universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

"The Time-Dependent Vehicle Routing Problem"

Verfasserin

Irena Ilić

angestrebter akademischer Grad

Magistra der Sozial- und Wirtschaftswissenschaften
(Mag. rer. soc. oec.)

Wien, Juni 2012

Studienkennzahl lt. Studienblatt:	A 157
Studienrichtung lt. Studienblatt:	Diplomstudium Internationale Betriebswirtschaft
Betreuer:	O. Univ.-Prof. Dipl.-Ing. Dr. Richard F. Hartl

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ACS	Ant Colony System
CVRP	Capacitated Vehicle Routing Problem
FCD	Floating Car Data
FIFO Property	First-In-First-Out Property
GA	Genetic Algorithm
LB	Lower Bound
MACS	Multi Ant Colony System
MACS-IH	Multi Ant Colony System - Insertion Heuristic
MDL	Minimum Delay Metric
MILP	Mixed Integer Linear Programming
TDVRP	Time-Dependent Vehicle Routing Problem
TDVRPTW	Time-Dependent Vehicle Routing Problem with Time Windows
TDTSP	Time-Dependent Traveling Salesman Problem
TS	Tabu Search
TSP	Traveling Salesman Problem
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPPD	Vehicle Routing Problem with Pickup and Delivery
VRPTW	Vehicle Routing Problem with Time Windows

1. Introduction

The vehicle routing problem (VRP) is a combinatorial optimization problem dealing with the distribution of goods between a depot and a set of customers. Various applications of this problem can be found in the real world, e.g. mail delivery, solid waste collection, street cleaning, etc. (Toth and Vigo (2002)).

Most vehicle routing models assume constant travel times throughout the day. In reality, however, travel times fluctuate because of two reasons: the variation in travel time may result from predictable events such as congestion during rush hours or from unpredictable events such as accidents (Ichoua et al. (2003)). The time-dependent vehicle routing problem (TDVRP) takes the first aspect into account by assuming that travel times depend on the time of the day.

This diploma thesis gives an overview of the TDVRP and presents the results of an experimental study.

The first part of this thesis introduces the VRP and different solution methods. This is followed by a detailed description of the TDVRP.

The second part presents an algorithm based on tabu search to solve the capacitated VRP (CVRP). In the next step, the best solutions of the CVRP are evaluated with five time-dependent scenarios, each representing a different degree of time-dependency. Afterwards, the original algorithm is adapted to solve the TD-CVRP. Finally, the results of the whole implementation are presented and analyzed. The last chapter provides a conclusion of the thesis.

2. The Vehicle Routing Problem

Since its introduction in 1959, the VRP has been studied extensively and many variants have been discussed. The following chapter gives an overview of the problem. It starts with a brief definition of the basic VRP. This is followed by a description of the most important variants.

2.1 Definition

This section is based on the work of Toth and Vigo (2002) who provide an extensive survey of the VRP.

The VRP deals with the distribution of goods between a depot and a set of customers who have known demands. The distribution is performed by a fleet of vehicles which are based at the depot. The aim of this problem is to minimize the total cost of a set of routes such that all constraints are satisfied. The VRP is known to be NP-hard, meaning that it is unlikely to be solved within polynomial time (Lenstra and Rinnooy Kan (1981)).

The VRP is described on a graph which represents the road network. It consists of road sections, customer locations, depots and road junctions. Road sections are represented by arcs which are either directed or undirected, depending on whether they can be traversed in only one direction (e.g. one-way streets) or in both directions. Each arc is associated with a certain cost which corresponds to its length. Customer locations, depots and road junctions are represented by nodes.

The basic version of the VRP is the CVRP. In this problem each vehicle has a limited capacity and each customer is associated with a fixed demand which is not allowed to be divided. The objective is to determine a set of routes with minimum cost such that i) each vehicle route originates and terminates at the depot, ii) each customer is visited exactly once by one vehicle route and iii) the sum of all customer demands corresponding to one vehicle route does not exceed the capacity limit.

A typical CVRP is presented in Figure 1. It shows a single depot and eight customers who are visited by three vehicles.

