APPROXIMATE MODELS AND SOLUTION APPROACHES FOR THE VEHICLE ROUTING PROBLEM WITH MULTIPLE USE OF VEHICLES AND TIME WINDOWS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

JEROEN WOUTER DE BOER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JUNE 2008

Approval of the thesis

"APPROXIMATE MODELS AND SOLUTION APPROACHES FOR THE VEHICLE ROUTING PROBLEM WITH MULTIPLE USE OF VEHICLES AND TIME WINDOWS"

submitted by **Jeroen Wouter de Boer** in partial fullfilment of the requirements for the degree of **Master of Science in Industrial Engineering, Middle East Technical University** by,

Prof. Dr. Canan Özgen ____________________
Director, **Graduate School of Natural and Applied Sciences**

Prof. Dr. Nur Evin Özdemirel ____________________
Head of Department, **Industrial Engineering**

Assoc. Prof. Dr. Haldun Süral ____________________
Supervisor, **Industrial Engineering, METU**

Examining Committee Members:

Assist. Prof. Dr. Sedef Meral ____________________
Industrial Engineering, METU

Assoc. Prof. Dr. Haldun Süral ____________________
Industrial Engineering, METU

Prof. Dr. Ömer Kırca ____________________
Industrial Engineering, METU

Assoc. Prof. Dr. Canan Sepil ____________________
Industrial Engineering, METU

Assist. Prof. Dr. Ferda Can Çetinkaya ____________________
Industrial Engineering, Çankaya University

Date: ____________________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name	:	Jeroen Wouter de Boer
Signature	:	

ABSTRACT

APPROXIMATE MODELS AND SOLUTION APPROACHES FOR THE VEHICLE ROUTING PROBLEM WITH MULTIPLE USE OF VEHICLES AND TIME WINDOWS

Boer, Jeroen Wouter de
M.Sc., Department of Industrial Engineering
Supervisor: Assoc. Prof. Dr. Haldun Süral

June 2008, 90 pages

In this study we discuss the Vehicle Routing Problem with multiple use of vehicles (VRPM). In this variant of the routing problem the vehicles may replenish at any time at the depot. We present a detailed review of existing literature and propose two mathematical models to solve the VRPM. For these two models and their several variants we provide computational results based on the test problems taken from the literature. We also discuss a case study in which we are simultaneously dealing with side constraints such as time windows, working hour limits, backhaul customers and a heterogeneous vehicle fleet.

Keywords: Vehicle Routing, Set-Covering, Time Windows, Pick-up and Delivery, Heterogeneous Vehicle Fleet

ÖZ

ÇOK SEFERLİ VE ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMİ İÇİN YAKLAŞIK ÇÖZÜM VEREN MODELLER VE ÇÖZÜM YÖNTEMLERİ

Boer, Jeroen Wouter de
Yüksek Lisans, Endüstri Mühendisliği Bölümü Bölümü
Tez Yöneticisi: Doç. Dr. Haldun Süral

Haziran 2008, 90 sayfa

Bu çalışmada çok seferli araç rotalama problemi işlenmiştir. Bu problemde araçlar günün herhangi bir anında depoya dönüp, tekrar sefere çıkabilirler. Problemde ilgili detaylı bir literatür taraması yapılmış ve problemi çözmek için iki matematiksel model önerilmiştir. Önerilen modeller ve varyasyonları, literatürden alınan test problemleri üzerinde denenmiş ve sayısal sonuçlar verilmiştir. Ayrıca, zaman penceresi, çalışma saati kısıtları, farklı müşteri tipi ve değişik tipte araçlar barındıran filo gibi özellikler içeren bir gerçek hayat uygulaması ele alınmıştır.

Anahtar Kelimeler: Araç Rotalama, Küme-Kapsama, Zaman Penceresi, Toplama ve Dağıtım, Heterojen Araç Filosu

To my supportive parents and sister

ACKNOWLEDGEMENTS

I would like to thank Assoc. Prof. Dr. Haldun Süral for guiding me through all stages in this study. His frankness, sincerity and enthusiasm were giving me plenty of motivation to successfully finish this study.

I would like to thank Assist. Prof. Dr. Sedef Meral for her guidance in the process of finding a suitable internship. She was always ready to assist me throughout my entire Master of science program.

I want to express my gratitude to Mr. Jason Will, Managing Director of Santa Fe Indonesia for giving me the opportunity to do an internship at his office. During the four months at Santa Fe he gave me the opportunity to look around in the company and granted all cooperation in collecting the necessary data for this study.

Finally I would like to thank my parents and sister for always supporting me. Even though I was usually far away from home you were always ready to encourage me.

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CLP	Constrained Logic Programming
CVRP	Capacitated Vehicle Routing Problem
GA	Genetic Algorithms
GIS	Graphical Information System
LTR	Longest Tour Ratio
MDVRP	Multi-Depot Vehicle Routing Problem
MIP	Mixed Integer Program
NN	Neural Networks
NP	Nondeterministic Polynomial
SA	Simulated Annealing
SVRP	Symmetric Vehicle Routing Problem
TS	Tabu Search
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPM	Vehicle Routing Problem with Multiple use of Vehicles
VRPTW	Vehicle Routing Problem with Time Windows

CHAPTER 1

INTRODUCTION

In this study, we will discuss the Vehicle Routing Problem (VRP) with multiple use of vehicles, also known as the multiple trip vehicle routing problem. The vehicle routing problem tries to minimize the total distance traveled by a set of vehicles while satisfying the demand of a given set of customers. In this classical VRP the assumption is made that each vehicle serves a single route during any planning period. In the vehicle routing problem with multiple use of vehicles we drop this assumption.

We start this study by reviewing some of the formulations and solution approaches of the capacitated vehicle routing problem (CVRP). This problem is frequently discussed in the literature because of its wide range of application areas as well as the mathematical complexity. We will demonstrate the applicability of the VRP by mentioning several important modifications of the CVRP.

Continuing this brief overview we will extensively study the literature related to the vehicle routing problem with multiple use of vehicles. We will in detail explain the solution approaches that are suggested and the possible drawbacks of each approach. We will also carefully look at the computational results of the benchmark test problems proposed in the literature as we will be making use of these problems as well.

Inspired by a real-life problem that concerns the scheduling of the vehicle fleet of Santa Fe Indonesia, we modify some of the existing algorithms in the literature to solve this problem. Since we want to solve a practical problem, we are dealing with many side constraints such as a heterogeneous vehicle fleet, time windows for the drivers, individual time windows for customers, inaccessibility of certain vehicle types for certain jobs, etc. It is our contribution here that we explore the fruitful features of some classical VRP approaches. We adapt and combine them to solve more complex practical routing problems.

1.1 Problem definition of the classical VRP

The classical VRP is a generalization of the Traveling Salesman Problem (TSP). A common used definition for the TSP is the following: given a graph with nonnegative arc weights, find a least weight Hamiltonian cycle such that all nodes (customers) are visited at least once.

In vehicle routing problem, instead of salesmen, we now have vehicles with restricted capacities that visit customers having demand. When demand of all customers exceeds the vehicle capacity, we need two or more vehicles. This implies that in the CVRP we have to find multiple Hamiltonian cycles such that each Hamiltonian cycle is not exceeding the vehicle capacity (see an illustration of VRP in Figure 1.1).

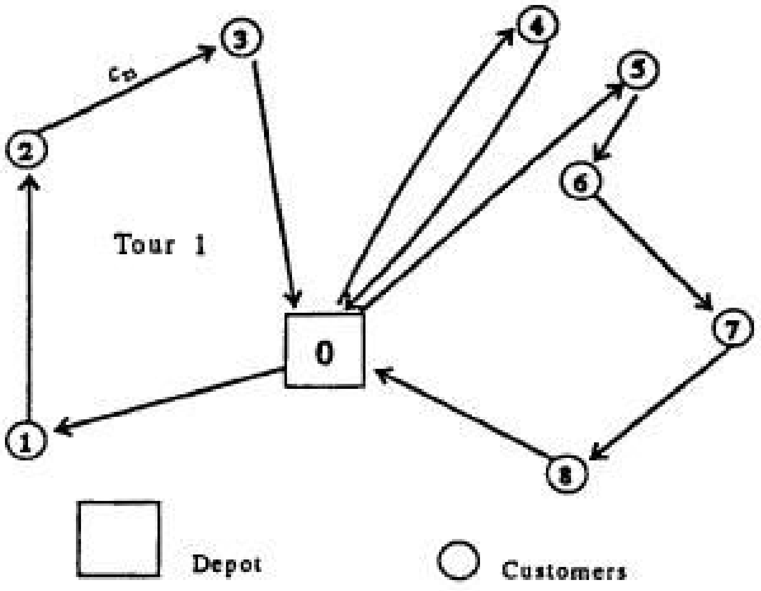

Figure 1.1: VRP example from Osman (1993), page 422

In the classical problem we make the following assumptions:

- The vehicle fleet is homogeneous, i.e. each vehicle has an equal capacity of C.
- Each customer has a positive demand q_i which has to be fully satisfied.
- The depot has no demand.
- All customers are served by a single vehicle.

We define a graph $G = (V, A)$ where $V = 0, \ldots, n$ is a vertex set with vertex 0 representing the depot and other vertices representing customers. Let A be an arc set (see equation (1.1)), where each arc is associated with nonnegative numbers c_{ij} and t_{ij} that denote travel cost and travel time, respectively. We make a distinction between symmetric and a-symmetric travel cost/time matrices. In the symmetric case the arc set is usually replaced by an edge-set E (see equation (1.2) with undirected edges). Solution approaches can vary significantly between these two cases.

$$A = \{(i,j) | i \in V, j \in V, i \neq j\} \tag{1.1}$$

$$E = \{(i,j) | i \in V, j \in V, i < j\} \tag{1.2}$$

1.1.1 Vehicle flow formulation for the VRP

In this section we will show the integer linear programming vehicle flow formulation for the asymmetric VRP, given by Toth and Vigo (2001). This model is minimizing the total cost or distance for all vehicles while completely satisfying all demands. The parameters and decision variables of this model are given below.

Parameters:

c_{ij}	$=$	the cost of travelling from node i to node j.
q_i	$=$	positive demand of customer i.
m	$=$	the number of vehicles available.
C	$=$	the vehicle capacity.

Decision variables:

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ travels from node } i \text{ to node } j \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{if vehicle } k \text{ satisfies the demand of node } i \\ 0, & \text{otherwise} \end{cases}$$

The mathematical representation:

$$\min \quad z = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} c_{ij}x_{ijk} \tag{1.3}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} y_{0,k} = m \tag{1.4}$$

$$\sum_{k=1}^{m} y_{ik} = 1 \qquad i = 1,\dots,n \tag{1.5}$$

$$\sum_{i=1}^{n} q_i y_{ik} \leq C \qquad k = 1,\dots,m \tag{1.6}$$

$$\sum_{j=1}^{n} x_{ijk} = \sum_{j=1}^{n} x_{jik} = y_{ik} \qquad i = 1,\dots,n \;\; k = 1,\dots,m \tag{1.7}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ijk} \leq |S| - 1 \qquad S \subset \{1,\dots,n\}, |S| \geq 2, k = 1,\dots,m \tag{1.8}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j,k, i \neq j \tag{1.9}$$

$$y_{ik} \in \{0,1\} \qquad \forall i,k \tag{1.10}$$

The objective function, equation (1.3), minimizes the total cost or distance while constraints (1.4)-(1.10) are satisfied. Constraint (1.4) ensures that exactly m vehicles will leave the depot, while constraint (1.5) guarantees that exactly one vehicle will visit each customer. Constraint (1.6) restricts total demand of each vehicle by its capacity. Constraint (1.7) and (1.8) will guarantee feasible routes without sub-routes or more than once visited customers. Finally equations (1.9) and (1.10) define decision variables.

$O(n^2m+nm)$ variables and the exponential number of subtour elimination constraints (1.8) make this model hard to solve. Because of this interesting property the CVRP and its modeling have attracted attention for several decades now. As a result, several other formulations and solution approaches have been proposed. We will discuss several of these in Chapter 2.

1.2 Problem definition of the VRPM

Throughout this study we mainly focus on the modification of the classical VRP where vehicles are allowed to be replenished at the depot, the so called VRP with multiple use of vehicles (VRPM). Each vehicle may now serve more than one route in a single planning period.

While studying the logistic activities of a (real-life) Jakarta based relocation company we observed that quite frequently a vehicle returned to the depot during the day for (un)loading. Their activities consist of inbound shipments arriving at air or seaport that have to be picked up from the concerning port. Once arrived at the depot the shipment is prepared to be shipped to its final destination. Outbound shipments travel in the reverse order, first the shipment is packed at origin from where it is brought to the depot. Here the shipments are prepared for long-haul sea or airfreight.

When choosing a model for their activities the classical VRP is clearly not appropriate and the VRPM is a more logical choice. While analyzing the available literature on this topic, we noticed that, despite its relevance to many practical applications, this topic is not well covered in the literature.

A trivial way of extending the vehicle flow formulation (1.3)-(1.10) would be to add a constraint that puts a limit to the maximum duration of a route. When defining this limit as half a working day, we could assign one route to each vehicle in the first half of the day and another route to each vehicle in the second half of that day. For some applications this method could work reasonable, but obviously there are more complex examples that require more sophisticated models. For instance, it could be the case that some vehicles serve more than one route, but other vehicles only serving one route. This case is obviously disregarded in the method above.

Basically the literature only covers the case where the vehicles are limited on their working day. A maximum size (in terms of total time or total distance) is defined and a bin-packing approach is used to "pack" the routes in these bins. In this formulation each route would be associated with a weight (duration or length) and the bins (working day) have to be utilized efficiently filling these routes. In Chapter 3 we will discuss the VRPM in more detail.

In the case-study we are simultaneously dealing with a heterogeneous vehicle fleet and individual time windows for the customers. The latter is discussed once (without details) in the literature, but the first is never before discussed in combination with the VRPM. Besides these two extensions we also extend the VRPM to two types of customers (namely, linehaul and backhaul).

Since all these extensions are of significant importance to the logistic activities of Santa Fe and these were not covered yet in the literature, we decided that this was a suitable subject for this study.

1.3 Chapter outline

The following chapters will all contribute to the completeness of our study.

In Chapter 2 we give a detailed literature survey. Because of the large amount of papers dedicated to the VRP and related topics we have to make a selection. We have selected important papers that all have contributed to our study. After a quick note on the rise of combinatorial optimization problems in general, we will start with a famous survey-paper. Following the classification of this survey, we will discuss a number of papers that propose different solution approaches. We end this chapter by discussing some papers related to a variety of VRP extensions.

In Chapter 3 we completely focus on the available literature about the vehicle routing problem with multiple use of vehicles. Since the number of papers about this topic is very limited, we will discuss all related papers about this topic. We look at the proposed solution approach, computational results, drawbacks, and their potential improvement directions.

In Chapter 4 we basically suggest two approaches to solve VRPM: an integer model based on the well-known route-first cluster-second approach and a new approach using set-covering principle. The first model can be considered as an approximate formulation where as the second one is a true representation of the problem as long as entire route set or optimal routes are incorporated into the formulation. Both approaches are enhanced using improvement routines. We will compare our results with the existing algorithms and discuss our findings.

In Chapter 5 we discuss a case study based on a real-life problem from a relocation company in South-East Asia. We modify the set-covering approach developed in Chapter 4 to deal with the presence of several practical side constraints such as a heterogeneous vehicle fleet, individual time windows and backhaul customers.

We conclude this study in Chapter 6 by discussing our findings and pointing out several possible follow-up studies.

CHAPTER 2

LITERATURE SURVEY

The Vehicle Routing Problem (VRP) is a generalization of the well known and widely studied Traveling Salesman Problem (TSP). The TSP is in mathematical terms a hard combinatorial problem. As Schrijver (2005) mentions in his working paper, the TSP has been discussed as early as 1832 by a group of Germans, but the first mathematician to pay attention to the problem was K. Menger in 1928. For a more detailed outline of the early evolution of the TSP and other combinatorial optimization problems we refer to Schrijver (2005).

As we already mentioned in the introduction a difficulty of solving the CVRP is its computational complexity. Innumerable studies have been performed on complexity of combinatorial problems such as the TSP and VRP. In this study we take for granted that the TSP and therefore also the VRP are NP-hard. We will no further discuss the details of the computational complexity, but for a reliable survey we refer to Lenstra and Rinnooy Kan (1981).

The NP-hard property is an important reason that TSP and VRP received much attention of researchers. Secondly researchers had an inquisitive attitude towards the VRP, because of the applicability to a wide range of areas. Think of only school bus routing, collection of milk, postal delivery network, supplying gas stations, moving personnel to off-shore oil platforms, etc. Researchers developed many exact and approximate algorithms in the literature over the last couple of decades to solve these problems. Laporte (1992, 2007) gives an overview of some important algorithms. In both overviews a classification is made between exact and approximation algorithms. We will start with discussing some exact algorithms.

2.1 Exact algorithms for the VRP

In this section we mention several exact algorithms proposed in the literature. The aim of these exact algorithms is to solve the problem to optimality. Laporte (1992) makes a rough classification of all exact algorithms to put into three categories,

- Direct tree-search methods
- Dynamic programming methods
- Integer linear programming methods

For each of these categories we will now show an example to illustrate how each class of algorithms works.

2.1.1 Direct tree-search methods

Christofides et al. (1981) have proposed a formulation for the symmetrical VRP. A symmetrical VRP is defined on a graph $G = (V, E)$ with edges instead of arcs. The idea presented in this paper is based on partitioning of the edge set E into four subsets:

- E_0: Edges not belonging to the solution
- E_1: Edges forming a k-degree center tree where the depot has degree $k, k = 2m - y$
- E_2: y edges incident to the depot.
- E_3: $m - y$ edges not incident to the depot.

The objective of their model is to minimize the cost of the edges selected such that all demand is satisfied. The parameters and decision variables are given below.

Parameters:

$c_l \quad = \quad$ the cost of edge l.

$m \quad = \quad$ the number of available vehicles.

Decision variables:

$$\epsilon_l^t = \begin{cases} 1, & \text{if edge } l \text{ is an element of } E_t \text{ for } t = 1, 2, 3 \\ 0, & \text{otherwise} \end{cases}$$

$y \quad = \quad$ the number of edges incident to the depot, $0 \leq y \leq m$.

Furthermore they define E^i as the set of all edges incident to node i; $(S, \bar{S})$ as the set of all edges with one node in subset $S \subset N$ and one node in the complement of S.

The mathematical representation:

$$
\begin{aligned}
\min \quad & \sum_{l \in E} c_l(\epsilon_l^1 + \epsilon_l^2 + \epsilon_l^3) & & (2.1)\\
\text{s.t.} \quad & \sum_{l \in (S,\bar{S})} \epsilon_t^1 \geq 1 & (S \subset V; |S| \geq 1) & \quad (2.2)\\
& \sum_{l \in E^1} \epsilon_l^1 = 2m - y & & (2.3)\\
& \sum_{l \in E} \epsilon_l^2 = n - 1 & & (2.4)\\
& \sum_{l \in E^1} \epsilon_l^2 = y & & (2.5)\\
& \sum_{l \in E \setminus E^1} \epsilon_l^3 = m - y & & (2.6)\\
& \sum_{l \in E_i} (\epsilon_l^1 + \epsilon_l^2 + \epsilon_l^3) = 2 & (i = 2, \ldots, n) & \quad (2.7)\\
& \epsilon_l^t \in \{0, 1\} & l \in E, t = 1, 2, 3 & \quad (2.8)\\
& 0 \leq y \leq m & \text{and integer} & \quad (2.9)
\end{aligned}
$$

The objective function, equation (2.1), is minimizing the cost of the selected edges. The first three constraints (2.2)-(2.4) define a k-degree center tree. Constraint (2.5) defines the y-edges, incident to the depot, that will form the set E_2. Constraint (2.6) states that $m - y$ edges not incident to the depot will form the set E_3. The model is completed with (2.7) that ensures each vertex to have degree 2. Finally (2.8) and (2.9) define the decision variables.

