AN EXACT ALGORITHM FOR THE VEHICLE ROUTING PROBLEM WITH BACKHAULS

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Cumhur Alper GELOĞULLARI

August 2001

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

Assoc. Prof. Dr. Osman Oğuz (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

Asst. Prof. Dr. Oya E. Karaşan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

Assoc. Prof. Dr. M. Selim Aktürk

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray
Director of Institute of Engineering and Science

Abstract

AN EXACT ALGORITHM FOR THE VEHICLE ROUTING PROBLEM WITH BACKHAULS

Cumhur Alper GELOĞULLARI
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Dr. Osman Oğuz
August 2001

We consider the Vehicle Routing Problem with Backhauls, in which a fleet of vehicles located at a central depot is to be used to serve a set of customers partitioned into two subsets of linehaul and backhaul customers. The objective of the problem is to minimize the total distance traveled by the entire fleet. The problem is known to be $\mathcal{NP}$-hard in the strongest sense and finds many practical applications in distribution planning. We present an exact algorithm for the Asymmetric Vehicle Routing Problem with Backhauls based on solving a relaxation of the problem. In a cutting plane fashion, the algorithm iteratively solves the relaxation while at each iteration, infeasible solutions are identified and seperated from the feasible set of the relaxation. The procedures to identify infeasible solutions are presented, and a set of cuts to eliminate these solutions is proposed. Local search procedures are incorporated to improve the algorithm. Computational tests on randomly generated instances, involving up to 90 customers, are given. The results show the effectiveness of the proposed approach.

Keywords: Vehicle Routing Problem, Vehicle Routing Problem with Backhauls, Subtour Elimination Constraints, Valid Inequalities, Local Search Heuristics.

Özet

DAĞITIM VE TOPLAMA GÜZERGAHI BULMA PROBLEMLERİ İÇİN EN İYİ ÇÖZÜMLÜ BİR ALGORİTMA

Cumhur Alper GELOĞULLARI
Endüstri Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Osman Oğuz
Ağustos 2001

Bu çalışmada, Dağıtım ve Toplama Güzergahı Bulma Problemi olarak bilinen ve bir merkezde konuşlandırılmış olan araçların, müşterilerin gereksinimlerini karşılamak amacı ile gitmeleri gereken en düşük maliyetli güzergahları bulma problemini inceledik. Bu problem çözümü zor bir problem olup dağıtım planlaması alanında bir çok uygulamayla karşımıza çıkmaktadır. Problemin simetrik olmayan uyarlaması için en iyi çözümünü veren bir algoritma sunduk. Bu yöntem, kesikli düzlem yönteminde olduğu gibi, en iyi çözümü bulana kadar problemin bir gevşetmesini tekrar tekrar çözmek ve asıl problemin olursuz çözümlerini uygun kesikler ile çözüm kümesinden ayırmak fikri üzerine kuruludur. Olursuz çözümleri belirleyen yöntemler ve bu olursuz çözümleri çözüm kümesinden ayıran kesikler önerdik. Yerel arama yöntemleri ile algoritmanın daha da verimli olabileceğini gösterdik. Rassal olarak oluşturulan problemler üzerinde algoritmayı test ettik. Sonuçlar önerilen yaklaşımın oldukça etkili olduğunu göstermektedir.

Anahtar Kelimeler: Dağıtım Güzergahı Bulma Problemi, Dağıtım ve Toplama Güzergahı Bulma Problemi, yerel arama, alttur kırıcı kısıtlar

To my parents Hamit and Gülseren Geloğulları
and to my sister Gülin Geloğulları

Thank you for your love and support

Acknowledgements

I would like to express my gratitude to my supervisor Assoc. Prof. Dr. Osman Oğuz for his guidance and encouragement throughout the development of this thesis.

I would like to thank Asst. Prof. Dr. Oya E. Karaşan for reading and reviewing this thesis. I am also grateful for her support in my future career.

I am also indebted to Assoc. Prof. Dr. M. Selim Aktürk not only for reading this thesis and his suggestions, but also for spending considerable time with me talking about my future career.

I would like to thank my close friends Güneş Erdoğan, Onur Boyabatlı, Çağrı Gürbüz, Çerağ Pinçe and Filiz Gürtuna for their support.

I would like to thank Ersin Gündoğdu for his keen friendship and morale support at my desperate times. It's great to know that I will have such a good friend throughout my life.

Finally, I would like to express my deepest thanks to Şengül Doğan for her love, patience, and for relieving me with her presence. I am grateful to her for the beautiful times we share and the happiness she brings into my life.

Contents

List of Figures

List of Tables

Chapter 1

Motivation

"Vehicle routing has been one of the great success stories of operations research in the last decade"

Arjang A. Assad [5], 1988.

Routing problems are problems of logistics concerned with allocation of customers to depots and formation of routes to service these customers. The term *logistics* is described in Encyclopædia Britannica as "the organized movement of materials and, sometimes, people". Council of Logistics Management, a trade organization based in the United States, defines logistics as "that part of the supply chain process that plans, implements, and controls the efficient, effective flow and storage of goods, services, and related information from the point of origin to the point of consumption in order to meet customers' requirements". More simply, it is the science (and art) of ensuring that the right products reach the right place in the right quantity at the right time to satify customer demand.

Logistics is now regarded as a means of cost-saving. Economic phenomena such as the oil crisis of the early 1970's, which resulted in increased interest rates and fuel costs, have stressed distribution as an area where substantial improvements can be achieved. Problems of logistics have become more and more important as the firms started to compete on service differentiation and widened the range of products they offer.

Logistics is often used as a blanket term, encompassing many different components of operations and influencing all aspects of business. One major activity of logistics is the distribution activity. Distribution constitutes a notable fraction of operating costs of individual firms, as well as a substantial portion of the economy of most developed nations. In a report prepared for the National Council of Physical Distribution Management, Kearney [45] estimates annual distribution costs in the United States in 1980 at $400 billion, and in 1983 at $650 billion, almost 21% of the U.S. gross national product. Kearney also reports that an average company can save 20% or more by adopting improvements in its distribution systems.

Therefore, the importance of routing problems is primarily because of the large cost of physical distribution. These problems are quite complex and frequently cannot be solved to optimality. However, small improvements can yield significant savings. This economic importance has motivated both companies and academic researchers to apply techniques of Operations Research/Management Science (OR/MS) to improve the efficiency of distribution systems.

One of the most important problems which play a central role in logistics is known to be the Vehicle Routing Problem with Backhauls (VRPB). The solution of vehicle routing problem with backhauls, which is the focus of this research, affects the overall distribution cost. By identifying individual elements of a distribution system, we can begin to examine trade-offs between them, and come up with an overall improved system.

In the following chapters, we provide information on characteristics and applications of vehicle routing problem, and propose an algorithm that solves it to optimality.

Chapter 2

Introduction

The Introduction consists of four sections. The first section gives a definition of the vehicle routing problem and discusses its variants. Then, VRP with backhauls is discussed. The next section includes applications of the VRPs in real-world. The chapter concludes with the outline of the thesis.

2.1 Routing Problems

The Vehicle Routing Problem (VRP) is an important management problem in the field of distribution and logistics. The problem appears in a large number of practical situations and is known in the literature also as the *vehicle scheduling* [24], *truck dispatching* [20], [30] or simply the *delivery problem*. Operations researchers have been intensively involved with the vehicle routing problem since it was first introduced by Dantzig & Ramser [30] in 1959.

Large number of VRP applications brings a challenge for one to design an algorithm that is flexible enough to meet all the variations faced in the real world. Unfortunately, this is a goal unachieved by any of the existing solution methods in the literature. This is because the problem is known to be $\mathcal{NP}$-hard, which means it is inherently a difficult combinatorial problem. The algorithm we propose here is general enough to satify many but not all different characteristics

of vehicle routing problems.

The *classical* or *basic* vehicle routing problem involves a set of delivery points with known demands to be serviced by a homogeneous fleet of fixed capacity vehicles from a central depot or distribution center. Then, the objective of the problem is to develop a set of routes such that all the delivery points are serviced once and only once by exactly one vehicle, the total demand of the points assigned to each route does not exceed the capacity of the vehicle which services the route, and the total distance traveled by all of the vehicles is minimized. Each route should start and end at the depot.

Figure 2.1 exhibits how a solution to a 4-vehicle and 19-customer VRP looks like. The solid circle stands for the depot, and the other circles represent customers.

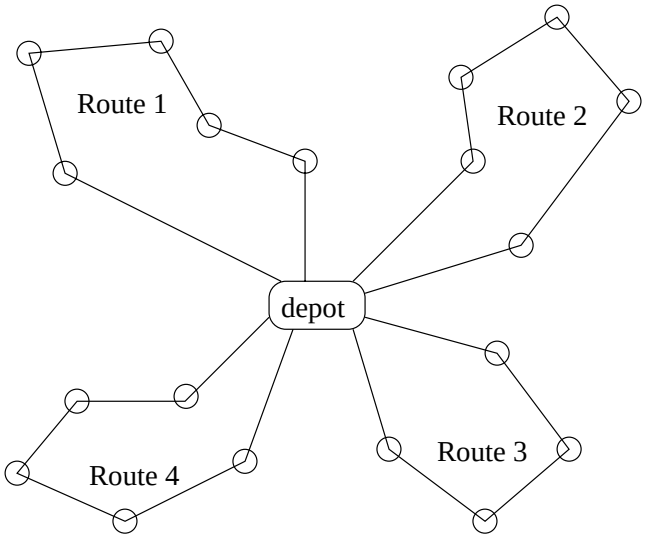

Figure 2.1: An Example of a solution to a VRP

The reason this problem is refered to as *basic* is that it is the core component of a variety of applications. Pure routing problems consist of a geographical component only; however, most real-world applications incorporate several side constraints, as well.

The Traveling Salesman Problem (TSP) is the simplest routing problem. It can simply be stated as follows: Given a set of customers and distances between them the objective is to find the shortest route that visits all customers exactly once. An extension to the TSP is the m-TSP which is similar to the ordinary TSP, but m routes starting and ending at a common depot, should be used. While the TSP has been an area of interest for researchers for many decades, study of the VRP began its rapid expansion only about 20 years ago. This motivation comes from the numerous real world applications and the potential for considerable savings that improved distribution systems represent. As will be explained later, TSP and m-TSP are special cases of VRP.

As stated before, vehicle routing problems exhibit a wide range of real world applications. This variety comes from the fact that every distribution system posseses its own side constraints. In addition, there are some parameters of the basic VRP, which further increase the number of variations. The objective of a routing problem can be to minimize number of vehicles that can serve all the customers or to minimize total distance traveled by the entire fleet. The fleet can be composed of a single vehicle or multiple vehicles. Vehicles can be identical or different types of vehicles can constitute a heterogenous fleet. Depending on the nature of the distribution system, a single depot or multiple distribution centers can serve as a basement for the vehicles. Generally, each vehicle is supposed to operate one route per period (i.e. per day); however, a vehicle can go on a trip several times during a given day. Demand of each customer may or may not be known in advance. In real life, the distance between a customer and another is generally not equal in both directions. In such cases the problem is referred to as Asymmetric VRP (AVRP). However, in most of the cases the underlying graph is considered to be symmetric. (i.e. for all customers i and j, distance from i to j is equal to the distance from j to i). A partial list of these parameters and their domains is presented in Table 2.1.

TSP is a well known $\mathcal{NP}$-hard problem. It is clear that VRP is a generalization of the m-TSP which is a generalization of the TSP. In the m-TSP, if each customer has an associated demand and there is an upper limit on

Parameter	*Domain*
Objective	Minimize distance / traveling time / # of vehicles
Fleet size	one vehicle / multiple vehicles
Fleet type	homogenous / heterogenous
# of depots	single depot / multiple depots
# of routes per vehicle	one route / multiple routes
Type of demand	deterministic / stochastic
Vehicle capacity	finite / infinite
Type of service	delivery / pick-up / mixed / split
Underlying graph	directed / undirected , symmetric / asymmetric

Table 2.1: Parameter settings for the general VRP

the sum of the demands a route can serve, then the resulting problem is a basic VRP. Therefore, VRP is also $\mathcal{NP}$-hard. The reader is referred to the paper by Lenstra and Rinnooy Kan [55] for the $\mathcal{NP}$-hardness of routing problems including the VRP.

The VRP may contain several real-world constraints which complicate an already difficult problem. Common side constraints that real vehicle routing problems include beyond the basic model are as follows.

1. Total time or distance restrictions: Safety considerations and government regulations prohibit drivers from driving more than a time or distance limit. Therefore, the length of each route should be designed to be less than some predetermined value.

2. Time Windows: The time of delivery to a customer may be constrained to fall within a "time window". For example, a store may be open between 7:00 a.m. and 9:00 p.m., which means the vehicles can visit that store between these hours. In such cases, the problem is refered to as *Vehicle Routing Problem with Time Windows* (VRPTW). Time window constraints appear frequently in practice.

3. Precedence Constraints: These constraints impose a partial ordering of the customers. For example, some customers have to be the first or the last one in a route.

4. Site Dependencies: Sometimes, each site (customer) can be serviced by some, but not necessarily all, vehicle types. Customers with high demands may require large vehicles.

5. Delivery and/or Pick up: Besides the delivery aspect of the routing problems, there is a pick up aspect, as well. The next section describes the vehicle routing problem in more detail, when pick up operation is also incorporated into the distribution system.

2.2 Vehicle Routing Problem with Backhauls

As stated in the previous sections, in the basic VRP a set of delivery customers with known demands is to be serviced by a homogenous fleet of fixed capacity vehicles from a single depot. Typically, vehicles leave the depot almost fully loaded, and come back to the depot, after the completion of deliveries, when they become empty.

An extension of the basic VRP, which has received less attention, is the Vehicle Routing Problem with Backhauls (VRPB). VRPB, also known as the *linehaul-backhaul problem*, [17], [41], concerns the routing of vehicles over a set of mixed customers. Some customers are delivery or linehaul points while the others are pick up or backhaul points. Linehaul points are sites that are to receive a quantity of goods from the depot. Backhaul points are sites that send a quantity of goods back to the depot; when a vehicle visits such a point, some quantity of goods are loaded on to the vehicle. Such a partitioning of customers is very frequent. Large retail companies have many outlets to be supplied from the depot, and at the same time, the depots must be resupplied by the vendors located in the same region. A good example is the grocery industry. In this case, supermarkets are linehaul customers, and grocery suppliers such as the vegetable and fruit vendors are the backhaul customers.

VRPB replaces the deadhead trip back to the depot with a profitable activity. That is, linehaul-backhaul problem reduces the distribution costs by making use

of the unused capacity of a vehicle on the trip back to the depot. Therefore, in recent years, backhauling has been widely recognized as a means of significant savings. For example, the Interstate Commerce Commission estimated that the yearly savings obtained by the USA grocery industry due to the introduction of backhauling is almost $160 millions. (see Toth and Vigo, [69]). Kearney's report [45] includes a summary of programs implemented by companies in the period from 1978 to 1983 for improving productivity in logistics. The number one program, utilized by 83% of the survey respondents was coordination of inbound and outbound freight to provide private fleet backhauls.

Like the VRP, VRPB is $\mathcal{NP}$-hard. VRP is a special case of VRPB when the number of linehaul customers is zero. Paper by Yano et al. [71] states that "On the surface, the problem may appear to be a standard vehicle routing problem. However, the special constraints, the presence of both delivery and pick up requirements, and the necessity to consider common carrier alternatives make it complex and interesting."

Since the trucks are assumed to be rear-loaded, backhaul customers are supposed to be visited after the linehaul customers. Many of the solution algorithms are designed to do so. However, different types of trucks with multiple doors for loading and unloading make it possible to construct routes in which linehaul and backhaul customers are located in any sequence.

2.3 Applications of the VRP

There are many applications of the vehicle routing problem in many industries, resulting from the different parameter settings and a bundle of side constraints that real world distribution systems face. These were explained in the previous sections. The delivery operations of many consumer products, such as bread, beer, gasoline and soft drinks, from a central warehouse to retail outlets involve some variant of the vehicle routing problem. The following operations fit into the models of the VRP.

1. Dial-a-ride Problem: This problem concerns dispatching of vehicles to satisfy the demands from the customers who call for a service request. One application from the home health care industry requires the scheduling of a nurse from home to several patients that call for some treatment, and back home, subject to some feasibilty contraints. Another example is in the public transportation industry where taxies are called by the customers. Different versions of the dial-a-ride problem are found in everday practice. (see Teodorovic & Radivojevic [67], Stein [65], Psaraftis [62] and Kikuchi [46])

2. School Bus Routing: A group of spatially distributed students must be provided with public transportation from their residences to their schools and back to their residences after the school is over. This problem generally involves a school district with a number of schools each of which is assigned a number of students, and a given time window for the student pickup and delivery. With the time window restrictions, the problem can be modeled as a VRPTW. The objective is to minimize the fleet size and travel time of the students. (see Bowerman et al. [14] and Braca et al. [15])

3. Inventory Routing: This problem (Christiansen [19], Reiman [63], Federgruen et al. [34]) addresses the problem of allocating some resource available at a central depot among customers such as retail stores. The customers keep some amount of the resource as their own inventory but they experience a random demand pattern. Each day a fleet of vehicles has to be routed within a subset of the customers. Therefore, which customers are to be visited and in what order is to be decided.

4. Waste Collection: A waste management company has to design a set of routes within a city in order to collect the garbage. This problem is actually an arc routing problem because each street, for example, should be traversed for the collection of garbage from every waste basket. (see for example, Shih & Lin [64] and Tung & Pinnoi [70])

5. Package Delivery and Pick up: Package service companies like UPS and FedEx try to efficiently determine delivery/pick up routes. Packages should be collected from the customers and sent to their destinations.

6. Meal and Soft Drink Delivery: A large meal delivery company servicing a large territory would like to design minimum cost and/or time routes to its customers. Such companies like the ones providing meals to airline companies, have to deliver products within some time since meals are not durable for a long time. Soft drink companies like Coca-Cola also try to construct economic routes for delivering their products to supermarkets, restaurants or stores.

7. Machine Scheduling Problems: If the term *vehicle* is interpreted as a machine, and the term *customer* is thought to be any kind of demand, then scheduling problems can be modeled as a vehicle routing problem. (see Chan et al. [18])

8. Automated Guided Vehicle Scheduling Problems: Automated guided vehicles in a production environment should be routed among the production stations. (see e.g. Aktürk & Yılmaz [2])

The above is just a partial summary of the application areas of VRPs. See also, Christofides et al. [21], Bodin et al. [13], Magnanti [57] and most recently Fisher [35] for the applications and classifications of vehicle routing problems.

2.4 Outline of the Thesis

The remainder of this thesis has the following structure: Chapter 3 discusses the existing literature on the VRPs and related problems, the TSP and m-TSP. It gives an overview of formulations and exact and heuristic methods proposed for these problems. Following this review, Chapter 4 demonstrates the algorithm we propose which is based on iteratively solving a relaxation of the VRP. This

chapter discusses some feasibility check and seperation procedures. The chapter then explains further improvements to the original algorithm. An illustrative numerical example demonstrates how the algorithm works. Chapter 5 exhibits the results of some computational experiments with randomly generated problem instances, and discusses some of the implementation details. Finally, Chapter 6 gives conclusions on the experiments and introduces some ideas that can be used for future research.

Chapter 3

Literature Review

3.1 The TSP and m-TSP

Given a set of customers represented by the nodes of a graph, traveling salesman problem is the problem of finding the shortest route which visits each customer once. The multiple traveling salesman problem, on the other hand, is defined as the problem of finding a set of routes originating and terminating at a single depot, where each node is visited once by exactly one salesman.

TSP was extensively studied by researchers and there is a huge literature on it. The reader is directed to Burkard [16] and Lawler et al. [54] for comprehensive surveys on the TSP.

3.1.1 Mathematical Formulations of the m-TSP

In terms of graph theory terminology, the m-TSP can be stated as follows: Given a graph $G = (V, A)$ where $V = (1, 2, \ldots, n)$ is the set of nodes and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of edges, and let $C = (c_{ij})$ be a distance matrix of A, find a minimum cost collection of m node disjoint circuits in the graph G where each circuit starts and ends at the depot. The problem is said to be symmetric if $c_{ij} = c_{ji}$ for all $(i, j) \in A$, and asymmetric otherwise.

The mathematical formulations of the m-TSP are based on the assignment model. These models are, essentially, extended versions of the models for the TSP. This section summarizes some of the formulations in the literature.

Motivated by the definition above, the multiple Traveling Salesman Problem can be modeled as an integer linear program (ILP) as follows. Let

$$x_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \text{ is in the optimal solution} \\ 0, & \text{otherwise} \end{cases}$$

then we would like to find the x_{ij}'s which are to become 1, i.e. finding the arcs that the salesmen should go through, for the distance traveled to be minimized.

Miller Tucker and Zemlin's Formulation

It seems that the first formulation of the m-TSP was given by Miller, Tucker and Zemlin [58]. Their formulation allows the salesman to turn back to the origin, denoted by 0, t times.

$$\text{Minimize} \quad \sum_{i=0}^{n} \sum_{j=0, i\neq j}^{n} c_{ij} x_{ij}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{i=1}^{n} x_{i0} = t && && (3.1)\\
& \sum_{i=0}^{n} x_{ij} = 1, && j = 1,2,\ldots,n \quad i \neq j && (3.2)\\
& \sum_{j=0}^{n} x_{ij} = 1, && i = 1,2,\ldots,n \quad j \neq i && (3.3)\\
& u_i - u_j + p x_{ij} \leq p-1 && 1 \leq i \neq j \leq n && (3.4)\\
& x_{ij} \in \{0,1\} && \forall i,j && \\
& u_i \quad urs && &&
\end{aligned}$$

The constraints (3.1) forces the salesman turn back to the origin t times. The constraints (3.2) and (3.3) are the usual degree constraints of an assignment problem. The constraints (3.4) prohibit the formation of the *subtours*, tours that do not include the depot. These constraints are generally called *subtour elimination constraints* or *SEC* in short. p denotes the maximum number of nodes that a salesman is allowed to visit.

