approach is presented next.

Chapter 7

A Tabu Search with Vocabulary Building Approach for the Vehicle Routing Problem with Split Demands[1]

7.1 Introduction

The vehicle routing problem (VRP) seeks optimal routes over which similar vehicles deliver demands to geographically dispersed customers. The VRP presumes all vehicles leave from and return to a common depot. Customer demand is fully met by the vehicle visiting that customer. The split delivery vehicle routing problem (SDVRP), first introduced by Dror and Trudeau (1989) and then Dror and Trudeau (1990), is a variant of the VRP. The SDVRP relaxes vehicle restrictions so that customer demands can be supplied by one or more vehicles. The intent of the relaxation is so the SDVRP approach can yield more efficient route structures.

Mathematically, the SDVRP is defined on an undirected, fully connected graph $G = (V, E)$ where $V = \{0, 1, ..., n\}$ is the set of $n + 1$ nodes of the graph, and $E = \{(i, j) : i, j \in V,\ i < j\}$ is the set of edges connecting the nodes. Node 0 represents a depot where a fleet $M = \{1, ..., m\}$

[1]This chapter is found as Aleman and Hill (2008).

of identical vehicles with capacity Q are stationed, while the remaining node set $N = \{1, ..., n\}$ represents the customers. A non-negative cost, usually the inter-node distance, c_{ij}, is associated with every edge $(i, j) \in E$. Each customer $i \in N$ has a demand of q_i units and is located at a point (x_i, y_i) in the two-dimensional space. The coordinates (x_0, y_0) indicate the depot location. This notation is used throughout this chapter to describe the proposed SDVRP solution approach.

The SDVRP solution potentially reduces the operational cost of the fleet, especially when the average customer demand exceeds 10% of the vehicle capacity (as stated by Dror and Trudeau, 1989). In their worst-case analysis of the SDVRP, Archetti, Savelsbergh, and Speranza (2006) show that the reduction in delivery costs obtainable by allowing split deliveries is at most 50%, and this reduction bound is tight. Archetti et al. (2008) suggest that the benefits are mainly due to the reduction in the number of vehicles required to supply the customer demands. Their mathematical analysis proved that the maximum reduction in the number of vehicles is 50% and the largest reduction occurs when the mean customer demand is between 50% and 70% of the vehicle capacity with demand variances relatively small.

Although the literature on SDVRP is limited, different solution techniques have been implemented to solve the problem. These techniques include branch and bound, column generation, dynamic programming, lagrangian relaxation, mixed integer programming, constructive approaches, local, tabu and scatter search, variable neighborhood descent, memetic algorithms, and simulated annealing. Most of these solution approaches have been developed in the last four years showing an increasing interest in this routing problem. The reader is referred to Archetti and Speranza (2007), Chen et al. (2007) and Aleman et al. (2008) for a thorough literature review on properties, applications, and algorithms available. A state-of-the-art compilation of this literature is presented in Table 7.1 showing the author(s), the solution methodologies deployed, and some remarkable information describing the study. This compilation includes the very recent work of Rubrico et al. (2004), Nakao and Nagamochi (2007), Schmid (2007), Archetti, Speranza, and Savelsbergh (2008), Belfiore et al. (2008), Bolduc et al. (2008), and Nowak et al. (2008).

Table 7.1: Existing literature on SDVRP.

Author(s)	Year	Solution methodology & Remarks
Dror and Trudeau	1989	Local search
Dror and Trudeau	1990	Properties and complexity
Frizzell and Giffin	1992	Constructive heuristic, grid network distances
Bouzaiene-Ayari et al.	1993	Adapted Clarke and Wright, stochastic demands
Dror et al.	1994	Branch and bound, properties
Frizzell and Giffin	1995	Constructive heuristic, grid network distances, time windows
Mullaseril et al.	1997	Adapted Dror and Trudeau (1989) for time windows
Sierksma and Tijssen	1998	Column generation, crew exchange
Belenguer et al.	2000	Lower bounds
Song et al.	2002	Newspaper allocation
Ho and Haugland	2004	Tabu search, time windows
Rubrico et al.	2004	Fast heuristics, grid distances
Archetti et al.	2005	Complexity; vehicles with capacity of k units
Liu	2005	Two-stage algorithm, valid inequalities
Nowak	2005	Dynamic program, pickup and delivery
Archetti, Hertz, and Speranza	2006	Tabu search
Archetti, Savelsbergh, and Speranza	2006	Worst-case analysis and potential savings
Belfiore et al.	2006	Scatter search, heterogeneous fleet, time windows
Gendreau	2006	Properties and review
Lee et al.	2006	Dynamic programming and shortest path
Yu et al.	2006	Lagrangian relaxation, inventory routing
Aleman et al.	2007	Variable neighborhood descent, adaptive memory concepts
Ambrosino and Sciomachen	2007	Local search, clustering procedure
Archetti and Speranza	2007	Survey on the SDVRP
Boudia et al.	2007	Memetic algorithm with population management
Chen et al.	2007	Mixed integer program, heuristic algorithm
Jin et al.	2007	Exact method with valid inequalities
Mota et al.	2007	Scatter search
Nakao and Nagamochi	2007	Dynamic program, set of items per customer
Schmid	2007	Exact and heuristic approaches, multi-depot, heterogeneous fleet, time windows
Tavakkoli-Moghaddam et al.	2007	Simulated annealing, heterogeneous fleet
Wilck and Cavalier	2007	Loaded travel cost objective
Aleman et al.	2008	Diversification strategies, variable neighborhood descent
Archetti, Savelsbergh, and Speranza	2008	Empirical analysis; benefits of split deliveries
Archetti, Speranza, and Savelsbergh	2008	Integer program, heuristic search
Belfiore et al.	2008	Scatter search, time windows
Bolduc et al.	2008	Tabu search, time horizon, production, inventory
Jin et al.	2008	Column generation, lower bounds
Nowak et al.	2008	Heuristic approach, tabu list, pickup and delivery

This chapter presents a learning procedure, called Tabu Search with Vocabulary Building Approach (TSVBA) for solving the SDVRP. TSVBA is a population-based search approach that constructs an initial set of solutions and then uses the set of solutions to find attractive solution attributes with which to construct new solutions. As the search progresses, the solution set evolves; better solutions move into the set while bad solutions are removed. The initial set is constructed by varying the critical angle of the constructive approach described in Aleman et al. (2007). New solutions are also created using an adaptation of the well-known savings algorithm of Clarke and Wright (1964) that uses an improved version of the variable neighborhood descent presented in Aleman et al. (2007). The remainder of this chapter is organized as follows. Section 7.2 describes the proposed approach. Section 7.3 provides the procedure to construct the initial set of solutions while the generation of new solutions is presented in Section 7.4. A diversification strategy is outlined in Section 7.5, computational results are provided in Section 7.6, and finally conclusions are given in Section 7.7.

7.2 Tabu Search with Vocabulary Building Approach

Tabu search with vocabulary building (TSVBA) is a learning procedure based on generating an initial set of solutions whose characteristics are then used to construct new solutions of higher quality replacing existing solutions of less quality. The method used to construct the new solutions adapts the savings algorithm of Clarke and Wright (1964). Edges from previous solutions are stored in a short-term memory structure to diversify the search and avoid getting trapped in a local optima. Some of the edges are also kept in an elite list. This elite list functions as an aspiration criteria for those edges within it. Within the framework of tabu search, this helps to intensify the search in regions where good solutions may exist. The solutions generated are improved with the variable neighborhood descent (VND) procedure of Aleman et al. (2007). The VND modifies standard operators used for the VRP and helps improve SDVRP solutions. Since TSVBA is population based, a fast local search is needed to improve the solutions at a relatively low computational effort.

Thus, the VND of Aleman et al. (2007) was modified to create Fast-VND which drastically reduces the required processing time. The improved solutions obtained using Fast-VND replace selected solutions in the set. The set of solutions is continually improved and evolved during the search until a completion criteria is met. Once a final set is obtained, the search is intensified by again improving each solution with another version of the original VND of Aleman et al. (2007), called Slow-VND. The proposed TSVBA is outlined in Table 7.2.

7.3 Initial Set of Solutions

The initial set is formed using various solutions generated with the constructive approach of Aleman et al. (2007) augmented with a solution constructed using an adaptation of the parallel version of the savings algorithm of Clarke and Wright (1964), called Clarke and Wright with split demands (CW-SD). The inclusion of this latter solution in the set augments the solution attributes found in solutions obtained with the Aleman et al. (2007) approach, particularly in problems with clustered customers. Both constructive approaches are described below.

7.3.1 Constructive Approach of Aleman et al. (2007): CA

The CA creates a list of customers sorted according to their distance from the depot. The farthest customer is inserted into the solution to initiate a new route. Subsequent customers are inserted in an existing route or a new route is initiated. The triangular inequality (i.e., $c_{ik} + c_{kj} \geq c_{ij}$, for all i, j, k) favors insertions in existing routes.

The CA adds additional considerations to keep routes from spreading spatially. Potential customers should be placed in routes that are not spread out. The CA employs a route angle control mechanism (RAC) which penalizes the insertion of customers into routes whose angles exceed a critical angle value. The angle of a customer is defined as the angle between the line connecting the depot with the customer and the horizon (i.e., 0 degrees), while the angle of a route is defined as the

Table 7.2: Tabu Search with Vocabulary Building Approach: TSVBA.

Step 1: Generate an empty list of elite edges.

Step 2: Generate an initial set of solutions.

Step 3: Improve all solutions in the set with the Fast-VND.

Step 4: Find the common edges among the solutions in the set and generate a savings list based on the savings measure used in the Clarke and Wright (1964) algorithm.

Step 5: Generate a final savings list with the common edges, elite edges, and the classical savings list of Clarke and Wright (1964), in that order (i.e., concatenate the lists).

Step 6: Construct a solution with the CW-SD and the final savings list.

Step 7: Use the Fast-VND and the route addition local search to improve the constructed solution and generate S_n.

Step 8: Update the short-term memory with the edges existing in solution S_n.

Step 9: If the worst solution in the set, S_w, was improved by S_n, remove from the list of elite edges those not existing in S_n.

Step 10: Compare S_n with the best solution in the set, S_b, and identify those edges in one solution but not in the other.

Step 11: Calculate the degree of attractiveness for each of those edges found in Step 10.

Step 12: If the degree of attractiveness of an edge is greater than 0.5, add the edge to the list of elite edges.

Step 13: If S_n is better than S_w, replace S_w with S_n in the set.

Step 14: If a predefined number of iterations without improving S_w has not been reached, go to Step 4.

