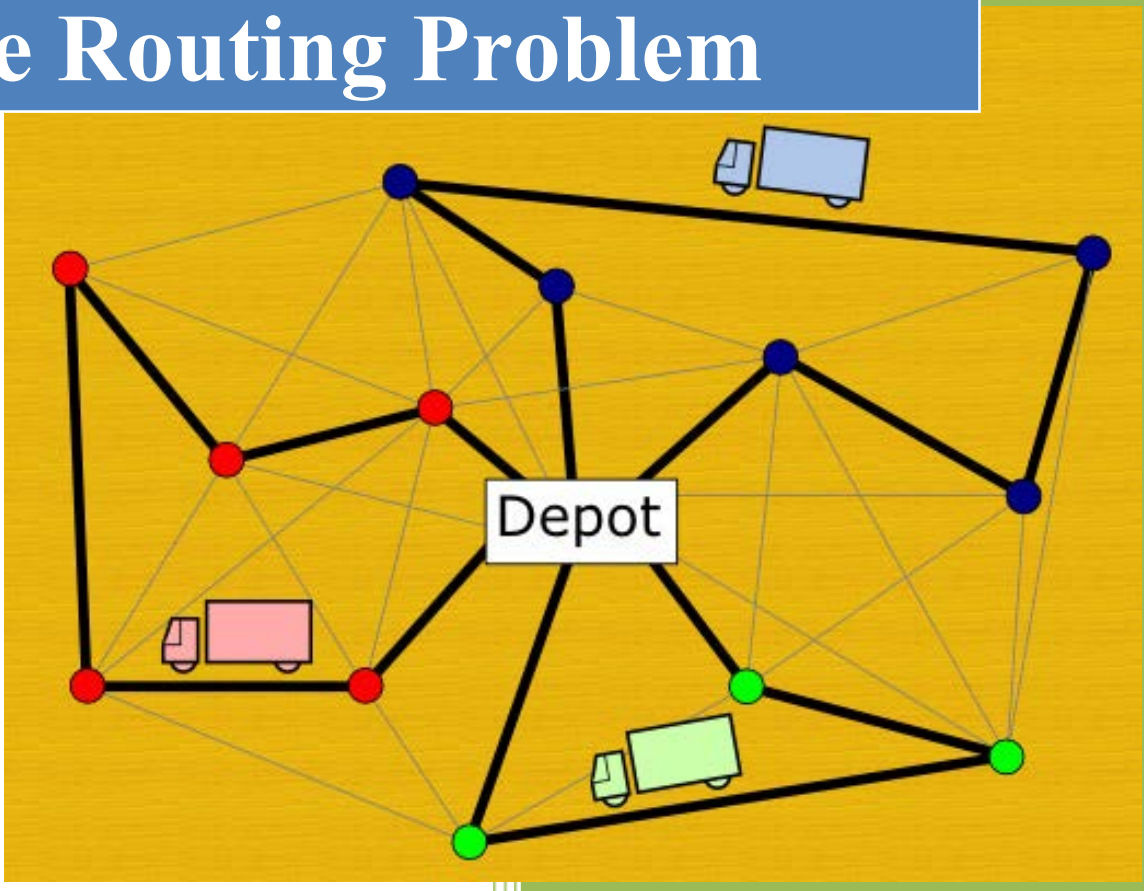
BMI thesis

The Multiple Trip Vehicle Routing Problem



Bilal Singer

Vrije Universiteit Amsterdam

29-09-2008

BMI thesis

The multi-trip vehicle routing problem

Author: Bilal Singer

Student number: 1426702

Supervised by: Marco Bijvank

Vrije Universiteit Amsterdam

29-09-2008

Preface

One of the final compulsory subjects of the master study Business Mathematics and Informatics (BMI) is the BMI thesis. In this thesis a problem in the field of BMI is assessed using existing literature. The subject addressed in this paper is the Multiple Trip Vehicle Routing Problem (MTVRP). This is a variant on the standard Vehicle Routing Problem (VRP) which has been extensively covered in literature. Although the MTVRP is a real-life problem, not a lot of literature can be found on this subject. That is why it is interesting to give a summary of the solutions that are available and give a basic comparison of these solutions.

In January 2007, during the Optimization of Business Processes project an algorithm to solve the MTVRP was developed. After the completion of this project, the members wondered to what extend their solution would match up to available solutions in literature. The algorithm developed during that project will therefore also be discussed and compared to the other available solutions.

Finally I would like to thank my supervisor Marco Bijvank for his encouragement and support. I have written this thesis with pride and joy and I have learned a lot. I hope that you, the reader, will also read this thesis with pleasure. Hopefully, I can share the acquired knowledge with you this way.

29-09-2008

Bilal Singer

Summary

The subject addressed in this paper is the Multiple Trip Vehicle Routing Problem (MTVRP). This is a variant on the standard Vehicle Routing problem (VRP) which has been extensively covered in literature. Although the MTVRP is a real-life problem, not a lot of literature can be found on this subject. That is why this subject was chosen for my BMI thesis .

The objective of this thesis is:

To research the available techniques for solving Multi-trip VRP and give a comparison of these techniques.

A number of solution techniques are described in this thesis. These solution techniques have different characteristics and produce different results. The conclusions that were taken are:

- Greedy algorithms work better than non-greedy when solving the MTVRP
- The Adaptive Memory Procedure of Olivera and Viera [21] produces the best results for the MTVRP.

The goal of each of the discussed MTVRP algorithms is to minimize the total driving time of all routes. However, in practice the total driving time is not the most important quality indicator of a MTVRP solution. Many real-life companies are interested in minimizing the number of used vehicles and not necessarily in minimizing the total driving time. It would therefore be better to look at the number of used vehicles of a solution when comparing MTVRP algorithms.

The self-developed algorithm provided poor results, which was somewhat anticipated. However, this algorithm can be used as a starting point for further research in the field of MTVRP algorithms. The self-developed algorithm also provides a solution for incorporating a heterogeneous fleet. This solution technique can also be used in other (MT)VRP algorithms that want to incorporate a heterogeneous fleet.

Index

Chapter 1

Introduction

In today's business world, transportation costs form a major share of the total logistics expenses of companies. That is why many companies are giving special attention to the transportation costs of goods in order to minimize their expenses. Decreasing transportation costs can be achieved through better utilization of resources such as people and vehicles. Companies try to improve their transportation by using rational manners and effective tools. Finding an improved planning for transportation is a problem which is frequently addressed in literature. The objective of these problems is to make a planning for the routes executed by people or vehicles, such that the transportation cost is minimized. In literature such problems are referred to as *routing problems*.

The most well-known routing problem is the *Travelling Salesman Problem* (TSP) [32]. In this problem a salesman has to visit a number of cities and afterwards the salesman has to return to the location where (s)he started. The objective in a TSP is to construct a route such that the travel distance is minimized. A TSP can be generalized to a *m-TSP* where *m* salesmen have to cover the given cities. Each city must be visited by exactly one salesman. Every salesman starts from a starting point, the depot, and must also return to this depot at the end of his/her journey. The objective in an *m*-TSP is to minimize the sum of the total distances travelled by the salesmen.

A *Vehicle Routing Problem* (VRP) [33] can be considered as a *m*-TSP where a demand is associated with each city. In a VRP a number of vehicles are stationed at one or several depots and these vehicles need to visit a number of geographically dispersed clients. A specific set of routes for each of the vehicles has to be determined in order to serve all the clients. The aim is to design these routes in order to meet certain (capacity or time) constraints and to minimize a given objective function.

This thesis focuses on the *Multi-Trip VRP* (MTVRP) [6]. This is a specific variant of the VRP in which each vehicle can drive more than one route consecutively. A vehicle that returns from a short route may be used a second time or even a third time as long as the maximum driving time isn’t exceeded.

The MTVRP is very relevant because in practice many transportation companies have vehicles that can drive multiple routes on a day. Despite the fact that the MTVRP occurs a lot in everyday practice, there has not been much research on the MTVRP. Therefore, hardly any literature is available on this subject. That is the main reason for focusing this study on the MTVRP. Another reason for choosing the MTVRP is the fact that during a university project last year an algorithm for solving the MTVRP was developed by myself together with three other BMI-students. It is interesting to see how that algorithm compares to the already available solutions for the MTVRP.

The objective of this research is:

To research the available techniques for solving Multi-trip VRP and give a comparison of these techniques.

The remainder of this thesis is organized as follows. In Chapter 2 the VRP and it's different variants will be discussed. In Chapter 3 a mathematical formulation of the MTVRP is given. In Chapter 4 the currently available techniques for solving the MTVRP are discussed. After that, the algorithm that was developed during the university project will be presented in Chapter 5. Finally a comparison will be made between all discussed techniques in Chapter 6 and a conclusion will be given in Chapter 7.

Chapter 2

Vehicle Routing Problems

A Vehicle Routing Problem (VRP) is a general name for problems to assign a set of clients to vehicles by means of routes. To solve the VRP a set of routes is designed in order to serve the given clients. Each of these routes belongs to a certain vehicle (see figure 1).

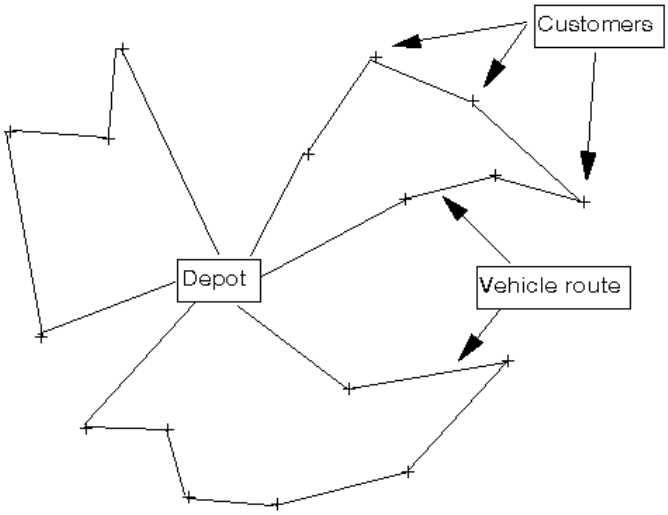


Figure 1: A simple VRP example.

The VRP is defined more than 50 years ago [1] and is considered one of the most challenging combinatorial optimization tasks. This is because VRP's belong to the category of NP hard (Non-deterministic Polynomial-time Hard) problems. This means that the computational effort required to solve this problem increases exponentially with the problem size. Consequently, these kinds of problems are often solved with approximate solutions.