The model can be solved most efficiently by relaxing the constraint-set (2.7) by introducing Lagrangean multipliers, because its relaxation decomposes the problem into three separate subproblems. By the time of writing the authors were able to successfully solve several VRP's up to twenty-five nodes with this method.

Fisher (1994) improves this method. He claims that the lower bound used by Christofides et al. (1981) is a weak link in their approach. Instead of finding a k-degree center tree, Fisher redefines the formulation in order to find a m-tree which is a set of $n+m$ arcs that span the complete set of $n+1$ nodes with degree $2m$ on the depot. On a set of test problems with n

ranging between 25 and 199 nodes he finds that this method obtains lower bounds of 98% of the optimum, compared to a performance of 85% by the method of Christofides et al. (1981).

Other algorithms that are members of this class are all sorts of branch-and-bound and branch-and-cut algorithms. Branch-and-bound algorithms are frequently used to solve all sorts of MIP-formulations. Here the idea is to relax the integer property of the decision variables. The solution of the polynomially solvable LP-problem is then used as our starting point. For each decision variable we then branch by adding additional constraints in which the variable is fixed to either nearest integer and solve the LP again. By continuously updating lower and upper bounds we try to reduce the size of our branching tree.

Branch-and-cut algorithms are the latest development in the exact solution approaches for the symmetric VRP (SVRP). Branch-and cut is a quite competitive algorithm for MIP formulations with an exponential number of constraints such as the TSP and VRP. We will explain the idea of the branch-and-cut method on the basis of a mathematical representation of the SVRP, given by Laporte (2007). The parameters and decision variables of this model are given below.

Parameters:

c_{ij}	=	the cost of travelling from node i to node j.
q_i	=	positive demand of customer i.
m	=	the number of vehicles available.
C	=	the vehicle capacity.

Decision variables:

x_{ij}	=	number of times edge (i, j) appears in the solution.

The mathematical representation:

$$\min \quad z = \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{2.10}$$

$$\text{s.t.} \quad \sum_{j\in V\setminus\{0\}} x_{0j} = 2m \tag{2.11}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad k \in V\setminus\{0\} \tag{2.12}$$

$$\sum_{i \in S, j \notin S} x_{ij} + \sum_{i \notin S, j \in S} x_{ij} = 2 \sum_{i \in S} q_i / C \qquad S \subset V \backslash \{0\} \tag{2.13}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in V \backslash \{0\} \tag{2.14}$$

$$x_{0j} \in \{0, 1, 2\} \qquad j \in V \backslash \{0\} \tag{2.15}$$

where the objective function, equation (2.10), minimizes the total cost of the selected edges. Equation (2.11) ensures that the degree of the depot equals $2m$, where equation (2.12) defines the degree of all other nodes to be 2. Equation (2.13) ensures that the number of vehicles serving an arbitrary subset of customers is larger than a certain lowerbound which is usually defined as $\sum\limits_{i \in S} q_i / C$. Equation (2.14) and (2.15) define the decision variable that can equal to 2 referring to the case that customer j is the only customer served on a route.

Recall that in the branch-and-bound method, we relax the integer-property of the decision variable and solve the resulting linear program. The objective function value of this linear program is a lower bound on the optimal objective function value of the original problem. However, when the number of constraints, which is growing exponentially with n is too large to solve the linear program, the branch-and-bound method shows deficiency.

The branch-and-cut method based on a cutting plane technique tries to resolve the deficiency of the branch-and-bound method. We define P as the complete set of constraints (2.13). We then take only a subset $\hat{P}$ of reasonable size and solve the linear relaxation of the resulting model. When the obtained solution is feasible for (2.10)-(2.15) this is also the optimal solution, otherwise we need to find one or more constraints of the set (2.13) that are violated. These constraints (cutting planes) are then added to the subset $\hat{P}$ and we repeat this process. According to Laporte (2007), these branch-and-cut methods have successfully solved VRP's up to 135 customers.

2.1.2 Dynamic programming methods

In the early days that the VRP was studied, a few researchers tried to apply dynamic programming to solve the VRP problem. An example of such a formulation is given by Laporte (1992). Here $V(S)$ is defined as the cost of a TSP visiting all customers in $S \subseteq V$. Then $f(k, S)$ is defined as the minimum cost of serving all customers in S with k vehicles.

$$f(k,S) = \begin{cases} V(S) & k = 1 \\ \min_{L \subset S \subseteq V} f(k-1, S \backslash L) + V(L) & k > 1 \end{cases} \tag{2.16}$$

The optimal solution is then given by $f(m, V)$. Since it requires excessive amounts of computations this method gives very poor performance on big problems in terms of computation time. The size of computations can, however, be reduced by adding two more constraints:

$$\sum_{i \in S} d_i - (k-1)D \leq \sum_{i \in L} d_i \leq D \tag{2.17}$$

$$\sum_{i \in V} d_i - (m-k)D \leq \sum_{i \in S} d_i \leq kD \tag{2.18}$$

Note that D is the capacity of a single vehicle. This reduction is useful, but it remains a poor method for larger problems. In recent papers we could not find any work related to solving a VRP with the use of dynamic programming.

2.1.3 Integer linear programming methods

Balinski and Quandt (1964) propose a method based on a set-partitioning formulation. The idea is simple; they prepare a set J with all feasible routes j. The model is then to select a subset of routes from J such that each customer is visited and the cost of the routes in this subset is minimized. The parameters and decision variables of their model are given below.

Parameters:

c_j	=	the cost (or distance) of route j.
a_{ij}	=	equals one if customer i is included in route j.
m	=	the number of vehicles available.

Decision variables:

$$x_j = \begin{cases} 1, & \text{if route } j \text{ is selected in the solution} \\ 0, & \text{otherwise} \end{cases}$$

The mathematical representation:

$$\min \quad z = \sum_{j \in J} c_j x_j \tag{2.19}$$

$$\text{s.t.} \quad \sum_{j \in J} a_{ij} x_j = 1 \qquad i = 1, \ldots, n \tag{2.20}$$

$$\sum_{j \in J} x_j \leq m \tag{2.21}$$

$$x_j \in \{0, 1\} \qquad \forall j \tag{2.22}$$

The objective, equation (2.19), is minimizing the cost of the selected routes. Equation (2.20) will ensure that each customer is visited and (2.21) is a limitation on the number of vehicles. Finally (2.22) defines the binary decision variables.

The bottleneck of this approach is the size of set J which grows exponentially with n. For larger problems this is very impractical and for this reason a column generation technique is used by several researchers. The idea is here to define (2.19)-(2.22) as the master problem and then define a second problem in which we find the column that will maximize improvements on z when it is added to set J.

The objective function of the sub-problem (the second problem) is defined as the reduced cost of route j. If there is a route with negative reduced cost, we add this route to the set of initial routes. The process is continued until no new routes can be found to further reduce the objective function value of the master problem. The column generation phase is still time consuming and therefore the initial set of routes will be of importance the CPU-time.

Since even this sub-problem is NP-hard, it received much attention in the literature. Bramel and Simchi-Levi (2001) propose to generate a large and good set of initial routes J and then solve the linear relaxation of (2.13)-(2.15) for a small subset $\hat{J} \subseteq J$. They look at the reduced cost of the routes in J and add the routes with negative reduced costs to $\hat{J}$. Repeat these steps till no routes can be added to $\hat{J}$.

This approach have been applied to many types of MIP formulations. A successful application to the VRP has been carried out by Sierksma and Tijssen (1998). In their paper they solve a real-life VRP by means of an integer linear programming formulation in combination with a column generation technique.

2.1.4 Remarks

The drawback of all these exact algorithms is that so far they have been proved to be successful only for relatively small problem instances. A recent review of some successful exact approaches by Toth and Vigo (2002) concludes that over the years the largest solvable instances have grown from about 25 customers to over a 100 customers. This means relatively important progress has been made, but problems of reasonable size remain unsolvable. They report some Euclidean capacitated VRP instances with 75 customers that are still unsolved.

Despite this fact we have seen successful applications in recent studies. Successful meaning that such exact algorithms gave optimal or near optimal solutions in acceptable CPU time.

We expect that as long as new approaches are developed and computers grow stronger, larger instances can be solved in the future. Toth and Vigo (2002) suggest several possible studies that are not (well) covered so far. An example they give are Dantzig-Wolfe decomposition based approaches (also known as branch and price algorithms).

2.2 Heuristic algorithms for the VRP

Heuristic approach cannot guarantee optimal solutions, but their goal is to find a "good" solution in reasonable time. Not all of the heuristics discussed were originally designed to solve the VRP, but were modified for this purpose.

In this section we will discuss some of the heuristic methods that are proposed in the literature to solve the VRP. According to Zanakis et al. (1989) who thoroughly scanned the literature (442 papers in 37 journals) for the use of different conventional heuristic methods, we can roughly make the following classification. However, we will not discuss all twelve classes mentioned in their paper. We only discuss the classes that according to us have proved to be of significant importance for solving the VRP and its extensions:

1. Construction heuristics
2. Decomposition and partitioning heuristics
3. Improvement heuristics

Now we will give an example of a heuristic for each of these classes and discuss its properties. Note that for a good solution approach a sequential use of several heuristics might be necessary in order to obtain satisfactory solutions. For example, constructive heuristics are frequently used to find initial feasible solutions which then can be improved using one or more improvement heuristics.

2.2.1 Construction heuristics

This class of heuristics gradually constructs solutions by adding nodes or arcs to the solution following a predefined set of rules. The famous nearest neighbor heuristic that was originally developed for the TSP, can also be used for solving the VRP. The heuristic works as follows. Let S be the set of all customers that are not routed yet. We randomly pick a customer as the starting point of our tour. Then we look in S for the closest node to the starting point. We add this customer to our route and remove it from S. We continue till $S = \emptyset$ and connect the last added node with the starting point and we obtain a Hamiltonian cycle. It tends to perform well in the beginning, but while adding the last customers to the tour, some expensive arcs have to be used. Even though it is highly unlikely to find the optimal solution in this way, it is possible to find a reasonable solution in polynomial time.

To modify this heuristic for the VRP, we simply add the rule that every time a vehicle is full we end the tour and start a new tour from the depot (starting point of the tour). Note that we select the node for insertion by the rule of nearest neighbor of the last inserted note. We could modify this rule into: cheapest insertion, farthest insertion, random insertion, etc.

Another important heuristic based on a constructional principle is the saving heuristic introduced by Clarke and Wright (1964). This heuristic starts with n tours that all serve a single customer. Then the cost that can be saved by merging route i and j for each pair i, j are defined as s_{ij}.

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \tag{2.23}$$

Here c_{i0} is the cost between node i and node 0 (the depot). We merge the routes with the highest savings, given that the capacity restrictions are satisfied and merging is applied to the nodes next to the depot in each route. We continue till no further savings can be obtained. Computational results of different methods are presented by Laporte and Semet (2001), see Table 2.1, given at the end of this chapter.

2.2.2 Decomposition and partitioning heuristics

This class includes both decomposition and partitioning heuristics as the difference is sometimes hard to expound. Decomposition heuristics solve a sequence of smaller sub-problems; the output of the previous sub-problem is used as the input for the next. Partition heuristics are quite similar as they also partition the original problem and solve these sub-problems independently from each other. Some examples of this class of heuristics are,

1. Cluster first, route second heuristic
2. Route first, cluster second heuristic
3. Petal heuristic

The cluster-first, route-second heuristic first makes up a set of rules to divide the customer set in several clusters. The goal of this first phase is to find m clusters of entire customer set, each of which is a disjoint set from others, and with equally distributed demand between the clusters. In the second phase a TSP is solved for each cluster individually.

The route-first, cluster second heuristic is doing the exact same steps, but in reverse order. First we try to find a giant tour by solving the TSP problem for all customers. Second we try to find an optimal partition of this tour to obtain feasible vehicle routes in terms of capacity and time limitations. Beasley (1983) is the first source to apply this idea to the VRP. He proposes several modifications of this approach to obtain good quality solutions for the classical VRP.

The Sweep Heuristic is a modification of a route-first, cluster second heuristic. It was first discussed by Gillett and Miller (1974). The idea is based on the following assumptions. We assume that all nodes have known coordinates on a plane and the distances are Euclidean. Then we calculate for each customer the polar coordinate angle with respect to the depot and order them in terms of these angles. After reordering we start assigning customers to vehicles in such a way that we start with the first customer on the list and keep adding customers to the route while keeping the route feasible. Once this is no longer possible, we finish the route and start a new route.

The last, but probably the most interesting heuristic of this class is an algorithm first discussed by Foster and Ryan (1976) in an attempt to employ the fact that many optimal solutions show a petal or an almost petal structure. In this approach the authors start similar as in the sweep method by assigning polar coordinate angles and reorder all customers accordingly. In the next step they list all feasible routes with a petal structure. In the last step they solve a linear program to optimality in which they select a set of feasible petal routes (a spanning petal) such that each customer is visited and the total traveled distance is minimized. A major advantage for this method is that additional constraints can easily be added, but the drawback is the CPU time.

This method has proved to give near optimal results for several problems with size ranging from 21 to 100 customers. In the year this paper was written these were good results. However to further improve this promising method, Ryan et al. (1993) propose an alternative method to find the optimal petal solution. This method is based on a shortest path technique. The authors first define a cyclic petal directed graph where the nodes correspond to customers and the arcs to generalized petals. In this they define a generalized petal as the petals obtained when reordering the customers in a non-radial cyclic order before generating the petals. Then they claim that the problem reduces to finding a shortest path on this cyclic petal directed graph. Computational results of different methods are presented by Laporte and Semet (2001), see Table 2.1, given at the end of this chapter.

2.2.3 Improvement heuristics

This class of heuristics is based on the idea to find improvements on feasible solutions that we have obtained in a different way. In each step of the improvement heuristic we try to find an exchange or transfer of customers on the tours that will improve our solution. A famous heuristic by Lin and Kernighan (1973) is called k-opt heuristic for TSP. Despite the fact that this paper is written more than 3 decades ago, the idea is still frequently used in new algorithms for the improvement steps.

In their paper the authors apply their heuristic to the TSP, but they already mention it can be modified to solve other combinatorial optimization problems. For the TSP the idea is the following. Assume we have a solution T with objective value $z(T)$ and any tour $T^{'}$ with length $z(T^{'}) < z(T)$. Suppose that these tours differ by k arcs. The algorithm then attempts to find two disjoint set of arcs $X = \{x_1, \ldots, x_k\}$ and $Y = \{y_1, \ldots, y_k\}$ such that if the arcs of set X are replaced by the elements of set Y, the result is a new tour $T^{'}$ with a lower objective function value.

Note that this algorithm is a generalization of the 2-opt and 3-opt algorithms frequently mentioned in the literature. The k-opt heuristic just assumes that the number of arcs that will be interchanged is a variable instead of an input parameter.

Computational results of different methods are presented by Laporte and Semet (2001), see Table 2.1, given at the end of this chapter.

2.3 Metaheuristics for the VRP

The latest development in solving combinatorial optimization problems is the use of Metaheuristics. Where conventional approximation methods are no longer sufficient, metaheuristics can be very helpful. They have been developed since early 1980's. Osman and Laporte (1996) define a metaheuristic as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space. Learning strategies are used to structure information in order to find efficiently near-optimal solutions. An important property is that in most metaheuristics we allow non-improving and infeasible solutions as intermediate steps to avoid local optima. Some classes of metaheuristics are

1. Constrained logic programming, CLP
2. Evolutionary Computing (Genetic Algorithms), GA
3. Neural Networks, NN
4. Simulated Annealing, SA
5. Tabu Search, TS
6. Nonmonotonic search trajectories
7. Threshold algorithms
8. Hybrid methods

Not all of these methods have (frequently) been used to solve the VRP. Therefore we will only discuss four of these classes of metaheuristic in this paper. Based on the frequency that these methods are applied to the VRP, we will discuss: GA, SA and TS. We note that metaheuristics frequently make use of conventional heuristics to find good starting points of the search trajectory.

2.3.1 Evolutionary Computing (Genetic Algorithms), GA

The principles of Genetic Algorithms are based on the fact that we maintain a population of solutions by an iterative reproduction process. In each step we select "parent" solutions to produce so called "offspring" solutions with some features of every parent. The quality of each offspring solution is measured by the objective function value. In general the number of offspring solutions that survive, i.e. the solutions with the best objective function values, is equal to the number of parent solutions chosen in order to keep the population size at the same level.

The question how a parent solution can represent a VRP solution is not straightforward. Baker and Ayechew (2003) choose for a representation of size n with values in the range of $[1, m]$ that represent which vehicle visits each customer. Note that for this representation the total distance traveled is not immediately known, but by solving m TSP's the optimal objective function for every solution can be obtained. A different representation could be a string of size $n + m$ with values in the range of $[0, n]$ that are ordered in the way the customers are visited. Each time a new vehicle is used, we separate this by adding a zero in the string. This way it is not necessary to solve m TSP's at each iteration. See Figure 2.1 for two different representations of the same VRP solution.

Also for creating offspring solutions there does not exist a clear set of rules that is to be used. Depending on the problem structure we can create offspring solutions by cross-over, randomness, etc. Computational results are scarce, but Baker and Ayechew (2003) claim that the GA is performing good, although it is beaten by TS in terms of solution quality. However, when they make some small adjustment (hybrid GA) they find solutions almost similar to the best known solutions. More often than the classical VRP, the GA has been used to solve the VRP with time windows.

parent	3	4	3	1	4	2	3	2	1				
parent	0	4	9	0	6	8	0	1	3	7	0	5	2

Figure 2.1: Different representations of a VRP solution

2.3.2 Simulated Annealing, SA

Simulated Annealing (SA) starts from an initial solution and at each iteration a solution in neighborhood of previous solution is selected. When an improvement is found we continue with this solution and redefine neighborhood as such, however when a worse solution is picked we continue with this solution with probability p_t where t is the iteration counter. The probability that a worse solution is picked will usually diminish over time, because when approaching the global optimum we would like to avoid decreasing steps.

Osman (1993) reports a successful implementation of SA for VRP. He uses a λ-interchange generation mechanism to determine neighborhood of a given solution. First two routes are selected and then we choose a subset of customers from each route such that the size of these sets is less than or equal to λ. When the interchange of these subsets is feasible in terms of capacity, the move is feasible. Note that the subsets of customers are allowed to be empty and therefore "donation" is also included in this method. Donation means that a customer is moved to a different route without returning another customer. Computational results show several good results for problem instances ranging in size from 30 to 200 customers.

2.3.3 Tabu Search, TS

Tabu Search (TS) works similar as SA, we also start with an initial solution and then search neighborhood of this solution. However, in TS we are always looking for the best solution in neighborhood which is then used as our next solution. To avoid cycling we put the "old" solutions in a Tabu-list. Members from this list cannot be selected anymore. TS has been proven to be very effective when solving the VRP.