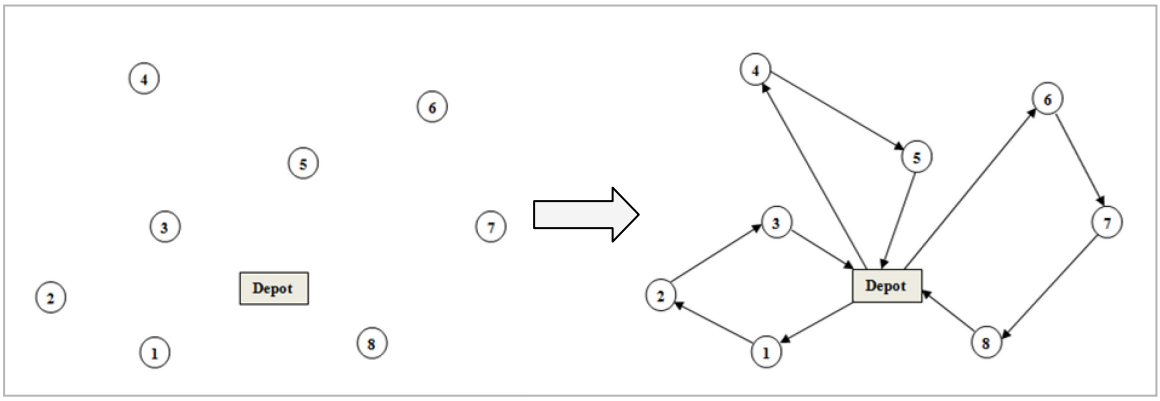


Figure 1: CVRP solution

2.2. Variants of the VRP

The following section describes five VRP variants which have been widely studied in the past. The section is based on Toth and Vigo (2002).

2.2.1 The Distance-Constrained VRP

The distance-constrained VRP imposes a limit on the total distance of a route. The objective consists in minimizing the total length of the routes such that i) each route originates and terminates at the depot, ii) each customer is visited exactly once by one vehicle route and iii) the limit on the total distance is not exceeded.

2.2.2 The VRP with Time Windows

The VRP with Time Windows (VRPTW) restricts the service at customer i by the time window $[e_i, l_i]$, where e_i represents the earliest start of service and l_i represents the latest start of service at customer i. The duration of the service at customer i is denoted by s_i.

Time windows can be either soft or hard. Soft time windows allow time window violations at a certain penalty cost. Hard time windows, on the other hand, do not accept any violation. The objective of the VRPTW is to minimize the total cost over all routes such that i) each route starts and ends at the depot, ii) each customer is visited exactly once by one route, iii) the vehicle capacity is satisfied iv) the service of customer i begins within time window $[e_i, l_i]$ and v) lasts for s_i time instants.

2.2.3 The VRP with Backhauls (VRPB)

The VRP with Backhauls (VRPB) is concerned with the delivery and the collection of goods. There are two groups of customers involved in this problem: linehaul customers demand the delivery of goods whereas backhaul customers require the collection of goods. The objective of the VRPB is to determine a least-cost set of routes such that i) each route begins and ends at the depot, ii) each customer is visited exactly once by one route, iii) the sum of the demands of linehaul and backhaul customers belonging to one route does not exceed separately the vehicle capacity and iv) the set of linehaul customers is always visited first.

2.2.4 The VRP with Pickup and Delivery

The VRP with Pickup and Delivery (VRPPD) is based on the assumption that each customer i is associated with a quantity d_i which has to be delivered and a quantity p_i which has to be picked up. For each customer i, O_i indicates the origin of the delivery demand and D_i signifies the destination of the pickup demand. A precedence constraint clarifies that the delivery is always performed prior to the pickup. The objective is to design a set of least-cost routes such that i) each route originates and terminates at the depot, ii) each customer is visited exactly once by one route, iii) the current load of the vehicle is nonnegative and does not exceed capacity limit C, iv) customer O_i, if different from the depot, must be visited before customer i and v) customer D_i, if different from the depot, must be visited after customer i.

3. Solution methods

Since the introduction of the VRP, numerous exact and heuristic solution methods have been developed. A typical exact method is the branch and bound algorithm which determines the optimal solution of a VRP. A major drawback of this method is its immense computational effort, i.e. it can only tackle small problems. Larger problems are solved with heuristic methods which provide good quality solutions within reasonable computation times (Laporte and Semet (2002)). However, there is no guarantee that the obtained solutions are optimal.