The goal of a VRP-algorithm is to minimize a given objective function, usually characterized by the total length of the routes. The length of a route can be measured in distance (e.g. miles) or in time (e.g. hours). Another objective could be to minimize the number of routes of the number of vehicles needed.

There are also a number of different constraints that can be applied to a VRP. The most basic constraint is the time constraint for the vehicles. This constraint means that each vehicle only has a limited amount of time T it can drive each day. The VRP has to meet this constraint when minimizing the objective function.

There are a number of different variants on the VRP. Each of the specific variants of the VRP has some different constraints or characteristics that have to be taken into account.

§2.1 Capacitated Vehicle Routing Problem

The most elementary version of the VRP is the *Capacitated Vehicle Routing Problem* (CVRP). The CVRP has a number of specific characteristics and constraints. There is only one depot from which the fleet of vehicles originates. Each vehicle in that fleet has a limited capacity Q available to deliver goods. Each vehicle also has a limited amount of time T it can drive each day. There are a number of clients and each client that is visited by a vehicle has a certain demand q that occupies a portion of the capacity of that vehicle. It is not possible to split the delivery of the goods intended for one client. A CVRP solution is therefore a collection of routes with a minimum total driving distance. These routes have to visit each customer exactly once, the total demand of each route is at most Q and the total length of each route is at most T. The capacity and maximum driving time are identical for each vehicle, so all vehicles are of the same type. When all vehicles of a fleet are identical, it is called a *homogeneous fleet*.

For the CVRP a number of exact algorithms like Branch & Bound, Branch & Cut and dynamic programming are known for solving this problem. However, these algorithms can only find a solution within modest computing times for a maximum of 100 clients [25]. This is not of great practical use so these methods are typically not used. There are also a number of heuristics known which provide good results also on large problem instances, for instance the Clarke and Wright Savings algorithm [2] and the Sweep algorithm [3].

Since the Capacitated VRP is the most basic version of the VRP it is usually referred to as VRP. In the remainder of this thesis the CVRP will also be referred to as VRP.

§2.2 Other VRP variants

Some of the other VRP variants and their characteristics will be briefly discussed in this paragraph. They all share the same basic restrictions as the CVRP.

2.2.1 Multiple Depot VRP

A *Multiple Depot VRP* (MDVRP) is similar to a CVRP with the only difference that a number of different depots are available from which the vehicles can depart to service the clients. In this problem it is not sufficient to link vehicles to clients but it is also necessary to determine from which depot the vehicle should originate.

If there is a clear clustering of clients around the depots, then it is simple to model this problem as a set of independent VRPs. Otherwise, it requires more effort to solve the MDVRP. This is usually done by a cluster-first, route-second method. In this method first an allocation of clients to depots is made and then a number of regular VRPs are solved.

2.2.2 Split Delivery VRP

In a *Split Delivery VRP* (SDVRP) it is allowed for different vehicles to serve the same client. This is a relaxation of the regular VRP assumption where each client can only be served by one vehicle. The overall cost of a SDVRP are less than in a VRP, especially if the size of the client

orders are (nearly) as big as the capacity of the vehicles.

2.2.3 Period VRP

In a basic VRP the routes are usually determined over a period of one single day. In a *Period Vehicle Routing Problem* (PVRP), the basic VRP is generalized by extending the planning period to *M* days. Each client can require to be visited more than once during this planning period.

2.2.4 Stochastic VRP

In a *Stochastic VRP* (SVRP) one or several components of the problem is random and can be stochastically represented. Usually three kinds of components that can be stochastic:

- Stochastic customers: each customer is either present with probability *p* or absent with probability *(1-p)*.
- Stochastic demand: the demand of each customer is a stochastic random variable.
- Stochastic time: the service and travel times are stochastic random variables.

A possible way to solve a SVRP is to use two stages. The first stage determines a solution to the problem before knowing the realizations of the random variables. In the second stage a corrective action can be taken when the values of the random variables are known.

2.2.5 VRP with Pick-up and Delivering

The *VRP with Pick-up and Delivering* (VRPPD) adds the possibility for a client to return some goods. These goods can either be returned to the depot or can be used for delivery to another client. Therefore it is necessary to take into account that the returned goods of a client must also fit into the vehicle. This makes the planning problem more complicated and leads to a worse performance of the algorithm compared to a basic VRP (increased travel distance or more vehicles).

If the restriction is added that all deliveries are completed before any pickups are made and that all picked-up goods are brought to the depot, it is called a VRP with Backhauls (VRPB).

2.2.6 VRP with Time Windows

The *VRP with Time Windows* (VRPTW) adds a restriction that each client is associated with a specific time window at which the vehicle must visit the client. This problem has received a lot of attention in literature [4][5]. This is mostly due to the wide applicability of time window constraints in real-world cases like bank deliveries, postal deliveries or school bus routing.

2.2.7 VRP with a Heterogeneous fleet

In a basic VRP all vehicles in the fleet have the same capacity. At a *VRP with a Heterogeneous fleet* (HVRP) the vehicles can have different capacities. Another difference with the basic VRP is that the number of vehicles available in the fleet is limited and known beforehand.

2.2.8 Multiple Trip VRP

In a *Multiple Trip VRP* (MTVRP) a vehicle can drive more than one route consecutively. Each route still has a maximum capacity it can carry. However, a vehicle that returns from a short route may be used a second time or even a third time as long as the maximum driving time is not exceeded. This is an important characteristic, especially in urban areas, where the vehicle capacity is the limiting factor and the travel times are rather short.

It is possible to combine some of these variants. For instance a CVRP with Pick-up and Deliveries and Time Windows (CVRPPBTW) or a MDVRP with Time Windows and a heterogeneous fleet (MDHVRPTW). This research however focuses on the Multiple Trip VRP with only a single depot and a homogeneous fleet. In the next chapter we present a review of the available solution heuristics in the literature designed for the MTVRP.

Chapter 3

Model description and notation

In this chapter a mathematical formulation of the MTVTP is presented so it can be used in the remainder of this thesis.

Consider a number of locations where each vehicle can drive to/from. A location can be either a depot or a customer. There is a single depot, $i = 0$. There are n customers, where each customer i, $(i=1,\ldots,n)$, has a demand q_i and a non-negative service-time s_i. There are m vehicles, each vehicle v has a capacity Q_v. The fleet of vehicles is denoted by V, so $v \in V=\{1,2,\ldots,m\}$. The driving time between location i and location j is denoted by t_{ij}.

The assumptions and restrictions in this problem are:

- each route starts and ends at the single depot, $i = 0$.
- every client has to be served by exactly one vehicle.
- the sum of the demands of the clients served by a single vehicle in one trip may not exceed the capacity of the vehicle.
- the maximum driving time of each vehicle is T.
- a vehicle may drive more than one route as long as the total driving time of that vehicle does not exceed T.
- the fleet is homogeneous, since each vehicle has the same capacity (so $Q_v=Q$ for all v).

Other notation needed for this problem:

- the minimal 'resting' time for a vehicle between driving two routes is t_0. This time can be used to load the truck.
- a route $R = (i_1, \ldots, i_k)$ is a sequence of clients visited by a vehicle, including the travel from the depot to client i_1 and from client i_k to the depot.
- an assignment of a route R to a vehicle is denoted by $v(R)$.
- the total demand of all clients in route R is denoted by $q(R)$ and the total driving time is denoted by $t(R)$.
- a schedule S is a set of feasible routes $R \in S$, such that every client is contained in exactly one route.
- a schedule S is feasible if there exists an assignment $v(R) \in V$ of the routes $R \in S$ to vehicles such that:
    - every route is assigned to a vehicle and the total time needed to drive the routes assigned to a vehicle, T_v, should not exceed the maximum driving time of that vehicle:

$$T_v = \sum_{R:v(R)=v} [t(R)+t_0] \leq T+t_0, \ \forall \ v \in V \qquad (1)$$

    - every route is only assigned to a vehicle that has enough capacity:

$$q(R) \leq Q_{v(R)} \ \forall \ R \in S \qquad (2)$$

Equations (1) and (2) are called the feasibility conditions.

- a shuttle route, R^S is a route containing only a single customer. $R_1^S = (i_1)$ is the shuttle route containing only client 1. All shuttle routes must be feasible. If a shuttle route isn't feasible because of the time or capacity restriction, then there exists no solution for that MTVRP.

Heterogeneous fleet:

Some algorithms allow for the possibility of having a heterogeneous fleet. In a heterogeneous fleet the capacity Q_v of the vehicles does not have to be the same. Each vehicle v has a different capacity Q_v.

When a route R is to be assigned to a vehicle, it is necessary to know to which vehicles in the heterogeneous fleet, this route can be assigned to, to result in a feasible schedule. This can be facilitated by creating different classes of vehicles:

- first the vehicles are ordered according to decreasing capacity:
  $Q_1 \geq Q_2 \geq \ldots \geq Q_m$.
- after that, the vehicles are grouped into classes of identical capacity. Vehicle class C_1 is the class of vehicles that have the largest capacity. In general, C_i is the class of vehicles that have the i-th largest capacity. The capacity of each vehicle v is equal to the capacity of the vehicle class it belongs to: $Q_v = Q_{C_k} \ \forall \ v \in C_k$.

Using this notation, it is possible to find for each route R the vehicle class of the smallest feasible vehicle, $c(R)$. This is the class with the smallest capacity (the highest class number) that has a capacity equal to or larger than the capacity needed for that route.
The class of the smallest feasible vehicle is: $c(R) = \max\{k : Q_{C_k} \geq q(R)\}$

In the next chapter the solution techniques found in the available literature are described.

Chapter 4

Solution techniques

In this chapter all solution techniques for the MTVRP known in the literature will be presented. Each of these solution techniques has some unique extra notation or assumptions only needed for that solution. These will be described in the respective paragraphs of the solutions. The bin-packing assignment heuristic is used in several of the techniques to solve the MTVRP. Therefore, we first explain the basics of a bin-packing assignment heuristic.

§4.1 Bin-Packing assignment heuristic

In every MTVRP solution technique, at some point, an assignment of routes to vehicles has to be performed. A number of the MTVRP solution techniques, that will be discussed in the remainder of this chapter, use the same heuristic for assigning a number of routes to a fleet of vehicles. They use the *bin-packing assignment* heuristic.