Osman (1993) and Taillard (1993) have reported high quality solutions for most of the instances from the VRP-library for both symmetric and asymmetric cases. CPU-times are very high compared to conventional heuristics, but still acceptable. For instances up to 200 customers, both authors report CPU-times of sometimes close to 100 minutes.

2.4 Extensions of the classical VRP

As mentioned before it is possible to drop or add certain assumptions in order to obtain a new VRP. In the literature the most studied modification is the vehicle routing problem with time windows (VRPTW). The modification that is of most interest to our study is the vehicle routing problem with multiple use of vehicles. In the next chapter we will give a detailed description of this problem. In this section we will briefly discuss several other modifications that can be of importance to our and other case studies.

1. The mix-fleet vehicle routing problem
2. The vehicle routing problem with stochastic demands
3. The multi-depot vehicle routing problem (MDVRP)
4. The vehicle routing problem with backhauls (VRPB)
5. The vehicle routing problem with pick-up and delivery
6. Dynamic location-routing problems
7. Split delivery vehicle routing problems
8. The vehicle routing problem with time windows (VRPTW)

The first modification drops the assumption that the vehicle fleet has identical characteristics for each vehicle. Most common is a fleet of vehicles with capacity C_k for each vehicle k, but there are also multi-commodity applications where some vehicles might have refrigerated compartments and other vehicles that do not have this property. Either way, it should be clear that in some applications a mix of vehicles with different capacities or properties can be more useful than the use of a single vehicle type. When vehicle fleet composition is assumed to be fixed in advance, a small change in the formulation can solve this problem. A much more interesting question is what the optimal composition of the vehicle fleet should be. This is discussed in detail by Salhi and Rand (1993).

The second modification takes into account that customer demands are usually not known in advance. This means that we have to design routes in such a way that the probability that the capacity of a vehicle serving a route is not sufficient is minimized while the total distance traveled is minimized at the same time. For many applications this is a very realistic assumption. Consider distribution of consumer goods; we know that demands may change according to weather and other factors.

The third modification drops the assumption that a single depot is operated. The goal of this problem is to construct a set of routes in such a way that all the equations of the classical VRP hold true as well as that each route starts and ends in the same depot. Renaud et al. (1996) give a detailed formulation of the MDVRP. A trivial solution approach would be to assign each customer to the nearest depot and independently solve for each depot a VRP. However more sophisticated approaches are proposed in the literature to give better quality solutions. Renaud et al. (1996) propose a TS algorithm for this modification and report to find the best known solutions on 20 out of 23 benchmark problems.

The fourth modification is the VRP with backhauls. This problem considers that besides the deliveries to a set of customers, a second set of customers requires a pick up. In this problem it is assumed that each vehicles will first visit all customers that require delivery (linehaul customers) before it visits the customers that require pick-up (backhaul customers).

A generalisation of this variant is the VRP with pick up and delivery. Now we drop the assumption that each vehicle first visits all the linehaul customers. In other words combined effort of pick-up and delivery of goods is considered. The difficulty here is the continuously changing load of the vehicle that should be less than the capacity of the vehicle at each stop. Therefore when a full vehicle leaves the depot, it first has to serve one or more deliveries before a pick-up can be served. In other words the available capacity of vehicles determines whether or not a next stop is feasible.

The sixth modification considers simultaneously the decision problems of locating depots and designing routes to customers. Laporte and Dejax (1989) discuss an exact approach for small-scale problems and an approximate approach for bigger-scale problems. In both approaches the idea is to introduce a binary variable that equals 1 when the depot representing that variable is used. Associating that variable is a constant cost of operating that depot, which is used in the objective function.

The seventh modification of the classical VRP drops the assumption that every customer has to be visited exactly once by a single vehicle. In the split delivery VRP it is therefore now possible that the total demand of a single customer is delivered by multiple vehicles. Archetti et al. (2006) propose a TS algorithm for this variant of the VRP and report to find better solutions than previous methods.

Finally we mention the vehicle routing problem with time windows (VPRTW). This variant of the VRP is probably the variant that has received the most attention in the literature. This could be explained by the practical importance of time windows. Time windows occur when customers require pick-up or delivery within pre-specified times. In the literature a distinction is made between soft time windows that can be violated against a penalty-cost and hard time windows that cannot be violated. For a few decades the VRPTW has been discussed in the literature. An overview of the early published papers is given by Solomon (1987). He mentions and compares several approximation heuristics that are all modifications of well known heuristics for the classical VRP.

2.5 Computational results

Most approaches discussed in this chapter have been tested on benchmark problems from the literature. All benchmark problems are named as follows X111-11y where "X" is defining the character of the problem instance, like

- "E" for Euclidean symmetric VRP instances
- "A" for asymmetric capacitated VRP instances
- "D" for symmetric distance constrained VRP

Following X (i.e. 111) is the number of nodes of the instance, which is followed by the number of vehicles (i.e. 11). Finally "y" is the character that identifies the source where the problem data are introduced, like

- "c" refers to Christofides, Mingozzi, and Toth (1979)
- "e" refers to Christofides and Eilon (1969)

- "f" refers to Fisher (1994)

We have already briefly discussed the performance of several heuristics and metaheuristics, but in this section we will try to compare some of the results. In Table 2.1 we have collected solutions and their CPU times from different methods. In the table, the entries under z refer to the solution value of the associated problem instance, whereas the entries under CPU are the CPU-time in seconds.

When comparing the CPU times we have to take the difference, in the computer used to solve that problem instance, into account. For this reason we present the characteristics of the different computers in Table 2.2. Given these different computers it is very hard to draw any solid conclusions regarding the CPU time, but a rough conclusion is that conventional heuristics need less CPU than metaheuristics. Looking at the solution quality we notice that metaheuristics are in general performing better, in particular the TS algorithm of Taillard (1993).

Usually the CPU time and solution quality are the only two properties taken into account when evaluating a solution approach. However, Laporte (2007) is mentioning two other important properties that are of importance when assessing algorithms. These are simplicity of implementation and flexibility. The first one meaning that researchers often put too much weight in the accuracy of their algorithm by introducing lots of user-controlled parameters, which can be a handicap for later users. Flexibility means the ease in which the algorithm can adopt important side constraints encountered in practical applications.

Table 2.1: Computational results

	heuristics							
	C&W		Sweep		Petal		C&W + 3-opt	
Problem	z	CPU	z	CPU	z	CPU	z	CPU
E051-05e	584.64	<0.2	532	12.2	531.90	0.10	578.56	<0.2
E076-10e	900.26	<0.2	874	24.3	885.02	0.07	888.04	<0.2
E101-08e	886.83	<0.2	851	65.1	836.34	0.32	878.70	<0.2
E101-10c	618.40	<0.2	560	11.4	560.08	0.09	616.66	<0.2
E121-07c	975.46	<0.2	933	23.8	968.89	0.07	974.79	<0.2
E151-12c	973.94	<0.2	888	58.5	877.80	0.25	968.73	<0.2
E200-17c	1287.64	<0.2	1230	134.7	1220.20	0.26	1284.63	<0.2
D051-06c	1538.66	<0.2	1518	238.5	1515.95	0.35	1523.24	<0.2
D076-11c	1596.72	<0.2	1776	85.5	1773.69	0.26	1587.93	<0.2
D101-09c	875.75	<0.2	949	53.6	894.77	0.17	868.50	<0.2
D101-11c	1395.74	<0.2	1389	252.2	1406.84	0.41	1386.84	<0.2
D121-11c	1133.43	<0.2	1079	142	1070.50	0.41	1128.24	<0.2
D151-14c	833.51	<0.2	937	50.8	824.77	0.21	824.42	<0.2
D200-18c	1071.07	<0.2	1266	104.3	1252.84	0.61	1049.43	<0.2
average:		<0.2		89.8		0.26		<0.2

	metaheuristics						
	SA		GA		TS		best
Problem	z	CPU	z	CPU	z	CPU	
E051-05e	528	167.4	**524.61**	23.00	**524.61**	49	**524.61**
E076-10e	838.62	6434.3	838.89	617.00	**835.26**	53	**835.26**
E101-08e	829.18	9334.0	829.47	717.00	**826.14**	580	**826.14**
E101-10c	826.00	632.0	**819.56**	1285.00	**819.56**	3800	**819.56**
E121-07c	1176	315.8	1046.27	1483.00	**1042.11**	3000	**1042.11**
E151-12c	1058	5012.3	1034.80	1961.00	**1028.42**	17	**1028.42**
E200-17c	1378	2318.1	1327.78	5261.00	1298.42	51	**1291.45**
D051-06c	**555.43**	3410.2	**555.43**	429.00	**555.43**	1100	**555.43**
D076-11c	909.68	626.5	909.68	449.00	909.68	1100	**909.63**
D101-09c	866.75	957.2	867.68	1904.00	**865.94**	-	**865.94**
D101-11c	890	305.2	867.13	585.00	**866.37**	-	**866.37**
D121-11c	1545.98	7622.5	1546.31	1063.00	**1541.14**	340	**1541.14**
D151-14c	1164.12	84301.2	1166.27	2242.00	**1162.55**	3900	**1162.55**
D200-18c	1417.85	5708.0	1425.27	6433.00	1397.94	1500	**1395.85**
average:		9081.8		1746.6		1290.8	

Table 2.2: Hardware properties

Method:	Source:	Year	Avg. CPU
C&W	Laporte and Semet (2001)[1]	2001	<0.2
Sweep	Laporte and Semet (2001)[2]	1979	89.8
Petal	Laporte and Semet (2001)[3]	1996	0.3
C&W + 3-opt	Laporte and Semet (2001)[4]	2001	<0.2
SA	Osman (1993)[5]	1993	9081.8
GA	Baker and Ayechew (2003)[6]	2003	1746.6
TS	Gendreau et al. (2001)[7]	1996	1290.8

[1] Sun Ultrasparc 10 workstation (42 Mflops)
[2] CDC6600
[3] Sun Sparcstation 2 (210.5 Mips, 4.2Mflops)
[4] Sun Ultrasparc 10 workstation (42 Mflops)
[5] VAX 8600 computer
[6] Pentium 266Mhz (83.7 Mflop/s)
[7] Silicon Graphics 4D/35

2.6 Remarks

In this chapter we have given an extensive survey where we discussed several solution approaches in terms of exact and heuristic algorithms for the VRP. In Table 2.3 we have given an overview of some important papers we have discussed.

We have seen that there is no exact algorithm capable to solve VRP's of realistic size in polynomial time. Therefore we have discussed the use of heuristic search methods that can help us in finding near optimal solutions in polynomial time. We have seen that some heuristics perform better than others, but in general we can conclude that the conventional heuristics usually do not find acceptable solutions.

For this reason the development of metaheuristics is necessary. In these heuristics we perform an iterative process of replicating solutions and by some set of rules accept new solutions and delete old solutions. These methods are build in such a way that local optima are avoided. The best reported results are calculated with TS and GA.

Table 2.3: Overview of important VRP papers

Category	Author	Year	Topic
History	Schrijver, A.	2005	History of combinatorial optimization
Survey	Solomon, M.	1987	Survey of VRPTW
	Zanakis, S.	1989	Categorized survey heuristics
	Laporte, G.	1992	Exact and approximate algorithms
	Toth, P. and Vigo, D.	2002	Branch & bound algorithms
	Laporte, G.	2007	Exact and approximate algorithms
Exact approaches	Balinski, M.	1964	Set-partitioning
	Christofides, N.	1981	Spanning tree
	Fisher, M	1994	Minimum K-trees
Heuristics	Clarke, G.	1964	Saving heuristic
	Lin, S. and Kernighan, B.W.	1973	Improvement heuristic
	Gillet, B.	1974	Sweep heuristic
	Beasley, J.	1983	Route-first, Cluster-second
	Ryan, D.	1993	Petal heuristic
Meta-heuristics	Osman, I.	1993	SA & TS
	Taillard, E.	1996	TS
	Baker, B. and Ayechew, M.	2003	GA
Modifications	Laporte, G. and Dejax, P.J.	1987	Dynamic location-routing
	Salhi, S. and Rand G.	1993	Vehicle fleet composition problem
	Renaud et al.	1996	Multi-depot vehicle routing
	Archetti, C.	2006	Split delivery VRP
Applications	Sierksma, G. and Tijssen, G.	1998	Routing helicopters
	Pamuk et al.	2004	Product delivery system

CHAPTER 3

THE VEHICLE ROUTING PROBLEM WITH MULTIPLE USE OF VEHICLES

In the previous chapter we have seen that many researchers have been working on the vehicle routing problem and its modifications. In this study we are predominantly interested in the vehicle routing problem with multiple use of vehicles.

Contrary to the classical VRP the vehicles are now allowed to be replenished at the depot. This implies that we have to adjust our model such that the depot, besides being the start and end node of a tour, can also be an intermediate node in the same tour. We will discuss several related formulations and the associated solution methods in the literature. We will also look what happens with the problem when we add additional constraints such as time windows, backhaul customers, heterogeneous vehicle fleet, etc.

This topic is usually overlooked in the literature despite the fact that it is relevant to many real-life applications. Our main goal is to give a complete survey of all published articles related to this topic and to discuss how we can approach practical problems based on the methods related with the VRP with multiple use of vehicles. To illustrate the latter we will discuss a real life case study in Chapter 5.

3.1 The VRPM in the literature

Surprisingly this topic received very little attention in the literature. It is surprising because of the number of applications it can be used for. In our opinion models based on the VRPM can be applied to many real life situations.

A trivial example would be the increasing e-commerce sector in Europe and the United States. Consider the case where we have to serve many customers in a small geographical area; home deliveries in a big city from a warehouse on the outskirts of the city. It is now plausible that the use of small vehicles each of which serves several routes on a single day is more efficient than the use of less bigger trucks on longer routes. Furthermore, in most European cities the streets are narrow and the only vehicles that are small in size can serve on such streets.

In this example we are dealing with a low-volume commodity that has to be transported a short distance. The use of vehicles with smaller capacity in this example is easy to justify, since in many big cities the use of larger vehicles is very inefficient and sometimes even restricted with aggravating time windows. Of course for practical problems where commodities have to be transported over long distances the use of a VRP with multiple use of vehicles seems not very likely.

Now, we first look at the already published papers in the literature. The first published paper about the VRPM is from Taillard et al. (1996), however they admit that Fleischmann was the first to explicitly define this problem in 1990.

To avoid misunderstandings we make a clear distinction between a route and a tour. A route is a single trip for a vehicle starting and ending at the depot without replenishing. A tour consists of one or more routes and encompasses a working day of a vehicle.

3.2 Heuristic algorithms for the VRPM

As mentioned Fleischmann was the first to propose a solution approach for the VRPM. His approach is built up from two phases. In the first phase a saving based heuristic is used to construct routes and in the second phase a bin packing algorithm is used to construct feasible working days for each of the vehicles.

Taillard et al. (1996) propose a tabu search algorithm based on the idea of Rochat and Taillard (1995). They start with mentioning a trivial heuristic that is not further discussed, because it gives rather poor results. The idea here is to artificially lower the maximum duration of a route. For instance, if the vehicles are allowed to work 10 hours a day, we could set the maximum duration to 5 hour and assign the routes to vehicles using a bin-packing algorithm. Of course such an approach can give very poor results depending on the problem structure. In some cases it might even not be practical if every vehicle is forced to make at least two or more trips in each day. The case that some vehicles make 1 trip and other vehicles make multiple trips is excluded in this approach.

The tabu search algorithm they discuss in more detail consists of three steps. The first step involves the generation of several feasible VRP solutions using the algorithm proposed by Taillard (1993). The second step consists of finding a subset of routes from the solutions found in step 1 using an enumerative algorithm. The last step uses a bin packing algorithm to create feasible working days to create a feasible solution for the VRPM.

The authors tested this algorithm on a number of generated problems, with customer sets varying in size between 50 and 199. They run each of these instances with several different values for the number of vehicles. This way they create 52 replicated VRPM instances out of 9 VRP instances, see Table 3.1.

Table 3.1: Characteristics of benchmark test problems

Problem number	Source	m	z^{RT} [1]
1	E051-05e	1,...,4	524.61
2	E076-10e	1,...,7	835.26
3	E101-10c	1,...,6	826.14
4	E151-12c	1,...,8	1028.42
5	E200-17c	1,...,10	1291.44
6	E121-07c	1,...,5	1042.11
7	E101-08e	1,...,6	819.56
8	E072-04f	1,...,3	241.97
9	E135-07f	1,...,3	1162.96

[1] Optimal objective function value by Rochat and Taillard (1995)

Note that Rochat and Taillard (1995) solved the VRP with an unspecified number of vehicles. In such a VRP the objective is simply to minimize the total distance regardless of the number of vehicles. For the VRPM this is an acceptable lower bound since in the VRPM multiple routes can be served by the same vehicle as long as the combined duration does not exceed the maximum allowable driving time of a vehicle. Note that these values are not optimal solution values even to classical VRP, but only an estimate of the "lower bound" for VRPM. It is therefore possible to find objective values lower than this z^{RT} value.

In Taillard et al. (1996) the maximum allowable driving time T for a single vehicle is determined by considering the number of vehicles available and an estimate (z^{RT}) of total length (time) of all routes. For this reason they define two estimates, T_1 and T_2 specified below, where $[x]$ is the value of x rounded to the nearest integer.

$$T_1 = [1.05z^{RT}/m] \tag{3.1}$$

$$T_2 = [1.1z^{RT}/m] \tag{3.2}$$

The authors try to find solutions that are feasible with respect to T_1 and T_2. This means that an infeasible solution exceeds the maximum allowable driving time, which implies the presence of overtime. The authors penalize this overtime with a factor 2 in order to give the cost of the infeasible solutions. A second measure that is introduced to measure the quality of the infeasible instances is the longest tour ratio (LTR).

$$\text{LTR} = \frac{\text{length of longest tour in the solution}}{T} \tag{3.3}$$

In equation (3.3), T can be either T_1 or T_2. It is obvious that ratios closer to 1 can be considered as better solutions. The authors conclude that this algorithm can find high quality solutions to the VRPM within reasonable computation times. Computational time for this algorithm for the above introduced VRPM instances vary between 300 and 4500 seconds. However, not for all of the 52 instances, the algorithm produces feasible solutions. Using T_1 they solved 38 out of 52 instances, while using T_2 gives a performance of 48 out of 52. We have to notice that the authors run this algorithm 5 times for each instance, sometimes only 1 run produced a feasible solution. They state that likelihood of finding a feasible solution grows for problems with more customers. This is due to the fact that the first step of the procedure produces more routes when n increases.

They do not compare the computational results of this algorithm with any other algorithms (for instance, with the algorithm proposed by Fleischmann) so it is not possible to judge the results on quality. Another drawback of this approach is that they do not take any additional constraints into account such as time windows, multiple vehicle type, etc.

Golden et al. (1997) try to apply the idea of this approach to the minmax vehicle routing problem. These are problems in which for equity reasons the objective is to minimize the maximum distance traveled by any vehicle. Frederickson et al. (1978) and Franca et al. (1995) have discussed this variant of the classical vehicle routing problem. Golden et al. (1997) then claim to be the first to propose an algorithm for the minmax capacitated VRPM. Their algorithm is tested on the VRPM problem instances generated by Taillard et al. (1996). For a measure of solution quality they use the following statistic:

$$x = 100(\frac{m\bar{z}}{z^{RT}} - 1) \tag{3.4}$$

where $\bar{z}$ is defined as the length of the longest tour of a single vehicle. Again assuming z^{RT} here is functioning as a lowerbound, x will always be nonnegative. Observing the values of x, we notice that when m increases, the value of x is also increasing. This is easy to explain since it is plausible that when more vehicles are used to serve the same set of customers in less time tours will vary more in length.