Kulkarni and Behave's Formulation

Another formulation by Kulkarni and Behave includes two more constraints to the usual assignment model. These constraints provide all of the salesmen to be assigned to a tour. Their formulation is as follows, where the origin is node n:

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{i=1}^{n} x_{ij} = 1, && j = 1, 2, \ldots, n-1 \quad i \neq j && (3.5)\\
& \sum_{j=1}^{n} x_{ij} = 1, && i = 1, 2, \ldots, n-1 \quad j \neq i && (3.6)\\
& \sum_{i=1}^{n} x_{in} = m && && (3.7)\\
& \sum_{i=1}^{n} x_{ni} = m && && (3.8)\\
& u_i - u_j + L x_{ij} \leq L - 1 && 1 \leq i \neq j \leq n-1 && (3.9)\\
& x_{ij} \in \{0, 1\} && \forall i, j &&
\end{aligned}$$

Constraints (3.5) and (3.6) are the usual assignments constraints, whereas (3.7) and (3.8) ensure that all the m salesmen are assigned. Constraints (3.9) are the subtour elimination constraints where L is the maximum number of nodes a salesman is allowed to visit.

Bektaş's Formulation

Bektaş [10] discusses about the subtour elimination constraints for the TSP and m-TSP, and proposes a new formulation for the m-TSP based on the assignment model. This formulation is compared with the formulation proposed by Miller, Tucker and Zemlin. Computational study consists of asymmetric m-TSPs of sizes ranging from 60 to 150. The results impose that the new formulation is the best among these formulations, in terms of CPU time. This formulation is given as follows, where 1 is the origin:

$$
\begin{array}{lll}
\text{Minimize} & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
 & & \\
\text{subject to} & \sum_{i=1}^{n} x_{ij} = 1, & j = 2,\ldots,n \\
 & \sum_{j=1}^{n} x_{ij} = 1, & i = 2,\ldots,n \\
 & \sum_{i=1}^{n} x_{1i} = m & \\
 & u_i - u_j + (n-m)x_{ij} & \\
 & \quad +(n-m-2)x_{ji} \leq n-m-1, & i,j = 2,\ldots,n \quad i \neq j \\
 & u_i + (n-m-1)x_{1i} \leq n-m & i = 2,\ldots,n \\
 & u_i + x_{1i} \geq 2 & i = 2,\ldots,n \\
 & t_i - u_i \leq 0 & i = 2,\ldots,n \\
 & t_i - u_i - (n-m-1)x_{i1} \geq -n+m+1 & i = 2,\ldots,n \\
 & t_i - (n-m)x_{i1} \leq 0 & i = 2,\ldots,n \\
 & \sum_{i=2}^{n} t_i = n-1 & i = 2,\ldots,n
\end{array}
$$

First two constraints are the usual assignment constraints. Third constraint ensures that m circuits will be created. The remaining constraints are subtour elimination constraints and ensure that all the m tours include the depot node 1.

3.1.2 Solution Methods of m-TSP

Since the m-TSP is $\mathcal{NP}$-hard, it is highly unlikely that a polynomial time algorithm to solve it exists. This nature of the problem lead to two alternative methods for its solution. Exact methods to find an optimum solution require too much computation time, while heuristic approaches need much more less computational effort but do not guarantee optimality. Exact methods are mainly based on *branch & bound* and *branch & cut* methods. On the other hand, heuristic techniques use local search methods such as tabu search, simulated annealing, genetic algorithms and neural networks. For a detailed review of these solution methods, the reader is referred to Bodin et al. [13], Laporte [48] and Lawler et al. [54].

Exact Solution Methods for the m-TSP

Exact algorithms require too much computation time but they guarantee to obtain the optimum solution. The most promising exact methods seem to be *branch & bound* and *branch & cut* methods.

The branch & bound method's main idea is to use a relaxation of the IP to recursively partition its solution set so that ultimately, for each element of the partition, the solution of the initial IP restricted to that subset is either known exactly or known to be not optimal. Branch & bound remains as the method of 'first attack' on an IP. Most branch & bound methods use a relaxation of the m-TSP obtained by either relaxing the degree constraints, the integrality constraints or the subtour elimination constraints or a combination of them. Svestka & Huckfelt [66] introduced an ILP formulation based on that of Miller, Tucker and Zemlin's but with a new set of SECs. Then, they employed their formulation in a branch & bound framework. Gavish & Srikanth [38] applied a branch & bound method to the m-TSP, obtaining a lower bound through a Lagrangian relaxation of the problem.

There exist another traditional method to solve IPs to optimality: *the cutting plane algorithm.* Its main idea is to solve a sequence of LP relaxations of the initial problem to optimality; each time the solution is nonintegral, an inequality is added to the current relaxation, that is valid for the solution set of original IP but is violated by the optimal solution for the current relaxation. The first example of the cutting plane method was due to Dantzig, Fulkerson and Johnson [29] when they published a description of a method for solving the TSP and illustrated the power of this method by solving a 49-city TSP which was an impressive size for 1950s.

Branch & cut method can be thought of a combination of branch & bound method with the cutting plane algorithm. For each partition of the solution set of the LP relaxation several cuts are added to the current formulation to tighten the problem. That is, in the bounding step, instead of solving one relaxation, a sequence of relaxations is solved each time adding an inequality that is violated

by the current fractional solution.

A cutting plane algorithm due to Laporte & Nobert [50] uses subtour elimination and integrality constraints as cutting planes. This work proposes two ILPs for the symmetric and asymmetric cases of m-TSP. Their algorithm introduces necessary SECs only when the solution to the relaxation is integral.

Exact solution methods for the m-TSP are similar to those for the TSP. This is natural because of the similarities between the structures of the two problems. Many researchers studied on the transformations of m-TSP to TSP. By efficient transformation algorithms, methods for the TSP can be used to solve m-TSPs. Bellmore & Hong [11] showed that the asymmetric m-TSP with m salesmen and n nodes can be transformed to an asymmetric TSP with $n+m-1$ nodes. Jonker & Volgenant [44] improved the standard transformation of the symmetric m-TSP to a standard TSP with a sparse edge configuration.

Heuristic Solutions for the TSP and m-TSP

A heuristic is a solution strategy that produces an answer without any formal guarantee for optimality. Heuristic procedures produce near-optimal solutions in a reasonable amount of time. Many heuristics have been proposed for the TSP. On the other hand, m-TSP attracted less attention in terms of the number of heuristics proposed. Heuristics developed for the standard TSP are applied to the m-TSP by transformong it to a standard TSP. The heuristics proposed for the TSP and m-TSP can be classified as *tour construction* heuristics and *tour improvement* heuristics.

Tour construction heuristics involve construction of a tour from scratch following some construction criteria and stop whenever an initial tour is formed. In the *Nearest Neighbour* procedure, the salesman starts from a city and then visits the city nearest to him. From there he visits the nearest city that was not visited so far. *Insertion heuristic*, starts with a tour on small subsets like a trivial tour of one or two nodes and then extends the tour by inserting the remaining nodes. Clarke & Wright [24] introduced the famous *savings* algorithm

for the VRP, but it is also applied to TSP. In savings method, initially there are $|\mathcal{C}|$ tours each of which start from a base node, visit only one customer and end at the base node. Then, the savings that can be obtained by combining different tours are computed and the tours are combined starting from the combination that yields the largest saving.

Tour improvement heuristics try to improve the quality of a given tour by simple tour modifications. In other words, these algorithms search for the best tour among a neighborhood of the given feasible tour. This neighborhood depends on the tour modification procedure. A well known tour improvement heuristic is the *2-opt* procedure proposed by Croes [27]. This procedure removes two arcs from the initial tour and replaces two different arcs that improve the quality of the tour. The new arcs are chosen so that the new solution is still a tour. When such a modification is done, the new tour is treated as the initial tour and the modifications are seeked on this new solution. Algorithm terminates when there is no possible improvement. A famous procedure proposed by Lin & Kernighan [56] which considers *r-exchanges* for the improvement while r changes dynamically during the procedure.

Improvement heuristics may get stuck in local optima. To prevent this, several heuristics such as *simulated annealing* and *tabu search*, are proposed. Simulated annealing procedure moves from a given solution to a minimum cost solution by gradually changing the initial solution. However, sometimes, the initial solution is substituted by the new solution although the new solution is more costly. This increases the probability to of getting closer to the global optimum. Simulated annealing has been applied to TSP by several researchers including Rossier et al. (1986) and Nahar et al. (1989) (see Laporte [48]). Tabu search also tries to prevent getting stuck at local minima. In order to prevent cycling, the solutions that are already been examined are stored in a 'tabu list'. The success of this method depends on the careful choice of control parameters. Several researchers that applied tabu search to the TSP include Knox (1988), Malek (1988) and Fiechter (1990) (see Laporte [48]).

The heuristics proposed for the m-TSP are limited and includes exchange heuristics, tabu search, evolutionary programming, and neural networks.

3.2 Vehicle Routing Problem

We gave a definition for the basic VRP in §2.1 on page 4. Here, we give the notation we use for the mathematical formulations of the VRP.

We denote the set of customer locations by $\mathcal{C}$ ={1,2,. . . ,n} and the depot location by 0. Let $G = (N, E)$ be a complete directed graph representing the Vehicle Routing Network where $N = \mathcal{C} \cup \{0\} = \{0, 1, 2, \ldots, n\}$ is the set of nodes and $E = \{(i, j) : i, j \in N, i \neq j\}$ is the set of edges. Further, we adopt the following notation:

d_i = demand of customer i, $i \in \mathcal{C}$
m = number of delivery vehicles
Q_k = capacity of vehicle k, $k \in \{1, 2, \ldots, m\}$
c_{ij} = distance from location i to location j

We note that $c_{ii} = \infty$ for all $i \in N$. The VRP then consists of finding a minimum-cost collection of m simple circuits such that each vehicle performs exactly one circuit, each circuit visits node 0, each node different from node 0 is visited by exactly one circuit, and for a given circuit the sum of the demands of all the nodes in the circuit does not exceed the capacity of the vehicle servicing that circuit. The objective is to minimize the total distance traveled, defined as the sum of all the arcs belonging to the circuits.

The following sections demonstrate mathematical formulations and solution methodologies of the vehicle routing problems.

3.2.1 Mathematical Formulations of the VRP

This section does not give a comprehensive survey on the VRP formulations which are many and varied, but rather gives basic formulations which led to

different solution methods. The reader is referred to Christofides et al. 1979 [21], Magnanti 1981 [57], Bodin et al. 1983 [13], Golden & Assad 1988 [42], Laporte 1992 [49], and most recently to Fisher 1995 [35] for surveys on the VRP.

Formulation due to Fisher and Jaikumar

This formulation was given by Fisher and Jaikumar in 1981 [36].
Let

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ visits customer } j \text{ immediately after customer } i \\ 0, & \text{otherwise} \end{cases}$$

and

$$y_{ik} = \begin{cases} 1, & \text{if customer } i \text{ is visited by vehicle } k \\ 0, & \text{otherwise} \end{cases}$$

The basic VRP is then

Minimize $\sum_{i,j} c_{ij} \sum_k x_{ijk}$

subject to	$\sum_k y_{ik} = 1$	$i = 1, \ldots, n$		(3.2.1)
	$\sum_k y_{ik} = m$	$i = 0$		(3.2.2)
	$\sum_i d_i y_{ik} \leq Q_{ik}$	$k = 0, \ldots, m$		(3.2.3)
	$\sum_j x_{ijk} = \sum_i x_{jik} = y_{ik}$	$i = 0, \ldots, n$	$k = 1, \ldots, m$	(3.2.4)
	$\sum_{i,j \in S} x_{ijk} \leq \lvert S \rvert - 1$	for all $S \subseteq \{2, \ldots, n\}$	$k = 1, \ldots, m$	(3.2.5)
	$y_{ik} \in \{0, 1\}$	$i = 0, 1, \ldots, n$	$k = 1, \ldots, m$	
	$x_{ijk} \in \{0, 1\}$	$i, j = 0, 1, \ldots, n$	$k = 1, \ldots, m$	

Constraints (3.2.1) and (3.2.2) ensure that every customer is allocated to some vehicle, except for the depot which is visited by all of the m vehicles. Constraints (3.2.3) are the vehicle capacity constraints, constraints (3.2.4) ensure that a vehicle which visits a customer also leaves it, and (3.2.5) are the usual subtour elimination constraints of the TSP.

Formulation due to Laporte, Nobert and Desrochers

In 1985, Laporte, Nobert and Desrochers [53] adapted a formulation of the TSP to the VRP by adding extra variables, and a constraint to model the depot and vehicle capacities. In this formulation, x_{ij} represents the number of vehicles traveling directly between customers i and j. $V(S) = \lceil \sum_{i \in S} d_i/Q \rceil$ where $\lceil y \rceil$ denotes the smallest integer not less than y. That is, $V(S)$ is a lower bound on the number of vehicles needed to serve all the customers in S. All the vehicles are assumed to be identical and Q is the common vehicle capacity. They consider a symmetric VRP.

$$\text{Minimize} \quad \sum_{i<j} c_{ij} x_{ij}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{j<i} x_{ij} + \sum_{j>i} x_{ji} = 2 && i = 1, 2, \ldots, n && (3.2.6)\\
& \sum_{j} x_{0j} = 2m && && (3.2.7)\\
& \sum_{(i,j) \in S \times S} x_{ij} \leq |S| - V(S) && \text{for all } S \subset \{1, \ldots, n\} && (3.2.8)\\
& x_{ik} \in \{0, 1\} && 1 \leq i \leq j \leq n &&\\
& x_{0j} \in \{0, 1, 2\} && j = 1, 2, \ldots, n &&
\end{aligned}$$

Constraints (3.2.6) ensure that the degree of every node except the depot is two, meaning that there is an incoming and outgoing arc. Constraint (3.2.7) provide that m vehicles enter and leave the depot, so the degree of the depot is $2m$. Constraints (3.2.8) are the subtour elimination constraints, where $V(S)$ is a lower bound on the number of vehicles needed to serve the customers in the set $|S|$. The case $x_{0j} = 2$ corresponds to a route containing only customer j. If single customer routes cannot occur, x_{0j} can be restricted to be 0 or 1.

3.2.2 Solution Methods of the VRP

It is quite clear that the mathematical formulations of the VRP, exhibited in the previous section, are too complex in solving VRPs of non-trivial size. Since VRP is $\mathcal{NP}$-hard, solution methods are dominated by heuristic approaches. We will

discuss some of the heuristics and exact methods for the VRP in the following sections.

Exact Solution Methods for the VRP

Exact methods for the VRP are based on the formulations given before. As with any combinatorial problem, their success or failure is dependent on the degree to which they exploit problem structure. Exact methods for the VRP can be classified into three broad categories: Direct tree search techniques, dynamic programming and integer linear programming. We review a few examples to illustrate the variety of exact methods for the VRP.

Direct tree search methods typically embed a non-LP based lower bounding procedure within a branch & bound scheme. For example, Laporte, Nobert and Desrochers [53], used the formulation presented on page 21 but relaxed this formulation by dropping the capacity constraints. They added these constraints as they are violated since these are too numerous to specify apriori. Later Laporte, Mercure & Nobert [52] used a similar formulation in a branch & bound algorithm. The relaxation was obtained by dropping the capacity constraints which results in a formulation of m-TSP. Then m-TSP is transformed to a standard TSP. Throughout the branch & bound algorithm, they eliminate the solutions that violate the capacity constraints by branching on proper variables when an integral solution is achieved. (i.e. partition the search space by setting the variables, that are in an infeasible tour, to 0 or 1)

Another example to the direct tree search methods is due to Christofides, Mingozzi and Toth [22] in 1981. In a branch & bound procedure the quality of the lower bounds is extremely important for the efficiency. In this method, the lower bound is obtained from *k-degree center tree*. A *k-degree center tree* is a tree (that is, a subset of $n-1$ edges, T, such that T is a single connected component containing no cycles) where the degree of the depot is k. The lower bound on the VRP is obtained by shortest path computations using the tree. Problems from the literature ranging from 10 to 25 nodes were successfully solved by this

procedure. Later, in 1987 Kolen et al. [47] generalized this method for the VRP with time windows.

Dynamic programming was first proposed for the VRPs by Eilon et al. [33]. Let the number of vehicles m be fixed and $c(S)$ denote the cost of a vehicle route through node 0 and all the nodes of a subset S of $N \setminus \{0\}$. Also let, $f_k(U)$ be the minimum cost that can be achieved using k vehicles and delivering to a subset U of $N \setminus \{0\}$. Then the minimum cost can be found through the following recursion:

$$f_k(U) = \begin{cases} c(U), & k = 1 \\ min[f_{k-1}(U \setminus U^*) + c(U^*) \mid U^* \subset U \subseteq N \setminus \{0\}], & k > 1 \end{cases}$$

The cost of the solution is $f_m(N \setminus \{0\})$ and the optimal solution corresponds to the optimizing subsets U^* in the above recursion. It is clear that the $f_k(U)$ has to be computed for all subsets of U and all values of k. Therefore, the number of computations is too high. The authors propose techniques to reduce the number of states by means of a relaxation procedure, and by using feasibility or dominance criteria. By that way, instances of 10 to 25 nodes were solved.

Balinkski & Quandt [7] were first to propose a *set partitioning* formulation for the VRP. Let r denote a feasible route and the index set of all feasible routes be R. Also let a_{ir} be a binary coefficient equal to 1 if and only if node $i > 0$ appears on route r. Let c_r^* be the optimal cost of route r and x_r, a binary variable equal to 1 if and only if route r is used in the optimal solution. Then, the VRP can be stated as:

$$\begin{array}{lll} \text{Minimize} & \sum_{r \in R} c_r^* x_r & \\ \text{subject to} & \sum_{r \in R} a_{ir} x_r = 1 & i \in N \setminus \{0\} \\ & x_r \in \{0, 1\} & \forall r \in R \end{array}$$

The number of binary variables x_r in this formulation can reach to millions in real-life instances. In addition, it is difficult to compute c_r^*, the cost of each route. To find c_r^* of route r, which includes the nodes in S, one must solve a TSP within the node set S. However, if the objective is to minimize the number of vehicles

(i.e. $c_r^* = 1$ for all $r \in R$) and the number of variables is relatively small, the linear relaxation of the above set partitioning problem provides integral solutions.

A good way to overcome the difficulties underlying the set use of partitioning formulation is to use *column generation algorithm*. This technique is successfully applied to the VRP by Orlof [60] and Desrosiers et al. [32]. In column generation, a reduced problem which includes only a subset of all possible columns (variables) is repeatedly solved. The approach to solve a linear program requires in each iteration, the solution of a pricing problem to determine whether or not the current set of columns contains an optimal solution for the linear program. In the case of the VRP, the pricing problem involves finding a tour through a subset of the nodes for which the reduced cost of the associated column is negative, or proving that no such tour exists. This pricing model is equivalent to finding a negative cycle in an edge-weighted graph with the additional restrictions that the cycle pass through the depot, and the sum of the demands of the nodes in the cycle does not exceed the vehicle capacity.

Fisher & Jaikumar [36] developed an algorithm for the VRP based on the formulation they propose (see page 20). The algorithm is designed as a heuristic but it guarantees optimality in a finite number of steps. The algorithm is based on Benders' Decomposition. A generalized assignment problem (GAP) that assigns customers to vehicles is solved iteratively while the routes are formed by solving a TSP within the customers assigned to each route. The algorithm generates a feasible solution even if it does not run to completion. Therefore, it is sometimes called as *the generalized assignment heuristic*.

Perhaps the most promising algorithm to optimally solve combinatorial problems is branch & cut. We have explained branch & cut algorithm shortly in section 3.1.3. The success of branch & cut algorithm for the TSP encourages its use for the vehicle routing problems. Consequently, as the polyhedral structure of the VRP was explored (see, for example, Cornuejols & Harce [26]) successful implementations of this algorithm for the VRP were reported. Araque et al. [4] report the solution of instances up to 60 identical customers. Bard et al. [8]

report a branch & cut algorithm for the vehicle routing problem with satellite facilities. More recently, Corberán et al. [25] developed a branch & cut algorithm for the general routing problems.

Although there are a number of exact methods proposed for the VRP, VRPB attracted less attention. To our knowledge, there are two exact algorithms proposed for the VRPB. One is due to Mingozzi & Giorgi [59]. The authors present a new 0-1 program for the VRPB and compute a lower bound to the optimal solution cost by combining different heuristic methods for solving the dual of the LP-relaxation of the exact formulation. This algorithm solved symmetric instances up to 100 customers.

The other exact algorithm, proposed by Toth & Vigo [68], makes use of a new linear integer programming model and a Lagrangian lower bound which is strenghtened in a cutting plane fashion. The Lagrangian lower bound is then combined, with a lower bound obtained by dropping the capacity constraints, thus obtaining an effective overall bounding procedure. A branch & bound algorithm, reduction and dominance criteria are also described. Symmetric and asymmetric instances involving up to 100 customers are solved successfully.