The bin-packing assignment heuristic is used in order to determine an assignment of vehicles $v(R)$ for all constructed routes $R \in S$ that satisfy the capacity and time constraints. S is a given schedule that contains a number of feasible routes. When routes have to be assigned to a number of vehicles, an assignment problem has to be solved. This problem is solved by performing a bin-packing assignment heuristic.

There are a number of variants of the bin-packing assignment heuristic. The specific heuristic that is used in this context is the First-Fit-Decreasing (FFD) heuristic. The FFD heuristic is modified in order to solve the assignment problem of routes to vehicles.

The first step of the FFD algorithm is to sort the list of routes. The routes are sorted so that the most 'difficult' routes are placed at the top of the list. A route is more difficult to assign if it requires a larger vehicle class (lower $c(R)$) or it requires more driving time (higher $t(R)$). The routes are sorted in increasing order with respect to the vehicle class of the smallest feasible vehicle, $c(R)$. For equal $c(R)$ the routes are sorted in decreasing order with respect to driving time of that route $t(R)$. The FFD then assigns routes to vehicles, starting from the most difficult routes (top of the sorted list of routes). The routes are then sequentially put into the first available vehicle that has enough space available.

Assignment heuristic:

The assignment of the routes of a given schedule S to a given fleet V is done in the following way:

1) The driving time of vehicle v, T_v, is set to zero $\forall\ v \in V$.
2) Sort all $R \in S$ in increasing order with respect to $c(R)$ and, when $c(R)$ is equal, in decreasing order with respect to $t(R)$.

3) For every $R \in S$ starting from the top of the sorted list :

For every $v \in V_c$ $(c = c(R), c(R)\text{-}1, \ldots, 1)$:

If $\{ T_v + t(R) \leq T \}$

a) $v(R) = v$,

b) $T_v = T_v + t(R) + t_0$,

If R has not been assigned: stop, no feasible assignment found.

If all routes are assigned to a vehicle, a feasible assignment is made. This assignment is then presented as result.

This bin-packing assignment procedure is used in a number of different heuristics that solve the MTVRP. The MTVRP algorithm of Fleischmann [6] (§4.2) however, was the first to use this procedure for the MTVRP.

§4.2 Savings Procedure for Multiple Use of vehicles (SPMU)

Fleischmann [6] was the first author to address the MTVRP in 1990. He proposed a heuristic called the Savings Procedure for Multiple Use of vehicles (SPMU) to solve the MTVRP. This heuristic was based on the standard Savings algorithm for a basic VRP [2] and was extended to take the possibility of multiple trips into account. A description of the Savings algorithm can be found in Appendix A.

Extra notation for this algorithm:

- Two routes $R_1, R_2 \in S$, can be combined into one route R^*. If $R_1 = (i_1, \ldots, i_k)$ and $R_2 = (j_1, \ldots, j_l)$, then the combined route will be $R^* = R_1 \times R_2 = (i_1, \ldots, i_k, j_1, \ldots, j_l)$.
- A link between the two routes can only be made at the first or last client of the routes R_1 en R_2.
- When two routes are combined it yields a certain savings $C(i,j)$ dependent on the objective function.
- The savings of combining two routes is dependent on the first and last client of these routes. When clients i and j are the first and last customers of two different routes the savings of combining clients i and j is computed as:
$$C(i,j) = t_{0,i} + t_{0,j} - t_{i,j}$$
- In Fleischmann's algorithm [6] the total driving time of all vehicles is minimized.

Fleischmann's heuristic [6](§4.3) allows for the possibility of having a heterogeneous fleet. This is the only algorithm described in the literature that incorporates a heterogeneous fleet for the MTVRP.

4.2.1 The SPMU algorithm

The main idea of this algorithm is to start with a schedule S_0 consisting of all shuttle routes and then iteratively combine routes together in new routes based on the savings. The routes are then assigned to the vehicles using the bin-packing assignment algorithm (§4.1).

The SPMU algorithm for solving the MTVRP consists of two parts.

1) Initial step.
2) Savings iteration.

Each part will now be discussed in more detail:

4.2.2 Initial step

The SPMU algorithm starts with schedule S_0 consisting of all shuttle routes. It is likely that the initial schedule is not feasible with respect to the number of vehicles needed. In order to enforce a feasible assignment of S_0 a number of $\overline{m}$ *fictitious vehicles* is introduced in the initial step of the procedure. Each fictitious vehicle has a capacity equal to the capacity of the smallest feasible vehicle class (i.e. $c(R)$), belonging to shuttle route R^S that is assigned to that fictitious vehicle. The fictitious vehicles are numbered $m+1,\ldots,m+\overline{m}$ and augment the real vehicles V and the

appropriate vehicle classes C_k. The initialization procedure of the SPMU is as follows:

1) Create the initial schedule consisting of all shuttle routes; $S = S_0$.

2) Determine the minimum vehicle class $c(R)$ needed for every route $R \in S_0$.

3) Set the number of fictitious vehicles $\overline{m} = 0$.

4) Determine an assignment $v(R)\ \forall\ R \in S_0$ using the bin packing assignment heuristic described in §4.1.3. If this heuristic fails to find a vehicle for a route R, introduce a fictitious vehicle v by doing the following:

- **a)** Set $\overline{m} = \overline{m} + 1$, $v = m + \overline{m}$ and $k = c(R)$.
- **b)** $Q_v = Q_{C_k}$, $C_k = C_k \cup \{v\}$ *and* $V = V \cup \{v\}$.
- **c)** $v(R) = v$, $T_v = t(R) + t_0$.
- **d)** Continue the assignment algorithm.

4.2.3 Savings iteration

After the initial step, the iterative savings procedure is started and routes are combined based on the computed savings. The highest savings are processed first and only routes that can be can be assigned to one of the vehicles in the fleet are actually combined. The combined routes are assigned to vehicles using the bin-packing assignment algorithm (§4.1). As soon as a fictitious vehicle becomes idle during the savings procedure, it is withdrawn. This means that the number of fictitious vehicles never increases, except at the initial step. The iterative savings procedure is as follows:

1) Order all pairs (i,j) $(i = 2, \ldots, n,\ j = 1, \ldots, i\text{-}1)$ from the largest to the smallest value of $C(i,j)$.

2) Go to the first pair (i,j) in the sorted savings list such that i and j are the first or last customer in different routes $R_1, R_2, \in S$.

3) Test whether the new route $R^* = R_1 \times R_2$ can be assigned to a vehicle:

Test 1: Search if route R^* can be assigned to any of the two vehicles $v(R_1)$ or $v(R_2)$ relieved of routes R_1 and R_2, by doing:

set $v'(R) = v(R)$ for $R \neq R_1, R_2$ and $T'_v = T_v$ for $v \in V$;

$$T'_{v(R_i)} = T'_{v(R_i)} - t(R_i) - t_0 \quad (i = 1,2);$$

For i = 1,2:

If $c(R_i) \leq c(R^*)$ and $T'_{v(R_i)} + t(R^*) \leq T$:

$v'(R^*) = v(R_i)$;

$$T'_{v(R_i)} = T'_{v(R_i)} + t(R^*) + t_0;$$

go to step 4)

Test 2: Search if any other vehicle $v'(R^*)$ in the fleet can execute route R^*. If one is found go to step 4).

Test 3: Determine a new assignment $v'(R)$ for $R \in S \setminus \{R_1, R_2\} \cup R^*$ using the bin-packing assignment heuristic. If that is successful go to 4). If it fails, go to 5).

4) $v(R) = v'(R)$ $(R \in S)$;
$T_v = T'_v$ (v ∈ V);
Withdraw all unused fictitious vehicles from V and V_c.

5) If there are savings left in the savings list, proceed to the next pair (i,j) in the sorted savings list such that i and j are the first or last customer in different routes $R_1, R_2, \in S$; else stop the SPMU.

The SPMU results in a set of routes each assigned to a vehicle. If one or more fictitious vehicles remain that are assigned of routes, then the SPMU couldn't find a feasible result. If all fictitious vehicles are removed, then the SPMU has found a feasible result for the MTVRP.

§4.3 A Tabu Search algorithm

A totally different way of solving the MTVRP is by using a tabu search algorithm. For the basic VRP these algorithms produce very good results [7][8][9]. Tabu search is a mathematical optimization method, belonging to the class of local search techniques. Tabu search enhances the performance of a local search method by using memory structures: once a potential solution has been determined, it is marked as "taboo" ("tabu" being a different spelling of the same word) so that the algorithm does not visit that possibility repeatedly [27].

Taillard et al. [10] propose a heuristic to solve the MTVRP by using a tabu search algorithm based on the Rochat-Taillard principle [7]. This principle allows for diversification of the search process to take place by combining promising solutions. This is fairly similar to what is done in genetic algorithms [26].

The main idea of this algorithm is to create a number of different basic VRP solution and then transform these solutions into MTVRP solutions. The best MTVRP solution is then chosen presented as the final solution.

The Tabu Search algorithm for solving the MTVRP consists of three parts.

1) Generation of a large set of feasible vehicle routes.
2) Use the generated set to come to a number of basic VRP solutions.
3) Transform the basic VRP solution into a MTVRP solution using the bin-packing algorithm.

Each part will now be discussed in more detail:

Part 1: Generation of a large set of feasible vehicle routes.

The aim of the first part is to generate a number of feasible routes which can be used in Part 2. The generation is done in such a way that the selected routes are of high quality. This part has a number of steps which are described below:

1) *Creating a first set of feasible routes:*
   In this step a first set of feasible routes is generated. This is done by constructing a number of solutions for the basic VRP by using the tabu search algorithm of Taillard [9]. An unspecified number of vehicles is assumed here. Every VRP solution found in this step will be different because of the diversification of the search process in the Taillard tabu search algorithm [9].

   The individual vehicle routes of all found VRP solutions are then put in a set and each of the routes is assigned a label. This label is determined by the total driving time of the respective VRP solution, which is used as an indication of the quality of that route.

2) *Adding more routes to the set:*
   In this step some more feasible routes are generated using the set of feasible routes generated in 1). The reason for generating even more routes than already available from 1) is

because this step generates a much broader set of feasible routes. The generation of these routes is done as follows:

(i) A route is randomly selected from the set generated in 1) where the probability of selection is proportional to the inverse of the driving time of the route.