Heuristics can be classified into classical heuristics and metaheuristics. The first type performs a limited search of the solution space; the final solution is in most cases a local minimum. Metaheuristics, on the other hand, enable a deeper search in more promising regions and lead to better solutions than classical heuristics (Laporte and Semet (2002)).

The current chapter gives an overview of different solution methods discussed in the VRP literature. The first part provides a description of several classical heuristics; the second part presents the most important metaheuristics.

3.1 Classical Heuristics

According to Laporte and Semet (2002), classical heuristics can be further divided into three categories: constructive heuristics, two-phase heuristics and improvement heuristics. Constructive and two-phase heuristics create the initial solution of a VRP whereas improvement heuristics focus on further improvement of the initial solution.

3.1.1 Constructive Heuristics

Constructive heuristics are based on the principle of inserting customers into routes. This is either done in a sequential or in a parallel manner. Sequential approaches build routes one after another whereas parallel approaches construct several routes simultaneously. Typical constructive heuristics are the savings heuristic of Clarke and Wright (1964) and the sequential insertion heuristic of Mole and Jameson (1976).

3.1.1.1 The Savings Heuristic

The savings heuristic is based on the principle of merging two routes if it leads to a cost saving (see Figure 2).

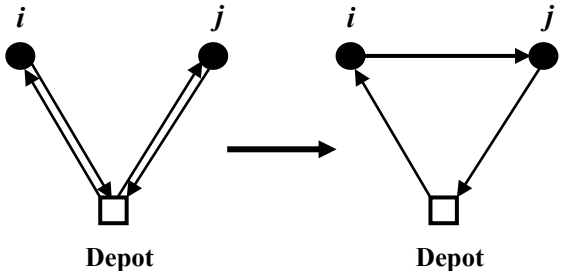


Figure 2: Savings Heuristic
(based on Bräysy and Gendreau (2005a))

Based on the formulations of Laporte et al. (2000), the savings heuristic is described as follows:

Let us consider Figure 2. At the beginning, the customers i and j belong to two separate routes. The total cost TC_{orig} of this solution is defined as follows:

$$TC_{orig} = c_{0i} + c_{i0} + c_{0j} + c_{j0} \quad (1)$$

In expression (1), c_{0i} and c_{i0} represent the costs of the two links between customer i and the depot whereas c_{0j} and c_{j0} represent the costs of the two links between customer j and the depot. Merging both routes would result in a total cost $TC_{new}$where c_{ij} represents the cost of link (i, j):

$$TC_{new} = c_{0i} + c_{ij} + c_{j0} \quad (2)$$

A merge results in a saving if TC_{new} is less than TC_{orig}. Consequently, saving s_{ij} is defined in the following way:

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \text{ for } i, j = 1, \dots, n \text{ and } i \neq j \quad (3)$$

The next paragraph gives an outline of the different steps performed during the savings heuristic (parallel approach):

The first step consists in computing the savings between all customer pairs and sorting them in decreasing order. Afterwards, each customer *i* is assigned to a separate vehicle route $(0, i, 0)$.

In the second step, the customer pair with the highest saving s_{ij} is selected and checked for a possible merge. A merge is only performed if i) the customers *i* and *j* belong to two different routes, ii) the customers *i* and *j* are located in the front or end position of their routes and iii) the merge does not violate any constraints. Provided that these conditions are satisfied, the routes of customers *i* and *j* are combined so that customer *j* is immediately visited after customer *i*. Then the next savings pair is selected and the same procedure is applied. This is done until no more feasible savings are left.

3.1.1.2 Sequential insertion heuristic

As the name already implies, this heuristic inserts unrouted customers in a sequential manner. According to Cordeau et al. (2007), the selection and insertion of an unrouted customer *k* is based on the following measure:

$$\alpha(i, k, j) = c_{ik} + c_{kj} - \lambda c_{ij} \tag{4}$$

In expression (4), c_{ik}, c_{kj} and c_{ij} represent the costs between the corresponding customers whereas λ is a user-controlled parameter. The cost $\alpha(i, k, j)$ describes the extra distance which arises from the insertion of customer k between the two adjacent customers *i* and *j*.

3.1.2 Two-phase heuristics

Two-phase heuristics consist of two components: clustering and routing. During the clustering phase customers are divided into routes; the customer sequence within each route is determined during the routing phase (Cordeau et al. (2007)). Depending on which phase is performed first, we can distinguish between cluster-first, route-second methods and route-first, cluster-second methods.