Yano et al. [71] proposed an exact algorithm for a special case of the VRPB where each route can have at most four points. This procedure uses set covering to find an optimal set of routes.

Heuristic Solutions for the VRP

Heuristic algorithms for the VRP are often derived from the algorithms for the TSP. The nearest neighbour algorithm, insertion algorithms and tour improvement procedures can be applied to the VRP almost without modification. The only difference is that, the routes constructed by the procedure should be checked for feasibility since VRPs contain several side constraints.

The *savings* algorithm proposed by Clarke & Wright [24] in 1964 starts with vehicle routes containing the depot and just one customer. At each step, two

routes are combined in the order of the largest savings that can be generated by combining the routes. More formally, the algorithm can be stated as follows. Compute the savings $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ for $i, j = 1, \ldots, n$ and $i \neq j$. Generate n routes $(1, i, 1)$ for $i = 1, \ldots, n$. Then, order the savings in a non-increasing fashion. Starting from the top of the list, merge the two routes containing nodes i and j into a new route $(0, i, j, 0)$. This step is repeated until no further improvement is possible.

The *sweep* algorithm proposed by Gillet & Miller [40] is a two-phase method and represents customers by their polar coordinates (θ_i, ρ_i) where θ_i is the angle and ρ_i is the ray length. Then the customers are ranked in increasing order of their θ_i. Then, an unused vehicle is chosen; starting from the unrouted customer with the smallest angle, customers are assigned to the vehicle as long as its capacity is not exceeded. If there are unrouted customers another vehicle is chosen and same steps are repeated. At the end, each vehicle route is optimized by solving the corresponding TSP.

Another two phase method is given by Christofides et al. [21]. Their method selects a seed node and constructs a route by including other nodes according to some insertion cost criteria until the capacity of the vehicle is reached. After all vehicles are used, the algorithm computes the insertion cost of a node into a feasible cluster relative to the seed of the cluster. The node with the minimum insertion cost is assigned to its corresponding cluster. In the second phase, TSP is solved for each of the cluster.

As discussed in the previous section, the two phase method of Fisher & Jaikumar [36] is an exact algorithm if allowed to run to completion. But it is generaly referred to as the generalized assignment *heuristic* since it generates feasible routes at each step. Baker & Sheasby [6] proposed an extension to the generalized assignment heuristic.

As in the case of the exact algorithms, the number of heuristics for the VRPB is less than those for the VRP.

Deif & Bodin proposed a modified savings algorithm for the VRPB where the

linehauls have to preceed backhauls in a given route (Casco et al. [17]). As the first step, the usual savings are computed but the condition that backhauls must occur after linehauls is also imposed. Therefore, once a backhaul customer is located at the end of a route, no linehaul customer is added to that route. This way, the routes become too short and therefore to have longer routes a penalty for the backhaul customers to be merged in a route is used.

Goetschalckx & Jacobs-Bella [41] propose a heuristic for the VRPB based on space filling curves developed by Bartholdi & Platzman [9]. Toth & Vigo [69] propose a cluster-first-route-second type heuristic which uses a new clustering method. The algorithm is applicable to both symmetric and asymmetric instances.

Tabu search, simulated annealing and genetic algorithms are recently being used to develop heuristic algorithms for the vehicle routing problems. The reader is referred to Gendreau et al. [39] for a detailed study on such recent heuristics for the VRPs.

Chapter 4

The Algorithm

In this chapter we propose an exact algorithm for the asymmetric vehicle routing problem with backhauls (AVRPB). Although the algorithm is designed for AVRPB, it can also be used for standard AVRPs (without backhauls) by simply setting the number of backhaul customers to 0. This chapter is organized as follows: Section 4.1 gives preliminaries including our notation and definitions. Section 4.2 describes the algorithm we propose for the VRPB. Section 4.3 discusses procedures that improve the proposed algorithm. Finally, section 4.4 demostrates the algorithm on a numerical example.

4.1 Preliminaries

We gave the notation we adopted for the VRP in §3.2 on page 19. Extended for the VRPB, we re-present our notation here. Additional notation will be introduced when necessary.

The set of linehaul customer locations is denoted by $\mathcal{L} = \{1, 2, \ldots, L\}$ and the set of backhaul customer locations by $\mathcal{B} = \{L+1, L+2, \ldots, L+B\}$ where L is the number of linehaul customers and B is the number of backhaul customers. Thus, the set of all customers is given by $\mathcal{C} = \mathcal{L} \cup \mathcal{B} = \{1, 2, \ldots, L+B\}$. The depot location is represented by 0. Let $G = (N, E)$ be a complete directed graph

representing the Vehicle Routing Network where $N = \mathcal{C} \cup \{0\} = \{0, 1, 2, \ldots, L + B\}$ is the set of nodes and $E = \{(i, j) : i, j \in N, i \neq j\}$ is the set of edges. Further, we adopt the following notation:

d_i = demand of (or amount supplied by) customer i, $i \in \mathcal{C}$

m = number of delivery vehicles

Q_k = capacity of vehicle k, $k \in \{1, 2, \ldots, m\}$

c_{ij} = distance from location i to location j, $(i, j) \in E$

We note that $c_{ii} = \infty$ for all $i \in N$ and $d_0 = 0$. The cost matrix is asymmetric; that is, $c_{ij} \neq c_{ji}$ for some $(i, j) \in E$. Whenever we are dealing with identical vehicles case, Q denotes the common vehicle capacity. The AVRPB then consists of finding a minimum-cost collection of m simple circuits such that each vehicle performs exactly one circuit, each circuit visits node 0, each node different from node 0 is visited by exactly one circuit, and for a given circuit the minimum capacity required to serve the nodes (i.e. deliver goods to linehaul customers and collect goods from the backhaul customers) on that circuit does not exceed the capacity of the vehicle servicing the circuit. The objective is to minimize the total distance traveled, defined as the sum of all the edges belonging to the circuits.

We define a *vehicle route* for the k^{th} vehicle as a sequence of locations $R_k = (i_1 = 0, i_2, i_3, \ldots, i_r = 0)$ beginning and ending at the depot, and all intermediate locations are distinct. We also define $q(R_k)$ as the capacity required for the route. In other words, it is the maximum amount of load on an infinite-capacity imaginary vehicle during its trip on the route.

As discussed in Chapter 2, in the literature, it is generally considered for the VRPB that the backhaul customers have to come after the linehaul customers in a route (see for example, Mingozzi & Giorgi [59], Toth & Vigo [68]). There are few heuristic examples with no obligation of this kind and to our knowledge, there is not an exact algorithm for the VRPB without this restriction. It is clear that with such precedence constraints, the capacities of vehicles can be reduced while still servicing the customers. In such cases, for a given route of linehaul customers preceeding backhaul customers, the capacity of the vehicle servicing

the route should be greater than or equal to the maximum of the sum of the demands of the linehaul customers and the sum of the amounts supplied by the backhaul customers. Consider the following example:

Consider the route $(0, 1, 2, 3, 4, 5, 0)$ and let the locations 1, 2 and 3 be linehaul, and 4 and 5 be backhaul customers. The figure below summarizes the route characteristics.

Route:	0	1	2	3	4	5	0
Type of customer:	-	L	L	L	B	B	-
Demand:	0	10	5	5	15	10	0

Figure 4.1: A route with backhauls after linehauls

It is clear that vehicle k which is assigned to this route should have a capacity of at least 25 units. The vehicle loaded with the goods to be delivered to the linehaul customers ($10 + 5 + 5 = 20$ units) starts its trip from the depot. But after it delivers all the goods, it should visit backhaul customers to pick up goods ($15 + 10 = 25$ units). Therefore, its capacity should be at least $q(R_k) = max\{ \sum_{i \in (R_k \cap \mathcal{L})} d_i \ , \ \sum_{j \in (R_k \cap \mathcal{B})} d_j\} = max\{10 + 5 + 5, 15 + 10\} = 25$.

On the other hand, consider that it is not obligatory for the backhaul customers to be visited after the linehaul customers. Customers of both type can be visited in any sequence in a route. Considering the same example, suppose that the route is now $(0, 4, 1, 2, 3, 5, 0)$. Now, the vehicle starts with a load of 20 units. But the first customer it should visit is a backhaul customer. That is, at customer 4 it should have empty space for 15 units. The capacity required by this route when it is at customer 4 is $20 + 15 = 35$ units. After visiting customers 1 and 2, 15 units will be delivered and the remaining empty space will be enough to collect goods from customers 3 and 5. Therefore, the capacity required by the entire route is 35 units. In the next sections, we give an algorithm to determine the capacity of a route where the linehaul and backhaul customers are in any sequence.

This simple example demonstrates that it is better to restrict the configuration of the routes so that the backhaul customers are visited after the linehaul customers, if it is desired to use smaller vehicles. But note that, the objective of the problem is to minimize the total distance traveled by the entire fleet. Without such a restriction, it is clear that the distance traveled will probably be less than that of restricted case (one of the constraints is now relaxed). This may be preferred considering the benefits in the long run.

As opposed to many of the algorithms proposed in the literature, the algorithm we propose here has no precedence relation between these two types of customers. The algorithms and heuristic methods proposed for the VRPB also generally allow formation of routes with only linehaul customers, commonly however, they prohibit the routes consisting of only backhaul customers. Our algorithm also allows the routes of linehaul or backhaul customers alone.

4.2 The Default Algorithm

In the previous chapters we explained that m-TSP is just a special case of the VRP. This is clear intuitively: m-TSP concerns with finding m tours within geographically dispersed customers where each tour starts and ends at the depot, and each customer is visited once. It is well known that when an additional constraint is added to a problem, its feasible set shrinks or stays the same since that constraint may be violated by some points within the original feasible set. In the m-TSP, if each customer has an associated demand and there is an upper limit on the sum of the demands a route can serve, then the resulting problem is a basic VRP with m vehicles. Note also that VRP is a special case of VRPB with number of backhaul customers equal to zero (i.e. $B = 0$).

Therefore, the m-TSP is a relaxation of the VRP and VRPB, obtained by dropping the capacity constraints. This implies that $X^{mTSP} \supseteq X^{VRP}$ where X^{mTSP} denotes the feasible set of the m-TSP and X^{VRP} denotes the feasible set of the VRP with m vehicles. This is equivalent to saying that any feasible

solution to the VRP is also a feasible solution to the m-TSP. This statement is not necessarily true in the reverse direction. That is, a feasible solution to the m-TSP may or may not be a feasible solution to the VRP.

This is the main motivation underlying the proposed approach for the solution of the VRP. One can make use of the fact that it is easier to solve m-TSP compared to VRPs. The core of the algorithm we propose to solve the VRP and VRPB is this: Solve the corresponding m-TSP obtained by dropping the capacity constraints of the VRP. Check the solution to the m-TSP and identify whether this solution is feasible for the VRP. If the solution is feasible for the VRP, it is also optimal for the VRP. If the solution is infeasible for the VRP then add necessary inequalities valid for the VRP but violated by the current m-TSP solution to the m-TSP formulation. After appending the inequalities, repeat the same steps.

Let x^*_{VRPB} and x^*_{m-TSP} denote the optimal solution for the VRPB and the corresponding m-TSP, respectively. Then, a more formal description of the default algorithm can be given as in *Figure 4.2*.

	The Default Algorithm
Step 1.	*Solve the corresponding m-TSP formulation for the VRPB. let* x^*_{m-TSP} *be its solution.*
Step 2.	*Check whether* $x^*_{m-TSP} \in X^{VRPB}$
Step 3.	*If* $x^*_{m-TSP} \in X^{VRPB}$ *stop,* $x^*_{VRPB} = x^*_{m-TSP}$. *else add inequalities valid for the VRPB but violated by* x^*_{m-TSP}. *Go to Step 1.*

Figure 4.2: The Default Algorithm

It is quite apparent that this is a finite algorithm since the number of solutions to the m-TSP is finite as in any combinatorial optimization problem. The algorithm will eventually find a feasible solution to the VRP, if of course the number and capacity of the vehicles are enough to service all the customers, or will declare infeasibility otherwise.

The above algorithm is just like cutting plane algorithms. The only difference is that, in the cutting plane algorithm, the LP relaxation of the IP is iteratively solved while at each iteration nonintegral solutions are chopped off by adding proper cuts. Both of the algorithms stop whenever the solution to the relaxation is feasible for the original problem (cutting plane algorithm stops when an integral solution is at hand).

It is clear that each step of the algorithm can be realized by different approaches. The following discussion includes the way we handle the steps of the algorithm.

4.2.1 Solution of the m-TSP

The heart of the default algorithm is the solution of the m-TSP formulation efficiently. Because m-TSP is solved again and again during the execution of the algorithm, fast algorithms should be used to solve it. Among the alternative formulations of the m-TSP in the literature, the formulation due to Bektaş [10] is reported to be the most effective for the asymmetric problems. This formulation was presented on page 14.

We propose solving the corresponding m-TSP for the VRPB by branch & bound which is quite effective for the asymmetric m-TSPs. We solve the problem with the subtour elimination constraints included in the formulation proposed by Bektaş [10]. Therefore, the optimal solution of the m-TSP denoted by x^*_{m-TSP} is integral.

4.2.2 Checking Feasibility for the VRPB

This section illustrates how it can be determined whether a given solution to the m-TSP is feasible for the VRPB or not.

Remember that the number of vehicles is represented by m and the capacity of each vehicle is denoted by Q_k, for all $k = 1, \ldots, m$. Note also that, the solution to the m-TSP is a set of m routes, each denoted by R_k, that does not include any

common node other than the depot. Finally, let Q_{max} be equal to the maximum of the capacities of the vehicles (i.e. $Q_{max} = max\{Q_k | k = 1, \ldots, m\}$).

Suppose that we are given a solution to the corresponding m-TSP, x^*_{m-TSP}. It is clear that one should try to assign vehicles to each of the m routes given in the solution. There are two situations. One should first check whether each route in this solution requires a capacity more than Q_{max} or not. Still, the vehicles may not be assigned to routes although each of the routes requires capacity less than or equal to Q_{max}.

Before discussing these two situations, we explain how the capacity required by route k, $q(R_k)$, can be computed.

Computation of $q(R_k)$:

We demonstrated how $q(R_k)$ can be computed for a route k in which backhauls come after linehauls in §4.1 and noted that we would explain an algorithm which computes the capacity required by a route in which backhauls and linehauls can be in any sequence. In this section we propose a simple algorithm for the computation of $q(R_k)$ for any route.

It is clear that a vehicle must be loaded with the goods it should deliver before it leaves the depot. Therefore, that vehicle should have a capacity of at least the sum of the linehaul customers in the route. The computation of $q(R_k)$ is simply keeping track of the maximum load on the vehicle during its trip: Starting with a load equal to the sum of the linehaul customers, at each linehaul customer decrease the load on the vehicle by the demand of that customer; and increase the load by the amount supplied by each backhaul customer. This simple procedure is depicted in *Figure 4.3*.

Consider the previous example:

The sum of the demands of linehaul customers in this route is 20 units. Starting with 20 units, the vehicle arrives backhaul customer 4 and picks up 15 units. The total load on the vehicle is now 35 units. Then comes linehaul

```
            Pseudo Code for Algorithm Compute q(R_k)
            Input: R_k = (i_1 = 0, i_2, i_3, ..., i_r = 0)
Step 1.     q(R_k) ← 0
            for i = i_1 to i = i_r
                if (i ∈ L)
                    q(R_k) ← q(R_k) + d_i
            maxq ← q(R_k)
Step 2.     for i = i_1 to i = i_r
                if (i ∈ L)
                    q(R_k) ← q(R_k) − d_i
                else
                    q(R_k) ← q(R_k) + d_i
            if maxq < q(R_k)
                maxq ← q(R_k)
Step 3.     q(R_k) = maxq
```

Figure 4.3: Algorithm *Compute* $q(R_k)$

Route:	0	4	1	2	3	5	0
Type of customer:	-	B	L	L	L	B	-
Demand:	0	15	10	5	5	10	0
Total Load:	20	**35**	25	20	15	25	0

Figure 4.4: Computation of $q(R_k)$, an example

customer 1 and 10 units of goods are delivered. Therefore, there are 25 units on the vehicle. The last row on the table exhibits the load on the vehicle during its trip. The maximum amount of load on the vehicle is after it visits customer 4, and is 35 units.

Feasibility Check, Case 1:

A solution to the m-TSP is a collection of m routes. As stated before, one should first check whether each route in the solution requires a capacity more than Q_{max} or not.

After computing the capacity required by each of the routes in the solution, it is easy to compare them with Q_{max}. Formally, the route $R_k = (i_1 = 0, i_2, i_3, \ldots, i_r = 0)$ is infeasible for the VRPB if $q(R_k) > Q_{max}$. This feasibility check should be applied to all of the m routes. For example, suppose that we have 3 vehicles of capacities 10, 15 and 20. Then, the route in *Figure 4.4* is infeasible because it requires a capacity of at least 25 which cannot be provided by any of the vehicles.

We will call this algorithm as *feasibility check algorithm 1.*

Feasibility Check, Case 2:

For a given solution, suppose that

$$q(R_k) \leq Q_{max} \qquad \forall k \in \{1, \ldots, m\}$$

or, in other words, all of the m routes require some capacity less than or equal to the capacity of the biggest vehicle. The solution at hand passes feasibility check algorithm 1 discussed in the previous section.

Still, we may not be able to assign vehicles to the routes, meaning that the solution is infeasible for the VRPB. Consider the following example:

Suppose that there are 3 vehicles of capacities 15, 20 and 30. Suppose also that the m-TSP solution is 3 routes such that $R_1 = \{0, 1, 2, 3, 4, 0\}$, $R_2 = \{0, 5, 6, 0\}$ and $R_1 = \{0, 7, 0\}$. Let $q(R_1) = 25$, $q(R_2) = 22$ and $q(R_3) = 12$. As explained in *Figure 4.5*, it is clear that vehicle 1 can be assigned to route 1, and vehicle 3 to route 3. But vehicle 2 cannot be assigned to route 2. Therefore, this solution is

R_k	Route #	$q(R_k)$		Q_k	Vehicle #
$\{0, 1, 2, 3, 4, 0\}$	1	25	$\rightarrow \surd \rightarrow$	30	1
$\{0, 5, 6, 0\}$	2	22	$\rightarrow \times \rightarrow$	20	2
$\{0, 7, 0\}$	3	12	$\rightarrow \surd \rightarrow$	15	3

Figure 4.5: Infeasibility Check, Case 2: An infeasible solution

infeasible for the VRPB.

We need to check the feasibility of the solution second time if it passes the feasibility check algorithm 1. We describe here what we call as *feasibility check algorithm 2*:

For simplicity, we assume that the vehicles are indexed so that

$$i < j \iff Q_i \geq Q_j \qquad i \neq j \quad i, j = 1, \ldots, m$$

(i.e. biggest vehicle has the smallest index) and the routes are indexed so that

$$i < j \iff q(R_i) \geq q(R_j) \qquad i \neq j \quad i, j = 1, \ldots, m$$

Then feasibility check algorithm 2 can simply be described as follows: Starting from vehicle 1, try to assign each vehicle to the route with the same index. If a route requires more capacity than the capacity of the corresponding vehicle, then the solution at hand is infeasible.

4.2.3 Cuts for the elimination of infeasible solutions

In the previous section we described the two cases which declare that a given collection of m routes is infeasible for the VRPB. In this section we introduce two types of cuts that are valid for the VRPB but seperate the infeasible solutions from the feasible set of m-TSP.

In this section $l(R_k)$ denotes the number of edges in route k.

Route Elimination Constraints

Note that route k, $R_k = (i_1 = 0, i_2, i_3, \ldots, i_r = 0)$, is a path of nodes starting and ending at the depot. Suppose that a given solution fails to pass feasibility check algorithm 1, or in other words there is at least one route, say route k, in this solution such that $q(R_k) > Q_{max}$. Then it can be eliminated by adding the

following *route elimination constraint* to the m-TSP formulation.

$$\sum_{\substack{i,j \in R_k \\ i \neq j}} x_{ij} \leq l(R_k) - 1 \tag{1}$$

where $l(R_k)$ corresponds to the number of edges in route k. Such a constraint forces one of the edges in a route not to be chosen for the solution, therefore prohibits the formation of the route. For example, assume that $Q_{max} = 30$. Then the route previously mentioned in *Figure 4.4* is infeasible because it requires a capacity of 35 units. This route is visualized in *Figure 4.6*, circles represent backhaul customers and squares represent linehaul customers. We add

$$x_{04} + x_{41} + x_{12} + x_{23} + x_{35} + x_{50} \leq 5 \tag{2}$$

to eliminate this particular route from the solution. Note that since the graph

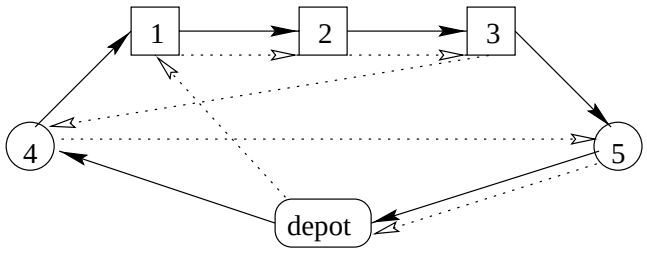

Figure 4.6: Two different routes among 5 customers

is directed (we have asymmetric VRPB), the permutations of this route, which may be feasible tours, are not eliminated by the addition of constraint 2. For example, the route $R = (0, 1, 2, 3, 4, 5, 0)$, depicted with the dashed lines above, is feasible since $q(R) = 25 \leq 30 = Q_{max}$. With the addition of 2, we can still have $x_{01} = x_{12} = x_{23} = x_{34} = x_{45} = x_{50} = 1$ which represents R.