(ii) In the set of routes generated in 1) a client is served by more than one route, because the set of routes result from a number of different VRP solutions. To ensure that in this step a client is not selected multiple times, all routes in the set that have clients in common with the already selected routes in (i) are discarded. If some routes remain in the set, go to (i) to select another route. If no routes remain in the set, go to (iii).

(iii) Using the routes that were selected in (i)-(ii) as a starting point, a tabu search is applied according to the algorithm of Taillard [9] to generate a new VRP solution. The individual routes found in this new VRP solution are appended to the set of 1) and labeled the same way as in 1). If two routes have the same set of clients, the route with the smallest driving time remains.

The result 1 is a large set of vehicle routes that are all feasible with respect to the capacity and time constraints. All selected routes are of high quality. The set of routes is given as input to Part 2.

Part 2: Generation of VRP solutions

In this part a number of feasible basic VRP solutions are created. These basic VRP solutions are found using an enumerative algorithm. The set of routes resulting from Part 1 is used as input for this enumerative algorithm. This enumerative algorithm uses a search tree [34] to find a large number of feasible VRP solutions.

The following steps are executed in order to find basic VRP solutions.

1) First a number of routes is selected from the input set. At most q routes (where usually $q >> m$) are selected in non-decreasing order of their driving time. These routes are put into a set J with a selection rule that each client must be selected at least once.

2) Then within a search tree, all feasible VRP solutions that can possibly be constructed by combining the routes in J are generated. In order to control the growth of the search tree, branching priority is always given to routes containing the largest number of clients. This process results in a set of K feasible VRP solutions.

These solutions will be transformed into MTVRP solutions in Part 3. It is however not certain that every VRP solution resulting from Part 2 can be transformed into a MTVRP solution in Part 3.

Part 3: Generation of solutions for the VRPM

In the last part of the heuristic the construction of a feasible solution for the MTVRP is attempted. This is done by solving a bin-packing problem for each of the K VRP solutions resulting from Part 2. This bin-packing algorithm is almost the same assignment heuristic as used in the SPMU of Fleischmann. Only one addition is made:

- In case a feasible assignment is not found, routes are repeatedly swapped between vehicles to try and find a feasible solution. It is however not guaranteed that a feasible solution is found in this stage.

The best MTVRP solution out of the K possible solutions is then selected and presented. The presented MTVRP solution is the final solution of this algorithm.

§4.4 Variations on the Tabu Search algorithm

After Taillard et al. [10] proposed their MTVRP algorithm a number of variations were developed, that also used a tabu search algoritm for solving the MTVRP. These algorithms were very similar to the Taillard et al. [10] algorithm. Three of these algorithm are described in this paragraph.

4.4.1 Minmax procedure

Golden et al. [11] made a small addition to Taillard et al.'s procedure [10] in order to incorporate a *minmax* objective function. E.g. to minimize the largest driving time of a vehicle. The minmax objective can be achieved by including a small addition at the end of the Taillard et al. algorithm [10]:

After all routes have been assigned to vehicles using the bin-packing assignment heuristic (§4.3, Part 3), an *interchange* procedure is applied. This procedure swaps routes between vehicles in order to improve the given minmax objective.

This is only a small addition and will only be used in case a minmax objective is necessary.

4.4.2 Zhao's extension

Zhao [17] proposed a solution for the MTVRP which is very similar to Taillard et al.'s heuristic [10]. Zhao's heuristic [17] has a significant difference with respect to the Taillard et al. algorithm [10] since it already constructs MTVRP solutions in Part 1 of the algorithm.

The differences in the procedure of Zhao's heuristic [17] compared to the Taillard et al. algorithm [10] are:

- At Part1, Step 1) when the first set of solutions is generated, Zhao uses a Savings algorithm [2] modified with a neighborhood function instead of Taillard's tabu search algorithm [9].
- Every VRP solution that is generated in Part 1, Step 1) is directly transformed into a MTVRP solution by using the bin-packing algorithm already in this stage. The label of each route that is put into the list is then determined by the total driving time of the MTVRP instead of the VRP. This addition does not change the kind of routes generated but only the value that each route gets.
- The same thing is done at Part 1, Step 2.(iii). There Zhao [17] also generates MTVRP solutions instead of basic VRP solutions as is done in Taillard et al.'s algorithm [10]. This is done to label each route with the MTVRP value instead of the VRP value.

Part 2 and 3 of Zhao's algorithm are the same as the Taillard et al. algorithm [10]. Zhao [17] added these variations to ensure that his algorithm accounts for the possibility of multiple trips per vehicle, in an earlier stage than the Taillard et al. algorithm [10].

4.4.3 Adaptive Memory Procedure

Oliveira and Viera [21] also proposed a solution to the MTVRP using the same general procedure as Taillard et al. [10]. However, they used an adaptive memory approach in order to find good MTVRP solutions.

The main difference with the algorithm of Taillard et al. [10] is that an adaptive memory is used to preserve only a number of solution when new VRP solutions are created in Part 2 of the Taillard et al. algorithm [10]. Part 2 of the Taillard et al. algorithm [10] is extended as follows:

Only a maximum of AMP^{size} top routes are kept in the list after each iteration in Part 2, Step 2). This is done to increase the quality of the produces solutions. The routes are ordered by their driving time and the routes having the highest driving time are discarded until only AMP^{size} routes remain in the list.

Another difference is that at Part 1, Step 1) of the Taillard et al. algorithm [10], Oliveira and Viera [21] use a Sweep algorithm in order to generate the first set of VRP solutions instead of the Taillard Tabu Search algorithm [9].

Another difference is that Oliveira and Viera [21] already construct MTVRP solutions at the end of Part 1 to label the routes that are added to the set. This is the same addition as Zhao's [17]. Part 1 of the Taillard et al. algorithm [10] is extended as follows:

Each of the VRP solutions created in Part 1, Step 2) (iii) is improved using a tabu search algorithm. This tabu search algorithm uses the bin-packing assignment algorithm (see §4.1) to create a MTVRP solution. The routes that are added to the set of routes are labeled by the driving time of their MTVRP solution.

§4.5 Brandao & Mercers technique

Brandao and Mercer [13] also proposed a solution which makes use of a tabu search approach. However, their approach is principally different from the Taillard et al. algorithm [10] and therefore is discussed separately. Brandao and Mercer [12] originally designed a heuristic for a real-life problem which included other requirements like time-windows, different types of vehicles, the hiring of extra vehicles and restricted access to some clients for some types of vehicles. After that, the authors have simplified their heuristic to a VRP which only adds the possibility of Multiple Trips, in order to compare their heuristic with the Taillard et al. algorithm [10]. The solution method of Brandao and Mercer [13] is different from many other MTVRP algorithms in that it does not use a bin-packing heuristic in order to assign routes to vehicles.

The approach of Brandao and Mercer [13] consists of two parts:

1) The first part is an algorithm based on a nearest-neighbour rule and an insertion criterion that provides an initial solution for the MTVRP.
2) The second part tries to improve this solution in a tabu search framework using two types of trial moves, namely insert and swap.

This heuristic will not be discussed in great detail because it would take too much explanation of sub-heuristics like nearest-neighbour and the specific tabu search heuristic. The general scheme of the procedure will be discussed below:

Part 1: Creating an initial solution

The first part is an algorithm based on a nearest-neighbour rule and an insertion criterion to provide an initial solution for the MTVRP. In this part two independent procedures are executed sequentially multiple times. The two independent procedures are:

1) The creation of a set of feasible routes where each route is assigned to a different vehicle. Such a set is called a layer.
2) Insertion of clients into the layers.

The combined execution of the two procedures is called a stage. The goal of a stage is to assign one route to each vehicle and to fill these routes as much as possible with clients. The procedure of a stage is as follows:

1) *The creation of a layer.*
   A route is created in a layer by assigning the unassigned client that lies farthest away from the depot to the vehicle with the largest driving time remaining that has not yet received a route in this stage.
2) *Insertion of clients into the layers.*
   The insertion procedure is applied in order to include more clients into the routes of the created layers. Clients are added into the routes by using a nearest-neighbour algorithm [35].

A stage results in a layer that is filled with routes that serve as much clients as possible. In each layer each route belongs to a different vehicle in the fleet. The maximum number of routes in each layer is m (i.e. the number of available vehicles). After a stage is performed, there could be a number of clients that are still not assigned to a route. In that case another stage is performed.

This means that a new layer is created on top of the other existing layers. Again clients are assigned to the routes in this layer, if possible. Each vehicle can thus have multiple routes, if the driving time restriction allows for it. Each of these routes would then be in a different layer.

The final solution given in Part 1 results from successively performing several stages. It is possible to use overtime for the routes in the last layer of each vehicle if there are still unrouted clients that could not be inserted using the standard maximum driving time *T*. This means that in Part 1 the MTVRP solution is allowed to become infeasible. This is corrected by allowing only feasible MTVRP solutions in Part 2.

Part 2: Tabu search algorithm

In this part a tabu search algorithm is executed in order to improve the initial MTVRP solution given in the previous part. If the solution from Part 1 wasn't feasible, the tabu search algorithm will try to make the solution feasible.

There are two kinds of trial moves in this tabu search heuristic, namely insert and swap moves:

- An insert move consists of removing one client from one route and putting it in another route.
- A swap move consists of exchanging two clients belonging to two different routes.

Part 2 results in a set of routes, each assigned to a specific vehicle. This is the final solution for the MTVRP.

§4.6 Multi-Phase heuristic

Petch and Salhi proposed a Multi-Phase heuristic [14] to solve the MTVRP. They developed a heuristic that tries to integrate the approach used by Taillard et al. [10] and that of Brandao and Mercer [13].

The framework of this heuristic is largely the same as the framework proposed by Taillard et al. [10]. However, it has a number of differences in every step. The main idea of this algorithm is to create a number of different basic VRP solutions and then produce a feasible MTVRP solution using the bin-packing assignment heuristic (§4.1) and some improvement techniques.

The Multi-Phase heuristic for solving the MTVRP consists of three parts.

1) Generation of VRP solutions using a modified savings.
2) Construction of a MTVRP solution and improvement.
3) Generation of more VRP solutions using a route population approach.

Each part will now be discussed in more detail:

Part 1: Generation of VRP solutions using a modified savings algorithm

In this first part a sample of VRP solutions is generated using a modification on the savings algorithm. The modified savings heuristic described by Yellow [15] is used to construct routes in a parallel way. A number of VRP solutions are generated using this heuristic. This results in a pool P of VRP solutions.