Brandao and Mercer (1997) propose a different Tabu Search algorithm for the VRPM. This algorithm was specially designed to be able to solve real distribution problems. In their paper they discuss a real life problem for Burton's Biscuits Ltd. Some of the problem specific constraints they took into account in this study are:

- Time windows
- Heterogeneous vehicle fleet
- Restricted access to certain customers by certain vehicle types
- Maximum daily driving time for drivers
- (Un)loading times

The conclusion of this paper was that the algorithm performed 20% better than the manual scheduling which was used till that time. However, since many problem specific constraints were used the algorithm could not be compared to the one of Taillard et al. (1996). To make such a comparison, they simplified this algorithm, (see Brandao and Mercer, 1998)

This simplified algorithm consists of two phases. In phase one they generate an initial solution with a constructive algorithm in which they use two independent procedures, nearest neighbor and insertion to find a layer. A layer is then defined as a set of routes such that each route belongs to a different vehicle. The solution of this phase is then used as the starting solution of the second phase. This phase is a tabu search algorithm with two sub-phases. The difference between these phases is that in the first one a solution is allowed to be infeasible in terms of allowable driving times.

In phase one at each iteration of the Tabu Search we have a set of routes N_r and a set of customers that are randomly chosen from each route to serve as candidates for a possible move. For moves they consider two possibilities, an insert or swap move. At the end of the iteration, they update the best solution found so far (feasible or infeasible) and repeat this process until a given number of iterations is attained without improving the best solution so far. During phase one, a table of frequencies is created to count the number that each customer has been moved between two routes. In the second phase, the set C of candidates for a possible move is fixed by the customers that have been moved below average in phase one. This phase also ends after a certain number of iterations without improving the best found solution.

The authors apply their heuristic on test problems introduced in Table 3.1. Taillard et al. (1996) are so far the only paper publishing results on these test problems. In order to compare their approaches, we refer to their results with TLG. Using T_1 as the maximum tour length for each vehicle, they solved 37 out of 52 instances, compared to 38 from TLG. However using T_2, all instances are solved compared to 48 of TLG. Then further details are given on the infeasible problem instances. They conclude that, in terms of total length, their approach performs 2.1% poorer than TLG, but in terms of LTR, they perform 1.2% better than TLG, meaning that their approach performs better in terms of balancing the tours. Computation times are ranging from 3 to 80 minutes, which is comparable with TLG. Again 5 runs are done for each instance and only the results of the best run is taken into account. On the other hand the strength of this approach is that the authors have proven it to be effective on real life examples after adding some problem specific side constraints.

Recently a new multi-phase heuristic is proposed by Petch and Salhi (2004). The algorithm is designed to generate a greater diverse set of routes which increases the probability of finding a more successful bin-packing solution. The phases are briefly shown in Figure 3.1. In step 1 an initial set of VRP solutions is created. In the next step we order the VRP solution according to the objective function value in nondecreasing order in a list L. In the third step, we select a solution from the set L which is then used to construct a VRPM solution with a bin-packing algorithm. After improving this solution, we compare the VRPM solution with the best known solution. If a better solution is found, the best known solution is updated and this process is continued till L is empty. The last phase of this algorithm is producing a second initial VRP solution sample and the process is repeated.

step 1	Generate VRP solution sample using a saving approach.
step 2	Order VRP solutions according to total driving time in a list L.
step 3	Take VRP solution with lowest total driving time from L.
	Construct VRPM solution with bin-packing algorithm.
step 4	Improve the VRPM solution.
step 5	Compare results with best solution found so far.
	Update best solution if applicable.
	While $L \neq \emptyset$ return to step 3.
step 6	More VRP solution are derived and placed in L.

Figure 3.1: Multi-phase heuristic of Petch and Salhi (2004)

They compare the computational results on the benchmark instances with the results from TLG and Brandao and Mercer (1998) (we will refer to these solutions with BM). For T_1 this multi-phase algorithm produces feasible solution for only 24 of the 52 instances compared to 38 from TLG and 37 from BM. Also for T_2, this algorithm is performing poorer than the other two, 47 compared with 48 from TLG and 52 from BM. However, when overtime is required the algorithm gives solutions with competitive overtime restrictions. To be precise they claim to decrease the average overtime against TLG by 29.6%, while compared to BM they find a 25.3% higher average overtime. The computation times for this algorithm are ranging between 60 and 2460 seconds.

To summarize, Petch and Salhi (2004) have presented a multi-phase algorithm to solve the

VRPM. In terms of solution quality this algorithm is not performing better than the existing TS-algorithms of Taillard et al. (1996) and Brandao and Mercer (1998). The computational times are better, but when taking the computational power of their computer into account this difference is negligible. We think the drawback of their approach is the lack of combining elements from different good VRP solutions together. Each VRPM solution is constructed with elements from only one VRP solution. We will come back to this point in Chapter 4.

Lin and Kwok (2006) proposed multi-objective metaheuristics for the VRPM. Instead of just routing decisions, they simultaneously take location decisions into account. This topic is too far out of the scope of this study and shall therefore not be discussed in more detail.

Another paper related to VRPM is Kim et al. (2006) about an application of waste collection in which they consider multiple disposal trips. In their paper they introduce a new set of benchmark problems and propose a solution approach for the commercial waste collection. In this application they take the following constraints into account:

- Hard time windows (i.e. earliest and latest starting time at each node)
- Driver lunch breaks
- Vehicle capacity
- Limited routing time per vehicle

Since these additional constraints are considered, the proposed algorithm cannot be used to solve the benchmark problems of TLG. For this reason their computational results are too far out of the scope of this study.

The last published paper in tackling the VRPM is from Olivera and Viera (2007). They propose an algorithm based on adaptive memory programming principle which was introduced by Rochat and Taillard (1995) as an enhancement of TS. Also this idea has been applied by Golden et al. (1997) to solve the minmax variant of the VRPM.

The idea is the following: a set of solutions is stored in the memory M. Now every time a new solution is constructed combining components of M and a local search procedure is applied to this new solution to obtain an improved solution. The components of this improved solution are stored in M and we continue this iterative process until a stopping criterion is met.

Their approach is actually comparable with the approach of TLG. However, the assignment of routes to vehicles is done in every iteration to obtain intermediate solutions for the VRPM while TLG only applies an assignment heuristic at the end of their approach. They report feasible solutions on several so far unsolved instances. For T_1 they have solved 46 out of 52 instances and for T_2 they have solved all instances, but computational times ranging from 10 to 320 seconds which are acceptable computational times.

3.3 Exact algorithms for the VRPM

Azi et al. (2007a,b) presented two papers about the VRPM. In one paper they discuss the single-vehicle case and in the other they generalize the same idea for the case where we have a vehicle fleet. The main difference with the previously discussed papers is that they try to solve the problem with an exact algorithm based on a mathematical programming formulation. Besides that the authors include some practical constraints such as time windows and service times. We start by introducing the parameters in Azi et al. (2007a).

Parameters:

c_{ij}	$=$	the cost of travelling from node i to node j.
q_i	$=$	positive demand of customer i.
a_i	$=$	the earliest starting time of service at customer i.
b_i	$=$	the latest starting time of service at customer i.
s_i	$=$	the service time for customer i.
σ_r	$=$	the set-up time for route r (loading at the depot).
C	$=$	the vehicle capacity.
β	$=$	coefficient for set-up times.
$t_{\max}$	$=$	maximum duration of a route.

Decision variables:

$$x_{ij}^r = \begin{cases} 1, & \text{if arc } (i,j) \text{ is in route } r \\ 0, & \text{otherwise} \end{cases}$$

$$y_i^r = \begin{cases} 1, & \text{if customer } i \text{ is in route } r \\ 0, & \text{otherwise} \end{cases}$$

t_i^r	=	the starting time of service at customer i in route r.
t_0^r	=	the starting time of route r.
t_{n+1}^r	=	the end time of route r.

Further symbols that have to be introduced before the mathematical representation can be presented are $N = \{1, \dots, n\}$ for the customer set and N^+ for $N \cup \{0, n+1\}$ where 0 and $n+1$ are representing the depot. Similarly A is the arc set with arcs $(i,j), i,j \in N$ and A^+ the extended arc set including all arcs to and from the depot. The single vehicle is serving a set K of routes $\{1, \dots, k\}$, some of these routes might be empty.

$$\min \quad \sum_{r \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^r \tag{3.5}$$

$$\text{s.t.} \quad \sum_{j \in N^+} x_{ij}^r = y_i^r \qquad i \in N, r \in K \tag{3.6}$$

$$\sum_{r \in K} y_i^r = 1 \qquad i \in N \tag{3.7}$$

$$\sum_{i \in N^+} x_{ih}^r - \sum_{j \in N^+} x_{hj}^r = 0 \qquad h \in N, r \in K \tag{3.8}$$

$$\sum_{i \in N^+} x_{0,i}^r = 1 \qquad r \in K \tag{3.9}$$

$$\sum_{i \in N^+} x_{i,n+1}^r = 1 \qquad r \in K \tag{3.10}$$

$$\sum_{i \in N} q_i y_i^r \le Q \qquad r \in K \tag{3.11}$$

$$t_i^r + s_i + t_{ij} - M(1 - x_{ij}^r) \le t_j^r \qquad (i,j) \in A^+, r \in K \tag{3.12}$$

$$a_i y_i^r \le t_i^r \le b_i y_i^r \qquad i \in N, r \in K \tag{3.13}$$

$$t_0^1 \ge \sigma^1 \tag{3.14}$$

$$t_{n+1}^r + \sigma^{r+1} \le t_0^{r+1} \qquad r = 1, \dots, k-1 \tag{3.15}$$

$$\sigma^r = \beta \sum_{i \in N} s_i y_i^r \qquad r \in K \tag{3.16}$$

$$t_i^r \le t_0^r + t_{\max} \qquad i \in N, r \in K \tag{3.17}$$

$$x_{ij}^r, y_i^r \in \{0, 1\} \tag{3.18}$$

Equation (3.5) is the objective function that is minimizing the overall traveled distance. Equation (3.6) and (3.7) state that each customer is visited exactly once. Equation (3.8) assures that exactly one vehicle is entering and leaving customer h. Equation (3.9) ensures that each route is leaving the depot where (3.10) does the opposite and ensures that each route is turning back at the depot. Equation (3.11) guarantees the vehicle capacity is not exceeded for each route r. Equation (3.12) defines the starting times of service at each customer and (3.13) guarantees that these starting times satisfy the time windows. Equation

(3.14) and (3.15) ensure that the routes cannot start before the vehicle is prepared (loaded) at the depot using the set-up times. Equation (3.16) defines the set-up cost as a linear function of the total service time on that particular route. Equation (3.17) defines the maximum time for each route. Finally equation (3.18) is defining the binary decision variables.

In practice it can occur that due to high demand not all customers can be served within their time windows. We could in this case hire an additional vehicle or consider a common carrier and solve the generalized model discussed later in this section. When this is not an option, we can modify our objective function to maximize the number of customers visited while minimizing the total distance traveled. In this case equation (3.5) would be replaced by (3.19), where $p(i)$ is the profit made when customer i is serviced.

$$z = \sum_{r \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^r - \sum_{i \in N} p(i) y_i^r \tag{3.19}$$

The authors try to solve this problem via an approach that is based on the elementary shortest path algorithm with resource constraints proposed by Feillet et al. (2004). In phase one a set of feasible non-dominated routes is constructed. In phase two, some of these routes will be selected to build a feasible working day. The algorithm cannot be tested on the VRPM problem instances introduced by Taillard et al. (1996) due to the absence of time windows. Instead, the algorithm is tested on VRPTW problem instances from Solomon (1987), deadlines are added and tests were run on a 900 MHz Sun Ultra III with 8GB of RAM. We notice that CPU times can vary from only a few seconds to a day. The CPU times are depending on problem size for obvious reasons and $t_{\max}$. When $t_{\max}$ is not tight enough, the number of feasible routes is exploding and strongly effecting the CPU.

The drawback of their study is that they only consider a single vehicle and therefore clearly not very useful for practical applications. For this reason, the authors are currently working on a generalisation of this approach where it is possible to have many vehicles. A working paper is already available, see Azi et al. (2007b).

3.4 Remarks

In this chapter we have seen that the VRPM is a very complex problem that is very hard to solve. The available literature is very limited compared to its importance in real life applications. Only four papers have been published that propose solution approaches for the classical VRPM and all of these approaches are tested on the same problem instances. Taillard et al. (1996), Brandao and Mercer (1998), and Olivera and Viera (2007) define meta-heuristic solution approaches. Petch and Salhi (2004) are so far the only authors proposing a multi-phase algorithm for the VRPM.

Olivera and Viera (2007) is clearly performing the best in terms of solution quality as well as computational times, but no comments are made for the use of this algorithm in practical applications. The only authors that explicitly discussed a real life example are Brandao and Mercer (1997). They conclude that compared with a manual scheduling system, their approach can save up to 20% on the daily driving times.

Furthermore we have seen the paper of Azi et al. (2007a) that is proposing an exact solution approach for the VRPM with a single vehicle. They include time windows in their approach, but since they only use one vehicle, the practical use is very limited.

In the next chapter we will discuss two models that can solve the VRPM. Our goal is to develop a model that can easily adopt several side-constraints. The model should also be solvable within reasonable CPU times.

CHAPTER 4

MATHEMATICAL MODELS FOR VRPM

In this chapter we will basically present two models to solve the VRPM. Our aim is to present a solution approach that is able to solve a practical application. This means that the approach should be capable of successfully adapting several side constraints that might occur within the application.

The first model is an integer formulation based on the route-first cluster-second principle. We have chosen for a conventional based heuristic since they are relatively simple and therefore require very low CPU times. It is also an approximate representation of the problem in an integer formulation. Second we adapt the classical set-covering approach such that it can solve the VRPM. Actually it is a true representation of the problem in an integer formulation, but only when the optimal set of routes is included in the input set. Both approaches are enhanced using improvement routines. We show that the set-covering approach can easily be adapted to handle side constraints frequently observed in real-life applications.

Both approaches are tested on benchmark problems given in the literature and we compare the results with the previously published results.

4.1 Mathematical model for the route-first cluster-second approach

The route-first cluster-second heuristic for the classical VRP is already introduced in Chapter 2, recall that in the first step we solve a TSP problem to obtain a giant tour. In the second step of the algorithm we assign customers to individual routes such that each route is satisfying the vehicle capacity and the combined length is minimized. Recall from Chapter 2 that Ryan et al. (1993) proposed to use a shortest path technique in the second phase. In this integer formulation we enhance this idea for the VRPM.

We start with presenting a mathematical representation of a shortest path model. The parameters and decision variables are given below.

Parameters:

c_{ij} = the cost of travelling from node i to node j.

Decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if arc } (i,j) \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

The mathematical representation (which we will refer to as SP1):

$$[\text{SP1}] \quad \min \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{j=0}^{n} x_{0,j} = 1 \tag{4.2}$$

$$\sum_{j=0}^{n} x_{j,n} = 1 \tag{4.3}$$

$$\sum_{i=0}^{n} x_{ij} = \sum_{i=0}^{n} x_{ji} \qquad \forall j = 1, \ldots, n-1 \tag{4.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j = 0, \ldots, n \tag{4.5}$$

In the objective, equation (4.1), the total cost of the arcs selected in the optimal solution is minimized. We are interested in a shortest path starting at node 0 and ending at node n. Equation (4.2) is ensuring one arc is leaving the starting point and equation (4.3) is defining one arc to enter the end point of the shortest path. Equation (4.4) are the connectivity constraints that ensure inclusion of all nodes into the resulting set of individual (for vehicle) tours. Finally equation (4.5) is defining the binary decision variables. In this formulation the optimal solution will represent the shortest path from node 0 to node n disregarding whether all other nodes are visited or not.

step 1	Solve the giant TSP for n customers.
step 2	Create shortest path graph.
step 3	Solve shortest path model.
step 4	Improve solution.

Figure 4.1: Route-first, cluster-second based solution approach for the VRPM

In Figure 4.1 we have systematically showed the steps of our approach. In step 1 we solve the giant TSP using Concorde (Cook, 2008). Concorde is a software package that can solve TSP. It has solved to optimality, 104 out of 110 TSP's from the TSPLIB, with the largest instance solved having 15,112 customers. In step 2 we create a shortest path graph. The resulting graph is acyclic. We order the nodes and re-index every customer in the order as they appear on the giant TSP solution. Look at Figure 4.2, here $i = 0$ represent the depot and $i = 1$ the node that is visited first on the giant tour, etc. Finally node $i = n$ is representing the node that is visited last in the giant tour.

Because of the technical issues, we define an arc (i, j) as follows. Arc (i, j) represents a route starting at the depot, traveling to node $i+1, i+2, \ldots, j$ and back to the depot. This means that each arc can be interpreted as a route itself. The total length of this route is given by c_{ij}. When defining the arc we make a major assumption. We define arc (i, j) if and only if $i < j$, because it is an acyclic graph. Therefore arc (i, j) is only defined if and only if $\sum_{h=i+1}^{j} q_h \leq C$ where C is the vehicle capacity.

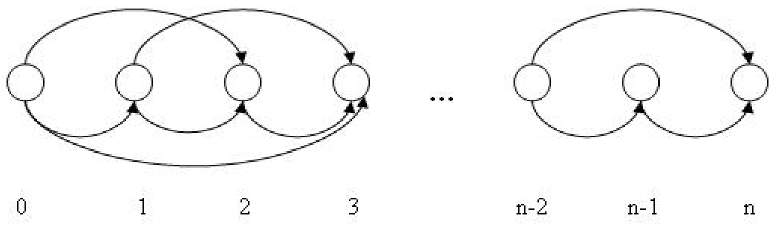

Figure 4.2: Shortest path graph

In step 3 we solve the shortest path with a modified version of the standard shortest path model SP1. Since we assume in this model that each vehicle can make several routes on a single day, we need an additional constraint ensuring that the combined length of these routes is not exceeding a prescribed limit T. In this study we assume that T is equal for all vehicles. Since our approach is solved in a framework of the route-first cluster-second algorithm, we assume, while solving the shortest path problem, that the order in which the customers are visited is fixed. This means that the order in which the customers are visited is an input for this model. The parameters and decision variables of this model are given below.