Multiple Routes Elimination Constraints

Consider a collection of m routes such that each route passes feasibility check algorithm 1, but the set fails feasibility check algorithm 2. Remember that the

routes and vehicles are ordered in a nondecreasing fashion according to their capacities. Without loss of generality, we can assume that k^{th} route cannot be assigned to a vehicle. Then the following *multiple routes elimination constraint* should be added to the m-TSP formulation.

$$\sum_{\substack{(i,j)\in R_1 \\ i\neq j}} x_{ij} + \sum_{\substack{(i,j)\in R_2 \\ i\neq j}} x_{ij} + \ldots + \sum_{\substack{(i,j)\in R_k \\ i\neq j}} x_{ij} \leq \Big(\sum_{i=1}^{k} l(R_k)\Big) - 1 \tag{3}$$

Such a constraint prohibits simultenous formation of the first k routes, by restricting the sum of the number of edges belonging to these routes. Consider the example given in *Figure 4.5*. First two routes should not occur together. So, we add

$$x_{01} + x_{12} + x_{23} + x_{34} + x_{40} + x_{05} + x_{56} + x_{60} \leq 7 \tag{4}$$

This constraint will eliminate either route 1 or route 2, or both. Note again that addition of constraint 4 will not eliminate any feasible route or set of routes.

In the case of the identical vehicles where $Q_1 = Q_2 = \ldots = Q_m = Q_{max}$, we do not need to apply feasibility check algorithm 2. Because since the capacities of vehicles are same, if each route requires capacity less than the identical capacity, it is clear that these m routes will pass feasibility check algortihm 2.

The main steps of the Default Algorithm are described above. However, the algorithm can be improved significantly. The next section discusses some procedures that accelerates the Default Algorithm.

4.3 Acceleration Procedures

The Default Algorithm solves the m-TSP iteratively, at each iteration checking the feasibility of the solution and adding the cuts, discussed in the previous section, for the elimination of infeasible routes. At a given iteration, the solution of the m-TSP is within some neighbourhood of the solutions obtained at the previous iterations. This means that, once a solution for the m-TSP is obtained, by searching a neighbourhood of this solution defined by some

local search operators, we can generate routes that will possibly come out as solutions of the next iterations of the Default Algorithm. So, we can check the feasibility of these routes and prevent their formation if they are infeasible for the VRPB by adding proper cuts explained before. We will call such cuts as *route prevention constraints* (rather than route elimination) because they are generated for potential infeasible solutions. The details are as follows:

First of all, for any pair of customers i and j of the same type, if $d_i+d_j > Q_{max}$ then i and j cannot belong to the same route. Thus, x_{ij} and x_{ji} need not be defined.

Approximation algorithms for the vehicle routing problems usually have two phases: a *construction phase*, in which an initial feasible solution is constructed, and a *local search phase*, in which an attempt is made to improve that initial solution by repeatedly searching a specified neighbourhood for a better one.

Most neighbourhoods that are being used in the context of VRPs are based on the well-known *k-exchange* procedures, which were originally proposed for the TSP. A *k-exchange* procedure for the TSP selects k edges in a given tour and replaces them by different k edges while keeping the solution as a tour shorter than the previous one. The techniques developed for the TSP have to be modified in order to handle multiple routes and various side constraints. Traditional *k-exchange* procedures can be used to improve a VRP solution by considering the routes one at a time. However, the multiple-route structure offers additional opportunities. In the next section, we focus on how we can generate potential routes, from the existing set of routes, using edge-exchange operators.

4.3.1 Edge-Exchange Neighbourhoods

As stated before, edge-exchange procedures search a neighbourhood of a solution for a better one. In our case, at each iteration, we solve the m-TSP to optimality. The solution is generally not feasible for the VRPB but provides a lower bound for the optimal objective function value of the VRPB. We add cuts to eliminate

the infeasible solutions and solve the m-TSP again. The objective value of the new solution will be greater than or equal to the previous objective value. In other words, the objective value of the m-TSP we are solving increases or stays the same after each iteration. Therefore, we should search for candidate solutions worse than the current solution.

Suppose we try to apply *k-exchange* procedures to the first m-TSP solution. Since this solution is optimal, we cannot find a better solution. Also suppose that we try to apply *k-exchange* procedures to the solution of the m-TSP which now contains a number of tour elimination constraints. It is clear that the *k-exchange* procedures may identify better solutions for the m-TSP since current solution has a cost much greater than the original m-TSP cost. But these solutions will not appear in the following iterations because the optimal cost of the m-TSPs in the following iterations cannot be less than the cost of the current solution. (As we add additional cuts, the cost of the optimal solution will increase.) Therefore, to generate potential solutions that will appear in the following iterations, we should search the neighbourhood of the current solution and select worse solutions rather then better ones.

It makes no sense to check the feasibility of every candidate solution obtained by local search. If the local search generates a solution whose cost is very high compared to the current solution, we may not check its feasibilty since it will not probably come out as a solution before the algorithm stops. We try to identify solutions worse than the current solution but have costs less than or equal to some percentage of the cost of the current solution. To be formal, we try to find $x^{candidate}_{m-TSP}$, such that $cx^{*}_{m-TSP} \leq cx^{candidate}_{m-TSP} \leq \lambda \cdot cx^{*}_{m-TSP}$ where cx^{*}_{m-TSP} is the current m-TSP solution and $\lambda > 1$ is fixed in advance. For example, if the cost of the current solution is 200 and $\lambda = 1.05$ then we try to find candidate solutions whose costs are between 200 and $200 \cdot 1.05 = 210$. Note, the larger values of λ the more generated candidate solutions we will check, but most of them will be unnecessary. We have chosen $\lambda = 1.02$ in our experiments given in the next chapter.

In the following section, we focus on the algorithms that provide edge-exchanges within a single route and between multiple routes.

Representation of the set of routes

The two prerequisites for a neighbourhood search approach are a representation for the problem and a set of operators which can be applied to that representation. The VRP as shown in *Figure 2.1* can be transformed so that it can be represented by a single vector. (see *Figure 4.7*) The depot is replaced with four copies of itself, each copy located between the routes. In this way, the four routes can be represented as one large string. This representation has a structure (e.g.

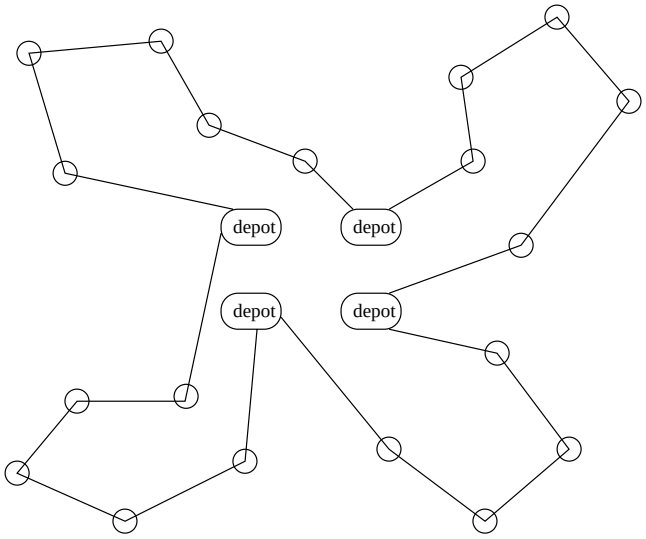

Figure 4.7: Representation of the routes as a single string

the routes are directed). We need operators that will preserve the structure while generating permutations of the string in order to allow us to explore its neighbourhood. Two of such operators are illustrated in *Figures 4.8* and *4.9*.

The *Swap* operator identifies two nodes, i and j, of the string and swaps their positions. This operator realizes a *4-exchange* each time it is called. If predecessor of i is denoted by pre_i and suc_i represents successor of i, *swap* operator excludes

edges (pre_i, i), (i, suc_i), (pre_j, j), (j, suc_j) from the string and introduces the edges (pre_i, j), (j, suc_i), (pre_j, i), (i, suc_j).

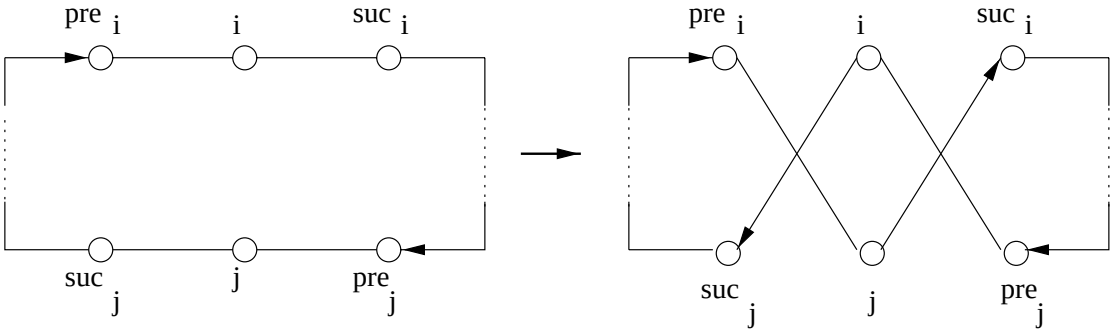

Figure 4.8: Swap operation

The *Relocate* operator, on the other hand, removes a node, i, and replaces it between two adjacent nodes, j and suc_j. This operator realizes a *3-exchange* as illustrated in *Figure 4.9*.

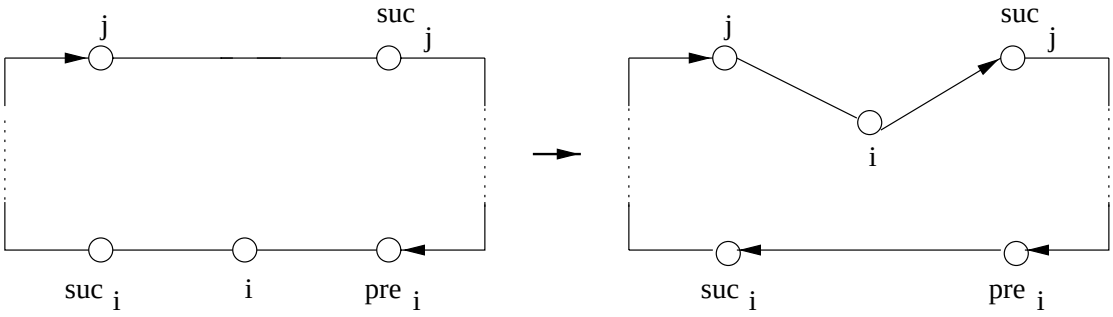

Figure 4.9: Relocate operation

Note that, these operators realize intra-route node exchanges when i and j are in the same route, and inter-route node exchanges when i and j belong to different routes. These operators can easily be used for asymmetric problems since they preserve the orientation of the parts of the string that remain the same.

Without using the representation described above, we also incorporate another neighbourhood search operator which is the *Crossover* operator. Given two different routes, *Crossover* operator divides each of the routes into two paths and forms two new routes by changing these paths between the routes. *Figure 4.10* clarifies this operation.

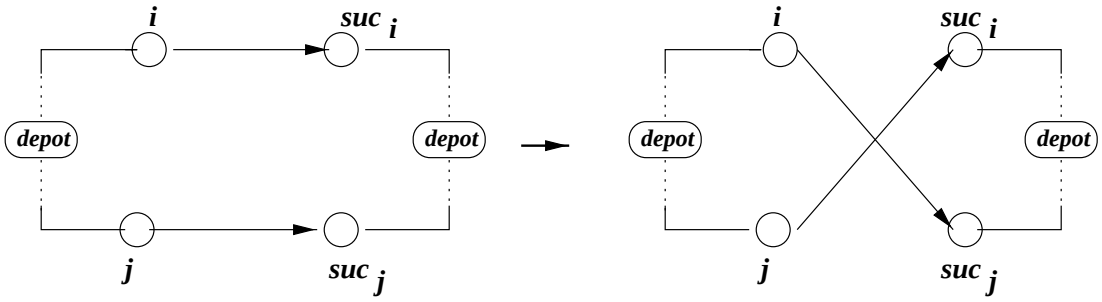

Figure 4.10: Crossover operation

Infeasible solutions generated by these operaters are eliminated by appending proper cuts as explained before. If any operator identifies a feasible solution for the VRPB, then this solution is given as an initial solution for the m-TSP. This solution will provide an upper bound for the m-TSP of the next iteration and hence reduce the size of the branch & bound tree.

The next section demonstrates the proposed algorithm on a numerical example.

4.4 A Numerical Example

Consider the example below. There are 10 customers of which 5 are linehaul customers. For simplicity, we have identical vehicles and we use only *Relocate* operator for local search and set $\lambda = 1.01$. This means that, a solution encountered during the local search will be considered only if its cost is worse than the cost of the current solution by at most 1%. There are 3 vehicles. Problem

data is summarized below:

Customer No:	1	2	3	4	5	6	7	8	9	10
Type:	L	L	L	L	L	B	B	B	B	B
Demand/Supply:	39	20	99	51	90	92	52	95	47	65

Vehicle No:	1	2	3
Capacity:	230	230	230

That is, $Q_1 = Q_2 = Q_3 = 230 = Q_{max}$. The distance matrix is given in *Table 4.1.*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
0	-	26	94	33	66	48	2	53	41	28	97
1	56	-	66	20	7	53	73	99	15	22	73
2	24	25	-	27	48	47	50	47	22	72	9
3	46	50	49	-	11	73	82	94	67	26	86
4	12	48	1	22	-	31	23	25	39	25	71
5	64	42	75	11	80	-	61	50	31	59	28
6	10	15	39	91	82	10	-	86	93	39	87
7	69	65	60	39	47	13	42	-	11	33	54
8	70	23	27	74	10	43	34	51	-	88	25
9	81	41	48	62	75	16	93	28	63	-	46
10	8	43	33	91	35	54	29	48	94	48	-

Table 4.1: Distance matrix for the example problem

Iteration 0:

The m-TSP solution for the given distance matrix is:

$$\begin{aligned} R_1: &\quad 0-9-7-5-3-4-0 && q(R_1) = 339 \\ R_2: &\quad 0-6-0 && q(R_2) = 92 \\ R_3: &\quad 0-1-8-2-10-0 && q(R_3) = 160 \end{aligned}$$

with a cost of 200.

Clearly, route 1 is not feasible for the VRPB, since it requires a capacity of 339 units, which is more than $Q_{max} = 230$. Routes 2 and 3 are feasible routes. So, add

$$x_{09} + x_{97} + x_{75} + x_{53} + x_{34} + x_{40} \leq 5$$

to the m-TSP formulation.

From the routes above, the *Relocate* operator identifies the following set of routes:

$$\begin{aligned} \bar{R}_1: &\quad 0-9-7-5-3-4-2-0 && q(\bar{R}_1)=359 \\ \bar{R}_2: &\quad 0-6-0 && q(\bar{R}_2)=92 \\ \bar{R}_3: &\quad 0-1-8-10-0 && q(\bar{R}_6)=74 \end{aligned}$$

The cost of this solution is 202. First route is infesible, so also add

$$x_{09}+x_{97}+x_{75}+x_{53}+x_{34}+x_{42}+x_{20}\leq 6$$

to the m-TSP formulation.

Iteration 1:

Resolving the m-TSP after adding the cuts found above gives the solution

$$\begin{aligned} R_1: &\quad 0-9-7-5-3-0 && q(R_1)=288 \\ R_2: &\quad 0-6-0 && q(R_2)=92 \\ R_3: &\quad 0-1-8-4-2-10-0 && q(R_3)=166 \end{aligned}$$

with a cost of 207. Route 1 is infeasible, we add

$$x_{09}+x_{97}+x_{75}+x_{53}+x_{30}\leq 4$$

to the m-TSP formulation. Local search does not give a candidate m-TSP solution this time.

Iteration 2:

The m-TSP solution is

$$\begin{aligned} R_1: &\quad 0-1-9-5-3-4-0 && q(R_1)=287 \\ R_2: &\quad 0-6-0 && q(R_2)=92 \\ R_3: &\quad 0-7-8-2-10-0 && q(R_3)=212 \end{aligned}$$

with a cost of 218. We add

$$x_{01}+x_{19}+x_{95}+x_{53}+x_{34}+x_{40}\leq 5$$

to the m-TSP formulation. *Relocate* operator generates two solutions. Among these, the following routes appear for the first time:

$$\begin{aligned}
\bar{R}: &\quad 0-1-9-5-3-4-2-0 && q(\bar{R}) = 307\\
\bar{R}: &\quad 0-7-8-10-0 && q(\bar{R}) = 212\\
\bar{R}: &\quad 0-1-9-7-5-3-4-0 && q(\bar{R}) = 339\\
\bar{R}: &\quad 0-8-2-10-0 && q(\bar{R}) = 160
\end{aligned}$$

The first and third routes are infeasible, so we add

$$x_{01} + x_{19} + x_{95} + x_{53} + x_{34} + x_{42} + x_{20} \leq 6$$

and

$$x_{01} + x_{19} + x_{97} + x_{75} + x_{53} + x_{54} + x_{40} \leq 6$$

to the m-TSP formulation.

Iteration 3:

The m-TSP solution is

$$\begin{aligned}
R_1: &\quad 0-9-7-5-3-2-10-0 && q(R_1) = 308\\
R_2: &\quad 0-6-0 && q(R_2) = 92\\
R_3: &\quad 0-1-8-4-0 && q(R_3) = 63
\end{aligned}$$

with a cost of 221. We add

$$x_{09} + x_{97} + x_{75} + x_{53} + x_{32} + x_{2,10} + x_{10,0} \leq 6$$

to the m-TSP formulation. Local search gives no solution.

Iteration 4:

The m-TSP solution is

$$\begin{aligned}
R_1: &\quad 0-1-9-7-5-3-4-2-0 && q(R_1) = 359\\
R_2: &\quad 0-6-0 && q(R_2) = 92\\
R_3: &\quad 0-8-10-0 && q(R_3) = 160
\end{aligned}$$

with a cost of 222. We add

$$x_{01} + x_{19} + x_{97} + x_{75} + x_{53} + x_{34} + x_{42} + x_{20} \leq 7$$

to the m-TSP formulation. Local search gives no solution.

Iteration 5:

The m-TSP solution is

$$
\begin{array}{lll}
R_1: & 0-3-9-7-5-10-0 & q(R_1)=189 \\
R_2: & 0-6-0 & q(R_2)=92 \\
R_3: & 0-1-8-4-2-0 & q(R_3)=166
\end{array}
$$

with a cost of 224. This solution is feasible for the VRP and hence optimal for it.

When the local search is not incorporated into the algorithm, the example above is solved after 9 iterations. Notice that we used just one local search operator. The use of several operators will be more successful in identifying more candidate solutions.

Chapter 5

Computational Experiments

We now report the results of experimenting with the algorithm described in Chapter 4. We implemented our algorithm in C programming language on a SUN Enterprise 4000 work station operating at CPU clock of 248 Mhz and 1024 real memory. We use CPLEX Linear Optimizer 5.0[1] as the IP solver for the m-TSP. The interface between the C code and the optimizer is realized by using CPLEX Callable Library [43] routines. The code is available in Appendix C.

We generated to sets of asymmetric VRPB instances by our random problem generator, and tested the proposed algorithm on these instances. A total of 720 problems are tested. The first set consists of problems with identical vehicles and the second set includes instances with heterogenous fleet.

[1] Copyright ©1997 ILOG

Instances with a homogenous fleet

There are 540 instances in this set. These instances are generated as follows:

- Number of customers (problem size) range from 10 to 90 in increments of 10.
- For a given problem size, 3 instances are generated so that the percentage of backhauls customers is 0%, 20% and 50%. For each pair of problem size and backhaul percentage value, 5 instances are generated.
- As proposed in Laporte et al. [52], the distances c_{ij} and the customer demands/supplies d_i are generated from a uniform distribution over $[0, 100]$.
- The common vehicle capacity, Q, is determined also as in [52], that is:

  $$Q = (1 - \alpha)(max\{d_i| \quad i \in \mathcal{C}\}) + \alpha \sum_{i \in \mathcal{C}} d_i$$

  then, lowest feasible value for the number of vehicles is:

  $$m = \left\lceil \frac{\sum_{i \in \mathcal{C}} d_i}{Q} \right\rceil$$

  where α is parameter chosen in the interval $[0, 1]$. For each instance of size less than or equal to 50, we ran the algorithm by setting α equal to 0.25, 0.50, 0.75 and 1.00. This way, the number of vehicles are generated as 4, 2, 2 and 1, respectively. Note that, the smaller the value of α, the harder the problem to solve. Because small values of α yield more vehicles with less capacities whereas large values give less number of vehicles with more capacities. We set α equal to 0.50, 0.75 and 1.00 for instances of size greater than 50.

The generated problems are accepted so that the utilization of the fleet is likely to be higher. For example, the cases $\frac{\sum_{i \in \mathcal{C}} d_i}{Q} = 3.12$ and $\frac{\sum_{i \in \mathcal{C}} d_i}{Q} = 3.85$ will both have $m = 4$ but the solution to the former will have a utilization less than that of the later.