Step 15: If a split savings list is being used by CW-SD, go to Step 16. Otherwise, generate a split savings list and replace the classical savings list in Step 5. Restart the iterations and go to Step 4.

Step 16: Use the Slow-VND to improve the solutions in the final set.

Step 17: Stop. The final solution corresponds to S_b in the final set.

maximum angle between the lines connecting the depot with any two customers in the route. In the CA, the critical angle, θ^*, is estimated as the ratio of the circumference (360 degrees) and the fleet size. This divides the region of the problem into equal slices so each vehicle ideally services the customers located inside each slice. The insertion of customers into existing routes is penalized if the route angles after the insertions exceed θ^*; otherwise, the insertion cost corresponds to the extra distance required to service the customer and resume the trajectory of the original route.

The CA performs reasonably well in problems where customers are uniformly spread about the solution region. The CA does not do as well on problems with clustered customers. In problems with clustered customers, some slices of the region may be without customers, or at least less than the expected number of customers. Changing θ^* helps CA find better solutions for such problems.

The TSVBA uses θ^* as a parameter to populate the initial set. The CA performs better in some problems with small θ^* values while better in other problems with large θ^* values. The TSVBA uses θ^* values in the range $[\alpha\theta^*, \beta\theta^*]$, where $0 < \alpha < 1$ and $1 < \beta$, to construct a variety of solutions. To construct the ith solution of the pool, TSVBA uses CA with critical alngle:

$$\theta_i^* = \theta_{min} + \frac{\theta_{max} - \theta min}{k} \times i \tag{7.1}$$

where $\theta_{min} = \alpha\theta^*$, $\theta_{max} = \beta\theta^*$, k is the desired maximum number of solutions to be generated, and $0 \leq i \leq k$. The constructed solutions form the initial set. The TSVBA also ensures only unique solutions are retained in the initial set.

7.3.2 Clarke and Wright (1964) Algorithm with Split Demands: CW-SD

The classical savings algorithm of Clarke and Wright (1964) (CW) is widely used to solve diverse routing problems because of its simplicity and efficiency. CW initially creates an exclusive route for each customer and then merges those routes producing the largest savings in the objective function

Table 7.3: Clarke and Wright (1964) Algorithm with Split Demands: CW-SD.

Step 1: Calculate the savings $s_{ij} = c_{0i} + c_{0j} - c_{ij}$ for each pair of customers (i, j).

Step 2: Sort the pairs (i, j) in descending order based on the savings s_{ij}. This generates a *savings list.* Process the *savings list* starting from the first item (largest s_{ij}).

Step 3: For the savings s_{ij} in consideration, insert the edge (i, j) in a route if no constraints are violated and any of of the following conditions holds:

- Neither customer i nor j are assigned to a route. In this case a new route is initialized with the edge (i, j). If the combined demand of i and j exceeds the vehicle capacity, the demand of the closer customer to the depot is split and a new customer demand is included in the problem whose value is equal to the un-serviced demand.
- Exactly one of the two customers i or j is assigned to an existing route and that customer is not interior in that route. A customer is interior if it is not the first or last customer in the route, excluding the depot. In this case the edge (i, j) is added to the existing route. If the demand of the customer not assigned to the existing route exceeds the vehicle capacity, its demand is split and a new customer demand is included in the problem whose value is equal to the un-serviced demand.
- Both customers i and j are included in existing routes and neither i nor j are interior in the routes. In this case the routes are merged through the edge (i, j). The sequence to service the customers in each route is considered before the merge. For simplicity, no split is allowed if the combined demand of the existing routes exceeds the vehicle capacity.

Step 4: If all customer demands are fully supplied, go to Step 6.

Step 5: If the savings list is not exhausted, return to Step 3 and process the next item in the list. Otherwise, create an exclusive route for each customer with any un-supplied demand.

Step 6: Stop. The solution consists of the routes created in Steps 3 and 5.

value. The savings produced by merging the routes of customers i and j is given by $s_{ij} = c_{0i} + c_{0j} - c_{ij}$. While CW only allows feasible route merges, the CW-SD developed in this study allows infeasible merges. When infeasible, meaning vehicle capacity is exceeded, the customer demand closer to the depot is split and the split demand is modeled as a new customer demand with the same location. The CW-SD is described in Table 7.3.

7.3.3 Adaptation of the VND of Aleman et al. (2007)

Solutions generated with both the CA and CW-SD form the initial set of solutions, which is then improved with an adaptation of the VND of Aleman et al. (2007). In its original version, the

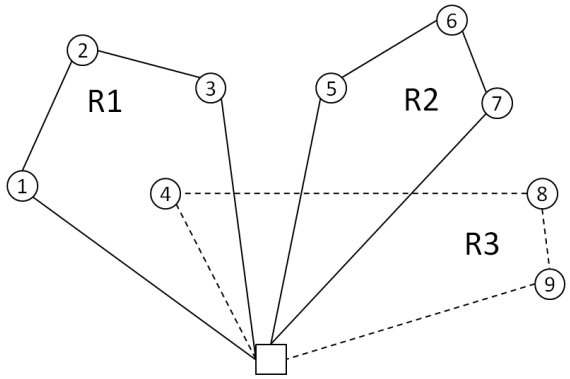

Figure 7.1: Illustration of neighbor routes.

VND exchanges customers between routes and considers the quantities delivered to include/remove split deliveries. This VND is modified for TSVBA to reduce the combinations of pairs of routes, and thus reduce the complexity of the procedure. Instead of using all possible combinations, only neighboring routes are considered. This modified VND is referred to as Fast-VND and we designate the original VND as Slow-VND simply to highlight its more aggressive neighborhood structure.

Because routes are constructed in parallel in both CA and CW-SD, routes with consecutive indices are not necessarily neighbors and thus a routine is used to identify neighbor routes. Two routes are considered neighbors if they are spatially contiguous or their trajectories cross. In Figure 7.1, routes R1 and R2 as well as routes R2 and R3 are neighbors because they are spatially contiguous; however, routes R1 and R3 are neighbors as well because their trajectories cross. At this point, Fast-VND and Slow-VND are used to improve solutions in the framework of the TSVBA.

A common practice in VRP and SDVRP algorithms is to evaluate only the closest customers. This reduces the number of candidates and allows local searches to run quicker. The Fast-VND embedded in the TSVBA considers the closest customers and those located within the region formed by those closest customers. In Figure 7.1, customers 1 and 3 are closest to customer 2; however, customer 4 is in the same region. The Slow-VND evaluates all customers, giving it a larger candidate list and thus a higher computational complexity. However, Slow-VND explores the search space more thoroughly and can provide better solutions than just using the Fast-VND.

7.4 Generation of New Solutions

The initial set is composed of solutions with different objective function values but often solution common attributes. Those attributes are used in the TSVBA to modify the savings list of the CW-SD and construct new – and different – solutions. Those edges that are common among all solutions in the set are utilized to generate a sub-list called *savings list of common edges*. Like the CW savings list, common edges are ordered according to their associated savings measure. Once generated and ordered, both the savings list of the common edges and the CW savings list are combined to create a final savings list with the common edges located in the top positions.

New solutions are generated using CW-SD and the final savings list. These solutions are improved using Fast-VND followed by the route addition local search described below. The edges included in the improved solution are stored in a short-term memory structure and considered tabu for a fixed number of iterations. Edges (i, j) and (j, i) are considered equivalent in the memory structure. Those edges with an active tabu status are ignored in Step 3 in Table 7.3. The tabu tenure is calculated deterministically and is equal to $\sqrt{n}$, where n is the number of customers in the problem. This value was found satisfactory during preliminary experiments and was used during the actual computational experiments.

7.4.1 Route Addition Local Search

Sometimes a customer demand appears in multiple routes. It may be possible to reduce the objective function value by consolidating the customer demand into a new route that supplies the customer exclusively. The TSVBA uses a local search based on this operator to reduce the objective function value (similar to the Splitabu of Archetti et al., 2006). For each split customer demand, this reduction corresponds to the savings produced by removing the demand from all the visiting routes minus twice the distance between the customer and the depot. To reduce the complexity of the local search, the savings for each customer with split demand are pre-processed and the savings of those customers involved in the route addition are updated if the objective function value is reduced.

7.4.2 Appearing/Disappearing and Elite Edges

As the solution set evolves, solution attributes change. Edges in new solutions, but not in the best solutions, are called *appearing edges*. Edges in the best solutions, but not in new solutions, are called *disappearing edges*. Both appearing and disappearing edges are managed in separate lists. The degree of attractiveness of the appearing and disappearing edges is defined as the ratio between the number of solutions containing the edge and the size of the solution set. Those edges with a degree of attractiveness greater than 0.5 are included in an *elite list*. In contrast, those edges in new solutions that already exist in the elite list but disappear when the new solutions are improved with Fast-VND are removed from the elite list.

The elite list is then used to modify the final savings list of the CW-SD. The final savings list is composed of the savings lists formed by the common edges, the elite edges, and the classical savings list as in Clarke and Wright (1964). In CW-SD, the common edges are the first edges inserted in the solution under construction, followed by the elite edges, and finally those edges in the classical savings list.

7.4.3 Conflicting Edges

When edges are added to both the list of common edges and elite list there may be conflicts with the edges already included in the lists. The conflict occurs when one of the two customers, either i or j, in the edge (i, j) to be added to the list are already in the list and have degree 2. The degree of a customer corresponds to the number of edges where it appears in a solution excluding the edges linking the depot. For example, in Figure 7.2 customer 26 has degree 1 (linked to customer 8) whereas customer 8 has degree 2 (linked to customers 26 and 48). Figure 7.2 illustrates some of the common edges – in weighted lines – and the edges selected as elite – in regular lines – during the execution of TSVBA to solve a problem with 50 customers.