Part 2: Construction of a MTVRP solution and Improvement

In this part a solution for the MTVRP is constructed using a pool P of VRP solutions as input. Every VRP solution in pool P is sequentially processed in this part. A VRP solution is transformed into a MTVRP solution using the bin-packing assignment heuristic (§4.1) and then a number of improvement techniques are executed to improve the MTVRP solution. The general procedure of this part is as follows:

1) The VRP solutions in the pool are ranked according to their value (total driving time). The highest ranking solution requires the least driving time and is the first solution to be processed.
2) Apply the bin-packing heuristic (§4.1) on the selected VRP solution in order to construct a MTVRP solution S. In this assignment overtime is allowed, meaning it is possible to produce a non-feasible MTVRP solution.
3) Improve the MTVRP solution by using a number of improvement techniques. These techniques are: Meiosis, VRP Partition, Donate, Exchange and Donate Exchange.
4) S is compared with the best known solution up till now, S_{best}. If the overtime of S is smaller than the overtime of S_{best}, $S_{best} = S$.
5) If S_{best} is MTVRP feasible (meaning no overtime), the heuristic is stopped and S_{best} is presented as result.
6) If there are still solutions in pool P, the next ranked solution chosen and processed from Step 2); else the heuristic is stopped and S_{best} is presented as result.

The result of this part is either a feasible MTVRP solution or a non-feasible MTVRP solution with the least overtime.

Part 3: Generation of more VRP solutions using a route population approach

Part 3 is not always executed in the Multi-Phase heuristic. First the pool of solutions found in Part 1 is processed in Part 2. If this results in a feasible MTVRP solution, the heuristic is stopped without executing this part. However, if no feasible solution is found using the pool of solutions found in Part 1, a new and larger pool of solutions is generated in this part. The new pool of solutions will then again be processed by Part 2.

In order to generate a new and larger pool of solutions a method is executed based on a route partition approach. This method generates a population of routes that satisfy the VRP constraints. The essence of this method is as follows:

1) The clients are ranked with respect to their angle with the depot.
2) For every client a number of feasible routes are generated, choosing that client as the first client on each route. A variant on the sweep method [16] is used for finding these routes. Each route is then improved using the 2-Opt and 3-Opt methods [29].
3) Using a search tree [34] VRP solutions are constructed by combining the generated routes. In this search tree data structures and reduction are used.

The result of the entire Multi-Phase heuristic is either a feasible MTVRP solution or a non-feasible MTVRP solution with the least overtime.

§4.7 Genetic Algorithms

Petch and Salhi [22] proposed a solution for the MTVRP using genetic algorithms (GA's). Genetic algorithms are search techniques that try to find an approximate solution to optimization problems. The main idea of a GA is that starting with an initial set of solutions, a new set of solutions can be created that give better results. Each solution is represented by a chromosome and a set of solutions is called a generation. Each time a new set of solutions is found, a new generation is created. The GA heuristic continues to iterate until the maximum number of generations, GEN^{max}, is reached

The genetic algorithm proposed by Petch and Salhi [22] is used to partition all clients in different clusters. The clients are partitioned in different clusters by using the angles of the clients with respect to a reference line (usually the x-axis). Each cluster starts at a certain angle and ends at the starting angle of the next cluster. All clients that are positioned between the begin and end angle of a certain cluster are assigned to that cluster. In this way a partition is made of clients into clusters. This is much like the sweep procedure [3]. The chromosomes of the GA represent the partition of clients into different clusters. Each chromosome is a sequence of begin angles of the different clusters. A generation represents a set of begin angles.

The GA procedure for solving the MTVRP consists of three parts:

1) Create an initial set of solutions.
2) Produce new populations.
3) Selection of the best solution.

Evaluation procedure:

Once a chromosome is constructed, an evaluation procedure is executed to evaluate the quality of that chromosome. A chromosome is evaluated by first creating a MTVRP solution using a combination of the savings algorithm [2] and the bin-packing assignment heuristic (§4.1) and then using the total driving time of all vehicles to label that chromosome. The details of the evaluation procedure are as follows:

1) Generate clusters belonging to those angles and assign the clients to their related clusters.
2) Generate routes for each cluster using the savings heuristic [2].
3) Allocate routes to vehicles using the bin-packing assignment algorithm as described in §4.1.
4) Compute the *fitness* of that solution. The fitness of a solution is determined by the total driving time of all vehicles in that solution.

This evaluation procedure is used in both Part 1 and Part 2 of the GA procedure. Each part of the GA procedure will now be discussed in more detail:

Part 1: Create an initial set of solutions

1) Create a number of Z chromosomes $\{X_i,\ldots,X_Z\}$.
2) For each chromosome do the following:
   a). Randomly select k angles from [0°,359°].
   b). Evaluate the chromosome using the evaluation procedure.

Part 1 results in a set of Z initial solutions for the MTVRP. This set is called the initial generation of chromosomes. This initial population is given as input to Part 2.

Part 2: Produce new populations

This is the main part of the recursive process where new population of chromosomes are created and evaluated. Using the initial population of chromosomes found in the previous part as a starting point, new and improved generations are created iteratively. Each iteration results in a new generation. This iterative procedure is as follows:

The creation of a new generation of chromosomes is done using four different types of GA operators:

- *Chromosome injection*:
  Randomly created chromosomes are inserted into the new generation.
- *Chromosome cloning*:
  The cloning mechanism is based upon maintaining a variety in the quality chromosomes. The chromosomes of the previous generation are ranked according to a fitness function and then partitioned into groups. After that, a proportion of chromosomes within each group is selected.
- *Extraction operator*:
  The operator extraction is the primary operator used to generate a new offspring. Given two chromosomes X_i and X_j, extraction produces a single offspring. Extraction is used to add a series of sectors, from a chosen partner X_j, into the chromosome X_i by overwriting the corresponding sectors. In other words, a sequence of angles of chromosome X_j is transplanted in the angle composition of chromosome X_i.
- *Mutation operator*:
  The mutation operator selects a number of chromosomes from the previous generation and adds a mutation of each of these chromosomes to the new generation. A chromosome is mutated by changing some of the angles of that chromosome randomly.

After the creation of a new set of chromosomes, the MTVRP solutions belonging to this set of chromosomes are determined using the evaluation procedure.

The result from Part 2 is a set of MTVRP solutions. This set is given as input to Part 3.

Part 3:Selection of the best solution

In this part, the best solution for the MTVRP is selected from the input set. This is done in the following way:

1) All solutions found are ranked according to their fitness function. The top 10% solutions are then chosen.
2) Some improvement modules are performed on the chosen solutions. These are the same techniques as used in the Multi-Phase heuristic of Petch and Salhi [14] (discussed in §4.5).
3) Finally the solution with the best fitness (smallest total driving time) is selected.

The selected solution from Part 3 is the final solution for the MTVRP.

§4.8 Insertion Heuristic Approach

Cambell and Savelsbergh [18] proposed a solution for the MTVRP in 2002 by using an insertion heuristic. Insertion heuristics are frequently used in solving a number of different VRP variants [19][20]. Cambell and Savelsbergh [18] have extended the insertion heuristic in order to incorporate multiple trips.

This heuristic tries to sequentially add every unrouted customer *j* to that specific vehicle and route which provides the highest profitability. Profitability is defined by the negative of the extra travel time needed for that route by inserting the unrouted customer *j* between the already routed customer (*i-1*) and *i*. Profitability of adding unrouted customer *j* between customer *(i-1)* and customer *i* is: $-(t_{(i-1),j} + t_{j,i} - t_{(i-1),i})$.

The insertion heuristic approach for solving the MTVRP consists of two parts:

1) Initialization.
2) Iteration.

Each part will now be discussed in more detail:

Part 1: Initialization

In the initialization phase, every vehicle in the fleet is given an empty route. In the iteration phase, clients are assigned to these routes. As soon as a client is assigned to an empty route of a vehicle in the iteration phase, a new empty route is created for that vehicle. In this way each vehicle can have multiple routes assigned to it, but always has one empty route available.

Part 2: Iteration

In this phase each client is sequentially assigned to a vehicle and a route. This is done by first computing all possible profitability's of that client and then assigning this client to the vehicle and route that have the highest profitability. The procedure of the iteration phase is as follows:

For each unassigned client *j* the following is done:

1) For every route *R* assigned to every vehicle $v \in V$ *v, (v(R) = v):*
   For every customer *i* in route *R*:
   - Determine whether inserting the unrouted customer *j* between *(i-1)* and *i* results in a feasible solution. Feasibility is checked with respect to capacity and time constraints.
   - If an insertion of *j* is feasible the profitability is computed for this insertion and this profitability is saved in a list.

2) Customer *j* is inserted between clients *(i-1)* and *i* that give the highest profitability. A new route is added if necessary, such that every vehicle has an empty route.

This results in a solution where each vehicle has a number of routes assigned to it. All customers are assigned to one of these routes. This is the final solution for the MTVRP.

§4.9 Route linking procedure

Goodson [23] proposed a heuristic for solving the MTVRP in his working paper in 2007. He does not use the bin-packing algorithm as many other MTVRP algorithm, but he proposes a different method of assigning routes to vehicles.

This algorithm uses an observation relating to the nature of linking routes in a single vehicle. A vehicle has to return to the depot for two reasons:

- if the vehicle load is near vehicle capacity.
- if the route driving time is near the maximum driving time.

If the vehicle has to return to the depot because of the capacity restriction, that vehicle can drive another route in the remaining time after the first route. If the vehicle returns to the depot because of the maximum driving time constraint, that vehicle cannot drive another route. This observation is used to link routes together in a certain vehicle.

The main idea of this algorithm is to first generate a large set of feasible routes and after that, link some of these routes together using the above observation. Finally a number of linked routes are assigned to vehicles using a set covering problem (SCP) [28] algorithm.

The route linking procedure for solving the MTVRP consists of three parts.

1) Route generation.
2) Route linking.
3) Selection of linked routes.

Each part will now be discussed in more detail:

Part 1: Route generation

In this part, a large number of potential routes is generated using the sweep approach [16]. The routes are generated the following way:

1) a certain customer *i* is chosen as starting point for the sweep approach [16].
2) all possible feasible routes, with customer *i* as starting point, are generated using the sweep approach [16]. This heuristic takes into account both the capacity constraint and the maximum driving time constraint.

This procedure is executed for every available customer. All routes generated for all customers are then put together in a list.