The results from this set of instances are presented in Appendix A. To observe the effectiveness of the acceleration procedures, we ran the *Default Algorithm* and the *Improved Algorithm*, which includes the acceleration procedures, on the same instances. Tables 5.1-5.3 exhibit a summary of these results. The column *Avg. Time* stands for the average computation time of 5 instances. Columns named *Avg. # iter*, *Avg. # TEC* and *Avg. # TPC* represent average values of the number of iterations, average number of the tour elimination constraints added, and average number of tour prevention constraints generated during the execution of the algorithm, respectively. Note that, for the identical case, there is no need for the *feasibility check algorithm 2*; hence, *multiple tour elimination constraints* are not reported. Notice also that for $\alpha = 1.00$, the problem reduces to a 1-TSP; no cuts will be added.

%B=0%		Default Algorithm			Improved Algorithm			
$\|\mathcal{C}\|$	α	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # TPC
10	0.25	1.00	5.40	5.80	0.97	4.60	4.80	5.00
	0.50	0.74	3.20	3.40	0.73	3.00	3.20	0.60
	0.75	0.42	1.20	1.20	0.43	1.00	1.00	0.40
	1.00	0.12	-	-	-	-	-	-
20	0.25	7.80	11.20	14.00	6.13	8.20	9.20	10.00
	0.50	7.68	6.80	5.60	6.16	6.20	6.20	4.20
	0.75	4.40	3.00	3.00	3.16	2.20	2.20	3.00
	1.00	0.45	-	-	-	-	-	-
30	0.25	70.69	15.80	17.40	43.46	10.40	11.40	18.80
	0.50	15.32	2.20	2.40	13.98	2.00	2.00	4.20
	0.75	9.67	0.60	0.60	8.71	0.40	0.40	1.60
	1.00	3.60	-	-	-	-	-	-
40	0.25	256.34	28.20	32.40	186.60	12.60	14.40	15.00
	0.50	46.14	5.80	6.40	44.61	5.20	5.60	4.00
	0.75	28.38	2.80	3.00	28.84	2.60	2.60	2.20
	1.00	12.32	-	-	-	-	-	-
50	0.25	502.79	18.80	23.20	391.69	12.60	13.80	29.80
	0.50	105.20	5.80	5.80	100.78	5.20	5.20	15.00
	0.75	15.87	0.60	0.60	14.40	0.60	0.60	1.60
	1.00	31.18	-	-	-	-	-	-
60	0.25	805.08	28.80	35.60	546.21	19.40	21.20	32.60
	0.50	446.90	8.20	8.20	181.58	4.20	4.20	8.00
	0.75	36.62	1.20	1.40	37.66	1.20	1.40	0.40
	1.00	41.59	-	-	-	-	-	-
70	0.25	1.383.60	21.00	23.60	1.193.22	17.60	17.80	32.40
	0.50	392.28	9.40	9.80	332.89	8.60	9.00	6.20
	0.75	27.71	1.00	1.00	28.07	1.00	1.00	0.40
	1.00	73.61	-	-	-	-	-	-
80	0.25	1.790.07	21.80	23.80	1.553.99	18.00	18.00	24.00
	0.50	645.59	14.20	15.40	497.18	11.40	11.60	19.40
	0.75	43.42	1.60	1.80	31.34	1.00	1.20	3.40
	1.00	139.86	-	-	-	-	-	-
90	0.25	2.496.71	23.40	24.20	2.227.70	18.40	18.40	34.60
	0.50	1.427.34	18.20	19.60	990.53	14.20	14.60	22.60
	0.75	77.09	0.80	0.80	78.56	0.80	0.80	4.00
	1.00	293.72	-	-	-	-	-	-

Table 5.1: Average Results for 5 instances from data set 1. ($\%B = 0$)

Average utilization of the fleet of vehicles is 0.92, 0.88, 0.79 and 1 for $\alpha = 0.25$, $\alpha = 0.5$, $\alpha = 0.75$ and $\alpha = 1.00$, respectively.

%B=20%		Default Algorithm			Improved Algorithm			
$\|\mathcal{C}\|$	α	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # TPC
10	0.25	1.02	4.60	4.60	0.90	4.00	4.00	1.60
	0.50	0.60	2.40	2.40	0.59	2.00	2.00	1.40
	0.75	0.42	0.80	0.80	0.34	0.60	0.60	0.60
	1.00	0.16	-	-	-	-	-	-
20	0.25	6.38	6.00	6.60	5.00	3.80	4.00	9.20
	0.50	2.00	2.40	2.40	1.87	1.80	2.00	3.00
	0.75	0.20	0.20	0.48	0.20	0.20	0.80	
	1.00	0.55	-	-	-	-	-	-
30	0.25	32.65	12.40	13.80	31.69	11.60	12.80	3.40
	0.50	17.74	5.00	5.00	16.72	4.20	4.40	2.40
	0.75	2.36	0.40	0.40	2.43	0.40	0.40	0.00
	1.00	2.08	-	-	-	-	-	-
40	0.25	59.09	9.40	10.00	48.92	7.20	7.60	5.40
	0.50	41.95	4.00	4.00	39.83	3.60	3.60	1.40
	0.75	12.63	0.60	0.60	12.40	0.60	0.60	0.40
	1.00	10.05	-	-	-	-	-	-
50	0.25	204.91	10.80	11.40	180.80	9.40	9.60	16.80
	0.50	89.27	2.60	2.60	76.63	2.20	2.20	2.80
	0.75	35.03	0.20	0.20	35.81	0.20	0.20	0.80
	1.00	34.95	-	-	-	-	-	-
60	0.25	769.28	18.80	23.40	407.83	12.40	13.40	26.60
	0.50	202.75	1.40	1.80	203.57	1.20	1.60	3.20
	0.75	94.19	0.40	0.60	94.97	0.40	0.60	0.60
	1.00	38.67	-	-	-	-	-	-
70	0.25	1.399.61	21.00	22.00	1.095.86	17.00	17.00	24.00
	0.50	227.61	3.20	3.20	220.50	2.80	2.80	2.80
	0.75	35.36	0.60	0.60	36.73	0.60	0.60	0.40
	1.00	91.55	-	-	-	-	-	-
80	0.25	1.572.84	18.60	20.40	1.478.81	16.20	16.60	18.80
	0.50	392.85	6.00	6.60	314.81	5.20	5.40	18.20
	0.75	45.02	1.40	1.60	46.08	1.40	1.60	3.20
	1.00	173.95	-	-	-	-	-	-
90	0.25	2.606.85	26.40	27.80	2.124.82	21.20	22.00	27.00
	0.50	371.21	5.60	6.60	332.21	5.00	5.60	11.00
	0.75	93.60	1.20	1.40	96.40	1.00	1.00	3.20
	1.00	365.29	-	-	-	-	-	-

Table 5.2: Average Results for 5 instances from data set 1. ($\%B = 20$)

Average utilization of the fleet of vehicles is 0.89, 0.82, 0.74 and 1 for $\alpha = 0.25$, $\alpha = 0.5$, $\alpha = 0.75$ and $\alpha = 1.00$, respectively.

%B=50%		Default Algorithm			Improved Algorithm			
$\|\mathcal{C}\|$	α	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # TPC
10	0.25	1.24	6.60	6.80	1.11	5.60	5.80	6.40
	0.50	0.24	0.60	0.60	0.26	0.60	0.60	0.80
	0.75	0.20	0.20	0.20	0.21	0.20	0.20	0.00
	1.00	0.17	-	-	-	-	-	-
20	0.25	7.40	8.60	9.40	6.72	7.20	7.60	6.00
	0.50	6.43	4.40	4.40	6.79	4.40	4.40	2.00
	0.75	2.63	2.20	2.40	2.67	2.20	2.40	1.40
	1.00	0.70	-	-	-	-	-	-
30	0.25	19.92	7.00	7.20	18.18	6.40	6.60	5.80
	0.50	11.05	2.60	2.80	11.02	2.20	2.40	2.20
	0.75	4.58	0.40	0.60	4.90	0.40	0.60	0.60
	1.00	4.08	-	-	-	-	-	-
40	0.25	123.88	10.40	11.40	108.98	8.00	8.20	12.40
	0.50	24.48	2.40	2.40	24.07	2.00	2.00	2.00
	0.75	7.80	0.00	0.00	8.10	0.00	0.00	0.00
	1.00	9.36	-	-	-	-	-	-
50	0.25	404.51	16.20	17.60	280.14	12.00	13.20	33.80
	0.50	36.63	2.20	2.40	35.40	1.40	1.40	4.40
	0.75	28.45	1.00	1.00	28.91	1.00	1.00	1.40
	1.00	19.11	-	-	-	-	-	-
60	0.25	664.97	12.00	12.40	222.58	3.80	4.20	12.60
	0.50	266.61	3.60	3.80	204.00	2.80	3.00	4.60
	0.75	43.66	0.40	0.60	43.55	0.40	0.60	1.60
	1.00	40.83	-	-	-	-	-	-
70	0.25	1.318.47	17.20	19.20	1.200.16	13.80	14.20	19.80
	0.50	224.13	2.80	3.00	223.14	2.80	3.00	3.80
	0.75	97.33	1.00	1.00	97.83	1.00	1.00	1.60
	1.00	102.74	-	-	-	-	-	-
80	0.25	1.428.46	13.40	14.20	1.125.90	11.00	11.00	23.60
	0.50	240.47	2.40	2.60	243.66	2.40	2.60	1.80
	0.75	119.45	0.80	0.80	120.45	0.80	0.80	1.20
	1.00	102.74	-	-	-	-	-	-
90	0.25	2.058.27	15.60	16.40	1.728.97	11.80	12.00	32.40
	0.50	367.36	3.00	3.60	330.20	2.60	2.80	9.40
	0.75	115.59	0.80	1.00	117.87	0.80	1.00	2.00
	1.00	217.31	-	-	-	-	-	-

Table 5.3: Average Results for 5 instances from data set 1. ($\%B = 50$)

Average utilization of the fleet of vehicles is 0.85, 0.80, 0.71 and 1 for $\alpha = 0.25$, $\alpha = 0.5$, $\alpha = 0.75$ and $\alpha = 1.00$, respectively.

The following observations are made for tables presented above:

- As expected, the problem gets harder to solve as α gets smaller. For a given instance of fixed backhaul percentage, the solution times for $\alpha = 0.25$ are quite reasonable. But it seems to increase sharply when compared to the

solution times for $\alpha = 0.5$, $\alpha = 0.75$ and $\alpha = 1.00$.

- The improved algorithm, which is the combination of the Default Algorithm and the acceleration procedures discussed in Chapter 4, reduces the computation time considerably as proposed. Table 5.4-5.6 exhibit the average results and % improvement in computation time for each α value.

$\%B = 0\%$	**Default Algorithm**			**Improved Algorithm**				
α	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # TPC	% Improvement in Time
0,25	741,93	18,02	20,58	625,07	12,70	13,42	21,72	15,75
0,5	310,31	7,44	7,72	218,28	6,06	6,22	8,58	29,66
0,75	27,48	1,42	1,49	25,69	1,20	1,24	1,89	6,52

Table 5.4: Averages and % Improvement in *Time*. ($\%B = 0$)

$\%B = 20\%$	**Default Algorithm**			**Improved Algorithm**				
α	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # TPC	% Improvement in Time
0,25	739,18	14,22	15,56	597,18	11,42	11,89	14,76	19,21
0,5	149,55	3,62	3,84	134,08	3,11	3,29	5,13	10,35
0,75	35,42	0,64	0,74	36,15	0,60	0,71	1,15	-2,06

Table 5.5: Averages and % Improvement in *Time*. ($\%B = 20$)

$\%B = 50\%$	**Default Algorithm**			**Improved Algorithm**				
α	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # TPC	% Improvement in Time
0,25	669,68	11,89	12,73	521,42	8,84	9,20	16,98	22,14
0,5	130,82	2,67	2,84	119,84	2,36	2,47	3,44	8,39
0,75	46,63	0,76	0,84	47,17	0,76	0,84	1,09	-1,15

Table 5.6: Averages and % Improvement in *Time*. ($\%B = 50$)

- For a given value of α, the computation time reduces as the backhaul percentage increases. This is because when all the customers are of type

linehaul (i.e. $\%B = 0$) the capacity required by the routes is higher than the cases when $\%B = 20$ and $\%B = 50$. Because when $\%B = 20$ and $\%B = 50$, backhaul customers are placed between linehaul customers which reduces the capacity required by the routes. This results in finding a feasible solution (set of m routes with $q(R_k) \leq Q_k$ for all $k = 1, \ldots, m$) earlier. In addition, since the capacity required by the routes reduces as the backhaul percentage increases, the utilization of the fleet also reduces.

Instances with a heterogenous fleet

There are 180 instances with heterogeneous fleet in the second set of problems. Exactly the same method is used to generate the problems in the second set. But, to have a heterogenous fleet of vehicles with different capacities, we adjusted the capacities of each vehicle so that they all differ while the total capacity of the vehicles ($m \cdot Q$) remains the same. For example, if we have $Q = 100$ and $m = 4$, we modify the capacities such that $Q_1 = 125$, $Q_2 = 113$, $Q_3 = 87$ and $Q_4 = 75$. We only give results for backhaul percentage of 0% and 50%, and $\alpha = 0.25$ and $\alpha = 0.50$.

The results from this set of instances is presented in Appendix B. Table 5.7 and 5.8 exhibit a summary of these results.

The results for the heteregenous case yields to the same observations as in homogenous case. Table 5.9 and 5.10 contain the average results and % improvement in computation time for each α value. The column *Avg. # MTEC* stands for the average number of tour elimination constraints idetified throughout the execution of the algorithm.

%B=0%		Default Algorithm				Improved Algorithm				
$\|\mathcal{C}\|$	α	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # MTEC	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # MTEC	Avg. # TPC
10	0.25	1.93	9.00	6.00	3.00	1.93	8.40	5.40	2.00	0.60
	0.50	1.77	4.60	4.60	0.00	1.62	4.20	4.20	0.00	0.60
20	0.25	11.23	15.20	9.60	5.80	9.18	13.80	8.20	5.80	5.80
	0.50	3.48	4.80	0.20	4.60	3.47	4.80	0.20	4.60	0.00
30	0.25	46.47	15.00	13.80	2.00	36.07	12.40	11.60	1.60	6.80
	0.50	11.68	3.75	3.00	0.75	8.61	2.75	2.50	0.50	4.00
40	0.25	137.60	14.60	12.80	2.20	111.85	11.60	10.80	1.20	12.80
	0.50	64.67	7.80	6.60	1.20	56.08	6.20	5.60	0.60	6.40
50	0.25	336.91	18.40	15.60	2.80	292.09	14.00	13.00	1.00	24.20
	0.50	94.71	6.20	5.40	0.80	84.24	5.00	4.40	0.60	13.60
60	0.25	863.48	27.40	24.60	2.80	675.12	19.00	17.80	1.20	28.80
	0.50	283.16	12.80	12.00	1.20	203.87	10.00	9.60	0.40	21.40
70	0.25	1.164.61	17.20	14.00	3.40	999.02	13.20	10.80	1.60	28.20
	0.50	461.07	12.00	10.00	2.00	403.81	9.60	8.00	1.60	8.80
80	0.25	1.823.69	19.40	17.20	2.80	1.558.03	15.40	14.00	1.40	27.80
	0.50	835.70	15.20	12.00	3.20	701.89	12.60	9.00	1.80	17.20
90	0.25	2.060.09	20.00	17.60	2.60	1.670.53	15.80	14.80	1.20	26.60
	0.50	1.506.83	17.20	14.40	3.20	1.292.92	13.20	11.60	2.00	22.20

Table 5.7: Average Results for 5 instances from data set 2. ($\%B = 0$)

Average utilization of the fleet is 0.92 and 0.87 for $\alpha = 0.25$ and $\alpha = 0.5$, respectively. Average utilization of the fleet is 0.87 and 0.83 for $\alpha = 0.25$ and $\alpha = 0.5$, respectively.

%B=50%		Default Algorithm				Improved Algorithm				
$\|\mathcal{C}\|$	α	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # MTEC	Avg. Time	Avg. # iter.	Avg. # TEC	Avg. # MTEC	Avg. # TPC
10	0.25	0.58	2.40	2.00	0.40	0.53	2.20	1.80	0.40	0.20
	0.50	0.36	1.00	1.00	0.00	0.37	1.00	1.00	0.00	0.40
20	0.25	13.69	12.40	12.40	0.60	11.85	11.00	11.00	0.60	3.00
	0.50	4.45	4.00	2.60	1.40	4.91	4.00	2.60	1.40	2.60
30	0.25	20.18	5.20	4.80	0.40	13.11	3.60	3.20	0.40	3.40
	0.50	7.44	1.80	1.40	0.40	5.32	1.40	1.00	0.40	2.60
40	0.25	100.11	12.40	11.00	1.80	80.62	8.60	7.60	0.80	10.60
	0.50	34.58	4.00	3.40	0.60	32.88	3.40	3.00	0.40	6.80
50	0.25	429.03	19.40	16.80	2.60	322.70	16.00	14.20	1.00	17.40
	0.50	33.94	5.80	4.20	1.60	31.62	4.40	3.40	1.00	10.20
60	0.25	479.10	12.20	10.80	1.00	383.87	9.40	8.60	0.80	17.60
	0.50	146.40	8.40	6.40	2.00	112.65	6.40	5.60	0.80	13.80
70	0.25	1.245.73	16.40	14.00	2.40	1.049.28	12.80	12.40	1.20	25.40
	0.50	286.46	8.80	7.00	1.80	268.50	7.00	6.00	1.00	11.40
80	0.25	1.565.90	17.60	15.00	2.80	1.386.98	14.00	12.40	1.60	23.80
	0.50	481.95	9.00	6.20	2.80	370.27	7.00	5.60	1.40	19.40
90	0.25	1.682.95	15.20	13.60	2.20	1.388.06	12.00	11.20	1.00	21.00
	0.50	1.413.27	12.80	10.20	2.60	1.151.94	9.80	9.00	1.00	24.00

Table 5.8: Average Results for 5 instances from data set 2. ($\%B = 50$)

$\%B = 0\%$	Default Algorithm				Improved Algorithm					
α	Avg. Time	Avg. # iter.	Avg. # MTEC	Avg. # TEC	Avg. Time	Avg. # iter.	Avg. # MTEC	Avg. # TEC	Avg. # TPC	% Impvmnt in Time
0,25	620,35	15,93	13,51	2,67	512,90	12,60	10,95	1,64	17,87	17,31
0,5	368,38	10,61	8,82	1,89	307,65	8,40	7,05	1,25	12,56	16,49

Table 5.9: Averages and % Improvement in *Time*. ($\%B = 0$) Heterogenous Fleet

$\%B = 50\%$	Default Algorithm				Improved Algorithm					
α	Avg. Time	Avg. # iter.	Avg. # MTEC	Avg. # TEC	Avg. Time	Avg. # iter.	Avg. # MTEC	Avg. # TEC	Avg. # TPC	% Impvmnt in Time
0,25	615,25	12,58	11,16	1,58	515,22	9,96	9,16	0,87	13,60	16,26
0,5	267,65	6,18	4,71	1,47	219,83	4,93	4,13	0,82	10,13	17,87

Table 5.10: Averages and % Improvement in *Time*. ($\%B = 50$) Heterogenous Fleet

Chapter 6

Conclusion

In this thesis, we discussed about the Vehicle Routing Problem (VRP) which is an important management problem in the field of distribution and logistics. We proposed an exact algorithm for a generalization of the problem, Vehicle Routing Problem with Backhauls (VRPB).

Many solution methods are proposed for the VRP and VRPB. These methods seem to be designed for just one version of the problem, and are generally obtained by modifying an already existing method. There are a number of exact algorithms for the VRP. On the other hand, we encountered just two exact algorithms for the VRPB [59], [68]. In both of the methods proposed, it is obligatory that the backhaul customers come after the linehaul customers in a given vehicle route. The algorithm we propose is unique in the sense that a combination of different aspects of the applications is handled. The algorithm works when the vehicle fleet is homogenous or heterogenous. Although the algorithm is designed for the VRPBs, it can be used to solve VRPs by simply setting the number of backhaul customers to zero. Finally, to our knowledge, we proposed the only exact algorithm which allows routes composed of linehaul and backhaul customers in any sequence in a vehicle route. We also allow routes of only backhaul or only linehaul customers.

The proposed method is based on iteratively solving a relaxation of the

VRPB, namely Multiple Traveling Salesman Problem (m-TSP). At each iteration, infeasible solutions for the VRPB are identified and seperated from the feasible set of the m-TSP by means of proper cuts. The procedures for the identification of infeasible solutions and cuts to eliminate these infeasible solutions were discussed. To improve the algorithm, several acceleration procedures which are based on local search methods are also presented. Chapter 4 gives the details of the algorithm.

The algorithm is tested on randomly generated problems, involving up to 90 customers. Although the problem is $\mathcal{NP}$-hard, the proposed algorithm is quite fast in finding the optimum solution. The acceleration procedures are proved to be very effective.

An area of further research may be the application of the proposed algorithm for different versions of the VRPs and VRPBs. The algorithm can be modified easily so that it can be used to solve VRPs with restrictions on the duration and/or distance of the routes. It can also be adapted or the VRPs with time windows. The number of infeasible solutions will be much more than the cases we examined but strong cuts that can eliminate several of them at the same time can be investigated.

Yet another area for further research can be on the acceleration procedures. We applied three local search operators. Additional operators may be useful in identifying more candidate solutions. We propose local search operators that only involve relocation of single nodes of the string. Instead of single nodes, relocation of paths can also be considered. Alternatively, Lin-Kernighan [56] type improvement heuristics, which allow *r-exchange* while r can change in each iteration, can be modified to be used for the algorithm we propose.