In Figure 7.2, the edge $(23, 48)$, with the dashed line, has a degree of attraction greater than 0.5, but its addition to the elite list would conflict with edges $(8, 48)$ and $(27, 48)$. This conflict must

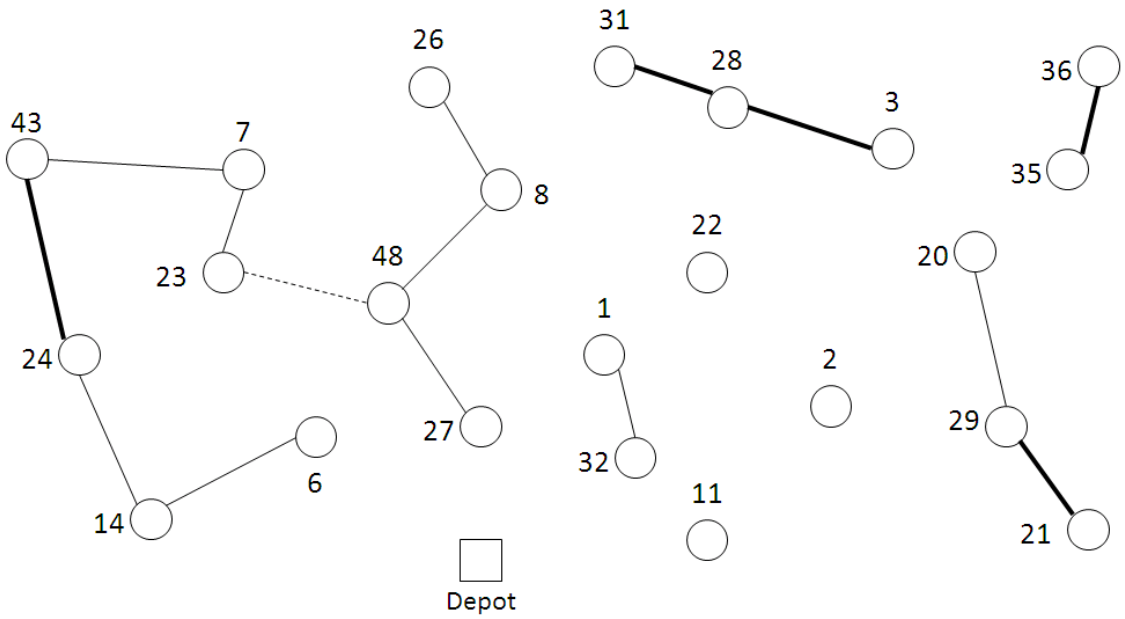

Figure 7.2: Conflicting edges.

be resolved. TSVBA is a learning process so it assumes that the common edges among the solutions in the set as well as the estimation of degree of attractiveness for elite edges are more reliable as the search progresses. Therefore, the conflicting edge first added to the list is removed while the new edge is added to the list. However, the TSVBA is more effective when a routine is used to remove from the list the conflicting edge, including the new one, with the lowest degree of attractiveness. This routine guarantees that common and elite edges are inserted in the solution under construction. In contrast, edges coming from the classical savings list may be ignored by the CW-SD (see Step 3 in Table 7.3) during the construction of solutions.

7.5 Split Savings List

Despite using a variety of local search and local improvement methods, search heuristics can still have problems finding really good solutions to hard problems. A diversification scheme aims to move the search process into new, hopefully unvisited, regions of the solution space. Once in those new regions, the search process resumes. The *BoneRoute* heuristic of Tarantilis and Kiranoudis

(2002) changes the number of solutions in the pool, or the bone-frequency, that must include an edge in their routes in order to vary the level that the user wants to intensify/diversify the search in the solution space. The short-term memory structure employed by TSVBA helps to use different edges each iteration. Although this does diversify the search, TSVBA uses an additional mechanism which consists of modifying the classical savings list of the CW-SD. When TSVBA commences, it uses the classical savings list proposed by Clarke and Wright (1964), adapted to the split demands, but modifies the classical savings list when no more new solutions are found. In the modified savings list, called *split savings list*, the savings associated to the pair of customers (i, j) is given by:

$$s_{ij} = \begin{cases} c_{0i} - c_{0j} - c_{ij}, & c_{0i} > c_{0j} \\ c_{0j} - c_{0i} - c_{ij}, & c_{0i} \leq c_{0j} \end{cases} \tag{7.2}$$

In the top case in Equation 7.2, demand of customer j is split and the initial exclusive route for servicing customer j is maintained after merging the two customers. In the lower case, demand of customer i is split and its initial exclusive route is similarly maintained. If the vehicle capacity is greater than the combined demand of i and j, both routes are merged without splitting any demand. As seen in Equation 7.2, the savings measure depends upon the location of the customers with respect to the depot. In this savings list, the TSVBA assumes a combined demand of customers i and j greater than the vehicle capacity and thus the demand of the customer closest to the depot is split. This assumption helps SDVRP in problems where the customer demands are at least 50% of the vehicle capacity.

7.6 Computational Results

This section presents the computational results for the proposed TSVBA. All tests were conducted on a P4, 2.8GHz, 512MB of RAM. The algorithm was implemented in `C#` and the problems sets of Archetti et al. (2006), Belenguer et al. (2000), and Chen et al. (2007) were used in the experiments. Tested problems are named using the notation `p-aaa-nnn`, where the first field, `p`, is an alphabetical character to identify the publication where the problem is presented (i.e., `a` for Archetti et al.,

b for Belenguer et al., and c for Chen et al.); the second field, aaa, is a string of variable length corresponding to the name of the instance adopted in the original publication; and the third field is a three-digit integer denoting the number of customers excluding the depot. Solutions obtained with TSVBA are compared to those found with algorithms available in the literature. The size of the solutions set is fixed to $k = 10$ (although the number of solutions can be smaller than this number, as discussed previously in Section 7.3.1); the critical angle θ^* of the CA is generated in the range $[0.01 \times \theta^*, 2 \times \theta^*]$. The predefined number of iterations without improvements is set to 5, and the maximum number of iterations is set 40.

Tables 7.4 and 7.5 show the results of TSVBA on the problems of Archetti et al. (2006) and Chen et al. (2007). For each problem, these tables show the size of the set, the worst (z_w) and best (z_b) solutions, the initial and final sets information, the number of iterations required to obtain the final set from its initial version, the running times, and the final solution produced by TSVBA. Running times correspond to those to produce the initial set, the final set after the initial set is produced, and the time required to improve the final set with the Slow-VND and pick its best solution. Note the difference in the set size from Table 7.4 to Table 7.5; this is a consequence of the problem type. Changing the critical angle θ^* in the CA does not produce many different solutions in the problems of Chen et al. (2007) due to the radial distribution of customers around the depot. Since solutions with a common objective function value are not inserted into the set of solutions, the size of the initial set is considerably lower than the expected size of 10 in these instances. Note that the best solution in the final set improves its counterpart in the initial set in most cases (see bold numbers in Tables 7.4 and 7.5). In contrast, the worst solution in the final set is always better than the worst solution in the initial set. This shows how the learning process in the TSVBA helps to improve the population of solutions.

Table 7.6 shows the best solution produced by TSVBA in the problem set of Archetti et al. (2006) and a comparison with other existing approaches. Results in this table and bold fonts show a clear dominance of TSVBA over the iVNDiv of Aleman et al. (2008), the ICA+VND of Aleman et al. (2007), and the scatter search (SS) of Mota et al. (2007) while the memetic algorithm with

Table 7.4: Computational results on problems of Archetti et al. (2006).

			Initial Set			Final Set				Final Solution	
Problem	Demand	Set Size	z_w	z_b	CPU	Iterations	z_w	z_b	CPU	z	CPU
a-01-050		6	577.14	539.29	12.75	16	556.27	**527.67**	13.83	527.67	23.27
a-02-075		9	915.23	859.14	25.30	25	878.75	**853.34**	61.36	853.20	59.13
a-03-100		8	968.72	849.55	66.83	28	854.72	**844.21**	107.72	844.21	120.67
a-04-150		9	1266.30	1087.20	134.28	37	1116.10	**1084.04**	851.28	1079.55	1232.11
a-05-199		10	1576.48	1374.02	365.92	35	1394.78	**1353.65**	1371.91	1339.49	2776.45
a-06-120		9	1257.40	1052.96	362.25	29	1085.96	1052.96	867.63	1051.24	714.31
a-07-100		8	1162.15	820.27	13.52	21	829.20	**819.60**	11.05	819.60	50.77
a-01-050	[0.01-0.10]	4	482.54	468.79	2.69	17	475.06	**466.74**	9.05	466.74	7.95
a-02-075	[0.01-0.10]	6	642.98	614.09	33.64	15	625.00	614.09	18.38	614.09	84.13
a-03-100	[0.01-0.10]	8	816.10	785.92	455.97	31	764.51	**741.91**	447.70	741.60	1040.42
a-04-150	[0.01-0.10]	9	987.30	893.23	372.83	30	928.59	**891.51**	623.02	891.10	1645.11
a-05-199	[0.01-0.10]	10	1203.71	1081.72	1267.83	34	1094.96	**1073.54**	2616.77	1069.24	7330.92
a-06-120	[0.01-0.10]	8	1263.94	990.59	348.73	24	1040.23	990.59	480.41	990.59	1907.20
a-07-100	[0.01-0.10]	5	970.08	676.60	239.50	26	666.40	**658.99**	59.67	658.99	162.58
a-01-050	[0.10-0.30]	10	853.87	773.49	3.45	36	780.47	**753.98**	14.77	753.98	4.95
a-02-075	[0.10-0.30]	8	1198.85	1098.56	10.94	40	1111.57	**1096.96**	50.95	1085.70	35.28
a-03-100	[0.10-0.30]	9	1492.20	1427.66	20.17	31	1433.55	**1420.57**	85.98	1416.35	54.80
a-04-150	[0.10-0.30]	10	2114.40	1965.39	97.33	40	1965.39	**1933.56**	372.22	1929.91	285.53
a-05-199	[0.10-0.30]	9	2749.45	2429.17	220.09	40	2454.50	**2416.62**	828.42	2408.16	495.84
a-06-120	[0.10-0.30]	10	3301.29	2758.72	49.94	31	2840.65	2758.72	225.28	2744.74	188.75
a-07-100	[0.10-0.30]	9	1702.91	1443.35	14.23	31	1470.21	1443.35	42.63	1441.48	41.45
a-01-050	[0.10-0.50]	9	1087.42	1048.78	2.02	32	1043.36	**1026.57**	11.34	1023.24	4.36
a-02-075	[0.10-0.50]	10	1616.61	1495.39	11.86	40	1477.03	**1458.59**	44.84	1458.59	10.95
a-03-100	[0.10-0.50]	10	2041.74	1917.33	17.58	40	1914.89	**1893.90**	85.38	1886.70	42.09
a-04-150	[0.10-0.50]	10	3029.51	2667.66	54.45	40	2674.71	**2651.31**	288.16	2647.17	127.73
a-05-199	[0.10-0.50]	10	3711.32	3299.42	179.47	40	3350.68	3299.42	619.73	3296.69	417.48
a-06-120	[0.10-0.50]	10	4401.14	4082.95	70.23	31	4113.26	**4010.80**	186.11	4010.80	84.19
a-07-100	[0.10-0.50]	10	2602.56	2026.14	11.47	40	2072.73	**2011.87**	40.84	2010.00	32.19

Continued on next page

Table 7.4: Computational results on problems of Archetti et al. (2006) (*Continued*).