The list with all possible routes is given as input to Part 2.

Part 2: Route linking

The generated routes from Part 1 are now linked together to form *linked-routes*. A linked-route is a combination of a couple of routes that can be executed sequentially by a single vehicle of the

homogeneous fleet. The sum of the driving times of the routes combined in the *linked-route*, does not exceed the maximum driving time.

If all potential linked-routes are created by combining the routes in every possible way, it will result in a huge amount of linked-routes. This will also require huge computational effort. In order to reduce the number of generated linked-routes Goodson [23] uses his route-linking procedure. Goodson proposed to link the routes using the two observations he made, relating to the nature of linking routes. He formulated these two observations as rules that are used in his algorithm:

1) Rule 1: do not link two routes if the sum of the capacity is less than the capacity of the vehicle. If two of such routes would be linked, unnecessary additional distance is added. This happens because it is not necessary to return to the depot between driving these two routes since the capacity constraint is not violated.
2) Rule 2: a series of r linked-routes must consist of at least r-1 *full* routes and at most 1 *non-full* route. A *full* route is a route that can accommodate no more than b additional customers without exceeding the vehicle capacity or the route duration limit. A non-full route is a route that is not *full*. This rule was formulated by using the observation that a vehicle returns to the depot when its current route cannot accommodate another customer.
   For instance: if $b = 2$, all routes that can accommodate more than 2 customers without exceeding the vehicle capacity or the route duration limit, are *non-full*. All other routes are *full*. To test whether a route can accommodate a number of clients, the sweep heuristic [16] is used.

Both these rules ensure that less linked-routes are generated and that the computational effort is also reduced. The choice of parameter b determines the number of *full* routes and therefore influences the number of potential route links. The procedure for executing this part is as follows:

All routes generated in Part 1) are divided in *full* and *non-full* routes based on parameter b. After that, the following iterative procedure is applied to generate the linked-routes:

1) Full routes are linked with other *full* routes until additional route links are infeasible in terms of route duration. All possible combinations of *full* routes are created this way.
2) At most 1 *non-full* route is added to each of the generated series of full routes in 1). All possible combinations of a series of *full* routes and the addition of a *non-full* route are generated.

Throughout the procedure, routes are only linked if they satisfy Rule 1 and if they contain no common customers, so that a customer is not visited more than one time.

The result from Part 2 is a large set of linked routes. Each of these linked routes can be assigned to one of the available vehicles in the homogeneous fleet. The set of linked routes is given as input to Part 3.

Part 3: Selection of linked routes

In this part a subset of the linked-routes generated in Part 2 will be selected such that each client is visited once. Each of the selected linked-routes is then assigned to a vehicle. The selection of linked-routes is done by formulating a *Set Covering Problem* (SCP) [28]. The SCP is a classical question in computer science and complexity theory. The input of a basic SCP is a collection of several sets. Each set contains a number of elements and the different sets may have some elements in common. The goal of a SCP is to select a minimum number of these sets so that the selected sets contain all elements exactly once.

Goodson [23] represents each linked-route by a set in the SCP model and each client that has to be served is represented by an element in the SCP model. The goal of this SCP is then to minimize the number of chosen linked-routes such that all customers are visited. Goodson uses the Lagrangian-based heuristic proposed by Caprara et al. [24] to solve the SCP.

In this way a number of linked routes are selected. Each linked route is then assigned to one of the vehicles in the fleet. This is the final solution for the MTVRP.

Chapter 5

Self-developed algorithm

During the course Project Optimization of Business Processes in 2007 we developed an algorithm to solve a MTVRP. This algorithm was developed without using any of the MTVRP algorithms described in the previous chapter. It was created by adding some modifications to the savings algorithm based on intuition to deal with multiple trips. This algorithm can also deal with a heterogeneous fleet of vehicles.

In many respects the algorithm developed is similar to the SPMU developed by Fleischmann [6]. It does however have a number of differences. The main idea of the self-developed algorithm and the differences with the Fleischmann's algorithm [6], will be described in §5.1 and §5.2. The general procedure will be described in mathematical terms in §5.3.

The notation will be the same as in Fleischmann [6] (see § 4.2).

§5.1 Part 1: initialization

The developed algorithm is initialized by creating the schedule S_0 consisting of all shuttle routes. First, all client are put in a list and are then sorted in decreasing order based on their demand. For each of these clients a shuttle route is made and each of these shuttle routes is assigned to a vehicle. This is done by sequentially assigning the unassigned shuttle route with the largest demand to the largest unused vehicle. If a shuttle route has a demand that is bigger than the capacity of the largest unused vehicle, split delivery is applied. Client i is split in two new clients, i_1 and i_2, where i_1 will have a demand equal to the capacity of the largest unused vehicle and i_2 will get a demand equal to the demand of i minus the demand of i_1. Client i_1 will be put into a shuttle route and is assigned to the largest unused vehicle. Client i_2 will be put in the list of clients that still have to be served in its appropriate position when looking at the demand. If there are no unused vehicles remaining, a fictitious vehicle with the smallest capacity Q_{min} is added to the fleet. All of the created shuttle routes must be feasible with respect to the driving time. If a shuttle route isn't feasible then there exists no solution for this problem.

In the initialization phase there are two main differences with the SPMU of Fleischman [6]. The first difference is the possibility of splitting clients when the demand it too big. The second difference is that only fictitious vehicles with the smallest capacity, Q_{min}, are added in contrast to Fleischmann's algorithm [6] in which fictitious vehicles can have larger capacity.

§5.2 Part 2: iteration

After the algorithm is initialized, the savings procedure is started and routes are combined. The self developed algorithm has a number of differences with respect to Fleischmann's algorithm [6].

5.2.1 Route combining

The self-developed algorithm combines two routes R_i and R_j in a combined route R^* using an improvement procedure. The order in which the clients of the combined route are visited is improved using a Traveling Salesman Problem (TSP) heuristic, namely Farthest Insertion (FI). This is an important difference with Fleischman's algorithm [6]. Fleischman [6] only tries to combine the routes of their first and last customers, so the order in which the clients are visited in the combined route does not change from their original routes. The self-developed algorithm combines two routes in a different manner using the Farthest Insertion heuristic. In this way the order in which the clients are visited in a combined route is improved.

5.2.2 Calculation of the savings

To determine which routes should be combined we calculate the savings. These savings are computed differently than the savings in Fleischmann's algorithm [6]. In the self-developed algorithm, a TSP heuristic, Farthest Insertion, is used in order to find the *TSP savings*. These savings are based on the difference in driving time when combining two routes using the Farthest Insertion heuristic. The *TSP savings* of combining routes R_i and R_j into route R^* are determined by:

$$S(i,j) = t(R_i) + t(R_j) - FI(R^*)$$

Where $FI(R^*)$ is the total driving time or route R^* when combining routes R_i and R_j into route R^* using the Farthest Insertion heuristic.

The *TSP savings* are savings based on combining entire routes and not only combining clients as is the case at Fleischmann's algorithm [6]. This means that if the set of routes is changed, by combining two routes into one route, our savings have to be computed again. The savings of Fleischmann's algorithm [6] only have to be computed once in the beginning.

5.2.3 Feasibility tests

In every iteration of the savings procedure we need to be test whether adding the combined route R^* and removing the routes R_i and R_j from the schedule provides a feasible solution. This is done by performing two tests. First we try to assign route R^* to one of the vehicles $v(R_i)$ or $v(R_j)$ relieved of the routes R_i and R_j (test 1) and after that we try to assign R^* to any other available vehicle in the fleet (test 2). Fleischmann [6] uses the same two tests, but he also adds a third test to this procedure. His third test comprises of performing a complete new assignment using the bin-packing assignment heuristic (§4.2). Our testing procedure may lead to a rejection of certain route combinations (savings) that would be accepted using the third test of Fleischmann [6].

5.2.4 Fictitious vehicles removal

If a fictitious vehicle isn't used any more, it isn't removed directly out of the fleet in the self-developed algorithm. Only at the final step of the algorithm, these fictitious vehicles are removed. Once the savings are processed, all routes are reassigned to vehicles using the same bin-packing heuristic used by Fleischmann. At the self-developed algorithm the fictitious, vehicles that aren´t used any more at a certain step of the iterative savings procedure, could be assigned to a route in subsequent steps of the iterative savings procedure. Fleischmann [6] removes the fictitious vehicles as soon as they become idle and these fictitious vehicles cannot be used anymore in subsequent steps of the iterative savings procedure.

§5.3 General procedure

The general procedure of the self developed algorithm is as follows:

Part 1: initial step

In this step an assignment $v(R)$ for all $R \in S_0$ is determined using the following procedure:

1) Create the initial schedule S_0 consisting of all shuttle routes. The routes in this schedule are not yet assigned to a vehicle.

2) Set the number of fictitious vehicles $\overline{m} = 0$.

3) Sort the routes in decreasing order based on their demand: $q(R_1) \geq q(R_2) \geq \ldots \geq q(R_n)$

4) Assign route R_i with the highest demand $q(R_i)$ to the largest available vehicle v with the highest capacity Q. Call this vehicle V^{max} with capacity Q^{Vmax}.
This assignment cannot be executed if $Q^{Vmax} < q(R_i)$ or if there isn't any unassigned vehicle left.

   If $Q^{Vmax} < q(R_i)$ split the delivery in two routes R_1 and R_2, where route R_1 will get a demand of Q^{Vmax} and R_2 will get a demand of $(d^{Ri} - Q^{Vmax})$. Assign route R_1 to vehicle V^{max} and add route R_2 to S_0 according to decreasing demand.

   If there are no more unassigned vehicles available, then introduce a fictitious vehicle V^{fic}:
   - Set $\overline{m} = \overline{m} + 1$ and set the capacity of this fictitious vehicle to the smallest capacity of a vehicle, Q_{min}.
   - Add this fictitious vehicle to the fleet of vehicles.
   - Assign route R_i to V^{fic} and apply split delivery if necessary.

5) Proceed with the next unassigned route in S_0 until all routes are assigned.

Part 2: iteration

In this step the savings are processed and the routes are combined in order to create a feasible MTVRP solution.

1) $S = S_0$.

2) Compute the savings $S(i,j)$ for combining all unassigned routes R_i and $R_j \in S$ using:
$$S(i,j) = t(R_i) + t(R_j) - FI(R^*)$$
If there are no unassigned routes, go to 6), else go to 3).

3) Put the savings in a list and sort them on a descending order.