Bibliography

[1] ACCUTHAN, N.R., and CACCETTA, L., 1991, 'Integer Linear Programming Formulation for a Vehicle Routing Problam', *European Journal of Operational Research, (52)*, 86-89.

[2] AKTURK, M.S.,YILMAZ,H., 1996, 'Scheduling of Automated Guided Vehicles in a Decision Making Hierarchy', *International Journal of Production Research, 34 (2)*, 577-591.

[3] APPLEGATE, D., BIXBY, R., CHVATAL, V., COOK, W., 1998, 'On the solution of Traveling Salesman Problems', *Documenta Mathematica, Extra Volume ICM, III*, 645-656.

[4] ARAQUE, J.R., KUDVA, G., MORIN, T.L., PEKNY, J.F., 1994, 'A Branch and Cut Algorithm for the Vehicle Routing Problems', *Annals of Operations Research (50)*, 37-59.

[5] ASSAD, A.A., 1988, 'Modeling and Implementation Issues in Vehicle Routing', In: Golden, B., Assad, A.(Eds), Vehicle Routing: Methods and Studies, North-Holland, Amsterdam, pp 7-45.

[6] BAKER, B.M., SHEASBY, J., 1999, 'Entensions to the Generalized Assignment Heuristic for Vehicle Routing', *European Journal of Opreational Research (119)*, 147-157.

[7] BALINSKI, M., QUANDT, R., 1964, 'On an Integer Program for a Delivery Problem', *Operatons Research (12)*, 300-304.

[8] BARD, J.F., LIU, H., MOSHE, D., PATRICK, J., 1998, 'Branch and Cut Algorithm for the VRP with Satellite Facilities', *IIE Transactions (30-9)*, 821-834.

[9] BARTHOLDI, J.J., PLATZMAN, L.K., 1988, 'Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space', *Management Science (34-3)*, 291-305.

[10] BEKTAŞ, T., 2000, 'Construction of the Subtour Elimination Constraints of the Traveling Salesman Problem and its Extensions', *Masters' Thesis, Başkent University, Ankara, TURKEY.*

[11] BELLMORE, M., HONG, S., 1974, 'Transformation of the Multisalesmen Problem to the Standard Traveling Salesman Problem', *Journal of Association for Computing Machinery (21-3)*, 500-504.

[12] BELLMORE, M., NEMHAUSER, G.L., 1968, 'The Traveling Salesman Problem: A Survey', *Operations Research (16)*, 538-558.

[13] BODIN, L.D., GOLDEN, B.L., ASSAD, A., BALL, M., 1983, 'Routing and scheduling of vehicles and crews. The state of the art.', *Computers and Operations Research (10)*, 69-211.

[14] BOWERMAN, R., HALL, B., CALAMAI, P., 1995, 'A multi-objective optimization approach to urban school bus routing: Formulation and solution method', *Tranportation Research Part A:Policy and Practice*, 107-123.

[15] BRACA, J., BRAMEL, J., POSNER, B., SIMCHI-LEVI, D., 1997, 'Computerized Approach to the New York City School Bus Routing Problem', *IIE Transactions (29-8)*, 693-702.

[16] BURKARD, R.E., 1979, 'Traveling Salesman and Assignment Problems: A "Survey",' In: P.L. Hammer, E.L. Johnson, and B.H. Korte, editors., Discrete Optimization 1 , Annals of Discrete Mathematics 4, North-Holland, Amsterdam, pp.

[17] CASCO, D., GOLDEN, B., WASIL. E., 1988, 'Vehicle Routing with Backhauls: Models, algorithms and case studies. In: Golden, B., Assad, A., editors, Vehicle Routing: Methods and Studies.' North-Holland, Amsterdam, pp 127-147.

[18] CHAN, L.M.A., MURIEL, A., SIMCHI-LEVI, D., 1998, 'Parallel Machine Scheduling, Linear Programming, and Parameter List Scheduling Heuristics', *Operations Research (46-5)*, 729-741.

[19] CHIRISTIANSEN, M., 1999, 'Decomposion of Combined Inventory and Time Constraint Ship Routing Problem', *Transportation Science (33-1)*, 3-16.

[20] CHRISTOFIDES, N. and EILON, S., 1969, 'An Algorithm for the Vehicle-Dispatching Problem', *Oper. Res. Quart. (20)*, 51-518.

[21] CHRISTOFIDES, N. , MINGOZZI, A., TOTH, P., 1979, 'The Vehicle Routing Problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C., editors, Combinatorial Optimization, chapter 11', John Wiley and Sons.

[22] CHRISTOFIDES, N. , MINGOZZI, A., TOTH, P., 1980, 'Exact Algorithms For the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations', *Math. Programming. (20)*, 255-282.

[23] CHRISTOFIDES, N., 1985, 'Vehicle Routing, In: Lawler, E.I., Lenstra, K., Rinnooy Kan, A. H. G., Shmoys, D. B., editors, The Traveling Salesman Problem', John Wiley & Sons, pp 431-448.

[24] CLARKE, G. and WRIGHT, J.W., 1964, 'Scheduling of Vehicles from a Central Depot to a Number of Delivery Points', *Operations Research (12)*, 568-581.

[25] CORBERÁN, A., LETCHFORD, A.N., SANCHIS, J.M., 2001, 'A Cutting Plane Algorithm for the General Routing Problems', *Mathematical Programming (Ser. A)*, DOI=10.1007/s101070100219.

[26] CORNUEJOLS, G, HARCE, F., 1993, 'Polyhedral Study of the Capacitated Vehicle Routing Problem', *Mathematical Programming (60)*, 21-52. 193-215.

[27] CROES, G.A., 1958, 'A method for Solving Traveling Salesman Problems', *Operations Research (6)*, 791-812.

[28] CROWDER, H., PADBERG, M.W., 1980, 'Solving Large-Scale Symmetric Traveling Salesman Problems to Optimality', *Management Science (26)*, 495-509.

[29] DANTZIG, G.B., FULKERSON, R., JOHNSON, S.M., 1954, 'Solution of a Large Scale Travelling Salesman Problem', *Management Science (6)*, 80-91.

[30] DANTZIG, G.B., RAMSER, J.H., 1959, 'The Truck Dispatching Problem', *Management Science (6)*, 80-91.

[31] DESROCHERS, M., LAPORTE, G., 1991, 'Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints', *Operational Research Letters (10)*, 27-36.

[32] DESROSIERS, J., SOUMIS, F., DESROCHERS, M., 1984, 'Routing with Time Windows by Column Generation', *Networks (14)*, 545-565.

[33] EILON, S., WATSON-GANDY, C.D.T., CHRISTOFIDES, N, 1971, 'Distribution Management: Mathematical Programming and Practical Analysis', Griffin, London.

[34] FEDERGRUEN, A., ZIPKIN, P., 1984, 'A Combined Vehicle Routing and Inventory Allocation Problem', *Operations Research (32-5)*, 1019-1037.

[35] FISHER, M., 1995, 'Vehicle Routing. In: Ball, M.O., Magnanti. T.L., Monma, C.L., Nemhauser, G.L., editors, Network Routing', Volume 8 Elseiver Science B.V., North Holland, pp.1-30.

[36] FISHER, M., JAIKUMAR, R., 1981, 'A Generalized Assignment Heuristic for Vehicle Routing', *Networks (11)*, 109-124.

[37] GAVISH, B., GRAVES, S.C., 1978, 'The Traveling Salesman Problem and Related Problems', *Working Paper GR-078-78, Operations Research Center, Massachusetts Institute of Technology.* 315-338.

[38] GAVISH, B., SRIKANTH, K., 1986, 'An Optimal Solution Method for the Large-Scale Multiple Traveling Salesman Problems', *Operations Research (34-5)*, 698-717.

[39] GENDREAU, M., LAPORTE, G., POTVIN, J.Y., 1997, 'Vehicle Routing: Modern Heuristics', In: Aarts, E., Lenstra, J.K., eds., Local Search in Combinatorial Optimization, John Wiley & Sons, pp. 311-336.

[40] GILLET, B., MILLER, L., 1974, 'A Heuristic Algorithm for the Vehicle Dispatch Problem', *Operations Research (22)*, 340-349.

[41] GOETSCHALCKX, M., JACOBS-BLESCHA, C., 1989, 'The Vehicle Routing Problem with Backhauls', *European Journal of Operational Research (42)*, 39-51.

[42] GOLDEN, B., ASSAD, A. (Eds), 1988, 'Vehicle Routing: Methods and Studies. North-Holland, Amsterdam,'.

[43] ILOG Inc.,1997, 'Using the CPLEX Callable Library', CPLEX Division, 930 Tahoe Blvd., #802-9436 Incline Village, NV 89451-9436, USA.

[44] JONKER, R., VOLGENANT, T., 1988, 'An Improved Transformation of the Symmetric Multiple Traveling Salesman Problem', *Operations Research (36-1)*, 163-167.

[45] KEARNEY, A.T., Inc., 1984, 'Measuring and Improving Productivity in Physical Distribution', *A report prepared for the National Council of Physical Distribution Management*, Oak Brook, II.

[46] KIKUCHI, S., 1984, 'Scheduling of Demand Responsive Transit Vehicles', *Journal of Tranportation Engineering (110)*, 511-520.

[47] KOLEN, A., RINNOOY Kan, A., TRIENEKENS, H., 1987, 'Vehicle Routing with Time Windows', *Operations Research (35)*, 266-273.

[48] LAPORTE, G., 1992, 'The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms', *European Journal of Operational Research (59)*, 231-247.

[49] LAPORTE, G., 1992, 'The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms', *European Journal of Operational Research (59)*, 345-358.

[50] LAPORTE, G., NOBERT, Y., 1980, 'A Cutting Plane Algorithm for the m-Salesmen Problem', *Operational Research Quarterly (31)*, 231-247.

[51] LAPORTE, G., NOBERT, Y., 1987, 'Exact Algorithms for the Vehicle Routing Problem', *Annals of Discrete Mathematics (31)*, 147-184.

[52] LAPORTE, G., MERCURE, H., NOBERT, Y., 1987, 'An Exact Algorithms for the Asymmetrical Capacitated Vehicle Routing Problem', *Networks (161)*, 33-46.

[53] LAPORTE, G., NOBERT, Y., DESROCHERS, M., 1985, 'Optimal Routing Under Capacity and Distance Restrictions', *Operations Research (33)*, 1050-1073.

[54] LAWLER, E.L., LENSTRA, J.K., RINNOOY KAN, A., SHMOYS, D.B., editors, 1985, 'The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization',Wiley, New York.

[55] LENSTRA, J., RINNOOY, KAN A., 1981, 'Complexity of Vehicle Routing and Scheduling Problems', *Networks (11)*, 221-228.

[56] LIN, S., KERNIGHAN, B.W., 1973, 'An Effective Heuristic Algorithm for the Traveling-Salesman Problem', *Operatons Research (21)*, 498-516.

[57] MAGNANTI, T.L., 1981, 'Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects', *Networks (11)*, 179-214.

[58] MILLER, C.E., TUCKER, A.W., ZEMLIN, R.A., 1960, 'Integer Linear Programming Formulations of Traveling Salesman Problems', *Journal of Association for Computing Machinery (7)*, 326-329.

[59] MINGOZZI, A., GIORGI, S., 1999, 'An Exact Method for the Vehcile Routing Problem with Backhauls', *Transportation Science (33-3)*, 315-329.

[60] ORLOF, C., 1976, 'Route-constrained Fleet Scheduling', *Transportation Science (10)*, 149-168.

[61] PADBERG, M.W., RINALDI, G., 1990, 'An Efficient Algorithm for the Minimum Capacity Cut Problem', *Mathematical Programming (47)*, 19-36.

[62] PSARAFTIS, H., 1980, 'A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-ride Problem', *Transportaton Scence*, 130-154.

[63] REIMAN, M.I., 1999, 'Heavy Traffic Analysis of the Dynamic Stochastic Inventory-Routing Problem', *Transportation Science (33-4)*, 361-380.

[64] SHIH, L., LIN, Y., 1999, 'Optimal Routing for Infectious Waste Collection', *Journal of Environmental Engineering (125-5)*, 479-484.

[65] STEIN, D., 1978, 'Scheduling Dial-a-ride Transportation System', *Tranportation Science (12)*, 232-249.

[66] SVESTKA, J.A., HUCKFELT, V.E., 1973, 'Computational Experience with an m-Salesman Traveling Problem Algorithm', *Management Science (19-7)*, 790-799.

[67] TEODOROVIC, D., RADIVOJEVIC, G., 2000, 'Fuzzy Logic Approach to Dynamic Dial-A-ride Problem', *Fuzzy Sets and Systems (116-1)*, 23-33.

[68] TOTH, P., VIGO, D., 1997 'An Exact Algorithm for the Vehcile Routing Problem with Backhauls', *Transportation Science (31-4)*, 372-385.

[69] TOTH, P., VIGO, D., 1999, 'A Heuristic Algorithm for the Symmetric and Asymmetric Vehicle Routing Problems with Backhauls', *European Journal of Operational Research (113)*, 528-543.

[70] TUNG, D.V., PINNOI, A., 2000, 'Vehicle Routing-Scheduling for Waste Collection in Hanoi', *European Journal of Operational Research (125-3)*, 449-468.

[71] YANO, C.A., Chan, T., RICHTER, L.K., MURTY, K.G., McGETTIGAN, D., 1987, 'Vehicle Routing at Quality Stores', *Interfaces (17)*, 52-63.

Appendix

Appendix A

Test Results for Instances with Homogenous Fleet

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
10	0	0,27	1	1	0,32	1	1	0
	0	0,73	5	6	0,64	3	4	6
	0	0,93	6	6	0,93	6	6	0
	0	0,75	5	5	0,89	5	5	6
	0	2,31	10	11	2,05	8	8	13
AVG:	0	1,00	5,40	5,80	0,97	4,60	4,80	5,00
	20	0,35	1	1	0,36	1	1	0
	20	3,28	14	14	2,67	11	11	5
	20	0,39	2	2	0,41	2	2	2
	20	0,79	4	4	0,75	4	4	1
	20	0,28	2	2	0,30	2	2	0
AVG:	20	1,02	4,60	4,60	0,90	4,00	4,00	1,60
	50	0,81	5	6	0,70	4	5	1
	50	2,75	12	12	2,17	10	10	6
	50	0,95	5	5	0,79	4	4	5
	50	0,25	2	2	0,32	2	2	5
	50	1,44	9	9	1,59	8	8	15
AVG:	50	1,24	6,60	6,80	1,11	5,60	5,80	6,40

Table A.1: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
10	0	0,58	3	3	0,49	3	3	0
	0	0,12	0	0	0,15	0	0	0
	0	0,81	4	4	0,75	3	3	3
	0	0,25	1	1	0,23	1	1	0
	0	1,95	8	9	2,01	8	9	0
AVG:	0	0,74	3,20	3,40	0,73	3,00	3,20	0,60
	20	0,23	0	0	0,24	0	0	0
	20	2,12	9	9	1,98	7	7	5
	20	0,21	1	1	0,22	1	1	1
	20	0,32	2	2	0,35	2	2	1
	20	0,12	0	0	0,15	0	0	0
AVG:	20	0,60	2,40	2,40	0,59	2,00	2,00	1,40
	50	0,30	1	1	0,33	1	1	0
	50	0,36	1	1	0,35	1	1	0
	50	0,22	1	1	0,29	1	1	1
	50	0,15	0	0	0,17	0	0	0
	50	0,16	0	0	0,18	0	0	3
AVG:	50	0,238	0,6	0,6	0,264	0,6	0,6	0,8

Table A.2: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
10	0	0,26	1	1	0,28	1	1	0
	0	0,16	0	0	0,13	0	0	0
	0	0,35	0	0	0,37	0	0	0
	0	0,16	0	0	0,21	0	0	0
	0	1,16	5	5	1,18	4	4	2
AVG:	0	0,42	1,20	1,20	0,43	1,00	1,00	0,40
	20	0,12	0	0	0,14	0	0	0
	20	1,54	4	4	1,06	3	3	2
	20	0,11	0	0	0,16	0	0	1
	20	0,19	0	0	0,20	0	0	0
	20	0,12	0	0	0,16	0	0	0
AVG:	20	0,42	0,80	0,80	0,34	0,60	0,60	0,60
	50	0,19	0	0	0,16	0	0	0
	50	0,40	1	1	0,41	1	1	0
	50	0,14	0	0	0,18	0	0	0
	50	0,09	0	0	0,14	0	0	0
	50	0,19	0	0	0,17	0	0	0
AVG:	50	0,20	0,20	0,20	0,21	0,20	0,20	0,00

Table A.3: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
10	0	0,11
	0	0,15
	0	0,16
	0	0,09
	0	0,10
AVG:	0	0,12
	20	0,13
	20	0,18
	20	0,15
	20	0,18
	20	0,14
AVG:	20	0,16
	50	0,20
	50	0,12
	50	0,21
	50	0,18
	50	0,12
AVG:	50	0,17

Table A.4: Results for 5 instances. ($\alpha = 1.00$)

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
20	0	7,33	8	11	6,57	7	10	9
	0	18,62	21	25	15,03	15	15	26
	0	2,90	7	8	2,95	7	8	0
	0	0,25	0	0	0,28	0	0	0
	0	9,90	20	26	5,83	12	13	15
AVG:	0	7,80	11,2	14	6,13	8,2	9,2	10
	20	1,18	2	2	0,52	1	1	3
	20	1,44	3	3	1,24	1	1	3
	20	18,61	11	13	14,92	7	8	23
	20	4,17	7	7	2,37	4	4	16
	20	6,48	7	8	5,96	6	6	1
AVG:	20	6,38	6,00	6,60	5,00	3,80	4,00	9,20
	50	0,98	1	2	1,18	1	1	5
	50	6,11	7	7	5,92	6	6	8
	50	7,53	15	17	6,27	12	13	6
	50	16,84	12	13	15,48	11	12	1
	50	5,55	8	8	4,73	6	6	10
AVG:	50	7,40	8,6	9,4	6,72	7,2	7,6	6

Table A.5: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
20	0	7,67	9	9	7,72	9	9	0
	0	23,07	15	15	15,10	12	12	16
	0	2,31	3	3	2,32	3	3	0
	0	4,83	6	0	5,03	6	6	4
	0	0,50	1	1	0,65	1	1	1
AVG:	0	7,68	6,8	5,6	6,16	6,2	6,2	4,2
	20	0,78	0	0	0,83	0	0	0
	20	0,29	0	0	0,30	0	0	0
	20	3,92	6	6	3,21	4	5	9
	20	1,65	3	3	1,56	2	2	6
	20	3,37	3	3	3,45	3	3	0
AVG:	20	2,00	2,4	2,4	1,87	1,8	2	3
	50	0,60	0	0	0,76	0	0	1
	50	4,11	4	4	5,05	4	4	7
	50	2,12	3	3	2,52	3	3	0
	50	21,16	11	11	21,31	11	11	2
	50	4,16	4	4	4,30	4	4	0
AVG:	50	6,43	4,4	4,4	6,79	4,4	4,4	2

Table A.6: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
20	0	0,40	0	0	0,37	0	0	0
	0	20,10	14	14	13,80	9	9	13
	0	0,30	0	0	0,35	0	0	0
	0	0,91	1	1	0,92	1	1	1
	0	0,29	0	0	0,35	1	1	1
AVG:	0	4,40	3	3	3,16	2,2	2,2	3
	20	0,77	0	0	0,80	0	0	0
	20	0,30	0	0	0,25	0	0	0
	20	0,31	0	0	0,33	0	0	0
	20	0,66	1	1	0,72	1	1	4
	20	0,30	0	0	0,32	0	0	0
AVG:	20	0,47	0,2	0,2	0,48	0,2	0,2	0,8
	50	2,12	4	5	2,06	4	5	5
	50	2,85	3	3	2,94	3	3	2
	50	0,68	0	0	0,72	0	0	0
	50	7,18	4	4	7,23	4	4	0
	50	0,34	0	0	0,39	0	0	0
AVG:	50	2,63	2,2	2,4	2,67	2,2	2,4	1,4

Table A.7: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
20	0	0,39
	0	0,72
	0	0,43
	0	0,37
	0	0,35
AVG:	0	0,45
	20	0,36
	20	0,35
	20	0,39
	20	1,28
	20	0,35
AVG:	20	0,55
	50	0,66
	50	0,45
	50	0,69
	50	1,27
	50	0,44
AVG:	50	0,70

Table A.8: Results for 5 instances. ($\alpha = 1.00$)

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
30	0	188,25	37	44	56,21	14	18	21
	0	80,31	9	9	78,60	8	8	17
	0	4,27	2	2	5,40	2	2	13
	0	30,02	16	17	24,78	15	16	19
	0	50,58	15	15	52,30	13	13	24
AVG:	0	70,69	15,8	17,4	43,46	10,4	11,4	18,8
	20	80,80	20	23	82,50	20	23	1
	20	8,81	6,00	7,00	8,92	6,00	7,00	0,00
	20	23,98	12	13	22,15	11	12	8
	20	15,86	10	11	14,01	8	8	7
	20	33,80	14	15	30,85	13	14	1
AVG:	20	32,65	12,4	13,8	31,69	11,6	12,8	3,4
	50	3,95	1	1	4,14	1	1	2
	50	45,23	9	9	32,59	7	7	15
	50	26,64	12	13	32,12	12	13	11
	50	8,78	5	5	8,88	5	5	0
	50	15,02	8	8	13,15	7	7	1
AVG:	50	19,92	7	7,2	18,18	6,4	6,6	5,8