			Initial Set			Final Set				Final Solution	
Problem	Demand	Set Size	z_w	z_b	CPU	Iterations	z_w	z_b	CPU	z	CPU
a-01-050	[0.10-0.90]	9	1589.20	1546.34	1.66	40	1553.21	**1532.91**	15.19	1530.81	2.27
a-02-075	[0.10-0.90]	10	2259.94	2193.23	6.72	40	2206.60	**2178.09**	42.02	2164.74	13.08
a-03-100	[0.10-0.90]	9	3064.75	2893.77	16.89	27	2928.20	2893.77	70.27	2874.86	38.13
a-04-150	[0.10-0.90]	10	4472.47	4179.40	71.88	40	4201.91	4179.40	285.41	4151.90	94.67
a-05-199 †	[0.10-0.90]	10	6316.18	5066.24	3.72	19	5706.29	5066.24	104.91	5066.24	0.00
a-06-120	[0.10-0.90]	10	7220.12	6324.80	140.41	25	6467.80	6324.80	144.98	6308.76	133.59
a-07-100	[0.10-0.90]	10	3767.95	3157.48	9.38	40	3212.29	3157.48	57.92	3157.48	30.28
a-01-050	[0.30-0.70]	10	1605.21	1537.72	1.41	40	1537.72	**1506.64**	14.73	1505.38	2.95
a-02-075	[0.30-0.70]	10	2305.95	2217.59	6.58	40	2213.23	**2189.16**	38.25	2182.33	10.34
a-03-100	[0.30-0.70]	10	3146.44	2941.41	17.72	40	2960.63	**2937.86**	90.09	2929.29	27.03
a-04-150	[0.30-0.70]	10	4472.47	4179.40	69.50	40	4201.91	4179.40	285.80	4151.90	94.05
a-05-199 †	[0.30-0.70]	10	6493.33	5281.55	3.82	19	5894.35	5281.55	115.23	5281.55	0.00
a-06-120	[0.30-0.70]	10	7697.91	6558.18	130.28	28	6610.42	**6536.24**	179.14	6511.08	127.38
a-07-100	[0.30-0.70]	10	3983.38	3200.62	8.45	19	3314.81	3200.62	32.98	3200.62	54.95
a-01-050	[0.70-0.90]	9	2303.72	2222.11	1.78	35	2259.90	2222.11	19.92	2219.32	2.70
a-02-075	[0.70-0.90]	10	3438.25	3278.33	6.80	40	3367.98	3278.33	56.08	3278.33	23.39
a-03-100	[0.70-0.90]	9	4558.68	4442.76	15.91	40	4507.72	4442.76	144.55	4435.56	25.09
a-04-150	[0.70-0.90]	10	6706.85	6425.31	58.06	33	6551.76	6425.31	376.91	6416.12	243.97
a-05-199 †	[0.70-0.90]	10	8754.87	8333.61	5.43	19	8709.99	8333.61	147.69	8333.61	0.00
a-06-120 †	[0.70-0.90]	10	11931.89	10186.06	1.99	25	10692.57	10186.06	28.33	10186.06	0.00
a-07-100	[0.70-0.90]	10	5614.21	5000.11	14.30	32	5088.87	5000.11	71.41	4996.88	67.22

z_w, z_b, and z denote worst, best, and final objective function value obtained, respectively.

CPU denotes running time in seconds on a P4, 2.8GHz, 512MB (0.0 implies no improvement applied to the final set).

†To avoid memory overflow, TSVBA does not use VND to improve any of the constructed solutions.

Table 7.5: Computational results on problems of Chen et al. (2007).

		Initial Set			Final Set				Final Solution	
Problem	Set Size	z_w	z_b	CPU	Iterations	z_w	z_b	CPU	z	CPU
c-SD01-008	2	29721.35	22828.43	0.05	5	25886.35	22828.43	0.03	22828.43	0.00
c-SD02-016	1	70828.43	70828.43	0.30	6	70828.43	70828.43	0.08	70828.43	0.02
c-SD03-016	3	49543.52	43058.22	0.11	18	44292.76	43058.22	0.33	43058.22	0.03
c-SD04-024	3	71830.11	63583.52	0.59	23	64062.14	**63104.90**	1.20	63104.90	0.08
c-SD05-032	2	143609.65	139056.83	0.41	8	140528.60	139056.83	0.48	139056.83	0.13
c-SD06-032	3	90543.08	83124.14	0.42	27	83881.10	83124.14	3.16	83124.14	0.14
c-SD07-040	1	364000.00	364000.00	0.41	8	364000.00	364000.00	0.30	364000.00	0.09
c-SD08-048	1	506828.43	506828.43	0.86	8	506828.43	506828.43	0.44	506828.43	0.14
c-SD09-048	2	212722.07	207102.79	3.22	17	208585.62	207102.79	7.13	207102.79	0.36
c-SD10-064	2	290016.47	274831.72	2.63	13	275250.18	274831.72	7.30	274783.08	0.89
c-SD11-080	1	1328000.01	1328000.01	1.91	10	1328000.01	1328000.01	2.38	1328000.01	0.41
c-SD12-080	2	727997.00	727482.00	2.00	31	722339.50	**721362.34**	15.42	721362.34	0.84
c-SD13-096	2	1017178.77	1011057.51	2.86	12	1012951.58	1011057.51	4.81	1011057.51	1.20
c-SD14-120	2	1089349.80	1082472.32	11.69	33	1080401.49	**1080287.10**	65.11	1080287.10	2.31
c-SD15-144	2	1523239.25	1515584.56	18.20	16	1515584.56	**1515345.27**	28.61	1515345.27	3.20
c-SD16-144	3	363422.72	347290.79	45.25	33	345079.16	**344643.28**	184.80	344643.28	7.59
c-SD17-160	2	2655992.75	2655477.75	12.13	34	2650906.21	**2649358.09**	68.13	2649356.48	7.27
c-SD18-160	2	1444513.63	1437428.11	38.03	20	1434948.74	**1434787.07**	147.34	1432304.04	27.95
c-SD19-192	3	2029507.42	2019119.29	25.66	38	2018269.12	**2015709.56**	152.42	2015709.56	11.95
c-SD20-240	2	3981348.58	3974471.10	52.42	38	3972400.27	**3972285.88**	324.23	3972285.88	11.02
c-SD21-288	3	1192375.34	1147667.29	680.41	35	1151770.98	1147667.29	1740.83	1145875.54	111.56

z_w, z_b, and z denote worst, best, and final objective function value obtained, respectively.

CPU denotes running time in seconds on a P4, 2.8GHz, 512MB. Zero implies less than 0.01 seconds.

Table 7.6: Computational results of TSVBA on instances of Archetti et al. (2006).

			TSVBA			iVNDiv		ICA+VND		SS		MA|PM	
Problem	Demand	m'	m	z	CPU[(a)]	z	CPU[(a)]	z	CPU[(a)]	z	CPU[(b)]	z	CPU[(c)]
a-01-050		5	5	527.67	49.84	524.61	54.91	**540.82**	10.89	**531.02**	24.80	524.61	8.53
a-02-075		10	11	853.20	145.78	851.24	83.28	**880.28**	9.81	839.75	61.66	823.89	35.72
a-03-100		8	8	844.21	295.22	**852.74**	319.33	**854.13**	43.50	835.82	108.80	829.44	34.59
a-04-150		12	12	1079.55	2217.67	1074.11	1361.16	**1088.91**	129.23	1056.92	261.28	1042.37	103.69
a-05-199		16	17	1339.49	4514.28	**1368.67**	3284.64	**1390.55**	534.83	**1340.44**	352.31	1311.59	353.84
a-06-120		7	7	1051.24	1944.19	**1201.83**	3414.41	**1223.28**	257.30	1042.97	131.34	1041.20	50.92
a-07-100		10	10	819.60	75.33	**824.78**	126.08	**824.82**	21.02	**820.92**	108.41	819.56	42.89
a-01-050	[0.01-0.10]	3	3	466.74	19.69	**471.92**	33.70	**473.22**	4.52	460.79	26.86	460.79	12.38
a-02-075	[0.01-0.10]	4	4	614.09	136.14	597.46	303.77	**617.65**	51.28	602.67	68.80	600.06	18.75
a-03-100	[0.01-0.10]	5	5	741.60	1944.09	**745.35**	2194.23	**789.16**	415.47	729.67	125.06	726.81	37.12
a-04-150	[0.01-0.10]	8	8	891.10	2640.95	**891.98**	3461.44	**893.49**	666.20	883.05	352.09	875.61	100.27
a-05-199	[0.01-0.10]	10	10	1069.24	11215.52	**1073.55**	15505.22	**1079.04**	3750.44	1039.51	963.84	1018.71	356.22
a-06-120	[0.01-0.10]	6	6	990.59	2736.34	**1087.80**	3952.67	**1101.14**	341.59	979.57	163.28	976.57	72.98
a-07-100	[0.01-0.10]	5	6	658.99	461.75	**673.54**	1207.42	**673.54**	222.42	633.80	80.56	649.73	34.97
a-01-050	[0.10-0.30]	10	10	753.98	23.17	**766.19**	19.77	**777.75**	1.59	**769.60**	26.31	751.41	10.22
a-02-075	[0.10-0.30]	15	15	1085.70	97.17	**1099.47**	73.05	**1099.47**	13.19	1074.01	86.02	1074.46	34.14
a-03-100	[0.10-0.30]	20	20	1416.35	160.95	**1425.90**	190.53	**1452.52**	34.09	**1416.48**	98.00	1392.85	78.06
a-04-150	[0.10-0.30]	29	30	1929.91	755.08	**1978.01**	878.55	**1978.01**	164.19	**1974.70**	10.06	1878.71	147.89
a-05-199	[0.10-0.30]	38	39	2408.16	1544.36	**2464.65**	1457.16	**2502.54**	248.83	**2435.08**	19.11	2340.14	347.14
a-06-120	[0.10-0.30]	23	24	2744.74	463.97	**2806.92**	558.56	**2806.92**	54.25	**2783.10**	11.33	2720.38	144.19
a-07-100	[0.10-0.30]	20	20	1441.48	98.31	1428.27	123.00	1428.27	22.56	1423.49	151.25	1417.28	43.27
a-01-050	[0.10-0.50]	15	15	1023.24	17.72	**1039.89**	18.16	**1045.93**	2.81	**1025.91**	3.84	988.31	12.49
a-02-075	[0.10-0.50]	22	23	1458.59	67.66	**1478.67**	67.80	**1503.02**	11.25	**1484.62**	6.09	1413.80	37.38
a-03-100	[0.10-0.50]	29	29	1886.70	145.05	**1956.13**	154.47	**1957.55**	25.16	**1926.15**	7.55	1845.30	28.39
a-04-150	[0.10-0.50]	43	45	2647.17	470.34	**2671.62**	625.83	**2685.33**	111.66	**2649.97**	16.17	2561.65	224.89
a-05-199	[0.10-0.50]	56	56	3296.69	1216.69	**3411.38**	2173.84	**3450.84**	339.36	**3310.71**	20.64	3191.25	436.20
a-06-120	[0.10-0.50]	34	35	4010.80	340.53	**4026.53**	358.56	**4085.36**	40.53	3996.29	63.80	3934.39	163.14
a-07-100	[0.10-0.50]	29	30	2010.00	84.50	2007.11	107.47	**2046.15**	11.92	**2022.30**	41.23	1994.59	51.31

Continued on next page

Table 7.6: Computational results of TSVBA on instances of Archetti et al. (2006) (*Continued*).