3.1.3 Improvement Heuristics

Improvement heuristics perform several route changes to create a neighborhood of solutions. The incumbent solution is replaced by a neighborhood solution if it leads to an improvement in the total cost. The route changes can be done within one route ('intra-route') or between two different routes ('inter-route').

Bräysy and Gendreau (2005a) describe several neighborhood operators:

The 2-opt exchange operator replaces two edges by two other edges in the same route: the edges $(i, i+1)$ and $(j, j+1)$ are replaced by the edges (i, j) and $(i+1, j+1)$. In this way the direction of the nodes between $i+1$ and j is reversed.

The relocate operator chooses a node from one route and inserts it into another: the edges $(i-1, i)$ $(i, i+1)$ and $(j, j+1)$ are replaced by $(i-1, i+1)$, (j, i) and $(i, j+1)$.

The exchange operator swaps two nodes between two routes. The edges $(i-1, i)$, $(i, i+1)$, $(j-1, j)$ and $(j, j+1)$ are replaced by $(i-1, j)$, $(j, i+1)$, $(j-1, i)$ and $(i, j+1)$.

The cross exchange operator first removes the two edges $(i-1, i)$ and $(k, k+1)$ from route 1 and then $(j-1, j)$ and $(l, l+1)$ from route 2. The segments $i-k$ and $j-l$ may consist of an arbitrary number of nodes. These nodes are swapped by creating the new edges $(j, j+1)$, $(i-1, j)$, $(l, k+1)$, $(j-1, i)$ and $(k, l+1)$. In this case, the route directions are not affected.

3.2 Metaheuristics

Metaheuristics combine different mechanisms to find solutions of good quality. In contrast to classical heuristics, they accept non-improving solutions during the search to overcome local optima. Usually, metaheuristics produce better results than classical heuristics at the expense of increased computation times (Gendreau et al. (2002)). The following sections present different metaheuristics for solving the VRP.

3.2.1 Simulated Annealing

Following Bräysy and Gendreau (2005b), simulated annealing (SA) is a stochastic relaxation technique based on the idea of heating a solid to a high temperature and then cooling it down so that it ends up in a low energy condition. In this case, a neighboring solution S' is accepted as new solution if $\Delta \leq 0$. The variable Δ measures the difference between the cost of the potential new solution S' and the cost of the current solution S: $\Delta = C(S') - C(S)$. If $\Delta > 0$, the acceptance depends on the probability $e^{-\frac{\Delta}{T}}$ where T describes the current temperature. During the process, the temperature is gradually cooled down.

3.2.2 Tabu Search

Bräysy and Gendreau (2005b) describe tabu search (TS) as a metaheuristic which moves at each iteration from the current solution to best neighborhood solution. It was introduced by Glover in 1986. It allows non-improving solutions to overcome local minima. In order to avoid recently visited solutions, certain solution attributes are declared forbidden or 'tabu' for a number of iterations. The tabu status of a move is ignored if satisfying an aspiration criterion, e.g. if a tabu solution is better than any solution achieved so far. More details about the TS algorithm will be provided in Chapter 5.

3.2.3 Genetic Algorithm

Gendreau et al. (2002) describe genetic algorithms (GA) as randomized global search techniques which mimic natural evolution. A population of chromosomes is created where each chromosome encodes a solution to a particular instance. In order to create new chromosome generations, different operators, such as reproduction and mutation, are applied.

3.2.4 Ant Algorithm

According to Donati et al. (2008), ant algorithms are based on the following idea: When ants are on their way to a food source, they deposit pheromones along their path. The more pheromones are laid on a path, the more attractive it becomes to other ants. In this way shorter paths are strengthened. Further details about the ant algorithm of Donati et al. (2008) will be provided in Chapter 4.