Table A.9: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
30	0	27,43	6,00	6,00	28,12	6,00	6,00	10,00
	0	36,41	2,00	3,00	27,60	1,00	1,00	4,00
	0	6,65	2,00	2,00	7,65	2,00	2,00	5,00
	0	4,28	1,00	1,00	4,44	1,00	1,00	2,00
	0	1,85	0,00	0,00	2,11	0,00	0,00	0,00
AVG:	0	15,32	2,20	2,40	13,98	2,00	2,00	4,20
	20	15,81	4,00	4,00	16,41	4,00	4,00	1,00
	20	2,68	1,00	1,00	2,69	1,00	1,00	0,00
	20	10,84	3,00	3,00	11,02	3,00	3,00	0,00
	20	7,13	4,00	4,00	5,29	2,00	3,00	6,00
	20	52,26	13,00	13,00	48,18	11,00	11,00	5,00
AVG:	20	17,74	5,00	5,00	16,72	4,20	4,40	2,40
	50	2,05	0,00	0,00	2,26	0,00	0,00	0,00
	50	8,45	2,00	3,00	9,12	2,00	3,00	5,00
	50	12,17	4,00	4,00	10,19	3,00	3,00	4,00
	50	28,41	4,00	4,00	29,56	4,00	4,00	0,00
	50	4,16	3,00	3,00	3,96	2,00	2,00	2,00
AVG:	50	11,05	2,60	2,80	11,02	2,20	2,40	2,20

Table A.10: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
30	0	4,17	0	0	4,12	0	0	0
	0	36,61	2	2	31,15	1	1	5
	0	3,47	1	1	4,16	1	1	3
	0	2,24	0	0	2,27	0	0	0
	0	1,87	0	0	1,86	0	0	0
AVG:	0	9,67	0,6	0,6	8,71	0,4	0,4	1,6
	20	6,04	1	1	6,23	1	1	0
	20	0,82	0	0	0,85	0	0	0
	20	2,59	0	0	2,42	0	0	0
	20	0,81	0	0	0,85	0	0	0
	20	1,53	1	1	1,79	1	1	0
AVG:	20	2,36	0,4	0,4	2,43	0,4	0,4	0
	50	2,20	0	0	2,32	0	0	0
	50	5,34	0	0	6,02	0	0	0
	50	9,85	2	3	10,42	2	3	3
	50	4,13	0	0	4,33	0	0	0
	50	1,36	0	0	1,43	0	0	0
AVG:	50	4,58	0,4	0,6	4,90	0,4	0,6	0,6

Table A.11: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
30	0	0,93
	0	7,18
	0	2,38
	0	2,04
	0	5,45
AVG:	0	3,60
	20	1,83
	20	2,25
	20	1,43
	20	3,95
	20	0,93
AVG:	20	2,08
	50	2,12
	50	5,54
	50	2,19
	50	4,74
	50	5,81
AVG:	50	4,08

Table A.12: Results for 5 instances. ($\alpha = 1.00$)

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
40	0	395,30	57	68	213,75	9	11	17
	0	40,19	9	13	41,59	9	13	4
	0	228,16	17	18	186,73	13	14	14
	0	257,62	19	21	180,79	11	11	21
	0	360,41	39	42	310,12	21	23	19
AVG:	0	256,34	28,20	32,40	186,60	12,60	14,40	15,00
	20	165,70	23	25	171,12	23	24	15
	20	33,78	4	4	34,69	4	4	0
	20	8,26	2	2	7,16	1	1	3
	20	40,50	8	8	15,18	3	3	2
	20	47,21	10	11	16,44	5	6	7
AVG:	20	59,09	9,40	10,00	48,92	7,20	7,60	5,40
	50	112,53	10	10	107,56	8	8	19
	50	270,41	20	23	255,53	16	16	21
	50	162,51	13	15	109,91	8	9	8
	50	63,02	7	7	60,23	6	6	13
	50	10,95	2	2	11,66	2	2	1
AVG:	50	123,88	10,40	11,40	108,98	8,00	8,20	12,40

Table A.13: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
40	0	28,11	4	4	29,01	4	4	2
	0	31,12	5,00	5,00	33,12	5,00	5,00	3,00
	0	36,79	4,00	6,00	35,83	3,00	4,00	6,00
	0	55,56	7,00	7,00	49,67	5,00	6,00	5,00
	0	79,12	9,00	10,00	75,41	9,00	9,00	4,00
AVG:	0	46,14	5,80	6,40	44,61	5,20	5,60	4,00
	20	6,02	2,00	2,00	5,91	1,00	1,00	3,00
	20	54,20	1,00	1,00	57,65	1,00	1,00	0,00
	20	25,51	2,00	2,00	24,12	2,00	2,00	1,00
	20	73,69	7,00	7,00	69,12	7,00	7,00	2,00
	20	50,32	8,00	8,00	42,33	7,00	7,00	1,00
AVG:	20	41,95	4,00	4,00	39,83	3,60	3,60	1,40
	50	4,92	2,00	2,00	4,11	1,00	1,00	5,00
	50	20,15	1,00	1,00	20,75	1,00	1,00	2,00
	50	14,47	5,00	5,00	10,54	4,00	4,00	3,00
	50	9,95	1,00	1,00	10,95	1,00	1,00	0,00
	50	72,93	3,00	3,00	73,99	3,00	3,00	0,00
AVG:	50	24,48	2,40	2,40	24,07	2,00	2,00	2,00

Table A.14: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
40	0	24,31	3	3	26,13	3	3	2
	0	26,30	3	3	27,19	3	3	1
	0	35,20	4	5	32,87	3	3	5
	0	49,66	4	4	51,12	4	4	3
	0	6,44	0	0	6,90	0	0	0
AVG:	0	28,38	2,8	3	28,84	2,6	2,6	2,2
	20	3,95	0	0	3,99	0	0	0
	20	30,18	0	0	30,03	0	0	0
	20	21,60	2	2	20,53	2	2	0
	20	5,41	1	1	5,56	1	1	2
	20	1,99	0	0	1,89	0	0	0
AVG:	20	12,63	0,6	0,6	12,40	0,6	0,6	0,4
	50	4,08	0	0	4,12	0	0	0
	50	11,58	0	0	12,02	0	0	0
	50	4,35	0	0	4,46	0	0	0
	50	6,36	0	0	7,21	0	0	0
	50	12,63	0	0	12,68	0	0	0
AVG:	50	7,80	0	0	8,10	0	0	0

Table A.15: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
40	0	4,57
	0	11,10
	0	19,34
	0	9,15
	0	17,44
AVG:	0	12,32
	20	8,10
	20	15,73
	20	14,38
	20	7,43
	20	4,61
AVG:	20	10,05
	50	5,99
	50	10,02
	50	7,72
	50	4,03
	50	19,04
AVG:	50	9,36

Table A.16: Results for 5 instances. ($\alpha = 1.00$)

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
50	0	75,57	5	5	81,45	5	5	24
	0	235,22	14	18	215,87	12	13	20
	0	841,14	23	26	819,10	20	21	32
	0	949,73	42	56	452,37	17	21	52
	0	412,30	10	11	389,65	9	9	21
AVG:	0	502,79	18,80	23,20	391,69	12,60	13,80	29,80
	20	21,35	3	3	22,61	3	3	4
	20	209,13	12	12	200,00	11	11	16
	20	184,91	13	13	144,60	10	10	37
	20	296,58	14	14	256,21	13	13	11
	20	312,59	12	15	280,56	10	11	16
AVG:	20	204,91	10,80	11,40	180,80	9,40	9,60	16,80
	50	35,83	4	4	17,73	2	2	10
	50	654,12	17	19	612,29	14	15	41
	50	127,53	9	9	120,14	8	8	16
	50	108,16	7	7	105,23	6	6	13
	50	1.096,91	44	49	545,31	30	35	89
AVG:	50	404,51	16,20	17,60	280,14	12,00	13,20	33,80

Table A.17: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
50	0	145,77	10	10	135,16	9	9	32
	0	36,49	3,00	3,00	39,10	3,00	3,00	7,00
	0	30,75	2,00	2,00	33,41	2,00	2,00	7,00
	0	226,13	10,00	10,00	205,10	8,00	8,00	15,00
	0	86,88	4,00	4,00	91,12	4,00	4,00	14,00
AVG:	0	105,20	5,80	5,80	100,78	5,20	5,20	15,00
	20	54,68	1,00	1,00	58,79	1,00	1,00	3,00
	20	80,22	3,00	3,00	77,05	3,00	3,00	3,00
	20	169,00	5,00	5,00	103,11	3,00	3,00	2,00
	20	46,87	0,00	0,00	45,02	0,00	0,00	0,00
	20	95,60	4,00	4,00	99,16	4,00	4,00	6,00
AVG:	20	89,27	2,60	2,60	76,63	2,20	2,20	2,80
	50	15,12	1,00	1,00	16,21	1,00	1,00	6,00
	50	25,84	2,00	2,00	24,99	0,00	0,00	2,00
	50	32,41	2,00	3,00	26,41	1,00	1,00	6,00
	50	23,60	0,00	0,00	24,54	0,00	0,00	0,00
	50	86,19	6,00	6,00	84,83	5,00	5,00	8,00
AVG:	50	36,63	2,20	2,40	35,40	1,40	1,40	4,40

Table A.18: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
50	0	7,11	0	0	7,16	0	0	0
	0	12,59	1	1	13,25	1	1	5
	0	7,19	0	0	7,25	0	0	0
	0	41,28	2	2	33,16	2	2	3
	0	11,20	0	0	11,16	0	0	0
AVG:	0	15,87	0,6	0,6	14,40	0,6	0,6	1,6
	20	30,89	0	0	34,16	0	0	2
	20	13,20	0	0	13,45	0	0	0
	20	15,84	0	0	15,01	0	0	0
	20	47,00	0	0	46,23	0	0	0
	20	68,21	1	1	70,19	1	1	2
AVG:	20	35,03	0,2	0,2	35,81	0,2	0,2	0,8
	50	5,27	0	0	5,32	0	0	0
	50	20,14	0	0	21,50	0	0	0
	50	24,57	1	1	25,36	1	1	4
	50	24,11	0	0	25,27	0	0	0
	50	68,17	4	4	67,11	4	4	3
AVG:	50	28,45	1	1	28,91	1	1	1,4

Table A.19: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
40	0	61,37
	0	17,55
	0	14,39
	0	23,97
	0	38,64
AVG:	0	31,18
	20	96,12
	20	11,72
	20	21,21
	20	16,21
	20	29,51
AVG:	20	34,95
	50	9,65
	50	22,35
	50	16,21
	50	28,10
	50	19,26
AVG:	50	19,11

Table A.20: Results for 5 instances. ($\alpha = 1.00$)

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
60	0	351,69	16	21	221,27	11	15	35
	0	107,24	6	9	90,55	4	4	19
	0	1.558,92	55	76	1.004,60	32	36	49
	0	1.387,73	46	51	990,64	36	37	33
	0	619,84	21	21	424,00	14	14	27
AVG:	0	805,08	28,80	35,60	546,21	19,40	21,20	32,60
	20	1.292,41	20	23	286,18	9	10	35
	20	144,41	8	9	120,60	6	7	15
	20	942,79	27	35	446,93	19	20	34
	20	446,68	9	14	301,18	7	7	12
	20	1.020,11	30	36	884,24	21	23	37
AVG:	20	769,28	18,80	23,40	407,83	12,40	13,40	26,60
	50	100,37	3	3	89,27	3	3	3
	50	646,00	9	9	170,36	1	1	5
	50	849,97	11	12	301,61	4	4	9
	50	783,71	22	23	264,54	9	10	35
	50	944,79	15	15	287,12	2	3	11
AVG:	50	664,97	12,00	12,40	222,58	3,80	4,20	12,60

Table A.21: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
60	0	1.807,21	26	26	586,58	10	10	22
	0	41,06	1,00	1,00	42,98	1,00	1,00	6,00
	0	256,82	11,00	11,00	145,24	7,00	7,00	11,00
	0	58,21	1,00	1,00	59,93	1,00	1,00	0,00
	0	71,21	2,00	2,00	73,15	2,00	2,00	1,00
AVG:	0	446,90	8,20	8,20	181,58	4,20	4,20	8,00
	20	215,91	1,00	1,00	219,35	1,00	1,00	3,00
	20	159,86	1,00	1,00	160,64	1,00	1,00	0,00
	20	284,09	2,00	2,00	278,64	1,00	1,00	4,00
	20	167,25	1,00	2,00	169,19	1,00	2,00	2,00
	20	186,64	2,00	3,00	190,02	2,00	3,00	7,00
AVG:	20	202,75	1,40	1,80	203,57	1,20	1,60	3,20
	50	741,89	10,00	10,00	475,92	8,00	8,00	3,00
	50	60,83	1,00	2,00	61,12	1,00	2,00	3,00
	50	256,23	5,00	5,00	205,61	3,00	3,00	6,00
	50	214,91	1	1	216,16	1,00	1,00	2,00
	50	59,19	1,00	1,00	61,21	1,00	1,00	9,00
AVG:	50	266,61	3,60	3,80	204,00	2,80	3,00	4,60

Table A.22: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
60	0	28,25	0	0	28,11	0	0	0
	0	38,33	0	0	38,15	0	0	0
	0	35,26	2	2	37,74	2	2	0
	0	41,27	2	2	43,13	2	2	0
	0	39,99	2	3	41,17	2	3	2
AVG:	0	36,62	1,2	1,4	37,66	1,2	1,4	0,4
	20	112,47	0	0	110,70	0	0	0
	20	81,15	0	0	83,00	0	0	0
	20	121,43	1	1	121,92	1	1	0
	20	49,56	0	0	51,12	0	0	0
	20	106,32	1	2	108,13	1	2	3
AVG:	20	94,19	0,4	0,6	94,97	0,4	0,6	0,6
	50	9,69	0	0	9,89	0	0	1
	50	30,80	0	0	31,12	0	0	0
	50	34,28	1	1	35,13	1	1	2
	50	93,49	0	0	92,14	0	0	0
	50	50,06	1	2	49,48	1	2	5
AVG:	50	43,66	0,4	0,6	43,55	0,4	0,6	1,6

Table A.23: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
60	0	14,70
	0	30,03
	0	62,02
	0	38,56
	0	62,62
AVG:	0	41,59
	20	14,56
	20	27,15
	20	50,54
	20	42,85
	20	58,26
AVG:	20	38,67
	50	62,18
	50	75,70
	50	24,48
	50	14,72
	50	27,06
AVG:	50	40,83

Table A.24: Results for 5 instances. ($\alpha = 1.00$)

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
70	0	1.142,41	10	11	846,59	8	8	15
	0	1.024,20	12	13	1.031,44	12	13	4
	0	987,90	12	12	906,03	11	11	21
	0	1.944,51	38	43	1.612,70	31	31	65
	0	1.819,00	33	39	1.569,34	26	26	57
AVG:	0	1.383,60	21,00	23,60	1.193,22	17,60	17,80	32,40
	20	1.387,08	21	23	1.039,20	18	18	19
	20	1.149,25	16	16	978,26	14	14	16
	20	1.429,25	22	23	1.021,36	18	18	28
	20	1.829,37	28	30	1.489,97	20	20	39
	20	1.203,09	18	18	950,50	15	15	18
AVG:	20	1.399,61	21,00	22,00	1.095,86	17,00	17,00	24,00
	50	1.490,50	16	18	1.209,10	14	14	21
	50	194,00	3	3	172,36	2	2	9
	50	1.819,57	29	30	1.680,17	21	22	29
	50	1.648,30	20	23	1.499,59	15	15	31
	50	1.440,00	18	22	1.439,57	17	18	9
AVG:	50	1.318,47	17,20	19,20	1.200,16	13,80	14,20	19,80

Table A.25: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
70	0	559,60	10	10	320,56	8	8	9
	0	581,17	18,00	18,00	583,21	18,00	18,00	5,00
	0	380,18	12,00	12,00	319,20	10,00	10,00	6,00
	0	160,05	2,00	3,00	161,31	2,00	3,00	7,00
	0	280,39	5,00	6,00	280,19	5,00	6,00	4,00
AVG:	0	392,28	9,40	9,80	332,89	8,60	9,00	6,20
	20	122,52	2,00	2,00	124,35	2,00	2,00	1,00
	20	149,64	3,00	3,00	131,19	2,00	2,00	6,00
	20	216,38	2,00	2,00	207,13	2,00	2,00	1,00
	20	304,45	4,00	4,00	291,73	3,00	3,00	3,00
	20	345,06	5,00	5,00	348,12	5,00	5,00	3,00
AVG:	20	227,61	3,20	3,20	220,50	2,80	2,80	2,80
	50	60,46	0,00	0,00	61,12	0,00	0,00	2,00
	50	159,00	1,00	1,00	149,00	1,00	1,00	3,00
	50	316,08	5,00	5,00	319,20	5,00	5,00	9,00
	50	315,23	4	5	316,25	4,00	5,00	2,00
	50	269,87	4,00	4,00	270,12	4,00	4,00	3,00
AVG:	50	224,13	2,80	3,00	223,14	2,80	3,00	3,80

Table A.26: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
70	0	20,34	0	0	21,01	0	0	0
	0	24,12	1	1	24,83	1	1	0
	0	31,29	2	2	31,88	2	2	1
	0	41,64	2	2	40,19	2	2	1
	0	21,16	0	0	22,45	0	0	0
AVG:	0	27,71	1	1	28,07	1	1	0,4
	20	31,12	0	0	32,23	0	0	0
	20	38,91	1	1	39,59	1	1	1
	20	29,50	0	0	31,56	0	0	0
	20	32,16	1	1	34,19	1	1	0
	20	45,10	1	1	46,08	1	1	1
AVG:	20	35,36	0,6	0,6	36,73	0,6	0,6	0,4
	50	59,12	0	0	58,01	0	0	1
	50	61,56	0	0	63,29	0	0	5
	50	146,30	2	2	149,25	2	2	1
	50	130,55	2	2	130,45	2	2	0
	50	89,10	1	1	88,16	1	1	1
AVG:	50	97,33	1	1	97,83	1	1	1,6

Table A.27: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
70	0	60,62
	0	68,50
	0	54,43
	0	75,82
	0	108,70
AVG:	0	73,61
	20	80,42
	20	115,56
	20	82,06
	20	68,81
	20	110,90
AVG:	20	91,55
	50	175,06
	50	118,06
	50	74,69
	50	99,74
	50	46,15
AVG:	50	102,74

Table A.28: Results for 5 instances. ($\alpha = 1.00$)

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
80	0	1.193,64	9	11	886,54	8	8	13
	0	1.449,05	13	15	1.264,60	11	11	16
	0	2.059,31	30	31	1.896,20	25	25	42
	0	1.789,15	19	19	1.565,81	16	16	25
	0	2.459,20	38	43	2.156,80	30	30	24
AVG:	0	1.790,07	21,80	23,80	1.553,99	18,00	18,00	24,00
	20	1.539,13	19	22	1.230,01	16	16	27
	20	1.405,90	14	15	1.407,00	14	14	18
	20	1.689,26	20	21	1.596,26	19	19	23
	20	2.410,67	33	35	2.340,13	25	25	21
	20	819,26	7	9	820,67	7	9	5
AVG:	20	1.572,84	18,60	20,40	1.478,81	16,20	16,60	18,80
	50	1.463,02	10	10	519,69	6	6	30
	50	1.194,19	12	12	1.076,41	11	11	13
	50	1.975,53	25	27	1.789,65	21	21	18
	50	1.064,00	8	8	1.025,00	7	7	46
	50	1.445,55	12	14	1.218,76	10	10	11
AVG:	50	1.428,46	13,40	14,20	1.125,90	11,00	11,00	23,60

Table A.29: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
80	0	628,27	12	13	456,90	10	10	8
	0	589,60	15,00	16,00	556,32	14,00	14,00	6,00
	0	875,21	21,00	21,00	653,04	16,00	16,00	29,00
	0	360,30	8,00	9,00	330,64	5,00	5,00	23,00
	0	774,59	15,00	18,00	489,00	12,00	13,00	31,00
AVG:	0	645,59	14,20	15,40	497,18	11,40	11,60	19,40
	20	259,69	5,00	5,00	261,20	5,00	5,00	12,00
	20	316,63	7,00	7,00	280,10	6,00	6,00	19,00
	20	294,60	3,00	3,00	296,30	3,00	3,00	14,00
	20	412,08	6,00	7,00	295,16	5,00	5,00	24,00
	20	681,26	9,00	11,00	441,27	7,00	8,00	22,00
AVG:	20	392,85	6,00	6,60	314,81	5,20	5,40	18,20
	50	18,12	0,00	0,00	19,00	0,00	0,00	0,00
	50	118,82	0,00	0,00	119,30	0,00	0,00	1,00
	50	412,36	3,00	3,00	413,65	3,00	3,00	5,00
	50	249,36	4	5	261,32	4,00	5,00	2,00
	50	403,70	5,00	5,00	405,05	5,00	5,00	1,00
AVG:	50	240,47	2,40	2,60	243,66	2,40	2,60	1,80