Parameters:

c_{ij}	$=$	the cost of a route serving customers $i+1$ up to and including j.
q_i	$=$	positive demand of customer i.
m	$=$	the number of vehicles available.
C	$=$	the vehicle capacity
T	$=$	the maximum cost (or driving time) of each vehicle in a planning horizon.

$$a_{ij} = \begin{cases} 1, \text{ if } \sum\limits_{h=i+1}^{j} q_h \leq C \\ 0, \text{ otherwise} \end{cases}$$

Decision variables:

$$x_{ijk} = \begin{cases} 1, \text{ if vehicle } k \text{ travels on arc } (i,j) \\ 0, \text{ otherwise} \end{cases}$$

The mathematical representation (which we will refer to as SP2):

$$[\text{SP2}] \quad \min \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} c_{ij}x_{ijk} \tag{4.6}$$

$$\text{s.t.} \quad \sum_{j=0}^{n}\sum_{k=1}^{m} x_{0,jk} = 1 \tag{4.7}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{j,n,k} = 1 \tag{4.8}$$

$$\sum_{i=0}^{n}\sum_{k=1}^{m} x_{ijk} = \sum_{i=0}^{n}\sum_{k=1}^{m} x_{jik} \qquad \forall j = 1,\dots,n-1 \tag{4.9}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ijk} \leq T \qquad \forall k = 1,\dots,m \tag{4.10}$$

$$x_{ijk} \leq a_{ij} \qquad \forall i,j = 0,\dots,n \ \& \ k = 1,\dots,m \tag{4.11}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j = 0,\dots,n \ \& \ k = 1,\dots,m \tag{4.12}$$

In the objective, equation (4.6), the total cost of the arcs selected in the optimal solution is minimized. We are interested in a shortest path starting at node 0 and ending at node n. Equation (4.7) is ensuring one arc is leaving the starting point and equation (4.8) is defining one arc to enter the end point of the shortest path. Equation (4.9) are the connectivity constraints, and equation (4.10) is the bin-packing constraint representing the working days of the vehicles. Finally equation (4.11) and (4.12) are defining the binary decision variables that can only equal 1 when part of the acyclic graph satisfying the capacity constraints.

The solution of this model will give us a set of arcs that represent routes and simultaneously give feasible working days for the vehicles each of which serves one or more routes. In step 4, we try to improve this solution by solving an individual TSP for each of the selected routes. In the next section we will test this model on the benchmark problems from the literature.

4.1.1 Computational results

All computations in this section are coded in C++ and were run on a Dell Inspiron 1300 with an Intel(R) Pentium(R) M 1.70GHz processor. The integer model SP is solved with CPLEX 10.0. Before we can modify our model to adapt several practical side constraints we want to compare its performance with the results published in the literature.

We start by solving our approach for $m = 1$ for the problem instances introduced in Table 3.1 (these are 9 of the 52 VRPM instances). Note that in this case our problem is equal to the classical VRP with an unspecified number of vehicles such as solved by Rochat and Taillard (1995). To compare the performance of our approach with the previous published results, we look at the statistic defined in equation 4.13 which shows the deviation of our solution value from that of Rochat and Taillard (1995) defined by z^{RT}.

$$x = \frac{z^{(\cdot)} - z^{RT}}{z^{RT}} * 100\% \tag{4.13}$$

where $z^{(\cdot)}$ is referring to the solution value of the used procedure $(\cdot)$.

We present the results for these 9 instances in Table 4.1. In the first column we see the problem instances also used by Taillard et al. (1996) with the corresponding number of customers in the second column. The third column shows the values obtained by Rochat and

Table 4.1: The results for Route-first, cluster-second algorithm

	n	z^{RT}	z^{SP1}	x (Eq. 4.13)	CPU(sec)
CMT-1	50	525	560	6.75	0.7
CMT-2	75	835	880	5.36	0.9
CMT-3	100	826	928	12.33	1.4
CMT-4	150	1028	1099	6.86	2.4
CMT-5	199	1291	1397	8.17	4.1
CMT-11	120	1042	1071	2.77	1.8
CMT-12	100	820	887	8.23	1.3
F-11	71	242	249	2.91	0.9
F-12	134	1163	1224	5.25	2.0
Avg.				6.51	1.7

Taillard (1995) while the fourth column is filled with our objective function values. The fifth column is the statistic introduced in equation 4.13, and finally the last column presents the time spent to find this solution (in seconds) for each instance.

We notice that this heuristic is on average performing 6.5% poorer than the TS algorithm of Rochat and Taillard (1995). We notice that this value is too high to find good solutions for the VRPM. Increasing m will only add extra constraints to our model meaning that results can only get poorer. However, the CPU-time of our algorithm is negligible, which is a strong aspect compared to Rochat and Taillard (1995), because it is a metaheuristic and their best solutions are the results of five replications of the algorithm.

We notice the following: after we have selected routes based on their cost (c_{ij}), we improve these c_{ij} values in step 4 by solving an individual TSP. However, this is only done for those routes that are selected in the optimal solution. We now suggest that if we solve an individual TSP for each arc (route) on our shortest path graph, we can further improve our solution quality. In the worst case scenario this would mean that we have to solve $\frac{n*(n-1)}{2}$ TSP's, however because of vehicle capacity constraints the number of actual TSP that is solved is only a small fraction of this number. To avoid misunderstandings, we would like to emphasize that the shortest path graph remains unchanged and only routing "within" the arcs is considered in this step. The results of the SP1 with modified arcs, denoted by SP2, are given in Table 4.2 for the nine test instances with $m = 1$.

In Table 4.2 we observe that the algorithm now performs on average 4.8% poorer compared to the Tabu search algorithm of Rochat and Taillard (1995) compared to the 6.5% we found before. It is an important improvement, but CPU times are no longer negligible. This is due to the additional computational time needed to solve $\frac{n*(n-1)}{2}$ TSP's.

Table 4.2: Enhancement of route-first, cluster-second algorithm

	n	z^{RT}	z^{SP2}	x (Eq. 4.13)	CPU(sec)
CMT-1	50	525	560	6.75	196
CMT-2	75	835	880	5.36	230
CMT-3	100	826	860	4.10	451
CMT-4	150	1028	1086	5.60	891
CMT-5	199	1291	1375	6.47	1110
CMT-11	120	1042	1064	2.10	1123
CMT-12	100	820	858	4.69	620
F-11	71	242	249	2.91	665
F-12	134	1163	1219	4.82	1414
avg.				4.75	744

So far we have only been solving the classical VRP with an unspecified number of vehicles. When we increase the number of vehicles m and define a maximum allowable driving time T for each vehicle, we have a VRPM. In these tests we use the same values for m and T as Taillard et al. (1996) in order to compare our results. Remind that T could be either T_1 or T_2.

$$T_1 = [1.05z^{RT}/m] \tag{4.14}$$

$$T_2 = [1.1z^{RT}/m] \tag{4.15}$$

It is clear that this setting of T's is artificial and there is nothing to do with representing time windows. Furthermore it cannot be possible to find feasible solutions even with $T = T_2$. Therefore we introduce an even more relaxed limit on T, as

$$T_3 = [1.15z^{RT}/m] \tag{4.16}$$

We present our results of these tests in a detailed way in Appendix A. We use a "$**$" sign to illustrate the instances for which we have found a feasible solution given the maximum allowable distance of a single vehicle. In Table 4.3 we compare the results with respect to the other results in the literature in terms of the number of instances for which a feasible solution is found. Our computations are based on the 52 test problem instances.

Table 4.3: The number of feasible solutions found for the benchmark test problems

	TLG^1	BM^2	PS^3	OV^4	$SP2^5$
T_1	38	37	24	**46**	11
T_2	48	**52**	47	**52**	42
T_3					50

^{1} results from Taillard et al. (1996)
^{2} results from Brandao and Mercer (1998)
^{3} results from Petch and Salhi (2004)
^{4} results from Olivera and Viera (2007)
^{5} results from our approach SP2 in section 4.1

We notice that our relatively simple algorithm cannot compete with the more sophisticated metaheuristics proposed in the literature when $T = T_1$, but it is competitive when $T = T_2$. We reckon that the drawback of our method is the restriction that the customers are ordered when solving the giant TSP. For this reason we drop this restriction and propose a set-covering approach in the next section to find a true representation of the problem.

4.2 Set-covering based mathematical model

In this section we discuss a set-covering model modified to solve the VRPM. We already discussed the set-partitioning model in Section 2.1.3, but now we will use the set-covering model. The parameters and decision variables of the set-covering model are given below.

Parameters:

c_j	=	The cost (in terms of distance or time) of route j.
a_{ij}	=	Equals 1 if customer i is included in route j for each j in the route set J.
m	=	the number of vehicles available.

Decision variables:

$$x_j = \begin{cases} 1, & \text{if route } j \text{ is selected in the solution} \\ 0, & \text{otherwise} \end{cases}$$

The mathematical representation (which we will refer to as SC):

$$\begin{aligned} \text{[SC]} \quad \min \quad & z = \sum_{j \in J} c_j x_j && && (4.17) \\ \text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j \geq 1 && i = 1, \ldots, n && (4.18) \\ & \sum_{j \in J} x_j \leq m && && (4.19) \\ & x_j \in \{0, 1\} && \forall j && (4.20) \end{aligned}$$

The objective function, equation (4.17), is minimizing the cost of the selected routes. Equation (4.18) ensures that each customer is visited at least once, and equation (4.19) puts a restriction on the number of vehicles. Finally equation (4.20) is defining the decision variables. Note that route j could be any feasible route.

Converting this model such that it can solve the VRPM involves the following changes, first we introduce a new index $k \in \{1, \ldots, m\}$ that represent the vehicles. Then we redefine the decision variable and give the modified mathematical representation.

Decision variables:

$$x_j^k = \begin{cases} 1, & \text{if route } j \text{ is served by vehicle } k \text{ in the optimal solution} \\ 0, & \text{otherwise} \end{cases}$$

The mathematical representation (which we will refer to as SCT):

$$\begin{aligned} \text{[SCT]} \quad \min \quad & z = \sum_{j \in J} \sum_{k=1}^{m} c_j x_j^k && && (4.21) \\ \text{s.t.} \quad & \sum_{j \in J} \sum_{k=1}^{m} a_{ij} x_j^k \geq 1 && i = 1, \ldots, n && (4.22) \\ & \sum_{j \in J} c_j x_j^k \leq T && \forall k && (4.23) \\ & x_j^k \in \{0, 1\} && \forall j, k && (4.24) \end{aligned}$$

All equations are same as those of model SC except (4.23). It is changed and this equation is now ensuring that for each vehicle the total duration of the routes that is served is less than the prescribed limit T. An advantage of this model is that we are no longer limited to a fixed order of routes in which the customers have to be visited in a prescribed order. We can now incorporate any route from any different source into set J and let the model pick the best mix of these routes.

This approach gives us the optimal solution to the model if J contains all possible feasible routes. Given the route set J (which could be only a small subset of the set of all possible feasible routes, because the entire set is so large) we also incorporate an improvement procedure to find a better solution.

4.2.1 Improvement on SCT

Let us assume that (4.21)-(4.24) results in an infeasible solution in terms of T. Most likely we have a solution where a few vehicles have a total driving time of more than T, while at the same time other vehicles have a total driving time less than T. In our improvement procedure we select a customer from a vehicle with $T_k > T$ and try to insert it in one of the other routes such that the new driving time remains below T and the capacity restriction is satisfied. For a stepwise summary of our solution approach, see Figure 4.3. In the next section we will present our computational results related to different settings of set J for SCT and improvement procedure in Figure 4.3.

4.2.2 Computational results

Again all computations in this section are coded in C++ and were run on a Dell Inspiron 1300 with an Intel(R) Pentium(R) M 1.70GHz processor. We used CPLEX 10.0 to solve the model SCT. Before we can modify our model to adopt several practical side constraints we want to compare its performance in its pure form with the results published in the literature.

We start by defining the subset $\hat{J}$. Remember from the integer-approach in Section 4.1 that we constructed an acyclic graph where each arc represented a route. This graph was created solving a giant TSP. Now we create a second graph using the sweep-method. We thus construct our route-set $\hat{J}$ as the set containing all routes from these two graphs combined. This means that we combine routes from both methods and at least it guarantees us

```
step 1   Derive any subset $\hat{J}$ from the entire route set $J$.
step 2   Solve (4.22)-(4.25) given this $\hat{J}$.
step 3   If a feasible solution is found, stop.
         Otherwise go to step 4.
step 4   Select the vehicle with the longest total driving time (>T).
step 5   List the customers served by this vehicle in order to seek savings
         when any one is removed from this vehicle.
step 6   Take item from the list and insert it in one of the other routes.
         Check feasibility and update driving times.
step 7   If feasible improvement is found, update solution and
         return to step 3.
         While list is not empty, return to step 6.
step 8   Stop.
```

Figure 4.3: Improvement procedure on the set-covering approach

no worse solutions.

Following the same methodology as in Section 4.1.2, we first solve our model for the nine test problem instances with $m = 1$ and we present the results in Table 4.4. z^{SCT} in this table refers to the optimal solution value of the model SCT given the subset of routes.

We notice that this approach is first of all performing only 2.9% poorer than the TS-algorithm of Rochat and Taillard (1995) for the VRP with an unspecified number of vehicles. The CPU-times are negligible as it only takes fractions of a second to solve the model to optimality using CPLEX version 10.0. Regarding the VRPM instances we expect that since we have a larger route-set the chance of finding a feasible solution is now higher compared with our MIP-model from Section 4.1.

In appendix B we have presented all results in detail, covering all 52 test problem instances, in the same manner as we did in appendix A. The symbol $**$ is illustrating for which instances we have found a feasible solution given the maximum allowable distance of a single vehicle. Note that if we have found a feasible solution for the T_1 instance, this automatically

Table 4.4: Results for the set-covering approach

	n	z^{RT}	z^{SCT}	x (Eq. 4.13)
CMT-1	50	525	525	0
CMT-2	75	835	879	5.27
CMT-3	100	826	827	0.12
CMT-4	150	1028	1086	5.54
CMT-5	199	1291	1361	5.34
CMT-11	120	1042	1064	2.11
CMT-12	100	820	827	0.85
F-11	71	242	249	2.91
F-12	134	1163	1211	4.13
Avg.				2.92

means we will find a feasible solution for T_2 and T_3. This may not be necessarily the same solution, there might be an improvement possible in terms of the total cost.

Although we see an improvement in the number of instances solved, the results are still not as good as the published results in the literature. However, since CPU-times of the set-covering model are so low, we come up with the following idea. We decide to generate more high quality routes with a modified version of the Clark and Wright saving heuristic introduced in Chapter 2 to enlarge the number of routes in set $\hat{J}$. Recall that the saving heuristic is initially serving all customers on an individual route and then step by step merges the routes in order to obtain a VRP solution. To determine which routes are to be merged we define:

$$s_{ij} = c_{0i} + c_{oj} - c_{ij} \tag{4.25}$$

We order these saving in a list in non-increasing order and start at the top of the list by looking if a merge is feasible. If so, we merge the routes and continue with the next element from the list. If not, we do not merge the routes and continue with the next element from the list. We continue till the list of savings is empty. Remind that a merge is only feasible when the combined capacity of the "old" routes is not exceeding C. Each time a feasible merge is found we add this "new" route into our route-set $\hat{J}$.

Note that this saving approach is deterministic in selection and gives us the same routes each time we run it. In order to obtain more "good" routes with the saving heuristic, we make a small modification and propose a stochastic version of the same saving heuristic. Here we select an item from the saving list with a probability p. We define the probability $P(s_{ij})$ as follows.

$$P(s_{ij}) > P(s_{tv}) \qquad \text{if } s_{ij} \geq s_{tv} \tag{4.26}$$

We have noticed that as the size of the route set is growing, the increase in CPU-time is quite significant. So we have to find a balance between the desired improvement of adding more routes into $\hat{J}$ against the increase in CPU-time. Preliminary experiments show us that when we apply the stochastic saving heuristic 20 times plus the deterministic saving heuristic and the routes given by the route-first cluster-second approach of Section 4.1, this will keep CPU-times reasonable. Note that we have excluded the routes from the sweep method here, because we noticed these routes were barely chosen and only affecting the CPU-time.

Table 4.5: Route generation results

	n	z^{RT}	routes	time(sec)	z^{SCT}	**x** (Eq. 4.13)
CMT-01	50	525	1861	612	521	**-0.76**
CMT-02	75	835	2462	771	864	**3.47**
CMT-03	100	826	3951	1384	824	**-0.24**
CMT-04	150	1029	6744	2501	1072	**4.18**
CMT-05	199	1292	8667	3979	1342	**3.87**
CMT-11	120	1042	6258	2626	1041	**-0.10**
CMT-12	100	820	4474	1781	827	**0.85**
F-11	71	242	3728	1871	249	**2.89**
F-12	134	1163	8525	4090	1139	**-2.06**
average			5186	2179		[-2.06,4.18]

In Table 4.5 we show the results of the new route generation phase for only the nine test instances. In the first column we find the problem instances, in the second column the accompanying number of customers. The third column gives us the results of the VRP for an unspecified number of vehicles by Rochat and Taillard (1995). The fourth column shows the size (cardinality) of the route set $\hat{J}$ and the fifth column shows the presolving time that is needed to calculate the c_j values for $j \in \hat{J}$. The next column shows the objective function value found with the set-covering model and in the last column we look at the statistic x as defined in equation (4.13). It is clear that this time we do even better than the benchmark results for some cases. The worst deviation is about 4%, which is observed for only two instances. These results are quite compatible with the results in the literature even if we only consider the quality of the solution and the solution effort needed as criteria.

In Table 4.6 we compare, for all 52 test problem instances, the results with respect to the other results in the literature in terms of the number of instances for which a feasible solution is found.

Table 4.6: The number of feasible solutions found for the benchmark test problems

	TLG^1	BM^2	PS^3	OV^4	$SP2^5$	SCT^6
T_1	38	37	24	**46**	11	34
T_2	48	**52**	47	**52**	42	45
T_3					50	50

^{1} results from Taillard et al. (1996)
^{2} results from Brandao and Mercer (1998)
^{3} results from Petch and Salhi (2004)
^{4} results from Olivera and Viera (2007)
^{5} results from our route-first, cluster-second approach SP2 in section 4.1
^{6} results from our set-covering approach SCT in section 4.2

We notice that this approach is significantly performing better than without the routes generated with the saving heuristic. To see if this enhancement also has a positive effect on the number of VRPM instances with a feasible solution we test this data on the VRPM instances. We presented our results in appendix C. To compare the results of our infeasible instances with the other authors we have computed the longest tour ratios (LTR) as defined in equation 3.3. We will now present two tables.

In Table 4.7 we present the LTR of the infeasible solution given by the set covering model (4.21)-(4.24). In this table T_1 is determined by equation (4.14) and T_2 by equation (4.15). When the LTR is smaller than 1 (< 1) this means we have found a feasible solution in terms of T_1 or T_2. In Table 4.8 we present again the same LTR, but this time after we have improved the infeasible solutions.

We notice that after we solve the set covering model, we obtain 25 instances (for T_1 and T_2 only) with an infeasible solution in terms of total driving time (or cost). After our improvement procedure, we have improved 20 out of these 25 instances. We see that this improvement step has a significant good effect on the instances to make the infeasible tours shorter.

4.3 Remarks

In this chapter we have derived two models to solve the VRPM. The first model based on the route-first cluster-second principle showed to be really good in terms of computation times. For each instance we found solutions within just a few seconds. On the other hand, we saw that assuming the customer must be visited in a prescribed order has a negative effect on the solution quality. So we can conclude that this approach is useful for large instances that need to be solved very quickly. However when solution quality becomes more important this approach is not very useful.

The second model is derived based on the set covering model. Here we drop the assumption that the customers have to be visited in the order as they appear on the giant tour and we take any feasible route as a candidate. We derive several good routes using three different procedures. The first being the same route-first cluster-second method as we use for the integer model. Second we derive a set of good petal routes using the idea of Ryan et al. (1993), and finally we derive a good set of routes using a stochastic version of the parallel saving heuristic of Clarke and Wright (1964). Results show very reasonable results in acceptable computation time.