3.2.5 Variable Neighborhood Search

Hansen and Mladenović (2001) describe variable neighborhood search (VNS) as a metaheuristic which changes the neighborhood systematically. The authors describe the individual steps of the VNS in the following way:

VNS starts with the selection of a set of neighborhood structures and the creation of an initial solution. While a stopping condition is not met, the following steps are repeated:

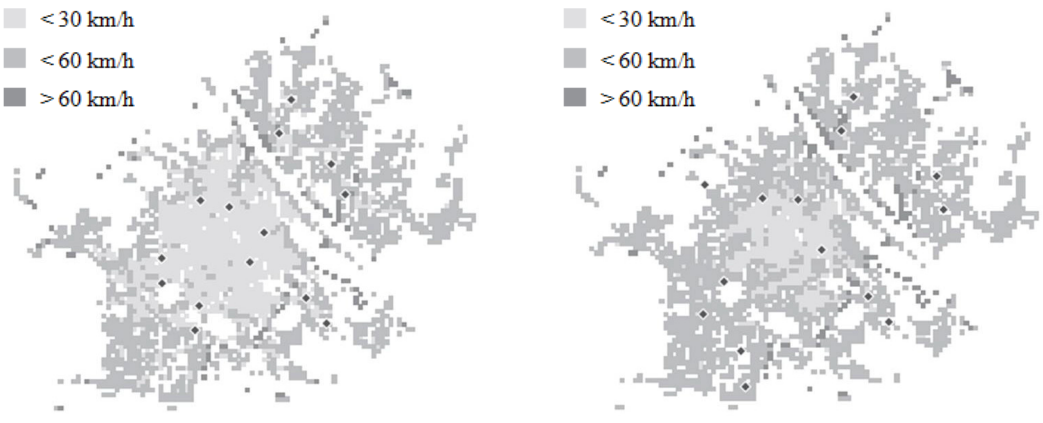
In the first step, the *shaking* procedure selects a random solution from the first neighborhood. After that, a *local search* procedure tries to improve the random solution. Finally, the *move or not* step compares the improved and the incumbent solution. In case the new solution is better, it replaces the incumbent solution and the search returns to the first neighborhood, otherwise it goes to the next neighborhood.

4. The Time-Dependent Vehicle Routing Problem

The following chapter presents a detailed overview of the TDVRP. First of all, the problem is introduced. In the next section, a short summary of related time-dependent problems is provided. This is followed by a description of different time-dependent travel time models. The following section is devoted to the problem formulation of the TDVRP. After that, the First-in First-out (FIFO) property is presented. The chapter concludes with a review covering literature from 1991 until 2012.

4.1 Introduction

The basic VRP relies on the assumption that travel times remain constant throughout the whole planning horizon. In reality, however, travel times may vary during the day. The variation in travel times may result from predictable events such as congestion during rush hours or from unpredictable events such as accidents (Ichoua et al. (2003)). The TDVRP takes these predictable events into account by assuming time-dependent travel times, i.e. the travel time between two locations depends on the distance between these two points and on the time of the day (Malandraki and Daskin (1992)). Figure 3 underpins this statement by showing two typical traffic situations in the city of Vienna. The left map depicts the average travel speeds at 8:30 am whereas the right map illustrates the average travel speeds at 11:30 pm.



Travel speeds at 8:30 am *Travel speeds at 11:30 pm*

Figure 3: Traffic situation in Vienna
(Schmid and Dörner (2010))

At 11:30 pm, the light grey area corresponding to the lowest travel speed is concentrated in the center of the city. However, this changes if we consider the traffic situation at 8:30 am. In this case, the light grey area expands such that the surrounding districts are also affected by lower travel speeds. This induces longer travel times for the same distance in the affected areas.

The two speed maps show that the assumption of constant travel times throughout the day is far from reality. According to Ichoua et al. (2003), the optimal solution of a VRP which assumes constant travel times may be suboptimal or infeasible for the time-dependent problem. For example, time windows might be missed because of late arrivals, transportation costs might increase because of overtime hours, etc. The TDVRP captures this aspect by taking predictable travel time variations into account. The aim of the problem is to construct feasible routes which minimize the total travel time and offer a higher reliability.

Furthermore, travel times are influenced by many other factors such as the direction of the trip. Also the day of the week or the season of the year might play a major role (Eglese et al. (2006)).

Time-dependent travel times (or travel speeds) are either simulated or derived from traffic monitoring systems. When analyzing the data from traffic monitoring systems, one can observe that most travel times follow a certain pattern during the day. This helps to predict future traffic situations. Especially travel time variations during rush hour congestion show a high predictability. This is proven by Eglese et al. (2006) who mention a study from the United Kingdom which examines tachograph analysis and data from a vehicle tracking and tracing system. It shows that on one road segment of a motorway the same commercial vehicle speeds vary in one week from 5 mph (at 8:45 am on a Monday) to 55 mph (at 8:15 pm on a Wednesday). When comparing the observed speeds over a period of ten weeks, the variation in speed for the same time of day and day of the week is less than 5%. According to Eglese et al. (2006), this small variation shows that travel speeds are highly predictable and can be forecasted for any road length and any time of the day.