			TSVBA			iVNDiv		ICA+VND		SS		MA|PM	
Problem	Demand	m'	m	z	CPU[(a)]	z	CPU[(a)]	z	CPU[(a)]	z	CPU[(b)]	z	CPU[(c)]
a-01-050	[0.10-0.90]	25	26	1530.81	19.11	1522.43	16.36	**1547.32**	2.83	**1580.77**	3.91	1467.06	21.42
a-02-075	[0.10-0.90]	37	39	2164.74	61.81	**2200.51**	71.11	**2212.93**	10.80	**2233.08**	6.64	2102.58	46.11
a-03-100	[0.10-0.90]	48	48	2874.86	125.28	2865.86	126.52	**2925.13**	19.00	**2932.34**	9.16	2780.95	84.38
a-04-150	[0.10-0.90]	73	74	4151.90	451.95	**4165.18**	671.36	**4192.50**	141.27	**4185.68**	25.03	4045.87	244.91
a-05-199 †	[0.10-0.90]	93	93	5066.24	108.63	**5184.57**	3650.59	**5192.06**	662.77	**5085.64**	71.09	4941.22	725.69
a-06-120	[0.10-0.90]	56	56	6308.76	418.98	**6364.87**	458.91	**6483.06**	42.00	**6361.46**	15.86	**6318.37**	196.14
a-07-100	[0.10-0.90]	48	48	3157.48	97.58	3156.31	96.98	**3178.28**	12.89	**3187.44**	9.08	3113.72	52.13
a-01-050	[0.30-0.70]	25	26	1505.38	19.09	**1540.39**	15.33	**1557.52**	2.20	**1568.04**	4.25	1477.01	24.53
a-02-075	[0.30-0.70]	37	38	2182.33	55.17	**2238.98**	80.30	**2241.59**	11.28	**2228.90**	7.14	2132.16	51.78
a-03-100	[0.30-0.70]	49	49	2929.29	134.84	**2941.64**	103.94	**2945.19**	15.14	**2986.33**	10.36	2858.87	100.16
a-04-150	[0.30-0.70]	73	74	4151.90	449.34	**4165.18**	675.39	**4192.50**	143.05	**4185.68**	19.38	4045.87	244.86
a-05-199 †	[0.30-0.70]	96	96	5281.55	119.04	**5363.65**	3026.22	**5366.06**	349.97	5265.01	120.28	5155.36	749.94
a-06-120	[0.30-0.70]	58	58	6511.08	436.80	**6545.50**	469.17	**6591.40**	59.20	6481.09	17.16	6424.71	271.39
a-07-100	[0.30-0.70]	49	50	3200.62	96.39	**3225.63**	110.05	**3318.08**	13.69	**3248.76**	9.73	3155.69	91.31
a-01-050	[0.70-0.90]	40	41	2219.32	24.41	2215.34	18.70	2215.34	2.59	**2312.48**	4.13	2154.35	22.91
a-02-075	[0.70-0.90]	60	60	3278.33	86.27	**3304.24**	58.05	**3341.26**	10.25	**3387.86**	7.66	3200.35	27.48
a-03-100	[0.70-0.90]	80	80	4435.56	185.55	4429.21	94.98	**4455.14**	14.31	**4580.98**	12.06	4312.95	55.75
a-04-150	[0.70-0.90]	119	119	6416.12	678.94	**6482.11**	584.84	**6513.36**	93.78	**6479.46**	131.91	6267.48	401.62
a-05-199 †	[0.70-0.90]	158	158	8333.61	153.12	8329.55	2124.66	**8368.35**	460.89	8323.72	165.28	8081.58	571.70
a-06-120 †	[0.70-0.90]	95	95	10186.06	30.32	**10302.16**	636.72	**10302.16**	59.28	10158.32	20.17	10063.47	298.08
a-07-100	[0.70-0.90]	80	80	4996.88	152.92	**5028.78**	178.19	**5058.76**	20.70	**5065.26**	9.19	4919.48	180.11

z denotes objective function value obtained

m denotes number of vehicles in TSVBA solutions. m' denotes number of vehicles in iVNDiv, ICA+VND, SS, and MA|PM solutions.

[(a)]P4, 512MB, 2.8 GHz; [(b)]P4, 1.0GB, 2.4 GHz; [(c)]PC 3.0 GHz.

†To avoid memory overflow, TSVBA does not use VND to improve any of the constructed solutions.

population management (MA|PM) of Boudia et al. (2007) produces the best known solutions in this problem set; TSVBA improves MA|PM in one case only. Results obtained with the tabu search (Splitabu) of Archetti et al. (2006) are not included in the table because there is no evidence about the equivalence of the tested problems, in spite of the fact that the problems were generated with the same code. In the case of the hybrid algorithm (EMIP+VRTR) of Chen et al. (2007), results were omitted because they solved 30 instances for each problem and published just the median solution values.

Except for iVNDiv, TSVBA is the approach with the largest processing times among those presented. TSVBA is usually faster than iVNDiv which shows the benefits of using both closest customers and neighboring routes to reduce the complexity of the variable neighborhood descent without deteriorating solution quality. It is important to highlight the fact that TSVBA produces more solutions than iVNDiv. For example, in problem a-04-150 with demands in the range [0.10-0.30] TSVBA produces 60 solutions (i.e., 10 forming the initial solution set, 40 to obtain the final solution set, and 10 after improving the final solution set, as shown in Table 7.4) while iVNDiv produces only 5 solutions (Aleman et al., 2008). The running times of TSVBA can be reduced by coding the algorithm more efficiently and using a programming language with lower resource requirements, but this is left for future work.

Tables 7.7 and 7.8 show the computational results on some TSPLIB instances and the random problems of Belenguer et al. (2000), respectively, using bold fonts to denote cases where TSVBA performs better. In the problems shown in Table 7.7, TSVBA finds better solutions than iVNDiv in most cases, improves the upper bound produced by the column generation approach of Jin et al. (2008), and dominates both ICA+VND and the branch-and-price (B&P) of Liu (2005). Note that the solutions values are calculated using integer inter-node distance in order to compare to the bounds of Belenguer et al. (2000) and the values obtained with (MA|PM). In the random problems of Belenguer et al. (2000) shown in Table 7.8, TSVBA improves the upper bounds of Belenguer et al. (2000) in most cases and dominates the solution values obtained with iVNDiv, ICA+VND, B&P, and the column generation of Jin et al. (2008). Both MA|PM and the hybrid algorithm

Table 7.7: Computational results of TSVBA on some TSPLIB instances.

	TSVBA		Belenguer et al. (2000)		MA|PM	
Problem	$z^{(a)}$	CPU$^{(b)}$	UB$^{(a)}$	LB	$z^{(a)}$	CPU$^{(c)}$
b-eil22-021	375	2.58	375	375.00	375	4.11
b-eil23-022	570	1.59	569	569.00	569	5.47
b-eil30-029	503	7.45	**510**	508.00	503	5.70
b-eil33-032	844	8.38	835	833.00	835	5.19
b-eil51-050	526	49.84	521	511.57	521	7.28
b-eilA76-075	847	145.78	832	782.70	828	35.94
b-eilB76-075	1027	91.36	1023	937.47	1019	13.09
b-eilC76-075	754	151.13	735	706.01	738	14.75
b-eilD76-075	691	122.52	683	659.43	682	23.12
b-eilA101-100	834	295.22	817	793.48	818	25.25
b-eilB101-100	1104	173.13	1077	1005.85	1082	21.81

Continued on next page

(EMIP+VRTR) of Chen et al. (2007), however, perform better than TSVBA.

Finally, Table 7.9 shows the results on the problems of Chen et al. (2007). TSVBA performs well on these problems and improves over all the presented approaches. In this set, it is also evident the difference in processing times, with TSVBA the fastest among all presented algorithms. In all cases, TSVBA uses the minimum possible fleet size, a key benefit in actual vehicle routing problems.

7.7 Conclusions

This chapter presented a learning procedure based on a population of solutions. The learning strategy consists of generating an initial set of solutions and finding common attributes among those to construct new solutions. New solutions contain those common attributes as well as new characteristics that can lead to better solutions. The solution set is evolved by replacing solutions with large objective function values with new solutions having lower values. The new solution characteristics are evaluated for each solution in the set to determine their degree of attractiveness and then these characteristics are included in an elite list of attributes if they are found attractive enough. The

Table 7.7: Computational results of TSVBA on some TSPLIB instances (*Continued*).

	TSVBA		iVNDiv		ICA+VND		B&P		Jin et al. (2008)		
Problem	z	CPU[(b)]	z	CPU[(b)]	z	CPU[(b)]	UB	LB	UB	LB	CPU[(d)]
b-eil22-021	375.28	2.58	375.28	4.19	375.28	0.70	**376.00**	373.60	-	-	-
b-eil23-022	569.75	1.59	569.75	3.42	569.75	0.59	**608.00**	564.30	-	-	-
b-eil30-029	505.01	7.45	**512.72**	14.47	**521.48**	2.22	**515.30**	**507.20**	-	-	-
b-eil33-032	843.64	8.38	**853.10**	14.03	**870.35**	1.86	**873.40**	830.20	-	-	-
b-eil51-050	527.67	49.84	524.61	54.91	**540.82**	10.89	**558.50**	507.60	-	-	-
b-eilA76-075	853.20	145.78	851.24	83.28	**880.28**	9.81	**900.70**	800.30	-	-	-
b-eilB76-075	1034.21	91.36	**1059.57**	79.00	**1059.57**	16.42	**1163.10**	965.70	**1063.75**	981.14	45084.00
b-eilC76-075	761.55	151.13	753.29	148.20	758.49	26.25	**809.30**	711.20	-	-	-
b-eilD76-075	695.96	122.52	**699.35**	140.83	**719.41**	29.92	**768.80**	652.30	-	-	-
b-eilA101-100	844.21	295.22	**852.74**	319.33	**854.13**	43.50	**910.20**	797.50	-	-	-
b-eilB101-100	1112.15	173.13	**1139.27**	185.84	**1142.02**	31.13	**1174.10**	1013.90	-	-	-

z denotes objective function value obtained. CPU denotes running time in seconds.