Table A.30: Results for 5 instances from data set 1. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
70	0	35,26	0	0	21,01	0	0	0
	0	37,16	2	2	22,30	1	1	3
	0	41,26	2	3	31,88	2	3	5
	0	64,50	4	4	41,38	2	2	9
	0	38,91	0	0	40,12	0	0	0
AVG:	0	43,42	1,6	1,8	31,34	1	1,2	3,4
	20	33,26	0	0	34,21	0	0	0
	20	45,62	2	2	46,31	2	2	6
	20	42,10	1	2	44,39	1	2	4
	20	48,91	2	2	49,25	2	2	3
	20	55,23	2	2	56,23	2	2	3
AVG:	20	45,02	1,4	1,6	46,08	1,4	1,6	3,2
	50	19,30	0	0	19,15	0	0	0
	50	106,91	0	0	109,13	0	0	1
	50	178,49	1	1	179,60	1	1	1
	50	156,30	2	2	158,20	2	2	1
	50	136,24	1	1	136,18	1	1	3
AVG:	50	119,45	0,8	0,8	120,45	0,8	0,8	1,2

Table A.31: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
80	0	115,19
	0	130,14
	0	103,42
	0	144,05
	0	206,52
AVG:	0	139,86
	20	152,81
	20	219,56
	20	155,91
	20	130,74
	20	210,72
AVG:	20	173,95
	50	43,36
	50	54,93
	50	141,91
	50	189,51
	50	87,69
AVG:	50	102,74

Table A.32: Results for 5 instances. ($\alpha = 1.00$)

$\alpha = 0.25$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
90	0	2.103,40	19	20	1.897,73	15	15	32
	0	2.203,61	18	20	1.846,50	14	14	21
	0	2.659,48	25	26	2.406,64	21	20	46
	0	3.067,83	31	31	2.397,66	20	21	51
	0	2.449,21	24	24	2.589,96	22	22	23
AVG:	0	2.496,71	23,40	24,20	2.227,70	18,40	18,40	34,60
	20	2.786,94	25	28	2.456,30	20	21	41
	20	2.698,46	26	26	2.356,95	23	23	16
	20	3.450,60	36	38	1.918,60	28	30	31
	20	2.218,67	29	31	2.469,17	25	25	33
	20	1.879,60	16	16	1.423,08	10	11	14
AVG:	20	2.606,85	26,40	27,80	2.124,82	21,20	22,00	27,00
	50	1.948,26	15	16	1.716,40	12	12	31
	50	1.658,69	12	14	1.546,31	10	10	17
	50	2.462,34	19	19	1.907,45	14	15	37
	50	2.062,18	16	17	1.872,36	13	13	26
	50	2.159,87	16	16	1.602,34	10	10	51
AVG:	50	2.058,27	15,60	16,40	1.728,97	11,80	12,00	32,40

Table A.33: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
90	0	725,60	14	15	665,32	11	11	16
	0	1.012,00	12,00	13,00	554,21	8,00	9,00	19,00
	0	1.543,20	19,00	21,00	1.456,82	18,00	18,00	8,00
	0	2.402,61	27,00	29,00	1.405,04	20,00	21,00	34,00
	0	1.453,27	19,00	20,00	871,26	14,00	14,00	36,00
AVG:	0	1.427,34	18,20	19,60	990,53	14,20	14,60	22,60
	20	389,12	6,00	6,00	301,15	5,00	5,00	11,00
	20	305,64	5,00	5,00	270,60	4,00	4,00	17,00
	20	258,60	2,00	3,00	261,12	2,00	3,00	9,00
	20	354,97	5,00	7,00	356,24	5,00	7,00	7,00
	20	547,73	10,00	12,00	471,94	9,00	9,00	11,00
AVG:	20	371,21	5,60	6,60	332,21	5,00	5,60	11,00
	50	164,60	1,00	1,00	166,31	1,00	1,00	9,00
	50	206,80	2,00	3,00	207,60	2,00	3,00	5,00
	50	354,08	2,00	2,00	358,13	2,00	2,00	6,00
	50	489,60	4	5	402,50	3,00	3,00	15,00
	50	621,73	6,00	7,00	516,48	5,00	5,00	12,00
AVG:	50	367,36	3,00	3,60	330,20	2,60	2,80	9,40

Table A.34: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.75$		**Default Algorithm**			**Improved Algorithm**			
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	Time	# iter.	# TEC	# TPC
90	0	96,89	1	1	97,11	1	1	2
	0	49,57	0	0	52,36	0	0	3
	0	116,59	2	2	118,14	2	2	4
	0	66,50	1	1	68,26	1	1	9
	0	55,91	0	0	56,91	0	0	2
AVG:	0	77,09	0,8	0,8	78,56	0,8	0,8	4
	20	85,54	1	1	88,91	1	1	4
	20	56,67	0	0	58,12	0	0	2
	20	46,29	1	1	49,30	1	1	4
	20	124,60	2	3	126,34	1	1	3
	20	154,92	2	2	159,32	2	2	3
AVG:	20	93,60	1,2	1,4	96,40	1	1	3,2
	50	67,89	0	0	68,91	0	0	0
	50	101,10	0	0	104,52	0	0	0
	50	89,46	0	0	91,21	0	0	0
	50	154,26	2	3	156,23	2	3	5
	50	165,23	2	2	168,49	2	2	5
AVG:	50	115,59	0,8	1	117,87	0,8	1	2

Table A.35: Results for 5 instances. ($\alpha = 0.75$)

$\|\mathcal{C}\|$	%B	Time
90	0	217,18
	0	241,89
	0	433,70
	0	273,30
	0	302,51
AVG:	0	293,7
	20	461,08
	20	442,51
	20	327,40
	20	274,54
	20	320,89
AVG:	20	365,3
	50	298,01
	50	184,15
	50	115,35
	50	91,05
	50	397,98
AVG:	50	217,3

Table A.36: Results for 5 instances. ($\alpha = 1.00$)

Appendix B

Test Results for Instances with Heterogenous Fleet

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
10	0	0,65	3	1	2	0,58	3	1	2	0
	0	3,69	15	10	5	4,00	15	10	0	0
	0	2,76	12	8	4	2,78	10	6	4	2
	0	0,50	3	2	1	0,51	3	2	1	0
	0	2,07	12	9	3	1,77	11	8	3	1
AVG:	0	1,93	9,00	6,00	3,00	1,93	8,40	5,40	2,00	0,60
	50	1,12	4	3	1	0,83	3	2	1	1
	50	0,24	1	1	0	0,25	1	1	0	0
	50	0,47	2	2	0	0,50	2	2	0	0
	50	0,39	2	1	1	0,39	2	1	1	0
	50	0,70	3	3	0	0,69	3	3	0	0
AVG:	50	0,58	2,40	2,00	0,40	0,53	2,20	1,80	0,40	0,20

Table B.1: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
10	0	0,26	0	0	0	0,27	0	0	0	0
	0	0,70	4	4	0	0,67	4	4	0	0
	0	6,19	12	12	0	5,66	11	11	0	2
	0	1,46	6	6	0	1,25	5	5	0	1
	0	0,25	1	1	0	0,23	1	1	0	0
AVG:	0	1,77	4,60	4,60	0,00	1,62	4,20	4,20	0,00	0,60
	50	0,45	1	1	0	0,49	1	1	0	2
	50	0,33	0	0	0	0,35	0	0	0	0
	50	0,34	2	2	0	0,36	2	2	0	0
	50	0,17	0	0	0	0,20	0	0	0	0
	50	0,50	2	2	0	0,45	2	2	0	0
AVG:	50	0,36	1,00	1,00	0,00	0,37	1,00	1,00	0,00	0,40

Table B.2: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
20	0	23,53	31	11	20	19,24	29	9	20	3
	0	4,21	8	4	4	3,99	8	4	4	0
	0	7,99	12	7	5	6,51	11	6	5	3
	0	6,34	10	11	0	5,95	9	10	0	6
	0	14,07	15	15	0	10,21	12	12	0	17
AVG:	0	11,23	15,20	9,60	5,80	9,18	13,80	8,20	5,80	5,80
	50	32,77	29	31	0	28,14	26	28	0	4
	50	5,63	6	7	0	4,98	5	6	0	6
	50	23,32	19	19	0	19,18	16	16	0	3
	50	2,64	3	3	0	2,71	3	3	0	1
	50	4,11	5	2	3	4,25	5	2	3	1
AVG:	50	13,69	12,40	12,40	0,60	11,85	11,00	11,00	0,60	3,00

Table B.3: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
20	0	6,71	10	0	10	6,59	10	0	10	0
	0	7,15	10	0	10	7,34	10	0	10	0
	0	0,39	0	0	0	0,43	0	0	0	0
	0	1,14	2	1	1	1,08	2	1	1	0
	0	2,03	2	0	2	1,89	2	0	2	0
AVG:	0	3,48	4,80	0,20	4,60	3,47	4,80	0,20	4,60	0,00
	50	3,16	3	0	3	3,47	3	0	3	3
	50	2,09	2	0	2	2,46	2	0	2	2
	50	2,16	2	2	0	2,59	2	2	0	3
	50	9,94	9	9	0	10,26	9	9	0	2
	50	4,89	4	2	2	5,79	4	2	2	3
AVG:	50	4,45	4,00	2,60	1,40	4,91	4,00	2,60	1,40	2,60

Table B.4: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
30	0	6,59	4	6	0	6,84	4	6	0	0
	0	11,03	7	7	2	10,51	6	6	1	5
	0	9,35	6	5	1	8,24	5	4	1	3
	0	121,10	36	33	4	109,10	33	30	4	12
	0	84,26	22	18	3	45,67	14	12	2	14
AVG:	0	46,47	15	13,80	2	36,07	12,4	11,60	1,6	6,80
30	50	30,82	4	4	0	27,29	3	3	0	1
	50	3,40	2	2	0	3,61	2	2	0	1
	50	52,73	12	12	0	26,59	9	9	0	6
	50	9,61	6	6	0	3,84	2	2	0	7
	50	4,35	2	0	2	4,21	2	0	2	2
AVG:	50	20,18	5,20	4,80	0,40	13,11	3,60	3,20	0,40	3,40

Table B.5: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
30	0	4,33	2	2	0	3,12	1	1	0	6
	0	5,36	2	1	1	3,45	1	1	1	2
	0	8,32	3	2	1	3,59	2	1	1	4
	0	24,91	7	6	1	19,25	5	5	0	7
	0	8,12	3	3	0	8,13	3	3	0	3
AVG:	0	11,68	3,75	3,00	0,75	8,61	2,75	2,50	0,50	4,00
30	50	4,89	0	0	0	4,91	0	0	0	0
	50	6,21	2	1	1	6,20	2	1	1	2
	50	12,39	3	2	1	6,05	2	1	1	3
	50	11,38	3	3	0	7,01	2	2	0	5
	50	2,31	1	1	0	2,41	1	1	0	3
AVG:	50	7,44	1,80	1,40	0,40	5,32	1,40	1,00	0,40	2,60

Table B.6: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
40	0	107,93	14	16	0	74,81	10	12	0	22
	0	46,50	2	1	1	47,21	2	1	1	3
	0	88,12	9	8	1	76,15	8	7	1	5
	0	186,20	19	17	2	159,57	14	12	2	16
	0	259,24	29	22	7	201,49	24	22	2	18
AVG:	0	137,60	14,60	12,80	2,20	111,85	11,60	10,80	1,20	12,80
40	50	31,03	3	3	0	32,59	3	3	0	2
	50	81,11	12	10	2	68,19	8	7	1	13
	50	201,54	25	20	5	146,37	15	12	2	17
	50	108,67	13	15	0	94,56	11	11	0	13
	50	78,19	9	7	2	61,41	6	5	1	8
AVG:	50	100,1	12,4	11	1,8	80,62	8,6	7,6	0,8	10,6

Table B.7: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
40	0	52,82	8	7	1	39,26	6	5	1	6
	0	31,59	5	4	1	32,11	5	4	1	2
	0	45,69	6	6	0	37,81	4	4	0	6
	0	71,60	11	8	3	61,04	8	7	1	11
	0	121,67	9	8	1	110,16	8	8	0	7
AVG:	0	64,67	7,8	6,6	1,2	56,08	6,2	5,6	0,6	6,4
40	50	23,12	2	2	0	24,16	2	2	0	2
	50	11,27	1	1	0	12,50	1	1	0	5
	50	37,54	4	3	1	33,45	3	2	1	6
	50	59,81	8	6	2	51,12	6	5	1	14
	50	41,16	5	5	0	43,19	5	5	0	7
AVG:	50	34,58	4	3,4	0,6	32,88	3,4	3	0,4	6,8

Table B.8: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
50	0	99,18	7	7	0	94,57	6	6	0	16
	0	167,17	11	8	3	147,51	7	7	0	18
	0	689,44	34	30	4	587,05	25	23	2	31
	0	527,38	25	21	4	441,34	20	18	2	29
	0	201,40	15	12	3	189,97	12	11	1	27
AVG:	0	336,91	18,40	15,60	2,80	292,09	14,00	13,00	1,00	24,20
50	50	128,68	8	6	2	114,23	6	2	0	8
	50	274,18	17	15	2	235,60	15	13	2	7
	50	976,19	42	36	6	847,72	35	32	3	37
	50	154,50	9	8	1	148,83	8	8	0	14
	50	611,61	21	19	2	267,12	16	16	0	21
AVG:	50	429	19,4	16,8	2,6	322,7	16	14,2	1	17,4

Table B.9: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
50	0	142,2	11	10	1	134,20	10	9	1	15
	0	94,68	4	4	0	96,21	4	4	0	8
	0	74,50	5	4	1	70,16	4	3	1	12
	0	124,87	8	6	2	85,52	5	4	1	25
	0	37,31	3	3	0	35,13	2	2	0	8
AVG:	0	94,71	6,2	5,4	0,8	84,24	5	4,4	0,6	13,6
50	50	22,50	3	3	0	24,56	3	3	0	8
	50	31,09	4	2	2	33,07	4	2	2	3
	50	32,51	7	5	2	31,16	5	4	1	14
	50	41,24	8	6	2	34,10	5	4	1	17
	50	42,37	7	5	2	35,20	5	4	1	9
AVG:	50	33,94	5,8	4,2	1,6	31,62	4,4	3,4	1	10,2

Table B.10: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
60	0	387,12	18	16	2	321,15	12	11	1	24
	0	1.489,62	49	45	4	1.100,30	32	31	1	46
	0	1.028,64	23	20	3	743,55	15	14	1	22
	0	1.125,50	39	35	4	986,48	30	28	2	37
	0	286,50	8	7	1	224,12	6	5	1	15
AVG:	0	863,48	27,40	24,60	2,80	675,12	19,00	17,80	1,20	28,80
50	50	86,00	4	2	1	80,45	3	2	1	11
	50	659,55	11	11	0	546,30	8	8	0	19
	50	978,21	28	25	2	711,42	21	20	1	32
	50	578,16	14	12	2	502,00	12	10	2	18
	50	93,57	4	4	0	79,16	3	3	0	8
AVG:	50	479,098	12,2	10,8	1	383,866	9,4	8,6	0,8	17,6

Table B.11: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
60	0	896,3	23	21	2	551,24	15	14	1	35
	0	91,24	5	4	1	88,20	4	3	1	26
	0	246,81	18	17	1	213,59	15	15	0	18
	0	112,67	8	8	2	95,21	6	6	0	17
	0	68,79	10	10	0	71,12	10	10	0	11
AVG:	0	283,162	12,8	12	1,2	203,872	10	9,6	0,4	21,4
60	50	124,24	8	8	0	108,17	6	6	0	15
	50	86,37	5	4	1	78,16	4	4	0	10
	50	271,31	16	12	4	200,14	13	12	1	17
	50	115,29	7	2	5	91,24	5	2	3	11
	50	134,81	6	6	0	85,55	4	4	0	16
AVG:	50	146,404	8,4	6,4	2	112,652	6,4	5,6	0,8	13,8

Table B.12: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
70	0	1.096,51	15	13	2	879,61	11	10	1	24
	0	1.234,10	18	17	1	986,27	14	12	2	26
	0	881,27	11	8	3	809,40	9	9	0	18
	0	846,59	10	4	6	796,14	8	2	2	22
	0	1.764,59	32	28	5	1.523,67	24	21	3	51
AVG:	0	1.164,61	17,2	14	3,4	999,02	13,2	10,8	1,6	28,2
70	50	1.304,20	18	16	2	1.102,30	15	14	1	25
	50	1.597,31	21	17	4	1.268,10	15	13	2	29
	50	1.870,60	24	20	4	1.615,21	20	22	2	38
	50	758,34	11	10	1	645,57	8	7	1	19
	50	698,21	8	7	1	615,20	6	6	0	16
AVG:	50	1245,73	16,4	14	2,4	1049,28	12,8	12,4	1,2	25,4

Table B.13: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
70	0	691,24	18	16	2	614,79	15	14	1	6
	0	647,19	16	12	4	596,21	15	11	4	5
	0	426,80	10	9	1	355,58	7	6	1	10
	0	394,51	11	11	0	321,19	8	8	0	14
	0	145,60	5	2	3	131,30	3	1	2	9
AVG:	0	461,068	12	10	2	403,814	9,6	8	1,6	8,8
70	50	215,35	3	3	0	218,20	3	3	0	3
	50	245,71	4	2	2	255,13	4	2	2	5
	50	311,59	12	10	2	226,15	7	6	1	24
	50	183,43	8	8	0	184,67	5	5	0	9
	50	476,23	17	12	5	458,37	16	14	2	16
AVG:	50	286,462	8,8	7	1,8	268,504	7	6	1	11,4

Table B.14: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
80	0	1.468,83	12	10	2	1.256,31	10	9	1	16
	0	2.454,37	34	28	6	1.925,64	27	25	2	39
	0	1.945,20	14	12	2	1.859,96	12	10	2	27
	0	1.293,25	9	10	1	1.102,50	7	7	0	15
	0	1.956,80	28	26	3	1.645,72	21	19	2	42
AVG:	0	1.823,69	19,4	17,2	2,8	1.558,03	15,4	14	1,4	27,8
80	50	1.285,64	11	10	1	1.098,00	9	9	0	24
	50	1.542,21	15	12	3	1.249,37	11	10	1	20
	50	1.876,24	27	25	3	1.724,54	20	18	2	41
	50	1.682,10	19	14	5	1.504,20	16	13	3	19
	50	1.443,33	16	14	2	1.358,81	14	12	2	15
AVG:	50	1565,9	17,6	15	2,8	1386,984	14	12,4	1,6	23,8

Table B.15: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
80	0	772,11	13	10	3	682,54	11	2	1	16
	0	1.027,25	21	19	2	955,25	18	16	2	13
	0	545,20	9	7	2	505,13	8	6	1	15
	0	896,45	15	10	5	665,42	12	10	2	23
	0	937,50	18	14	4	701,12	14	11	3	19
AVG:	0	835,702	15,2	12	3,2	701,892	12,6	9	1,8	17,2
80	50	105,23	2	1	1	108,20	2	1	1	9
	50	245,90	5	2	3	215,20	4	2	2	13
	50	609,52	9	6	3	396,11	7	6	1	19
	50	764,61	15	12	3	589,24	11	10	1	30
	50	684,49	14	10	4	542,60	11	9	2	26
AVG:	50	481,95	9	6,2	2,8	370,27	7	5,6	1,4	19,4

Table B.16: Results for 5 instances. ($\alpha = 0.50$)

$\alpha = 0.25$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
90	0	1.980,60	15	12	3	1.335,84	12	10	2	21
	0	2.153,20	19	17	2	1.648,11	15	15	1	16
	0	1.756,22	14	14	0	1.253,79	12	12	0	30
	0	2.350,35	24	21	3	1.964,53	19	18	1	34
	0	2,596,72	28	24	5	2.150,40	21	19	2	32
AVG:	0	2.060,09	20	17,6	2,6	1.670,53	15,8	14,8	1,2	26,6
90	50	1.562,56	12	8	4	1.005,80	8	8	0	24
	50	1.040,05	10	9	1	1.043,21	10	9	1	6
	50	2.019,94	21	22	2	1.952,34	14	13	2	37
	50	1.883,57	16	14	2	1.189,30	13	12	1	22
	50	1.908,61	17	15	2	1.749,65	15	14	1	16
AVG:	50	1682,946	15,2	13,6	2,2	1388,06	12	11,2	1	21

Table B.17: Results for 5 instances. ($\alpha = 0.25$)

$\alpha = 0.50$		**Default Algorithm**				**Improved Algorithm**				
$\|\mathcal{C}\|$	%B	Time	# iter.	# TEC	# MTEC	Time	# iter.	# TEC	$\#MTEC$	# TPC
90	0	1024,21	11	10	1	870,54	8	8	0	24
	0	2.159,30	24	21	5	1.795,30	19	15	4	16
	0	1.489,30	18	17	1	1.127,31	12	12	0	33
	0	1.359,32	16	12	4	1.307,17	12	10	3	20
	0	1.502,03	17	12	5	1.364,29	15	13	3	18
AVG:	0	1506,832	17,2	14,4	3,2	1292,922	13,2	11,6	2	22,2
90	50	1.140,20	9	8	1	998,31	7	7	0	23
	50	1.211,07	10	7	3	976,37	8	7	0	19
	50	1.352,74	12	11	1	1.215,45	10	10	0	25
	50	2.102,91	20	15	5	1.546,77	15	12	3	35
	50	1.259,41	13	10	3	1.022,80	9	9	2	18
AVG:	50	1413,266	12,8	10,2	2,6	1151,94	9,8	9	1	24

Table B.18: Results for 5 instances. ($\alpha = 0.50$)