4.2 Time-Dependency in related problems

The literature relating to the TDVRP is rather scarce when compared to other VRP variants. However, Ichoua et al. (2003) mention three related time-dependent problems which have been extensively studied:

The first problem mentioned is the time-dependent shortest path problem which was introduced in the late 1950s, e.g. Ford and Fulkerson (1958). It belongs to the earliest models dealing with time-dependency.
Marguier and Ceder (1984) consider a time-dependent path choice problem. In this case, a set of passengers is distributed in a transportation network consisting of overlapping bus routes with common stops. The passenger waiting times are represented by time-dependent distributions.
The last problem mentioned is the time-dependent traveling salesman problem (TSP) (e.g. Miller et al. (1964)). The problem concerns the determination of a least-cost route which visits each city exactly once. The travel cost between each city is time-dependent.

4.3 Classification of time-dependent models

According to Ichoua et al. (2003), TDVRP models can be classified into four main categories based on the type of travel time function:

The first category refers to models based on simple travel time functions. In this case, multiplier factors are used to integrate time-dependency.
Other TDVRP models assume discrete travel times (e.g. Malandraki and Daskin (1992)). For this purpose, the time horizon is partitioned into discrete time intervals and the travel time is defined as a step function of the starting time at the origin node. As the travel time changes appear in the form of jumps, it might happen that the FIFO property is not satisfied if the travel time in a consecutive time interval decreases. The FIFO property guarantees that if two vehicles depart from the same origin node for the same destination node, the one which started earlier will also arrive earlier. Further details about the FIFO property will be provided in Section 4.5.
Ichoua et al. (2003) also mention TDVRP models which are based on continuous travel time functions. According to the authors, it is very complicated to model this kind of

function because of its complexity. In order to make these models more tractable, many simplifications are required.

Finally, Ichoua et al. (2003) mention travel time functions based on Markovian formulations, e.g. Psaraftis and Tsitsiklis (1993) who solve a shortest path problem in a stochastic and dynamic setting.

4.4 Problem Formulation

In the TDVRP a fleet of m identical vehicles transports goods to a set of n customers. Each route originates and terminates at a single depot. Each customer is associated with a fixed demand and requires one visit by one vehicle. Several additional constraints can be imposed such as a capacity limit, time windows, etc. The objective is to minimize the total travel time. The time horizon is divided into p time intervals $T_1, T_2, \ldots, T_p$, each corresponding to a different time-dependent travel time or a time-dependent travel speed. Each interval T_k is restricted by a lower and an upper bound $T_k =]\underline{t_k}, \overline{t_k}]$. A simple way to divide the time horizon is to assume three time intervals T_1, T_2 and T_3 (e.g. Ichoua et al. (2003)) where the first and the third refer to morning and evening rush hours with lower travel speeds and the second one corresponds to higher travel speeds (see Figure 4).

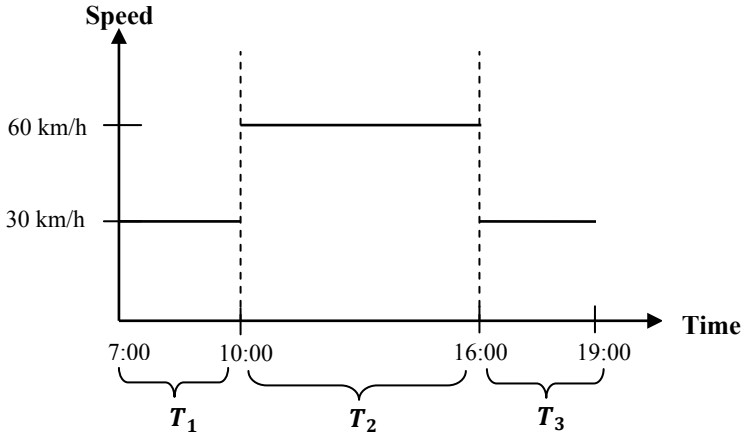


Figure 4: Model with three time intervals

After the definition of the time horizon, a travel time function needs to be specified. For this purpose we consider Figure 4. Let us suppose that a vehicle has to travel a distance

d_{ij} = 10 km, starting at 9:45. The associated travel speed v_{ij} would be 30 km/h. From this we can calculate its travel time c_{ij}:

$$c_{ij} = \frac{d_{ij}}{v_{ij}} \tag{5}$$

The resulting travel time would be 20 minutes, i.e. the arrival time would be 10:05. However, the travel time function ignores the speed change at time instant 10:00. The increase of travel speed in the next interval might lead to a violation of the FIFO property. This issue will be discussed in the following section.