TSVBA and MA|PM uses one more vehicle than the minimum fleet size on instance b-eil30-029.

[(a)] Objective function value obtained with euclidean distances truncated to the nearest integer.

[(b)]P4, 512MB, 2.8 GHz; [(c)]PC 3.0 GHz; [(d)]P4, 2GB, 2.8 GHz.

Table 7.8: Computational results of TSVBA on the random problems of Belenguer et al. (2000).

	TSVBA		Belenguer et al. (2000)		MA|PM	
Problem	$z^{(a)}$	CPU$^{(b)}$	UB$^{(a)}$	LB	$z^{(a)}$	CPU$^{(c)}$
b-S51D1-050	465	13.56	458	454	458	8.77
b-S51D2-050	715	31.66	**726**	676.63	707	7.44
b-S51D3-050	966	18.75	**972**	905.22	945	7.84
b-S51D4-050	1621	19.77	**1677**	1520.67	1578	11.98
b-S51D5-050	1357	15.39	**1440**	1272.86	1351	16.72
b-S51D6-050	2228	14.38	**2327**	2113.03	2182	9.92
b-S76D1-075	606	252.28	594	584.87	592	15.23
b-S76D2-075	1124	60.44	**1147**	1020.32	1089	30.50
b-S76D3-075	1466	51.13	**1474**	1346.29	1427	12.89
b-S76D4-075	2170	53.56	**2257**	2011.64	2117	8.76
b-S101D1-100	741	860.31	716	700.56	717	49.75
b-S101D2-100	1398	219.52	1393	†1270.97	1372	31.72
b-S101D3-100	1936	132.19	**1975**	†1739.66	1891	33.98
b-S101D5-100	2897	131.16	**2915**	†2630.43	2854	18.66

Continued on next page

search is intensified by using the common and elite attributes in the construction of new solutions, while the diversification is based on the use of a short-term memory structure and a modified cost function for the evaluation of candidates. Once a final solution set is obtained after a certain number of iterations, solutions are further improved with a variable neighborhood descent. The final solution is the best one found overall. The proposed learning procedure was tested on benchmark instances and performed well when its solutions were compared to those reported in the literature.

Table 7.8: Computational results of TSVBA on the random problems of Belenguer et al. (2000) (*Continued*).

	TSVBA		iVNDiv		ICA+VND		B&P		EMIP+VRTR		Jin et al. (2008)		
Problem	z	CPU[(b)]	z	CPU[(b)]	z	CPU[(b)]	UB	LB	z	CPU[(d)]	UB	LB	CPU[(e)]
b-S51D1-050	468.79	13.56	**471.92**	40.53	**473.22**	4.53	**513.90**	449.90	-	-	-	-	-
b-S51D2-050	718.69	31.66	**731.01**	28.34	**732.38**	4.05	**1296.50**	556.70	-	-	**723.37**	694.98	5978.00
b-S51D3-050	969.78	18.75	**1001.22**	14.70	**1001.22**	2.50	**986.00**	956.00	-	-	968.85	922.72	607.00
b-S51D4-050	1628.20	19.77	**1680.66**	16.53	**1708.00**	2.89	**1654.00**	1623.00	1586.50	201.74	**1657.61**	1505.35	260.00
b-S51D5-050	1362.19	15.39	**1389.40**	13.94	**1404.54**	1.80	**1434.00**	**1416.00**	1355.50	201.62	**1439.92**	1297.46	46.00
b-S51D6-050	2236.16	14.38	2218.23	16.83	2230.06	2.27	**2316.00**	**2270.00**	2197.80	301.90	**2300.21**	2108.59	243.00
b-S76D1-075	613.70	252.28	606.47	476.27	610.23	63.55	-	-	-	-	-	-	-
b-S76D2-075	1128.15	60.44	**1143.36**	46.94	**1169.80**	7.73	-	-	-	-	**1185.72**	1066.17	12806.00
b-S76D3-075	1472.92	51.13	**1490.08**	53.34	**1490.08**	12.23	-	-	-	-	**1504.94**	1397.43	2030.00
b-S76D4-075	2180.13	53.56	2173.61	51.84	**2220.87**	6.91	**2205.00**	2178.00	2136.40	601.92	**2219.07**	2019.91	1813.00
b-S101D1-100	749.93	860.31	749.19	2125.58	**765.48**	210.36	-	-	-	-	-	-	-
b-S101D2-100	1409.03	219.52	**1443.44**	217.91	**1444.96**	26.20	-	-	-	-	**1474.51**	1349.77	47658.00
b-S101D3-100	1947.62	132.19	**1988.78**	146.61	**1990.28**	27.84	-	-	-	-	**2012.86**	1837.33	7959.00
b-S101D5-100	2910.71	131.16	**2984.48**	104.05	**2999.31**	18.36	-	-	2846.20	645.99	**2954.96**	2725.50	847.00

†Earlier termination of the algorithm due to memory overflow (Belenguer et al., 2000). z denotes objective function value obtained.

CPU denotes running time in seconds.

[(a)] Obtained with euclidean distances truncated to the nearest integer.

[(b)]P4, 512MB, 2.8 GHz; [(c)]PC 3.0 GHz; [(d)]P4, 512MB, 1.7 GHz; [(e)]P4, 2GB, 2.8 GHz.

Table 7.9: Computational results of TSVBA on instances of Chen et al. (2007).

		TSVBA			iVNDiv		ICA+VND		EMIP + VRTR	
Problem	m'	m	z	CPU[(a)]	z	CPU[(a)]	z	CPU[(a)]	z	CPU[(b)]
c-SD01-008	6	6	228.28		228.28	0.19	228.28	0.06	228.28	0.70
c-SD02-016	12	12	708.28	0.02	708.28	1.48	708.28	0.22	**714.40**	54.40
c-SD03-016	12	12	430.58	0.03	430.58	0.58	430.58	0.17	**430.61**	67.30
c-SD04-024	18	18	631.05	0.08	**635.84**	2.31	**635.84**	0.55	**631.06**	400.00
c-SD05-032	24	24	1390.57	0.13	1390.57	5.55	**1390.57**	0.69	**1408.12**	402.70
c-SD06-032	24	24	831.24	0.14	**831.24**	2.95	831.24	0.94	831.21	408.30
c-SD07-040	30	30	3640.00	0.09	3640.00	8.13	3640.00	1.03	**3714.40**	403.20
c-SD08-048	36	36	5068.28	0.14	5068.28	11.91	5068.28	1.75	**5200.00**	404.10
c-SD09-048	36	36	2071.03	0.36	2071.03	19.73	**2071.03**	2.91	2059.84	404.30
c-SD10-064	48	48	2747.83	0.89	2742.84	33.27	2747.83	3.58	**2749.11**	400.00
c-SD11-080	60	60	13280.00	0.41	13280.00	35.16	13280.00	3.97	**13612.12**	400.10
c-SD12-080	60	60	7213.62	0.84	**7265.70**	43.13	**7279.97**	4.00	**7399.06**	408.30
c-SD13-096	72	72	10110.58	1.20	10110.58	50.97	**10110.58**	5.80	**10367.06**	404.50
c-SD14-120	90	90	10802.87	2.31	**10829.25**	141.77	**10893.50**	15.49	**11023.00**	5021.70
c-SD15-144	108	108	15153.45	3.20	**15168.28**	191.66	**15168.28**	18.33	**15271.77**	5042.30
c-SD16-144	108	108	3446.43	7.59	**3580.07**	2120.14	**3635.27**	39.71	**3449.05**	5014.70
c-SD17-160	120	120	26493.56	7.27	**26556.13**	179.61	**26559.93**	17.42	**26665.76**	5023.60
c-SD18-160	120	120	14323.04	27.95	**14372.80**	366.14	**14440.59**	40.38	**14546.58**	5028.60
c-SD19-192	144	144	20157.10	11.95	**20188.62**	330.06	**20191.19**	27.64	**20559.21**	5034.20
c-SD20-240	180	180	39722.86	11.02	**39803.13**	633.33	**39813.49**	63.18	**40408.22**	5053.00
c-SD21-288	216	216	11458.76	111.56	**11682.09**	9387.55	**11799.60**	738.49	**11491.67**	5051.00

z denotes objective function value obtained. Values scaled from Table 7.5 for comparison purposes with EMIP+VRTR.

m denotes number of vehicles in TSVBA solutions. m' denotes number of vehicles in iVNDiv and ICA+VND solutions.

No fleet size published for EMIP + VRTR.

[(a)] P4, 512MB, 2.8 GHz; [(b)] P4, 512MB, 1.7 GHz.

Part III

Summary

Chapter 8

Summary

8.1 Summary

This dissertation presented a study on the SDVRP that includes a thorough literature review covering representative research on the VRP, the SDVRP, and the VRPSC. Various solution methods are proposed to efficiently solve the SDVRP. Proposed techniques include a constructive approach based on a novel route angle control mechanism that help to produce thin routes with the minimum possible fleet size, an iterative constructive approach that uses adaptive memory concepts to modify the rules of the constructive approach, a variable neighborhood descent that uses operators specific to the SDVRP, a new diversification scheme based on concentric rings centered at the depot that partitions the original problem and solves resulting subproblems independently, and a tabu search with vocabulary building that creates an initial solution set to find attractive solution attributes and then generate new solutions to evolve the set. An empirical analysis is performed to compare the proposed techniques to existing solution techniques available in the literature. The results obtained demonstrate their effectiveness on the tested problems.

8.2 Contributions

The SDVRP is a variant of the classical VRP that allows to use multiple vehicles to supply the demand of single customers. The SDVRP has received little attention from researchers. According to the literature reviewed, only a few approaches prior to this study have been proposed to solve the problem. This dissertation provides multiple contributions. Part I provides a focused yet thorough literature review on the VRP, SDVRP, and VRPSC. The review includes problem properties, models, and solution algorithms. A new problem classification scheme is also presented in Part I useful for categorizing modern routing problems. This classification scheme is based on three criteria: staticism/dynamism of the problem parameters, the knowledge of the information relevant to the design of its solution, and the method to model the unknown data. Part II presents new algorithmic contributions, including: 1) a novel constructive approach, an iterative constructive approach that uses adaptive memory concepts, and a variable neighborhood descent found in Aleman et al. (2007); 2) a new solution diversification scheme found in Aleman et al. (2008) based on concentric rings centered at the depot that partitions the original problem and solves the resulting problems using a constructive approach; and 3) a population-based search approach, called tabu search with vocabulary building approach, that constructs an initial solution set and then uses the set of solutions to find attractive solution attributes with which to construct new solutions and evolve the set.