We notice looking at the costumer distributions in Appendix E, that our results are better for problems where the customers appear in clusters. The set covering approach seems to fail for larger instances, especially for the randomly distributed instances, compared with other published results in terms of LTR, however after incorporating an improvement procedure this failure is fixed and comparable results are found. However the comparisons made are quite poor. For instance the total tour length is so far not considered important in the literature. The main focus so far was to find feasible solutions (minimizing the longest tour). We think that in future research on this topic the total tour length will become of more importance.

Table 4.7: Results for infeasible instances before improvement

Problem:		m	T_1	T_2	SCT	TLG[1]	BM[2]	PS[3]	OV[4]
CMT-01	50	2	276		<**1**	<**1**	<**1**	1.004	<**1**
		3	184		1.027	1.115	1.041	1.026	**1.024**
		3		193	<**1**	1.05	<**1**	1.000	<**1**
		4	138		1.051	**1.027**	**1.027**	1.085	**1.027**
		4		144	<**1**	<**1**	<**1**	1.031	<**1**
CMT-02	75	6	146		1.041	1.032	1.031	1.019	<**1**
		7	125		1.056	1.073	1.088	1.064	**1.009**
		7		131	<**1**	1.023	<**1**	1.009	<**1**
CMT-03	100	5	173		<**1**	1.062	<**1**	1.052	<**1**
		5		182	<**1**	1.010	<**1**	<**1**	<**1**
		6	145		1.048	1.032	1.003	1.001	<**1**
		6		151	<**1**	1.012	<**1**	<**1**	<**1**
CMT-04	150	4	270		<**1**	<**1**	<**1**	1.005	<**1**
		5	216		1.051	<**1**	<**1**	<**1**	<**1**
		6	180		1.050	<**1**	<**1**	1.01	<**1**
		7	154		1.110	1.033	1.071	1.072	**1.002**
				162	1.056	1.010	<**1**	1.005	<**1**
		8	135		1.178	1.075	1.031	1.058	<**1**
				141	1.128	1.029	<**1**	<**1**	<**1**
CMT-05	199	5	271		1.052	<**1**	<**1**	1.007	<**1**
		6	226		1.049	<**1**	<**1**	1.000	<**1**
		7	194		1.046	<**1**	<**1**	1.008	<**1**
		8	170		1.094	<**1**	<**1**	1.015	<**1**
		8		178	1.045	<**1**	<**1**	<**1**	<**1**
		9	151		1.059	<**1**	1.056	1.024	<**1**
		9		158	1.013	<**1**	<**1**	<**1**	<**1**
		10	136		1.088	1.024	1.051	1.064	<**1**
		10		142	1.042	<**1**	<**1**	1.018	<**1**
CMT-11	120	2	547		<**1**	<**1**	<**1**	1.006	<**1**
		3	365		<**1**	<**1**	<**1**	1.006	<**1**
		4	274		<**1**	1.020	1.011	1.052	<**1**
		4		287	<**1**	<**1**	<**1**	1.002	<**1**
		5	219		<**1**	<**1**	<**1**	1.037	<**1**
CMT-12	100	4	215		<**1**	<**1**	1.012	<**1**	<**1**
		5	172		1.069	1.050	1.036	1.000	<**1**
		5		180	1.022	1.003	<**1**	<**1**	<**1**
		6	144		1.132	1.064	1.072	1.029	**1.014**
		6		150	1.087	1.014	<**1**	<**1**	<**1**
F-11	71	2	127		<**1**	1.031	1.011	1.020	<**1**
		3	85		1.047	1.075	**1.011**	1.020	1.020
		3		89	<**1**	1.027	<**1**	<**1**	<**1**

[1] results from Taillard et al. (1996)
[2] results from Brandao and Mercer (1998)
[3] results from Petch and Salhi (2004)
[4] results from Olivera and Viera (2007)

Table 4.8: Results for infeasible instances after improvement

Problem:		m	T_1	T_2	SCT	TLG^1	BM^2	PS^3	OV^4
CMT-01	50	2	276		**<1**	**<1**	**<1**	1.004	**<1**
		3	184		1.027	1.115	1.041	1.026	**1.024**
		3		193	**<1**	1.05	**<1**	1.000	**<1**
		4	138		1.029	**1.027**	**1.027**	1.085	**1.027**
		4		144	**<1**	**<1**	**<1**	1.031	**<1**
CMT-02	75	6	146		1.034	1.032	1.031	1.019	**<1**
		7	125		1.056	1.073	1.088	1.064	**1.009**
		7		131	**<1**	1.023	**<1**	1.009	**<1**
CMT-03	100	5	173		**<1**	1.062	**<1**	1.052	**<1**
		5		182	**<1**	1.010	**<1**	**<1**	**<1**
		6	145		1.020	1.032	1.003	1.001	**<1**
		6		151	**<1**	1.012	**<1**	**<1**	**<1**
CMT-04	150	4	270		**<1**	**<1**	**<1**	1.005	**<1**
		5	216		1.046	**<1**	**<1**	**<1**	**<1**
		6	180		1.011	**<1**	**<1**	1.01	**<1**
		7	154		1.064	1.033	1.071	1.072	**1.002**
		7		162	1.012	1.010	**<1**	1.005	**<1**
		8	135		1.089	1.075	1.031	1.058	**<1**
		8		141	1.043	1.029	**<1**	**<1**	**<1**
CMT-05	199	5	271		1.007	**<1**	**<1**	1.007	**<1**
		6	226		1.031	**<1**	**<1**	1.000	**<1**
		7	194		1.046	**<1**	**<1**	1.008	**<1**
		8	170		1.088	**<1**	**<1**	1.015	**<1**
		8		178	1.039	**<1**	**<1**	**<1**	**<1**
		9	151		1.059	**<1**	1.056	1.024	**<1**
		9		158	1.013	**<1**	**<1**	**<1**	**<1**
		10	136		1.074	1.024	1.051	1.064	**<1**
		10		142	1.028	**<1**	**<1**	1.018	**<1**
CMT-11	120	2	547		**<1**	**<1**	**<1**	1.006	**<1**
		3	365		**<1**	**<1**	**<1**	1.006	**<1**
		4	274		**<1**	1.020	1.011	1.052	**<1**
		4		287	**<1**	**<1**	**<1**	1.002	**<1**
		5	219		**<1**	**<1**	**<1**	1.037	**<1**
CMT-12	100	4	215		**<1**	**<1**	1.012	**<1**	**<1**
		5	172		1.064	1.050	1.036	1.000	**<1**
		5		180	1.017	1.003	**<1**	**<1**	**<1**
		6	144		1.090	1.064	1.072	1.029	**1.014**
		6		150	1.047	1.014	**<1**	**<1**	**<1**
F-11	71	2	127		**<1**	1.031	1.011	1.020	**<1**
		3	85		**1.011**	1.075	**1.011**	1.020	1.020
		3		89	**<1**	1.027	**<1**	**<1**	**<1**

[1] results from Taillard et al. (1996)

[2] results from Brandao and Mercer (1998)

[3] results from Petch and Salhi (2004)

[4] results from Olivera and Viera (2007)

CHAPTER 5

CASE STUDY

In this chapter we will discuss a real life example of an application of a vehicle routing problem with multiple use of vehicles. We will extend the set-covering model discussed in the previous chapter in order to develop a useful tool for the daily logistic decisions that have to be made in this case study.

Santa Fe Indonesia is a company that offers relocation services to individuals as well as companies. Santa Fe operates in South-East Asia with 26 offices located in 12 different countries, see Figure 5.1. Each office is independently operating a vehicle fleet for the operations in that area. In this study we put our focus on the logistic activities of the office in Jakarta, Indonesia, but it is plausible to assume that this study with some minor modifications can be used in the other offices as well.

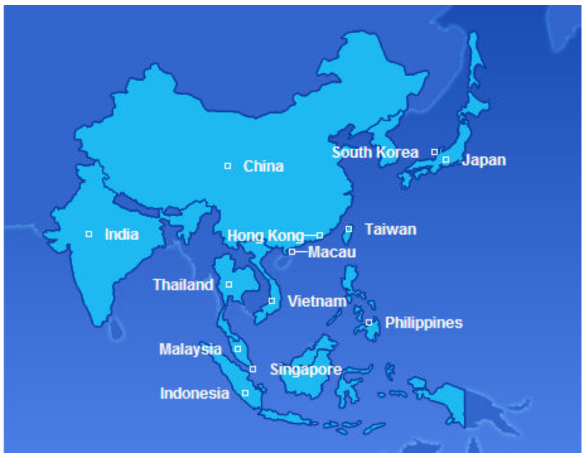

Figure 5.1: Countries with offices of Santa Fe

5.1 Problem introduction

Santa Fe is operating a warehouse in South-Jakarta from where all logistic operations are performed. Basically we can distinguish three different types of transportation.

1. Inbound shipments
2. Outbound shipments
3. Local moves

Inbound shipments arrive at the (air)port of Jakarta and once released from customs they will be brought to the warehouse. The shipment is then prepared to be transported to the final destination in Jakarta (or beyond). Most shipments arrive in containers at the port of Jakarta, located North-East of the city center. After custom formalities the shipment is released and transported to the warehouse. Goods are transfered into liftvans and delivered to final destination upon request of the customer.

Outbound shipments are packed into liftvans at the site of origin, before transported and collected at the warehouse. Here the shipment is prepared for air or sea freight transportation. At this point deadlines are really important since a shipment is usually booked onto a vessel and shipments need to arrive at the port before a deadline. Planning for this reason is essential and the use of time windows in our model is required.

The last type of transportation is local moves. Here both the origin and destination are located within the range of the Jakarta-office. Depending on the job-requirements, the shipment is transported directly to the destination or via the warehouse.

Most shipments involve household goods including consumer electronics and other fragile items. This requires much labor at origin and destination site. For valuable items, special custom-made wooden crates are made for transportation. For this reason one could argue for a joint crew and vehicle assignment approach since a job can not be transported before it is carefully packed. In our model, however, we do not take the crew assignment into consideration. We can justify this, because additional crew members can be hired from a labor office at all times at almost nil cost.

In Figure 5.2 we show some geographical information about Jakarta. The city can be partitioned roughly in five districts. North-Jakarta (yellow), East-Jakarta (pink), South-Jakarta (orange), West-Jakarta (green) and Central-Jakarta (purple). Also we have displayed the location of:

- The (main) airport, Soekarno-Hatta international airport.
- The (main) seaport, Tanjung Priok sea port.
- The Santa Fe warehouse.

Furthermore we should mention that the warehouse is located along the major toll-road circling around the city and offering a fast link with both the air and sea ports of Jakarta. Other roads are highly congested and driving times are difficult to estimate with a Graphical Information System (GIS). For this reason we are using time estimates from experts.

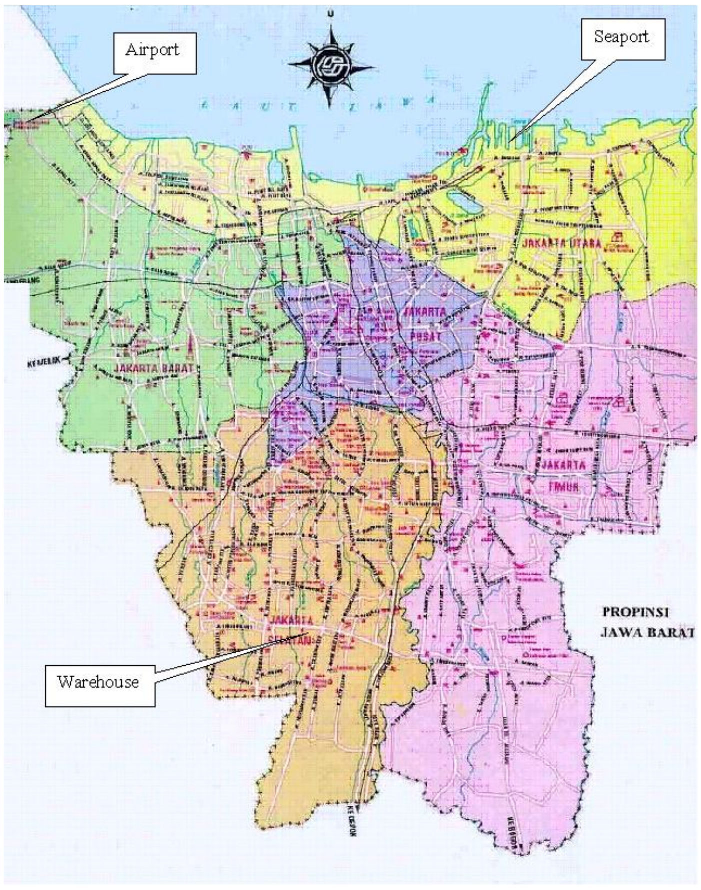

Figure 5.2: Districts of Jakarta

5.2 Problem definition

Let us briefly recall the set-covering model for the VRPM discussed in Section 4.2. We defined a route set J containing feasible routes. Each route $j \in J$ is associated with a positive route cost (time). We assume that we have a set of m vehicles that is operational for T hours each day. We assume T is equal for each vehicle. Furthermore we define the following parameters and decision variables given below.

Parameters:

c_j	=	The cost (in terms of distance or time) of route j.
a_{ij}	=	Equals 1 if customer i is included in route j, $j \in J$.
m	=	The number of vehicles available.
T	=	The maximum operational time for each vehicle.

Decision variables:

$$x_j^k = \begin{cases} 1, & \text{if route } j \text{ is served by vehicle } k \text{ in the optimal solution} \\ 0, & \text{otherwise} \end{cases}$$

α = The maximum allowable overtime, any positive numerical value.

The mathematical representation:

$$\begin{aligned}
\min \quad & z = \alpha && && (5.1)\\
\text{s.t.} \quad & \sum_{j\in J}\sum_{k=1}^{m} a_{ij}x_j^k \geq 1 && i = 1,\dots,n && (5.2)\\
& \sum_{j\in J} c_j x_j^k \leq T + \alpha && \forall k && (5.3)\\
& x_j^k \in \{0,1\} && \forall j,k && (5.4)\\
& \alpha \geq 0 && && (5.5)
\end{aligned}$$

Note that we have changed the objective function here. In addition an upperbound on α can be added to the model, in this case feasibility is no longer guaranteed. Equation (5.1) is now minimizing the maximum overtime. Equation (5.2) ensures that all customers are visited and equation (5.3) is the bin-packing constraint assigning the routes to the vehicles. Equation (5.4) is defining the binary decision variables, and equation (5.5) defines overtime as a positive variable.

We could easily modify the model to minimize the total overtime by introducing a decision variable α_k for each vehicle k and rewrite the objective function value as:

$$\min z = \sum_k \alpha_k \tag{5.6}$$

In this section we will step by step modify the set-covering model for the VRPM discussed in Section 5.1. First we incorporate a heterogeneous vehicle fleet, secondly we modify the model such that it can cope with backhaul customers. Finally we incorporate individual time windows for the customers.

5.2.1 Heterogeneous vehicle fleet

As we have seen in Section 2.4 this extension implies that the vehicle fleet consist of vehicles with different characteristics. Based on the vehicle fleet of Santa Fe we say that there are 3 different types of vehicles, vehicles having either small, medium or large capacity, but different compositions are possible as well. We define:

$$K = \{1, \ldots, k^1, k^1 + 1, \ldots, k^2, k^2 + 1, \ldots, k^3\}$$

Where k^1 is the number of vehicles with small capacity, $k^2 - k^1$ the number of vehicles with medium capacity and $k^3 - k^2$ the number of vehicles with large capacity. Note that earlier we defined the route set J given that all vehicles have equal capacity. Since this is no longer valid, we must partition the route set. We define J_1 as the route set given that all vehicles have the capacity of a small vehicle, J_2 as the route set given that all vehicles have the capacity of a medium vehicle and J_3 as the route set given that all vehicles have the capacity of a large vehicle. Now we notice that $J_1 \subseteq J_2 \subseteq J_3$, this is true since each route from J_1 that can be served by vehicles having a small capacity, can be served by a vehicle with a medium or large capacity as well. Because of this property we define the set J as

$$J = \{1, \ldots, r^1, r^1 + 1, \ldots, r^2, r^2 + 1, \ldots, r^3\},$$

where

- $J_1 = \{1, \ldots, r^1\}$
- $J_2 = J_1 \cup \{r^1 + 1, \ldots, r^2\}$
- $J_3 = J_1 \cup J_2 \cup \{r^2 + 1, \ldots, r^3\} = J$

Note that this is a capacity based classification. There could be another classification in terms of accessibility in narrow streets. In this case routes from the subset of large vehicles could also be served by small vehicles, but not the other way around, i.e. $J_1 \supseteq J_2 \supseteq J_3$. Here the smallest vehicles can go anywhere in the city, but the larger vehicles can go to only a small subset of streets. Anyway for the time being we stick to the first (capacity based) classification. Given these definitions we show the modified mathematical representation.

$$
\begin{aligned}
\min \quad & z = \alpha && && (5.7)\\
\text{s.t.} \quad & \sum_{j \in J} \sum_{k \in K} a_{ij} x_j^k \geq 1 && \forall i && (5.8)\\
& \sum_{j \in J} c_j x_j^k \leq T + \alpha && \forall k \in K && (5.9)\\
& x_j^k = 0 && j \in J \backslash J_1, k = \{1, \dots, k^1\} && (5.10)\\
& x_j^k = 0 && j \in J \backslash J_2, k = \{k^1 + 1, \dots, k^2\} && (5.11)\\
& x_j^k \in \{0, 1\} && \forall k \in K, j \in J && (5.12)\\
& \alpha \geq 0 && && (5.13)
\end{aligned}
$$

The objective function, equation (5.7) is minimizing the maximum overtime. Equation (5.8) is guaranteeing that all customers are visited, we do not allow unvisited customers at a penalty cost. Equation (5.9) is the bin-packing constraint that assigns routes to the vehicles such that the maximum (allowable) total driving time is not exceeded. Equation (5.10) is valid for the small vehicles ($k = 1, 2, 3$) and states that they cannot serve routes that need the capacity of a medium or large truck. Equation (5.11) is valid for the medium vehicles ($k = 4, 5$) and states that they cannot serve routes that need the capacity of a large vehicle. Finally, equation (5.12) is defining the decision variables, and equation (5.13) defines overtime as a positive variable.

5.2.2 Backhaul customers

This extension means that we do not only make deliveries to customers, but that we also have customers that require a pick-up. We assume that all pick-up (backhaul) customers have goods that are shipped to the warehouse. For this reason we define a set $N_1 = \{1, \dots, n_1\}$ with customers having positive demand and a second set $N_2 = \{1, \dots, n_2\}$ having negative demands (pick-up) for backhaul customers. In this case study we assume that on each route first the customers of N_1 are visited before visiting the customers of N_2.