4) Take the top unprocessed savings $S(i,j)$ in the sorted list and test whether routes R_i and R_j can be combined into the combined route R^*. This is done by checking whether route R^* can be assigned to a vehicle by performing the following two tests:

Test 1: Check whether the new route R^* can be assigned to one of the vehicles $v(R_i)$ or $v(R_j)$ relieved of routes R_i and R_j.

Test 2: Check whether the new route R^* can be assigned to any of the other vehicles in the fleet.

5) If the assignment was accomplished go to 2), if the assignment wasn't feasible and there are still savings in the list also go to 2), else go to 6).

6) All routes are now assigned to a vehicle and there are no more feasible savings available. In order to make sure that the assignment of routes to vehicles is optimal the routes are reassigned. All routes are first unassigned and then a bin packing assignment heuristic is executed. For details on this heuristic see § 4.2.1.

This results in a set of routes each assigned to a vehicle. If there are still fictitious vehicles in the solution then a feasible solution could not be found using this algorithm. If no fictitious vehicles are present in the solution then this algorithm presents a feasible solution for that MTVRP.

Chapter 6

Comparison of solution techniques

In this chapter a comparison is made between the MTVRP solution techniques discussed in the previous chapters. In §6.1 a number of discussed algorithms will be compared numerically. Not all algorithms can be compared, because not all numerical results are reported in the papers. In §6.2 some of the characteristics of the discussed algorithms are compared.

§6.1 Numerical results

Some of the papers discussed in Chapter 4, use the same problem instances to test the quality of the proposed algorithms. We also used these problem instances to test our the self-developed algorithm. The results from the tests of each algorithm are compared in order to give an indication of the quality of the algorithms.

Table 1 lists the algorithms for which all numerical results are available to perform the comparison.

Algorithm number	Algorithm name
1	A Tabu Search algorithm (§4.2)
2	Brandao & Mercers technique (§4.4)
3	Multi-Phase heuristic (§4.5)
4	Adaptive Memory Procedure (§4.3.3)
5	Genetic Algorithms (§4.6)
6	Self-developed algorithm (Chapter 5)

Table 1: list of algorithms that can be numerically compared.

There are 9 problem settings which are used. The first 5 settings come from problem 1-5 in Christofides, Mingozzi and Toth [29], 2 settings correspond to problem 11-12 of Christofides, Mingozzi and Toth [29] and 2 settings come from problem 11-12 in Fisher [30]. In each of these settings, the number of vehicles *m* changes according to Table 2. This results in 52 problem instances.

Problem number	Originated from:	Nr. Clients	*m*	*z**
1	Problem 1 of Christofides, Mingozzi and Toth	50	1,…,4	524,61
2	Problem 2 of Christofides, Mingozzi and Toth	75	1,…,7	835,26
3	Problem 3 of Christofides, Mingozzi and Toth	100	1,…,6	826,14
4	Problem 4 of Christofides, Mingozzi and Toth	150	1,…,8	1028,42
5	Problem 5 of Christofides, Mingozzi and Toth	199	1,…,10	1291,44
6	Problem 11 of Christofides, Mingozzi and Toth	120	1,…,5	1042,11
7	Problem 12 of Christofides, Mingozzi and Toth	100	1,…,6	819,56
8	Problem 11 of Fischer	71	1,…,3	241,97
9	Problem 12 of Fischer	134	1,…,3	1162,96

Table 2: characteristics of the base problems.

These problem instances are however for VRP problems. Therefore, the maximum driving time T corresponds to:

T^1 = [1.05 z*/m] and

T^2, = [1.1 z*/m],

where [x] is the value of x rounded to the nearest integer and z* is the value of a VRP solution to that problem obtained as in Rochat and Taillard [7] with an unspecified number of vehicles. This results in 104 problem instances.

Table 3 specifies the number of problem instances that are solved by the different MTVRP algorithms. For example, algorithm 1 finds a feasible solution for 5 out of 8 problem instances for base problem 1 and finds a feasible solution for 81 out of the 104 total problem instances.

Base problem nr.	# problem instances	Number of feasible problem instances for each algorithm:					
		1	2	3	4	5	6
1	8	5	6	4	6	5	1
2	14	11	12	12	12	8	3
3	12	8	11	10	12	8	0
4	16	12	14	13	15	8	1
5	20	19	18	14	19	9	5
6	10	9	9	8	9	5	7
7	12	8	9	11	11	10	5
8	6	3	4	4	5	3	1
9	6	6	6	6	6	6	3
Total	104	81	89	82	95	62	26

Table 2: number of feasible problem instances for each algorithm.

Table 3 shows that algorithm 4 (Adaptive Memory Procedure) finds a feasible solution for 95 problem instances. Algorithm 2 (Brandao & Mercers technique) also solves a big part of the problem instances (89 out of 104). The self-developed algorithm has the worst performance when looking at the number of problem instances solved (only 26 out of 104).

6.1.1 Comparing the quality of the feasible solutions

For only 3 algorithms the total driving time of the solutions is reported. The algorithms for which these results are available, are listed in Table 4. The total driving times are reported in Appendix B.

Algorithm number	Algorithm name
4	Adaptive Memory Procedure (§4.3.3)
5	Genetic Algorithms (§4.6)
6	Self-developed algorithm (Chapter 5)

Table 4: list of algorithms that present exact results for the feasible problem instances.

In Table 5 a summary of the results is presented. This table shows, for each algorithm, which percentage of feasible problem instances obtains better results (lower total driving time) by using that algorithm opposed to one of the other two algorithms. For example, the self-developed algorithm outperforms the GA in 53,85% of the instances and the Adaptive Memory Procedure in 7,69% of the instances. It should be noted, that only instances that provide a feasible solution for both algorithms are taken into account with this comparison.

Better \ Worse	*Self-developed algorithm*	*Genetic Algorithms*	*Adaptive Memory Procedure*
Self-developed algorithm	-	53,85%	7,69%
Genetic Algorithms	46,15%	-	3,23%
Adaptive Memory Procedure	92,31%	96,77%	-

Table 5: comparing the quality of the feasible solutions.

From Table 5 we can conclude that the Adaptive Memory Procedure generally produces results of a higher quality than the other two algorithms (92,31% and 96,77% of the problem instances have better results by using the Adaptive Memory Procedure [11]).

§6.2 Algorithm characteristics

The algorithms that are discussed in the previous paragraph have certain characteristics that make them more or less suitable for different practical circumstances. Some algorithm characteristics will now be discussed.

6.2.1 Homogeneous vs. heterogeneous

Each MTVRP algorithm can accommodate either a homogeneous fleet or a heterogeneous fleet. Only two of discussed algorithms accommodate a heterogeneous fleet. These two algorithms are the SPMU (§4.1) of Fleischmann [6] and the self-developed algorithm (Chapter 5). If an algorithm does not present the possibility for having a heterogeneous fleet, it could be changed in order to incorporate this possibility.

6.2.2 Greedy vs. non-greedy

A greedy algorithm is an algorithm that follows a problem solving heuristic of making the locally optimum choice at each stage with the hope of finding the global optimum. It iteratively makes one greedy choice after another, reducing each given problem into a smaller one. In other words, a greedy algorithm never reconsiders its choices. Greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not exhaustively consider all possible solutions. They can make commitments to certain choices too early which prevent them from finding the best overall solution later and they then get stuck in a local optimum.

A non-greedy algorithm looks at the entire problem simultaneously and at every stage of the algorithm there is a possibility of reconsidering previous choices. This way, is it possible to get out of a local optimum and find the global optimum.

Table 6 lists the discussed algorithms and whether these algorithms are greedy or non-greedy.

Algorithm name	Greedy / non- greedy
SPMU (§4.1)	Greedy
A Tabu Search algorithm (§4.2)	Non-greedy
Minmax Procedure (§4.3.1)	Non-greedy
Zhao's Extension (§4.3.2)	Non-greedy
Adaptive Memory Procedure (§4.3.3)	Non-greedy
Brandao & Mercers technique (§4.4)	Greedy/ Non-greedy
Multi-Phase heuristic (§4.5)	Non-greedy
Genetic Algorithms (§4.6)	Non-greedy
Insertion Heuristic Approach (§4.7)	Greedy
Route Linking Procedure (§4.8)	Greedy
Self-developed algorithm (Chapter 5)	Greedy

Table 6: list of algorithms and the respective algorithm type.

6.2.3 Performance of the self-developed algorithm

From the results can be concluded that the self-developed algorithm does not provide very good results for solving the MTVRP, in comparison to other available MTVRP algorithms. This is caused by a number of reasons:

- The standard savings algorithm, which is the base of the self-developed algorithm, performs relatively well with short computational time for the standard VRP. However, other algorithms are available that give better results when more computing time is available, like the algorithm proposed by Rochat and Taillard [7].
- The self-developed algorithm performs a type of "greedy" search. Once two routes are combines, this choice is never undone or changed. However, it isn't sure that this choice will provide an overall best solution.
- The choice for combining two routes is only based on the savings and not on some value that indicates how well the resulting route would fit in the fleet of vehicles.

Chapter 7

Conclusion

The MTVRP is very relevant because in practice many transportation companies have vehicles that can drive multiple routes on a day. However, not a lot of literature is available on this subject.

In this thesis, a number of algorithms are described that solve the MTVRP. This field of research is still in development and there exists no algorithm which is widely used. This thesis can be used as a summary for most MTVRP algorithms known up till know.

We can conclude that non-greedy algorithms work better than greedy algorithms. A greedy algorithm makes local choices based on some intuition regarding the characteristics of the MTVRP. These local choices tend to go to local optimum and not a global optimum and therefore do not present the best results for this type of problems. Non-greedy algorithms try to create a large sample of solutions and have a random component that result in solutions of higher quality.

The algorithm that presented the best results is the Adaptive Memory Procedure [11] described in § 4.3.3. This algorithm solves more test problem instances than any other MTVRP algorithm and the quality of the results of this algorithm is also higher than any of the other algorithms it was compared to.

The goal of each of the discussed MTVRP algorithms is to minimize the total driving time of all routes. However, in practice the total driving time is not the most important quality indicator of a MTVRP solution. Many real-life companies are interested in minimizing the number of used vehicles and not necessarily in minimizing the total driving time. It would therefore be better to look at the number of used vehicles of a solution when comparing MTVRP algorithms.