8.3 Future Research

The route angle control mechanism proposed in Chapter 5 is easy to implement and looks useful to solve the SDVRP, specifically in problems with large customer demands. Although the constructive approach tends to produce non-crossing routes, optimal solutions can have crossing as well as inner routes so it may be helpful to guide the CA to design such routes. Other local searches that have produced good results on the VRP, such as exchanging sequences of customer between routes, can be investigated for robustness and to find better solutions to problems with small customer demands.

Approaches presented in Part II can be implemented more efficiently and using a programming language with lower resource requirements, such as C++.

Bibliography

Aleman, R. E. and R. R. Hill (2008, December). A tabu search with vocabulary building approach for the vehicle routing problem with split demands. *Submitted to Computers & Operations Research*.

Aleman, R. E., X. Zhang, and R. R. Hill (2007, November). An adaptive memory algorithm for the split delivery vehicle routing problem. *Accepted for publication in the Journal of Heuristics*.

Aleman, R. E., X. Zhang, and R. R. Hill (2008, June). A ring-based diversification scheme for routing problems. *To Appear in the International Journal of Mathematics in Operational Research*.

Altinkemer, K. and B. Gavish (1991, May). Parallel savings based heuristics for the delivery problem. *Operations Research 39*(3), 456–469.

Ambrosino, D. and A. Sciomachen (2007, January). A food distribution network problem: a case study. *IMA Journal of Management Mathematics 18*(1), 33–53.

Archetti, C., A. Hertz, and M. G. Speranza (2006, February). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science 40*(1), 64–73.

Archetti, C., R. Mansini, and M. G. Speranza (2001). The split delivery vehicle routing problem with small capacity. *To appear in Transportation Science*.

Archetti, C., R. Mansini, and M. G. Speranza (2005, May). Complexity and reducibility of the skip delivery problem. *Transportation Science 39*(2), 182–187.

Archetti, C., M. W. P. Savelsbergh, and M. G. Speranza (2006, May). Worst-case analysis for split delivery vehicle routing problems. *Transportation Science 40*(2), 226–234.

Archetti, C., M. W. P. Savelsbergh, and M. G. Speranza (2008). To split or not to split: That is the question. *Transportation Research. Part E 44*(1), 114–123.

Archetti, C. and M. G. Speranza (2007, August). An overview on the split delivery vehicle routing problem. In *Operations Research Proceedings 2006*, Volume 2006 of *Operations Research Proceedings*, pp. 123–127. Springer Berlin / Heidelberg.

Archetti, C., M. G. Speranza, and M. W. P. Savelsbergh (2008, February). An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science 42*(1), 22–31.

Baker, B. M. and M. A. Ayechew (2003, April). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research 30*(5), 787–800.

Baldacci, R., E. Hadjiconstantinou, and A. Mingozzi (2004, September). An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research 52*(5), 723–738.

Beasley, J. E. (1983). Route-first cluster-second methods for vehicle routing. *Omega 11*(4), 403–408.

Belenguer, J., M. Martinez, and E. Mota (2000, Sept/Oct). A lower bound for the split delivery vehicle routing problem. *Operations Research 48*(5), 801–810.

Belfiore, P., H. Tsugunobu, and Y. Yoshizaki (2006, Sept/Dec). Scatter search for heterogeneous fleet vehicle routing problems with time windows and split deliveries. *Produção: uma publicação da Associação Brasileira de Engenharia de Produção 16*(3), 455–469.

Belfiore, P., H. Tsugunobu, and Y. Yoshizaki (2008, September). Scatter search for vehicle routing problem with time windows and split deliveries. In T. Caric and H. Gold (Eds.), *Vehicle Routing Problem*, Chapter 1, pp. 1–14. Vienna, Austria: IN-TECH.

Bell, W. J., L. M. Dalberto, M. L. Fisher, A. J. Greenfield, R. Jaikumar, P. Kedia, R. G. Mack, and P. P. J. (1983, December). Improving the distribution of industrial gases with an on-line

computerized routing and scheduling optimizer. *Interfaces 13*(6), 4–23.

Bent, R. W. and P. Van Hentenryck (2004, Nov/Dec). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research 52*(6), 977–987.

Berger, J. and M. Barkaoui (2003, December). A new hybrid genetic algorithm for the capacitated vehicle routing problem. *The Journal of the Operational Research Society 54*(12), 1254–1262.

Berman, O. and R. C. Larson (2001, May). Deliveries in an inventory /routing problem using stochastic dynamic programming. *Transportation Science 35*(2), 192–213.

Berman, O. and D. Simchi-Levi (1988, May). Finding the optimal a priori tour and location of a traveling salesman with nonhomogeneous customers. *Transportation Science 22*(2), 148–154.

Bertsimas, D. J. (1988). *Probabilistic combinatorial optimization problems*. Ph. D. thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA.

Bertsimas, D. J. (1989, August). Traveling salesman facility location problem. *Transportation Science 23*(3), 184–191.

Bertsimas, D. J. and L. H. Howell (1993). Further results on the probabilistic traveling salesman problem. *European Journal of Operational Research 65*(1), 68–95.

Bertsimas, D. J., P. Jaillet, and A. R. Odoni (1990, Nov/Dec). A priori optimization. *Operations Research 38*(6), 1019–1033.

Bertsimas, D. J. and G. Van Ryzin (1991, Jul/Aug). A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research 39*(4), 601–615.

Bodin, L. and B. Golden (1981). Classification in vehicle routing and scheduling. *Networks 11*(2), 97–108.

Bolduc, M.-C., G. Laporte, J. Renaud, and F. F. Boctor (2008, May). A tabu search heuristic for the split delivery vehicle routing problem with production and demand calendars. Tech-

nical Report CIRRELT-2008-13, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal.

Boudia, M., C. Prins, and M. Reghioui (2007, September). An effective memetic algorithm with population management for the split delivery vehicle routing problem. In *Hybrid Metaheuristics*, Volume 4771 of *Lecture Notes in Computer Science*, pp. 16–30. Springer Berlin / Heidelberg.

Bourjolly, J.-M., O. Gurtuna, and A. Lyngvi (2006, September). On-orbit servicing: a time-dependent, moving-target traveling salesman problem. *International Transactions in Operational Research 13*(5), 461–481.

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.

Bramel, J. and D. Simchi-Levi (1995, Jul/Aug). A location based heuristic for general routing problems. *Operations Research 43*(4), 649–660.

Bräysy, O. (2003, Fall). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing 15*(4), 347–368.

Bullnheimer, B., R. Hartl, and C. Strauss (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research 89*, 319–328.

Carlton, W. B. (1995). *A Tabu Search Approach to the General Vehicle Routing Problem*. Ph. D. thesis, The University of Texas at Austin, Austin, TX.

Chen, S., B. Golden, and E. Wasil (2007, July). The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks 49*(4), 318–329.

Christofides, N. and S. Eilon (1969, September). An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly 20*(3), 309–318.

Christofides, N., A. Mingozzi, and P. Toth (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi (Eds.), *Combinatorial optimization*,

Chapter 11, pp. 315–338. Chichester, England: John Wiley & Sons Ltd.

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

Cordeau, J.-F., M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany (2005). New heuristics for the vehicle routing problem. In A. Langevin and D. Riopel (Eds.), *Logistics Systems: Design and Optimization*, Chapter 9, pp. 279–297. New York, NY: Springer Science+Business Media.

Cordeau, J.-F., M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet (2002, May). A guide to vehicle routing heuristics. *Journal of the Operational Research Society 53*(5), 512–522.

Cordeau, J.-F. and G. Laporte (2003, July). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B 37*(6), 579–594.

Dantzig, G. and J. Ramser (1959, October). The truck dispatching problem. *Management Science 6*(1), 80–91.

Desrochers, M., J. K. Lenstra, and M. W. P. Savelsbergh (1990). A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research 46*(3), 322–332.

Doerner, K., M. Gronalt, R. F. Hartl, M. Reimann, C. Strauss, and M. Stummer (2002, April). Savingsants for the vehicle routing problem. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. R. Raidl (Eds.), *Applications of Evolutionary Computing: Proceedings EvoWorkshops 2002*, Volume 2279 of *Lecture Notes in Computer Science*, Chapter 2, pp. 11–20. Kinsale, Ireland: Springer Berlin / Heidelberg.

Dror, M., G. Laporte, and P. Trudeau (1994, May). Vehicle routing with split deliveries. *Discrete Applied Mathematics 50*(3), 239–254.

Dror, M. and P. Trudeau (1989, May). Savings by split delivery routing. *Transportation Science 23*(2), 141–145.

Dror, M. and P. Trudeau (1990, June). Split delivery routing. *Naval Research Logistics 37*(3), 383–402.

Dullaert, W., G. K. Janssens, K. Sörensen, and B. Vernimmen (2002, November). New heuristics for the fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society 53*(11), 1232–1238.

Eilon, S., C. D. T. Watson-Gandy, and N. Christofides (1971). *Distribution Management, Mathematical Modelling and Practical Analysis*. London: Griffin.

Fisher, M. and R. Jaikumar (1981). A generalized assignment heuristic for vehicle routing. *Networks 11*(2), 109–124.

Foster, B. A. and D. M. Ryan (1976). An integer programming approach to the vehicle scheduling problem. *Operational Research Quarterly 27*(2), 367–384.

Frizzell, P. and J. Giffin (1992). The bounded split delivery vehicle routing problem with grid network distances. *Asia-Pacific Journal of Operational Research 9*, 101–116.

Frizzell, P. and J. Giffin (1995, July). The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers and Operations Research 22*(6), 655–667.

Gaskell, T. J. (1967, September). Bases for vehicle fleet scheduling. *Operational Research Quarterly 18*(2), 281–295.

Gendreau, M. (2006, November). *Split Delivery Routing*. XIII CLAIO - Latin American Conference on Operations Research, Montevideo - Uruguay.

Gendreau, M., A. Hertz, and G. Laporte (1994, October). A tabu search heuristic for the vehicle routing problem. *Management Science 40*(10), 1276–1290.