The model (5.7)-(5.13) is still valid, but only the definition of the route set J needs to be modified. In Figure 5.3 we will give a stepwise overview of how J is constructed.

step 1a	for each $i \in N_1$ we define an individual route j
	we add this route to J_p^d depending on q_i $(p \in \{1,2,3\})$.
step 1b	for each $i \in N_2$ we define an individual route j
	we add this route to J_p^b depending on q_i $(p \in \{1,2,3\})$.
step 2a	for each $j_1, j_2 \in J^d$ we merge the routes
	where $J^d = J_1^d \cup J_2^d \cup J_2^d$ referring to routes with linehaul customers only.
	if merged route is feasible add to J_p^d, $(p \in \{1,2,3\})$.
step 2b	for each $j_1, j_2 \in J^b$ we merge the routes
	where $J^b = J_1^b \cup J_2^b \cup J_2^b$ referring to routes with backhaul customers only.
	if merged route is feasible add to J_p^b, $(p \in \{1,2,3\})$.
step 3	if no more feasible merges exist we define $J_p = J_p^d \cup J_p^b$
step 4	for each $j_1 \in J^d, j_2 \in J^b$ we merge the routes
	if merged route is feasible add to J_p, $(p \in \{1,2,3\})$.

Figure 5.3: Generation of route set J

In this procedure we define all routes consisting of just delivery customers in the set J^d. Similarly we define the set J^b as the set of routes with only backhaul customers. The set of all feasible routes J is the union of these sets plus the set of routes with both delivery and backhaul customers (step 4).

Note that we determine all feasible routes. We can do this in this case study, because of the fact that the number of customers served on a single route does not exceed 2 or 3 customers in practice. For an application with many customers on each route we could generate a subset of J using techniques discussed in Chapter 4 such as the route-first cluster-second approach or saving heuristic.

5.2.3 Time Windows

The most important modification are the individual time windows for each customer. In our case study we are dealing with preferences of customers, but more importantly deadlines at the air or sea port. We are given for each job (customer) a time window (a_i, b_i) in which delivery or pick-up is requested. When we add this property to the route set generation steps, we define e_j and l_j as the earliest time and latest time a route can start at the depot. Obviously a route is only feasible if $e_j \leq l_j$.

A difficult step in dealing with the time windows for the VRPM is that we need to "remember" the time that a second or third route by a specific vehicle can start. For this reason we define a new index $h = \{1, \ldots, H\}$ that represents the order in which the routes are served by a certain vehicle on its tour. So vehicle k always serves $h = 1$ before it can serve $h = 2$ etc. Now we redefine our decision variables accordingly and introduce a new decision variable t_k^h.

Decision variables:

$$x_{jk}^h = \begin{cases} 1, & \text{if vehicle } k \text{ serves route } j \text{ in the } h^{\text{th}} \text{ order within its tour} \\ 0, & \text{otherwise} \end{cases}$$

$$t_k^h = \begin{cases} 1, & \text{starting time of the } h^{\text{th}} \text{ route served by vehicle } k \\ 0, & \text{otherwise} \end{cases}$$

α = The maximum allowable overtime, any positive numerical value.

The mathematical representation:

$$\begin{aligned}
\min \quad & z = \alpha & & (5.14)\\
\text{s.t.} \quad & \sum_{j \in J} \sum_{k \in K} \sum_{h \in H} a_{ij} x_{jk}^h \geq 1 & \forall i \in N_1 \cup N_2 \quad & (5.15)\\
& \sum_{j \in J} x_{jk}^h \leq 1 & \forall h \in H, k \in K \quad & (5.16)\\
& x_{jk}^h > x_{jk}^{h+1} & \forall h \in H, k \in K, j \in J \quad & (5.17)\\
& t_k^h + c_j - M(1 - x_{jk}^h) \leq t_k^{h+1} & \forall h \in H, k \in K, j \in J \quad & (5.18)\\
& e_j x_{jk}^h \leq t_k^h \leq l_j x_{jk}^h & \forall h \in H, k \in K, j \in J \quad & (5.19)\\
& t_k^h + c_j - M(1 - x_{jk}^h) \leq T + \alpha & \forall h \in H, k \in K, j \in J \quad & (5.20)\\
& x_{jk}^h = 0 & j \in J \backslash J_1, k = \{1, \ldots, k^1\} \quad & (5.21)\\
& x_{jk}^h = 0 & j \in J \backslash J_2, k = \{k^1 + 1, \ldots, k^2\} \quad & (5.22)
\end{aligned}$$

$$t_k^h \geq 0 \qquad \forall h \in H, k \in K \tag{5.23}$$

$$x_{jk}^h \in \{0,1\} \qquad \forall h \in H, k \in K, j \in J \tag{5.24}$$

$$\alpha \geq 0 \tag{5.25}$$

In this formulation M is a large positive number. The objective function, equation (5.14) is minimizing the maximum necessary overtime. Equation (5.15) is assigning each customer from N_1 as well as N_2 to at least one vehicle. Equations (5.16) and (5.17) are necessary technical assignment restriction constraints. Equation (5.18) defines that route $h+1$ cannot start before route h has finished, but only if these routes are served by the same vehicle. Equation (5.19) defines for each route the earliest and latest starting times, where equation (5.20) is defining that each route should be finished before $T + \alpha$. Equation (5.21) ensures that small vehicles cannot serve on routes requiring a larger capacity and equation (5.22) does the same thing for medium vehicles. Finally equation (5.23) defines the decision variable for the starting time of all route h for vehicle k to be nonnegative and equation (5.24) defines the binary decision variables, and equation (5.25) defines overtime as a positive variable.

5.3 Decision support system

In this section we give some specific details about the case study and suggest a decision support system for Santa Fe. We suggest a decision support system because the decision maker has reported to us that the current vehicle-assignment process is out-of-date and time-consuming.

At Santa Fe each (outbound) job is guided by one of the sales employees of Santa Fe. This person is responsible for recruiting new customers and planning the entire move. Each move requires a professional estimation of the total load of the move, pointing out the items that require special care, a division between items that are to be shipped with air or sea transport, etc. Secondly a time schedule is proposed to the customer. The number of days that is needed for packing, a vessel and/or flight is selected and a partner relocation company at destination is selected.

Each inbound job requires less preparation, since all estimations are done at site of origin. The sales person from the site of origin is simply informing Santa Fe Indonesia about the date and time the shipment is arriving in Jakarta and all other job specifications. Santa

Fe will collect the shipment as soon as it arrives and stores the shipment at the warehouse in anticipation of the customer to select a delivery day and time.

Most shipments in the international relocation industry consist of two parts. The main part consists of high volume furniture, consumer electronics, etc. This part is usually packed into lift vans (wooden box of 3.5m to 5.2m long, 1.9m wide and 2m high) and shipped by sea. The possible second part consists usually of low-volume items of high importance such as clothes, papers, etc. This part is usually preferred to be sent by air because of duration advantages.

Santa Fe is currently using a deadline of 4pm for all employees to announce a "vehicle job" to the logistic manager regarding jobs for the next working day. Meaning that after 4pm the logistic manager is manually assigning customers to vehicles. This process takes up to 30 minutes and frequently needs to be repeated when last-minute announcements are done. Moreover when after the arrival of a last-minute job the vehicle fleet and/or packing-crew is not sufficient for the next day, this implies the fact that a rental-vehicle or additional crew-members have to be hired. We conclude that the planning-phase can cause vexation or be annoying for the logistic manager.

One difficulty in this case study is the data collection. Not the number of customers, but the fact that the customer sites are always changing makes it very difficult and time-consuming to construct distance or time matrices. In other practical application we have seen successful implementations of a Graphical Information System (GIS) (see for instance Pamuk et al. (2004)). Such a system can be used to extract coordinate and distance information of the customer set. Difficulties as different road-properties in different areas of the city can be manipulated in such a system.

Because of these difficulties we have tested the model with equations (5.11)-(5.21), for a selected of typical number of days. A typical working day may include a total number of 6 to 13 jobs divided over N_1 and N_2 that have to served with 6 vehicles. Two small capacitated vehicles (max. load: 1 lift van), three medium capacitated vehicles (max. load: 3 lift vans) and one large capacitated vehicle (max. load: 4 lift vans). We collected the following data for the four selected days, see Table 5.1 up to and including Table 5.4.

Table 5.1: Details for working day 1 of Santa Fe

Customer	Pick-up/ Delivery	load (lift van)	Time Window	Driving Time (min)	Service time (min)
1	Delivery	2	10am - 4pm	120	60
2	Delivery	2	8am - 7pm	120	120
3	Delivery	1	8am - 11am	45	60
4	Delivery	1	12am - 4pm	30	150
5	Delivery	3	8am - 6pm	60	60
6	Pick-up	4	8am - 6pm	75	60
7	Pick-up	1	9am - 4pm	30	180
8	Delivery	1	10am - 4pm	120	120

Table 5.2: Details for working day 2 of Santa Fe

Customer	Pick-up/ Delivery	load (lift van)	Time Window	Driving Time (min)	Service time (min)
1	Delivery	1	8am - 7pm	120	120
2	Delivery	1	11am - 4pm	30	140
3	Delivery	1	3pm - 7am	60	90
4	Delivery	1	8am - 11am	45	60
5	Delivery	2	2pm - 5pm	45	90
6	Pick-up	1	9am - 6pm	60	60
7	Pick-up	1	1pm - 5pm	60	120

Table 5.3: Details for working day 3 of Santa Fe

Customer	Pick-up/ Delivery	load (lift van)	Time Window	Driving Time (min)	Service time (min)
1	Pick-up	4	09am - 3pm	30	90
2	Delivery	3	8am - 11am	60	60
3	Pick-up	3	12am - 4pm	60	75
4	Pick-up	1	8am - 11am	30	60
5	Delivery	1	12am - 4pm	60	90
6	Pick-up	2	8am - 4pm	90	30
7	Pick-up	2	8am - 4pm	90	30
8	Pick-up	2	8am - 4pm	90	30
9	Delivery	3	8am - 4pm	30	30
10	Delivery	3	8am - 4pm	30	30
11	Delivery	1	10am - 6pm	90	60

Table 5.4: Details for working day 4 of Santa Fe

Customer	Pick-up/ Delivery	load (lift van)	Time Window	Driving Time (min)	Service time (min)
1	Delivery	2	10am - 4pm	90	120
2	Pick-up	1	10am - 4pm	120	30
3	Delivery	1	8am - 10am	30	30
4	Delivery	1	11am - 1pm	30	60
5	Delivery	3	8am - 4pm	45	120
6	Delivery	2	8am - 1pm	60	60
7	Delivery	2	8am - 1pm	60	60

In Table 5.1 up to Table 5.4 the first column specifies whether a customer is a delivery or backhaul customer. The second column denotes the corresponding load for this customer. In the third column an individual time window is given for each customer. The fourth column denotes the driving time from the warehouse to the customer location (one-way) and the last column denotes the service time necessary at the customer location. The last two columns are estimates in minutes. In addition to this data we collected information about intermediate travel times between all customers in order to create routes serving more than one customers. We coded the computations in C++ and ran on a Dell Inspiron 1300 with an Intel(R) Pentium(R) M 1.70GHz processor using CPLEX 10.0 as our MIP-solver.

Table 5.5: Results of Scheduling Santa Fe activities

Day	# Cust.	# Routes	$U_{\texttt{real}}$	$U_{\texttt{model}}$	CPU(sec)
1	8	20	51%	51%	1.6
2	7	18	38%	38%	1.2
3	11	22	66%	66%	6.2
4	7	18	35%	35%	1.0

In Table 5.5 we have summarized further results for each of the data instances provided by Santa Fe. In this table we displayed the number of jobs in the second column, the number of routes given by the route generation procedure in the third column. The fourth and fifth column give the utilization rate of the vehicles as a measure of performance of the scheduling. Here we assume that all vehicles are operational for 11 hours. We compare the utilization rate resulted from the actual schedule ($U_{\texttt{real}}$) against ($U_{\texttt{model}}$). The last column presents the CPU-time in seconds needed to solve (5.14)-(5.25).

We notice that the number of feasible routes given by the route generation process is very small. The number of routes being so small gives us a great advantage in solving the model, since the number of decision variables is therefore very limited. This confirms with Laporte (2007) who suggests that set-covering based models can be very helpful for practical applications since the various side-constraints restrict the number of feasible routes. Recall that the exponential number of feasible routes in theory was a drawback for the set-covering approach.

We noticed that using the decision support system in the used instances does not lead to better solutions, but it guarantees no poorer solutions. Unfortunately, we are not able to compare route lengths with those obtained by the logistic manager. We have only information about the number of vehicles used for each day. Given that our approach gives quite meaningful results we would expect reasonable time savings when exact travel time and distances are known. In Figure 5.4 we show an overview of the decision support system.

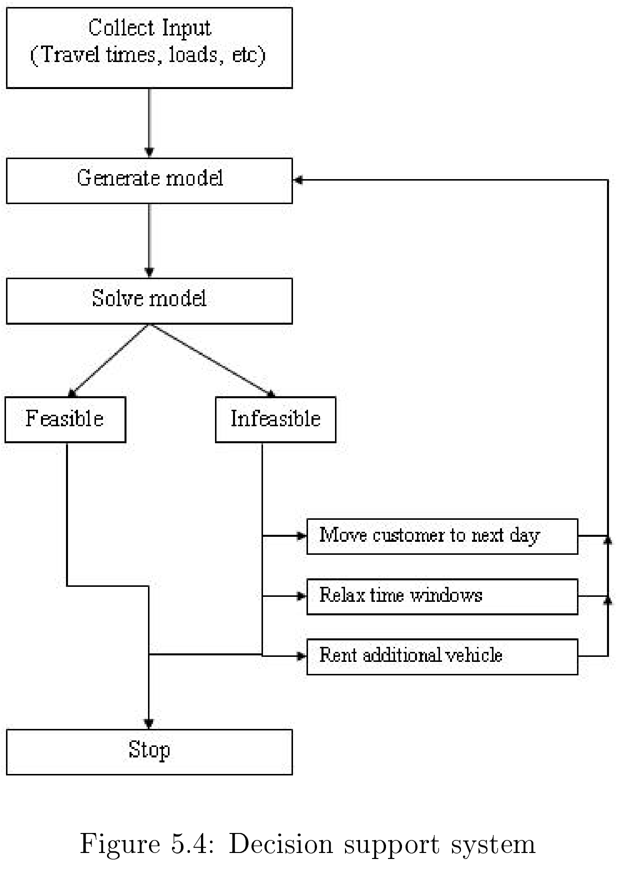

Figure 5.4: Decision support system

The main advantage of using a decision support system would be the time needed to find the optimal solution, meaning that the decision support system gives the optimal solution within seconds, which is a significant improvement for the logistic manager. Besides, additional last-minute jobs can be added to the data and instantly giving the logistic manager the knowledge whether or not this additional job is still feasible with the existing vehicle fleet. This information could then be used to possibly move this job to a later (more quite) day to avoid additional hiring costs.

5.4 Remarks

In this chapter we have seen an application of the set-covering model for VRPM discussed in Chapter 4. We have discussed additional side constraints such as a heterogeneous vehicle fleet, pick-up and delivery customers and individual time windows. For each of these additional constraints we made some minor modifications in the model.

We can conclude that the set-covering model is very flexible given that all these constraints can easily be incorporated in the model. We have seen that the problem instances from Santa Fe could all be solved within negligible CPU-time. Besides the small number of vehicles, it is due to the fact that the side constraints seriously reduce the number of feasible routes. Moreover we can say that a decision support system would be more beneficial when the number of customers in a period would be larger.

CHAPTER 6

CONCLUSION

In this study we basically propose two mathematical models to solve the VRPM. We first showed that an integer formulation based on the route-first, cluster-second method was performing really good in terms of CPU-time and is not inferior in terms of solution quality compared with other results in the literature. Next we adapted the set-covering formulation to solve the VRPM. The set-covering problem is a well studied problem and has been used to solve a wide-range of combinatorial optimization problems. We have tested the quality and speed of this approach on the available VRPM benchmark problems. We have seen that this approach is very compatible in terms of speed and is performing similar to the existing algorithms in terms of solution quality.

Given that all procedures in the literature are based on metaheuristics and their solution performances are based on the best outcomes of several replications, our simple approaches could be considered as an alternative to solve practical problem instances of moderate size. Moreover, our simple approaches could be a potential tool that can be embedded into a metaheuristic framework to solve practical problems of larger size.

In the last chapter we have modified the set-covering model to adopt several side constraints. We have showed that this model when applied to the VRPM can adopt a wide range of practical constraints such as time windows, a heterogeneous vehicle fleet and inaccessibility of certain vehicle types. We have seen that for problems with these additional side constraints the flexibility of the model is of significant importance. The set-covering model proved to be very flexible in this framework.

Future research areas may include the derivation of new models as well as practical applications. We think that mathematical models are of major importance these days for many companies operating a vehicle fleet. For example the total distance traveled by a vehicle fleet becomes more and more financially significant with oil prices that keep rising. Another reason is the growing demand of customers, consumers rely more and more on home delivery services, while they do not want to keep waiting all day for a delivery. Time windows are for this reason of great importance in such applications.

The VRPM can be of great help for these mathematical models since it can significantly increase the utilization of the fleet. It is possible to operate with a fewer number of vehicles, but with high utilization, if you operate as VRPM. Also vehicles with less capacity can be used in VRPM, rather than very large trucks.

REFERENCES

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

N. Azi, M. Gendreau, and J. Potvin. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, 178: 755–766, 2007a.

N. Azi, M. Gendreau, and J. Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. working paper, university of montreal. 2007b.

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

M. Balinski and R. Quandt. On an integer program, for a delivery problem. *Operations Research*, 12:300–304, 1964.

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

J. Bramel and D. Simchi-Levi. Set-covering-based algorithms for the capacitated vrp. In P. Toth and D. Vigo, editors, *Vehcile Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

J.C.S. Brandao and A. Mercer. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100:180–191, 1997.

J.C.S. Brandao and A. Mercer. The multiple-trip vehicle routing problem. *Journal of the Operational Research Society*, 49:799–805, 1998.

N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20: 255–282, 1981.

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

W. Cook. Concorde homepage:. *www.tsp.gatech.edu/concorde.html*, (Retrieved: April 2008), Georgia Institute of Technology, 2008.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44:216–229, 2004.

M. L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642, 1994.

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

P.M. Franca, M. Gendreau, G. Laporte, and F.M. Muller. The m-travelling salesman problem with minmax objective. *Transportation Science*, 29:267–275, 1995.

G.N. Frederickson, M.S. Hecht, and C.E. Kim. Approximation algorithms for some routing problems. *SIAM Journal Computing*, 7:178–193, 1978.

M. Gendreau, G. Laporte, and J. Potvin. Metaheuristics for the capacitated vrp. In P. Toth and D. Vigo, editors, *Vehcile Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

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

B. L. Golden, G. Laporte, and E.D. Taillard. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers & Operations Research*, 24: 445–452, 1997.

B. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33:3624–3642, 2006.

G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 1992.

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

G. Laporte and P. J. Dejax. Dynamic location-routeing problems. *Journal of the Operational Research Society*, 50(5):471–482, 1989.

G. Laporte and F. Semet. Classical heuristics for the capacitated vrp. In P. Toth and D. Vigo, editors, *Vehcile Routing Problem.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

J. Lenstra and A. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

C.K.Y. Lin and R.C.W. Kwok. Multi-objective metaheuristics for a location-routing problem with multiple use of vehicles on real data and simulated data. *European Journal of Operational Research*, 175:1833–1849, 2006.

S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21(2):498–516, 1973.

A. Olivera and O. Viera. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34:28–47, 2007.

I.H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.

I.H. Osman and G. Laporte. Metaheuristics: A bibliography. *Annals of Operations Research*, 66:513–623, 1996.

S. Pamuk, M. Köksalan, and R. Güllü. Analysis and improvement of the product delivery system of a beer producer in ankara. *Journal of the Operational Research Society*, 55: 1137–1144, 2004.