The self-developed algorithm provided poor results, which was somewhat anticipated. However, this algorithm can be used as a starting point for further research in the field of MTVRP algorithms. The self-developed algorithm also provides a solution for incorporating a heterogeneous fleet. This solution technique can also be used in other (MT)VRP algorithms that want to incorporate a heterogeneous fleet.

Appendix A:

References

[1] G. B. Dantzig and R.H. Ramser. The Truck Dispatching Problem. Management Science 6, 80–91. 1959.

[2] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12 #4, 568-581, 1964.

[3] B. E. Gillet and L. R. Miller. A Heuristic Algorithm for the Vehicle Dispatch Problem. Operations Research, 22:340-349, 1974.

[4] M. Desrochers, J. Desrosiers and M.Solomon. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research. N° 40, PP342-354, 1992.

[5] J. Desrosiers, Y. Dumas, M.Solomon, and F.Soumis. Time Constrained Routing and Scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser (Eds.), Network Routing, Volume 8 of Handbooks in Operations Research and Management Science. Amsterdam, The Netherlands. Pp 35 -139, 1995

[6] B. Fleischmann. The vehicle routing problem with multiple use of vehicles. Working paper, Fachbereich Wirtshaftswissenschaften, Universitat Hamburg, 1990.

[7] Y. Rochat and E. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. J. Heuristics 1, 147-167, 1995.

[8] M. Gendrbau, A. Hertz and G. Laporte. A tabu search heuristic for the vehicle routing problem. Mgmt Sci. 40, 1276-1290, 1994.

[9] E. Taillard. Parallel iterative search methods for vehicle routing problems. Networks 23,661-676, 1993.

[10] É. D. Taillard, G. Laporte and M. Gendreau. Vehicle Routing with Multiple Use of Vehicles. The Journal of the Operational Research Society, Vol. 47, No. 8. pp. 1065-1070, Aug. 1996.

[11] B. Golden, G. Laporte, E. Taillard. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. Comput. Oper. Res. 24 445–452, 1997.

[12] J. Brandao and A. Mercer. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. Eur J OplRes 100: 180-191, 1997.

[13] JCS Brandão and A. Mercer. The Multi-Trip Vehicle Routing Problem. The Journal of the Operational Research Society, Vol. 49, No. 8. pp. 799-805, Aug. 1998.

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

[15] P.C. Yellow, A computational modification to the savings method of vehicle scheduling, Oper. Res. Q.21 281–283, 1970.

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

[17] Q. Zhao, S. Wang, K. Lai and G. Xia. A vehicle routing problem with multiple use of vehicles. Advanced Modeling and Optimization 4(3), 21–40, 2002.

[18] A.M. Campbell and M.Savelsbergh. Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems. Transportation Science, Vol. 38, No. 3, pp. 369–378, August 2004.

[19] M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. 35 254–265, 1987.

[20] S. Salhi and G. Nagy. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. J. Oper. Res. Soc. 50 1034–1062, 1999.

[21] A. Olivera, O. Viera. Adaptive Memory Programming for the Vehicle Routing Problem with Multiple Trips. Computers & Operations Research Volume 34, Issue 1, Pages 28-47, January 2007.

[22] S. Salhi, and R. J. Petch. A GA Based Heuristic for the Vehicle Routing Problem with Multiple Trips. Journal of Mathematical Modelling and Algorithms, Volume 6, Number 4, p 591-613, December 2007.

[23] Justin Goodson. Vehicle Routing Problem with Multiple Trips. Working paper (2007).

[24] A. Caprara, M. Fischetti and P. Toth. A heuristic method for the set covering problem. Operations Research 47(5), 730–743, 1999.

[25]M. L. Fisher, Optimal Solution of Vehicle Routing Problems Using Minimum K-trees, Operations Research 42, 626-642, 1994.

[26] Genetic Algorithms:
http://en.wikipedia.org/wiki/Genetic_algorithm

[27] Tabu search:
http://en.wikipedia.org/wiki/Tabu_search

[28] Set Covering Problem:
http://en.wikipedia.org/wiki/Set_cover_problem

[29] 2-Opt and 3-Opt (in general k-Opt)
http://en.wikipedia.org/wiki/K-opt

[30] N. Christofides, A. Mingozzi, P. Toth and C. Sandi. Combinatorial Optimization. Wiley: Chichester, pp 313-318.

[31] M. L. Fischer. Optimal solution of vehicle routing problems using minimum K-trees. 1994, Opns Res. 42, 626-642.

[32] Travelling Salesman Problem:
http://en.wikipedia.org/wiki/Travelling_salesman_problem

[33] Vehicle Routing Problem:
http://en.wikipedia.org/wiki/Vehicle_routing_problem

[34] Search Tree:
http://en.wikipedia.org/wiki/Tree_(data_structure)

[35] Nearest neighbor algorithm
http://en.wikipedia.org/wiki/Nearest_neighbour_algorithm

Appendix B:

Results

Problem number	*m*	*T*	*Self-developed algorithm*	*Genetic Algorithms [22]*	*Adaptive Memory Procedure [11]*
1	1	551	x	546,28	524,61
	2	275	x	x	533
	3	184	x	x	x
	4	138	x	x	x
	1	577	567,03	547,14	524,2
	2	289	x	549,42	529,85
	3	192	x	560,26	552,68
	4	144	x	566,86	547,1
2	1	877	875,31	869,06	835,67
	2	439	x	865,48	843,13
	3	292	x	x	846,37
	4	219	x	856,77	838,71
	5	175	x	x	852,66
	6	146	x	x	x
	7	125	x	x	x
	1	919	875,31	869,73	844,26
	2	459	875,31	881,5	841,23
	3	306	x	869,11	836,77
	4	230	x	880,9	836,18
	5	184	x	883,29	844,28
	6	153	x	x	875,03
	7	131	x	x	872,64
3	1	867	x	845,33	830,77
	2	434	x	850,65	834,15
	3	289	x	x	831,16
	4	217	x	x	832,74
	5	173	x	x	851,47
	6	145	x	x	836,9
	1	909	x	845,33	829,69
	2	454	x	872,11	829,54
	3	303	x	869,48	829,45
	4	227	x	878	826,14

	5	182	x	901,3	833,15
	6	151	x	861,76	842,21
4	1	1080	x	1.064,06	1033,21
	2	540	x	1.065,86	1036,7
	3	360	x	x	1035,48
	4	270	x	x	1036,35
	5	216	x	x	1033,02
	6	180	x	x	1058,04
	7	154	x	x	x
	8	135	x	x	1064,97
	1	1131	1128,86	1.088,93	1041,77
	2	566	x	1.070,50	1047,02
	3	377	x	1.077,24	1038,98
	4	283	x	1.119,05	1038,88
	5	226	x	1.085,38	1044,09
	6	189	x	1.112,03	1033,02
	7	162	x	x	1062,89
	8	141	x	x	1064,56
5	1	1356	x	1.347,34	1323,13
	2	678	x	1.346,63	1341,41
	3	452	x	x	1317,58
	4	339	x	x	1330,63
	5	271	x	x	1329,17
	6	226	x	x	1337,05
	7	194	x	x	1340,91
	8	170	x	x	1327,09
	9	151	x	x	1342,5
	10	136	x	x	x
	1	1421	1390,32	1.340,44	1318,46
	2	710	1390,32	1.399,65	1314,09
	3	474	1390,32	1.409,37	1311,89
	4	355	1390,32	1.397,60	1338,52
	5	284	1390,32	1.411,19	1322,64
	6	237	-	1.377,07	1311,1
	7	203	-	1.394,73	1337,81
	8	178	-	x	1316,89
	9	158	-	x	1331,17
	10	142	-	x	1347,99
6	1	1094	1062,3	1.088,26	1073,34
	2	547	x	x	1073,07

	3	365	1062,3	x	1047,97
	4	274	x	x	x
	5	219	x	x	1049,81
	1	1146	1062,3	1.088,26	1044,35
	2	573	1062,3	1.110,10	1072,21
	3	382	1062,3	1.088,56	1043,17
	4	287	1062,3	x	1045,07
	5	229	1062,3	1.092,95	1045,85
7	1	861	830,33	819,97	820,96
	2	430	830,33	821,33	819,56
	3	287	x	826,98	819,6
	4	215	x	824,57	819,56
	5	172	x	x	845,37
	6	143	x	x	x
	1	902	830,33	819,97	819,56
	2	451	830,33	829,54	819,56
	3	301	830,33	851,16	819,56
	4	225	x	821,53	819,56
	5	180	x	833,85	824,78
	6	150	x	855,36	825,36
8	1	254	x	x	241,97
	2	127	x	x	252,13
	3	85	x	x	x
	1	266	264,61	254,07	243,25
	2	133	x	254,07	241,97
	3	89	x	256,53	260,63
9	1	1221	1191,17	1.190,21	1171,16
	2	611	x	1.194,24	1175,3
	3	407	x	1.199,86	1166,18
	1	1279	1191,17	1.183,00	1173,07
	2	640	1191,17	1.199,64	1173,18
	3	426	x	1.215,43	1167,43

Appendix C:

Savings algorithm

The Clarke and Wright savings algorithm [2] is one of the most known heuristic for the basic VRP. It was developed by Clarke and Wright in 1964 and it applies to problems for which the number of vehicles is not fixed (it is a decision variable), and it works equally well for both directed and undirected problems. When two routes $r_1 = (0,...,i,0)$ and $r_2 = (0,j,...,0)$ can feasibly be merged into a single route $r^* = (0,...,i,j,...,0)$, a distance saving $s_{ij} = c_{i0} + c_{0i} - c_{ij}$ is generated. (where c_{ij} is the distance from location i to location j)

The Clarke and Wright savings algorithm [2] works at follows:

Step 1: *Savings Computation*

1) Compute the savings $s_{ij} = c_{i0} + c_{0i} - c_{ij}$ for $i,j=1,...,n$ and $i \neq j$.
2) Create n shuttle routes $(0,i,0)$ for $i = 1,...,n$.
3) Order the savings in a non-increasing fashion.

Step 2: *Best Feasible Merge*

Starting from the top of the savings list, execute the following:

1) Given a saving s_{ij}, determine whether there the following two routes can feasibility be merged:
   - One starting with $(0,j)$.
   - One ending with $(i,0)$.
2) If the two routes can be merged, these two routes are combined by deleting the routes starting with $(0,j)$ and ending with $(i,0)$ and introducing a new combined route with (i,j) in the middle.