Gendreau, M., G. Laporte, and R. Seguin (1996, January). Stochastic vehicle routing. *European Journal of Operational Research 88*(1), 3–12.

Ghiani, G., F. Guerriero, G. Laporte, and R. Musmanno (2003). Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Opera-*

tional Research 151, 1–11.

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

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research 13*(5), 533–549.

Glover, F. (1998). A template for scatter search andpath relinking. In H. J-K, L. E., R. E., S. M., R. S. M. S. J.-K. SnyersHao J-K, Lutton E, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers (Eds.), *Artificial Evolution*, Lecture Notes in Computer Science, pp. 13–54. Springer Berlin / Heidelberg.

Hadjiconstantinou, E., N. Christofides, and A. Mingozzi (1995). A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxation. *Annals of Operations Research 61*(1-4), 21–43.

Helvig, C. S., G. Robins, and A. Zelikovsky (1998). Moving-target tsp and related problems. In G. Bilardi, G. F. Italiano, A. Pietracaprina, and G. Pucci (Eds.), *ESA '98: Proceedings of the 6th Annual European Symposium on Algorithms*, London, UK, pp. 453–464. Springer-Verlag.

Helvig, C. S., G. Robins, and A. Zelikovsky (2003, October). The moving-target traveling salesman problem. *Journal of Algorithms 49*(1), 153–174.

Ho, S. and D. Haugland (2004, October). A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers and Operations Research 31*(12), 1947–1964.

Ichoua, S., M. Gendreau, and J.-Y. Potvin (2000, November). Diversion issues in real-time vehicle dispatching. *Transportation Science 34*(4), 426–438.

Ichoua, S., M. Gendreau, and J.-Y. Potvin (2007). Planned route optimization for real-time vehicle routing. In V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, and I. Minis (Eds.), *Dynamic Fleet Management: Concepts, Systems, Algorithms & Case Studies*. Springer Sci-

ence+Business Media.

Jaillet, P. (1985). *Probabilistic traveling salesman problems*. Ph. D. thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA.

Jaillet, P. (1988, Nov/Dec). A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research 36*(6), 929–936.

Jaillet, P. (1993, Fall). Analysis of probabilistic combinational optimization problems in euclidean spaces. *Mathematics of Operations Research 18*(1), 51–70.

Jaillet, P., J. F. Bard, L. Huang, and M. Dror (2002, August). Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Science 36*(3), 292–300.

Jezequel, A. (1986). Probabilistic vehicle routing problems. Master's thesis, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA.

Jin, M., K. Liu, and R. O. Bowden (2007, January). A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. *International Journal of Production Economics 105*(1), 228–242.

Jin, M., K. Liu, and B. Eksioglu (2008, March). A column generation algorithm for the vehicle routing problem with split delivery. *Operations Research Letters 36*(2), 265–270.

Kawamura, H., M. Yamamoto, T. Mitamura, K. Suzuki, and A. Ohuchi (1998, June). Cooperative search based on pheromone communication for vehicle routing problems. *IEICE transactions on fundamentals of electronics, communications and computer sciences E81-A*(6), 1089–1096.

Kindervater, G. A. P. and M. W. P. Savelsbergh (1997). Vehicle routing: handling edge exchanges. In H. L. Aarts, E and J. K. Lenstra (Eds.), *Local Search in Combinatorial Optimization*, Chapter 10, pp. 337–360. Chichester, England: John Wiley & Sons Ltd.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983, May). Optimization by simmulated annealing. *Science 220*(4598), 671–680.

Kytöjoki, J., T. Nuortio, O. Bräysy, and M. Gendreau (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research 34*(9), 2743–2757.

Laporte, G. (2007, September). What you should know about the vehicle routing problem. *Naval Research Logistics 54*(8), 811–819.

Laporte, G., M. Gendreau, J.-Y. Potvin, and F. b, Semet (2000, July). Classical and modern heuristics for the vehicle routing. *Internation transactions in operational research 7*(4), 285–300.

Laporte, G., F. Louveaux, and H. Mercure (1994, May/Jun). A priori optimization of the probabilistic traveling salesman problem. *Operations Research 42*(3), 543–549.

Larsen, A., O. Madsen, and M. M. Solomon (2007). Classification of dynamic vehicle routing systems. In V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, and I. Minis (Eds.), *Dynamic Fleet Management: Concepts, Systems, Algorithms & Case Studies*. Springer Science+Business Media.

Lee, C.-G., M. A. Epelman, C. C. White, and Y. Bozer (2006, May). A shortest path approach to the multiple-vehicle routing problem with split pick-ups. *Transportation Research Part B 40*(4), 265–284.

Lin, S. and B. W. Kernighan (1973, March). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research 21*(2), 498–516.

Liu, K. (2005, December). *A study on the split delivery vehicle routing problem*. Ph. D. thesis, Mississippi State University.

Madsen, O. B. G., H. F. Ravn, and J. M. Rygaard (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research 60*(1-4), 193–208.

Malandraki, C. and M. S. Daskin (1992, August). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science 26*(3), 185–200.

Mazzeo, S. and I. Loiseau (2004, December). An ant colony algorithm for the capacitated vehicle routing. *Electronic Notes in Discrete Mathematics 18*(1), 181–186.

Miller, D. L. (1995, Winter). A matching based exact algorithm for capacited vehicle routing problems. *ORSA Journal on Computing 7*(1), 1–9.

Mladenović, N. and P. Hansen (1997, November). Variable neighborhood search. *Computers & Operations Research 24*(11), 1097–1100.

Mole, R. H. and S. R. Jameson (1976). A sequential route-building algorithm employing a generalised savings criterion. *Operational Research Quarterly 27*(2), 503–511.

Mota, E., V. Campos, and A. Corberan (2007, April). A new metaheuristic for the vehicle routing problem with split demands. In *Evolutionary Computation in Combinatorial Optimization*, Volume 4446 of *Lecture Notes in Computer Science*, pp. 121–129. Springer Berlin / Heidelberg.

Mullaseril, P. A., M. Dror, and J. Leung (1997, February). Split-delivery routing heuristics in livestock feed distribution. *The Journal of the Operational Research Society 48*(2), 107–116.

Nakao, Y. and H. Nagamochi (2007). A dp-based heuristic algorithm for the discrete split delivery vehicle routing problem. *Journal of Advanced Mechanical Design, Systems, and Manufacturing 1*(2), 217–226.

Nowak, M. (2005, August). *The Pickup and Delivery Problem with Split Loads*. Ph. D. thesis, Georgia Institute of Technology.

Nowak, M., O. Ergun, and C. C. White (2008, February). Pickup and delivery with split loads. *Transportation Science 42*(1), 32–43.

Or, I. (1976). *Traveling salesman-type combinatorial optimization problems and their relations to the logistics of regional blood banking*. Ph. D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research 41*(1-4), 421–451.

Paessens, H. (1988, March). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research 34*(3), 336–344.

Polacek, M., R. F. Hartl, K. Doerner, and M. Reimann (2004, December). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics 105*(6), 613 – 627.

Potvin, J.-Y., G. Lapalme, and J.-M. Rousseau (1989, November). A generalized k-opt exchange procedure for the mtsp. *INFOR 27*(4), 474–481.

Potvin, J.-Y. and J.-M. Rousseau (1995, December). An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society 46*(12), 1433–1446.

Prins, C. (2004, October). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research 31*(12), 1985–2002.

Psaraftis, H. (1995). Dynamic vehicle routing: status and prospects. *Annals of Operations Research 61*(1-4), 143–164.

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

Renaud, J., F. F. Boctor, and G. Laporte (1996, Spring). A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing 8*(2), 134–143.

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

Rubrico, J., J. Ota, H. Tamura, M. Akiyoshi, and T. Higashi (2004, Sept/Oct). Route generation for warehouse management using fast heuristics. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Volume 2, pp. 2093–2098. Sendai, Japan.

Ryan, D. M., C. Hjorring, and F. Glover (1993, March). Extensions of the petal method for vehicle routing. *The Journal of the Operational Research Society 44*(3), 289–296.

Schmid, V. (2007, November). *Trucks in Movement: Hybridization of Exact Approaches and Variable Neighborhood Search for the Delivery of Ready-Mixed Concrete*. Ph. D. thesis, Universitat Wien.

Sierksma, G. and G. Tijssen (1998). Routing helicopters for crew exchanges on off-shores locations. *Annals of Operations Research 76*(1-4), 261–286.

Solomon, M. M. (1987, March). Algorithms for the vehicle routing and scheduling problems with time windows constraints. *Operations research 35*(2), 254–265.

Song, S. H., K. S. Lee, and G. S. Kim (2002, July). A practical approach to solving a newspaper logistics problem using a digital map. *Computers and Industrial Engineering 43*(1-2), 315–330.

Taillard, E. D. (1993, December). Parallel iterative search methods for vehicle routing problem. *Networks 23*(8), 661–673.

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

Tarantilis, C. D. and C. T. Kiranoudis (2002, September). Boneroute: An adaptive memory-based method for effective fleet management. *Annals of Operations Research 115*(1-4), 227–241.

Tavakkoli-Moghaddam, R., N. Safaei, M. Kah, and M. Rabbani (2007, August). A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing. *Journal of the Franklin Institute 344*(5), 406–425.

Thompson, P. and H. Psaraftis (1993, September). Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research 41*(5), 935–946.

Toth, P. and D. Vigo (Eds.) (2002). *The Vehicle Routing Problem*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Toth, P. and D. Vigo (2003, Fall). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing 15*(4), 333–346.

Van Breedam, A. (1995, November). Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research 86*(3), 480–490.

Waters, C. D. J. (1989, December). Vehicle-scheduling problems with uncertainty and omitted customers. *Journal of the Operational Research Society 40*(12), 1099–1108.

Weintraub, A., J. Aboud, C. Fernandez, G. Laporte, and E. Ramirez (1999, July). An emergency vehicle dispatching system for an electric utility in chile. *The Journal of the Operational Research Society 50*(7), 690–696.

Wilck, J. and T. Cavalier (2007, May). Solving the split delivery vehicle routing problem with a loaded travel cost objective. In G. Bayraksan, W. Lin, Y. Son, and R. Wysk (Eds.), *Proceedings 2007 Industrial Engineering Research Conference*, Nashville, TN. Institute of Industrial Engineers.

Wren, A. and A. Holliday (1972, September). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly 23*(3), 333–344.

Xu, J. and J. Kelly (1996, November). A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science 30*(4), 379–393.

Yu, Y., H. Chen, and F. Chu (2006). Large scale inventory routing problem with split delivery: a new model and lagrangian relaxation approach. *International Journal of Services Operations and Informatics 1*(3), 304–320.