R.J. Petch and S. Salhi. A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133:69–92, 2004.

J. Renaud, G. Laporte, and F.F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229–235, 1996.

Y Rochat and E.D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal Heuristics*, 1:147–167, 1995.

D. M. Ryan, C. Hjorring, and F. Glover. Extensions of the petal method for vehicle routeing. *Journal of the Operational Research Society*, 44(3):289–296, 1993.

S. Salhi and G.K. Rand. Incorperating vehicle routing into the vehicle fleet composition problem. *European Journal of Operational Research*, 66:313–330, 1993.

A. Schrijver. On the history of combinatorial optimization (till 1960). *Working paper University of Amsterdam*, 2005.

G. Sierksma and G. A. Tijssen. Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research*, 76:261–286, 1998.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

E. D. Taillard, G. Laporte, and M. Gendreau. Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society*, 47:1065–1070, 1996.

E.D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23: 661–676, 1993.

P. Toth and D. Vigo. An overview of vehicle routing problems. In P. Toth and D. Vigo, editors, *Vehcile Routing Problem.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

P. Toth and D. Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123:487–512, 2002.

S. H. Zanakis, J. R. Evans, and A. A. Vazacopoulos. Heuristic methods and applications: A categorized survey. *European Journal of Operational Research*, 43(1):88–110, 1989.

APPENDIX A

RESULTS OF ROUTE-FIRST CLUSTER-SECOND APPROACH

		5%	10%	15%		
	m	=M1	=M2	=M3	z^{SP2}	x (Eq. 4.13)
CMT-01 (525)	1	-	**	**	560	6.746
	2	-	**	**	560	6.746
	3	-	-	**	560	6.746
	4	-	-	-	565	7.619
CMT-02 (835)	1	-	**	**	880	5.3564
	2	-	**	**	880	5.3564
	3	-	**	**	880	5.3564
	4	-	**	**	880	5.3564
	5	-	**	**	885	5.988
	6	-	**	**	892	6.8263
	7	-	-	**	890	6.5868
CMT-03 (826)	1	**	**	**	860	4.1162
	2	**	**	**	860	4.1162
	3	**	**	**	861	4.2373
	4	-	**	**	860	4.2373
	5	-	-	**	860	4.2373
	6	-	-	**	860	4.2373
CMT-04 (1029)	1	-	**	**	1086	5.5393
	2	-	**	**	1086	5.5393
	3	-	**	**	1086	5.5393
	4	-	**	**	1086	5.5393
	5	-	**	**	1086	5.5393
	6	-	**	**	1088	5.7337
	7	-	-	**	1087	5.6365
	8	-	-	-	1097	6.6084

CMT-05 (1292)	1	-	**	**	1375	6.5066
	2	-	**	**	1375	6.5066
	3	-	**	**	1375	6.5066
	4	-	**	**	1375	6.5066
	5	-	**	**	1375	6.5066
	6	-	**	**	1375	6.5066
	7	-	**	**	1375	6.5066
	8	-	**	**	1375	6.5066
	9	-	-	**	1378	6.6563
	10	-	-	**	1378	6.6563
CMT-11	1	**	**	**	1064	2.1113
	2	**	**	**	1064	2.1113
	3	**	**	**	1068	2.4952
	4	-	**	**	1064	2.1113
	5	-	**	**	1070	2.6871
CT-12	1	**	**	**	858	4.6341
	2	**	**	**	858	4.6341
	3	-	**	**	858	4.6341
	4	-	**	**	858	4.6341
	5	-	**	**	889	8.4146
	6	-	-	**	879	7.1951
F-11	1	**	**	**	249	2.9053
	2	**	**	**	249	2.9053
	3	-	**	**	249	2.9053
F-12	1	**	**	**	1219	4.8151
	2	-	**	**	1219	4.8151
	3	-	**	**	1219	4.8151

Figure A.1: The VRPM solutions using SP2-model

APPENDIX B

RESULTS OF SET-COVERING APPROACH I

In this table we present the results of the set-covering approach when we generate our initial set of routes using the route-first cluster-second method combined with petal routes from the advanced sweep method.

		5%	10%	15%		
	m	=M1	=M2	=M3	z^{SCT}	x (Eq. 4.13)
CMT-01	1	*	*	**	525	0
n=50	2	*	*	**	549	4.5714
z=525	3	-	*	**	549	4.5714
	4	-	-	-	-	-
CMT-02	1	-	*	**	879	5.2695
n=75	2	-	*	**	879	5.2695
z=835	3	-	*	**	879	5.2695
	4	-	*	**	879	5.2695
	5	-	*	**	879	5.2695
	6	-	*	**	888	6.3473
	7	-	*	**	908	8.7425
CMT-03	1	*	*	**	827	0.0012
n=100	2	*	*	**	827	0.0012
z=826	3	*	*	**	827	0.0012
	4	*	*	**	827	0.0012
	5	-	*	**	827	0.0012
	6	-	-	**	827	0.0012
CMT-04	1	-	*	**	1086	5.5394
n=150	2	-	*	**	1086	5.5394
z=1029	3	-	*	**	1086	5.5394
	4	-	*	**	1086	5.5394
	5	-	*	**	1086	5.5394
	6	-	*	**	1088	5.7337
	7	-	-	**	1087	5.6365
	8	-	-	-	-	-

CMT-05	1	-	**	**	1361	5.3405
n=199	2	-	**	**	1361	5.3405
z=1292	3	-	**	**	1361	5.3405
	4	-	**	**	1361	5.3405
	5	-	**	**	1361	5.3405
	6	-	**	**	1361	5.3405
	7	-	**	**	1375	6.5066
	8	-	**	**	1375	6.5066
	9	-	-	**	1378	6.6563
	10	-	-	**	1378	6.6563
CMT-11	1	**	**	**	1064	2.1113
n=120	2	**	**	**	1064	2.1113
z=1042	3	**	**	**	1068	2.4952
	4	-	**	**	1064	2.1113
	5	-	**	**	1070	2.6871
CMT-12	1	**	**	**	827	0.0085
n=100	2	**	**	**	827	0.0085
z=820	3	-	**	**	827	0.0085
	4	-	**	**	827	0.0085
	5	-	**	**	889	8.4146
	6	-	-	**	879	7.1951
F-11	1	**	**	**	249	2.9053
n=71	2	**	**	**	249	2.9053
z=242	3	-	**	**	249	2.9053
F-12	1	**	**	**	1211	4.1273
n=134	2	**	**	**	1212	4.2132
z=1163	3	**	**	**	1212	4.2132

Figure B.1: The VRPM solutions using SCT model

APPENDIX C

RESULTS OF SET-COVERING APPROACH II

In this table we present the results of the set-covering approach when we generate our initial set of routes using the route-first cluster-second method combined with routes from the saving heuristic.

	m	5% =M1	10% =M2	z^{SCT}	CPU(sec)
CMT-01	1	**	**	521	1
n=50	2	**	**	544	7
RT=525	3	-	**	549	4
	4	-	**	555	1
CMT-02	1	**	**	864	1
n=75	2	**	**	864	7
RT=835	3	**	**	864	5
	4	**	**	864	113
	5	**	**	864	249
	6	-	**	869	128
	7	-	**	875	831
CMT-03	1	**	**	824	1
n=100	2	**	**	824	1
RT=826	3	**	**	824	3
	4	**	**	824	28
	5	**	**	852	1252
	6	-	**	852	164
CMT-04	1	**	**	1072	2
n=150	2	**	**	1072	14
RT=1029	3	**	**	1072	453
	4	**	**	1072	358
	5	-	**	1072	677
	6	-	**	1077	807
	7	-	-		
	8	-	-		

CMT-05	1	**	**	1342	1
n=199	2	**	**	1342	37
RT=1292	3	**	**	1342	118
	4	**	**	1342	138
	5	-	**	1342	237
	6	-	**	1342	203
	7	-	**	1364	55
	8	-	-		
	9	-	-		
	10	-	-		
CMT-11	1	**	**	1041	1
n=120	2	**	**	1041	6
RT=1042	3	**	**	1041	10
	4	**	**	1053	85
	5	**	**	1041	18
CMT-12	1	**	**	827	1
n=100	2	**	**	827	1
RT=820	3	**	**	854	1326
	4	**	**	827	10
	5	-	-		
	6	-	-		
F-11	1	**	**	249	1
n=71	2	**	**	249	8
RT=242	3	-	**	249	11
F-12	1	**	**	1139	1
n=134	2	**	**	1139	1
RT=1163	3	**	**	1139	6

Figure C.1: The VRPM solutions using SCT model

APPENDIX D

DETAILS OF THE COMPUTER CODE

All codes are written in C++, we give some detailed information about the different used modules in Table D.1. Furthermore we have displayed the interaction between C++, CPLEX 10.0 and Concorde in Figure D.1.

Table D.1: C++ Modules

Name	# of lines	Details
Main	137	Main code.
Input	50	Reads input from text file.
Distance	31	Computes distance matrix.
Giant TSP	31	Solves TSP with Concorde.
Generate	69	Generates feasible routes given tour.
Saving	372	Generates feasible routes with saving heuristic.
Delete	161	Delete double routes from route list.
Cost	161	Calculates cost of each route.
Write	22	Writes route list to text files.
Improvement	460	Improvement of infeasible solutions.
Main	62	Main code for case study.
Generate1	39	Generates all routes with single customer of same type.
Generate2	95	Generates all routes with >1 customer of same type.
Generate3	175	Generates all routes with linehaul and backhaul customers.

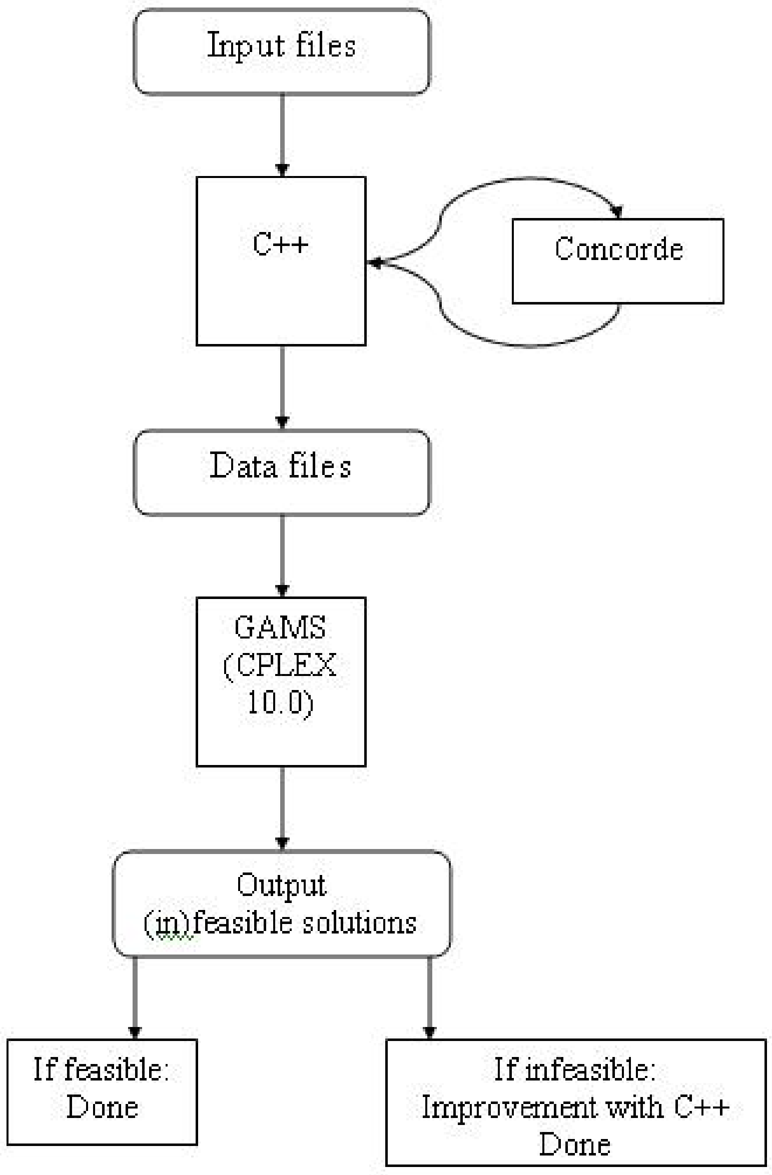
Input files
C++
Concorde
Data files
GAMS (CPLEX 10.0)
Output (in)feasible solutions
If feasible: Done
If infeasible: Improvement with C++ Done

Figure D.1: Systematic overview computer activities

APPENDIX E

TEST PROBLEM INSTANCE STRUCTURES

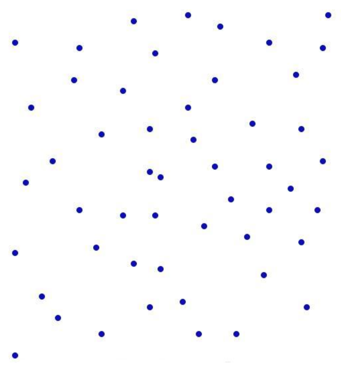

Figure E.1: CMT-01

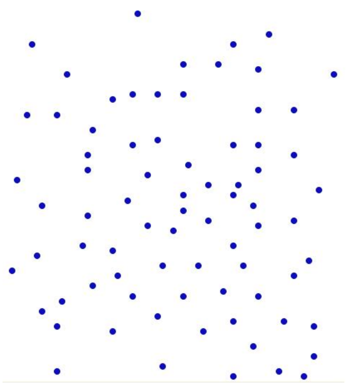

Figure E.2: CMT-02

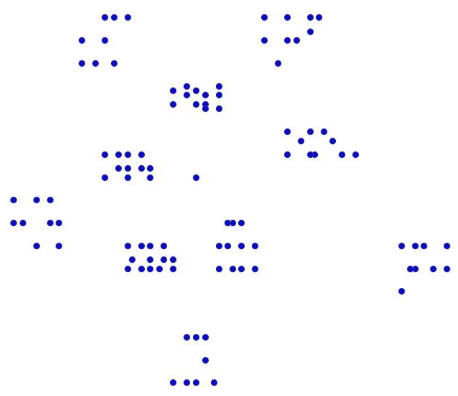

Figure E.3: CMT-03

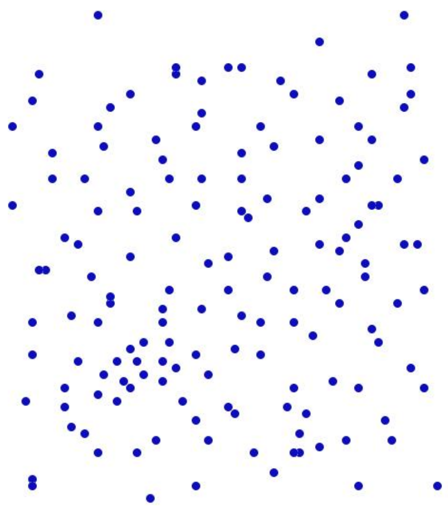

Figure E.4: CMT-04

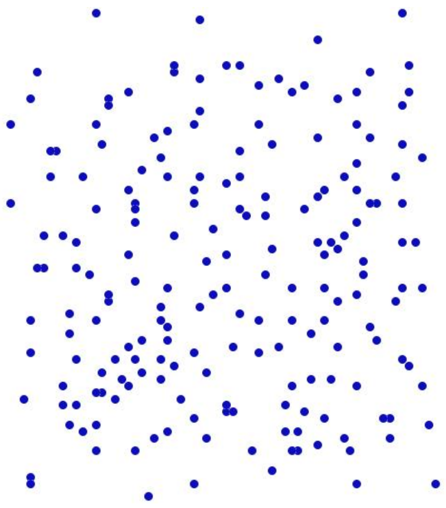
Figure E.5: CMT-05

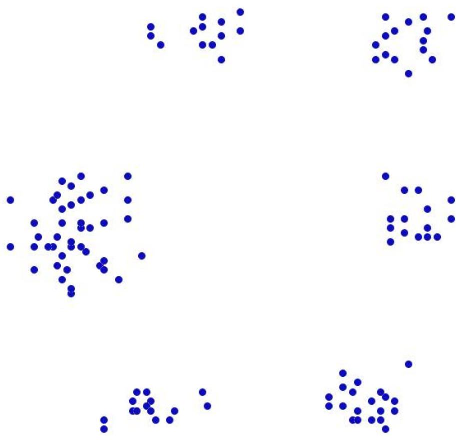
Figure E.6: CMT-11

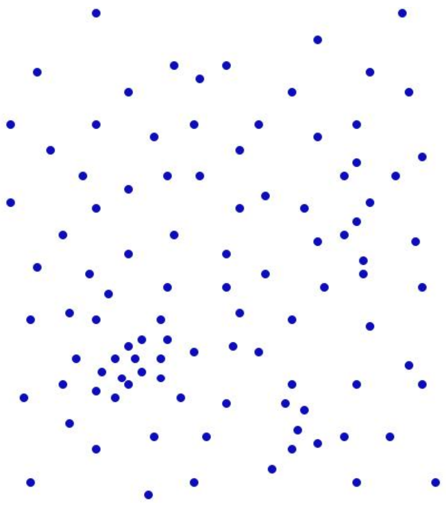
Figure E.7: CMT-12

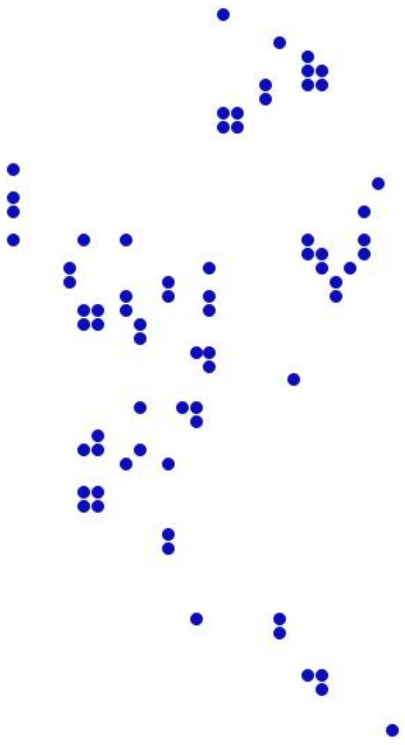
Figure E.8: F-11

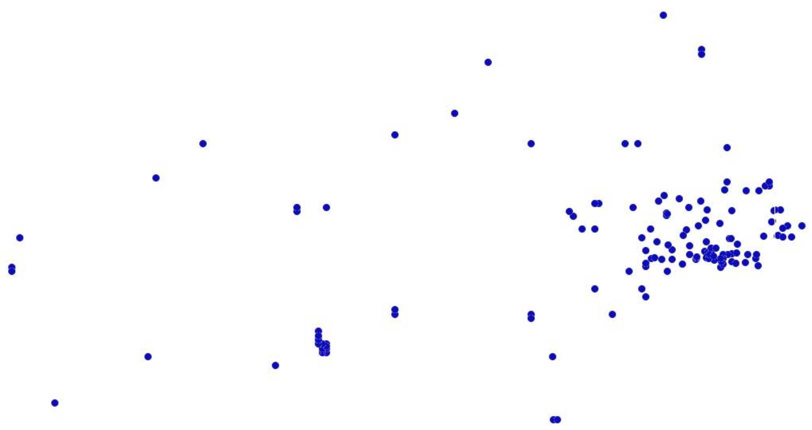

Figure E.9: F-12