4.5 The FIFO property

The FIFO property was formulated by Ichoua et al. (2003), other authors referred to it as "non-passing condition" (e.g. Ahn and Shin (1991)). The property guarantees that a vehicle traveling from node *i* to node *j* at time *T,* arrives earlier than any other identical vehicle traveling the same distance at a later time. Ichoua et al. (2003) criticize that several TDVRP models do not satisfy the FIFO property; Example 1 describes how this can happen:

Example 1: The time horizon of the following problem is divided into two time intervals $T_k =]\underline{t_k}, \overline{t_k}]$, where $\underline{t_k}$ and $\overline{t_k}$ represent the lower and upper bound of time interval T_k. The first time interval T_1 lasts from 8:00 to 10:00 o'clock and the second interval T_2 ranges between 10:00 and 12:00 o'clock. The average speeds are v_{T_1}= 20 km/h and v_{T_2} = 40 km/h. Two identical vehicles, $V1$ and $V2$, are supposed to travel a distance of d_{ij} = 40 km from origin i to destination j (see Figure 5). In this example the travel speed is assumed to remain constant over the entire length of link (i, j).

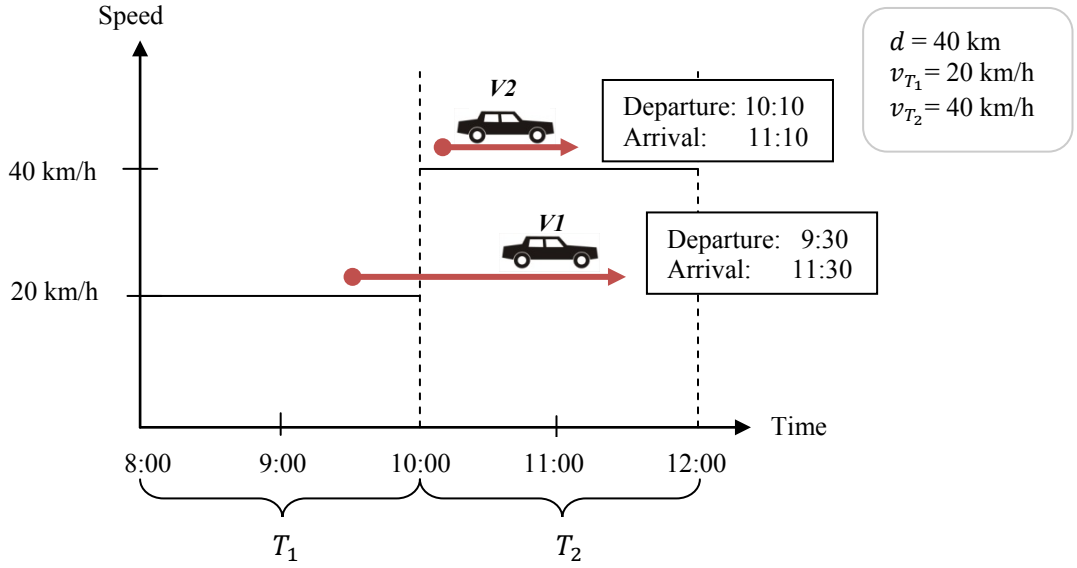


Figure 5: Passing situation

As shown in Figure 5, $V1$ departs at 9:30 and arrives at 11:30. As the departure time belongs to T_1, the related speed is v_{T_1}= 20 km/h. $V2$ starts at 10:10 and therefore the associated speed is v_{T_2} = 40 km/h. The corresponding arrival time is 11:10. As we can see, $V2$ arrives twenty minutes earlier than $V1$ (11:10 < 11:30) because of the speed increase in the second time interval. Thus, the FIFO property is not satisfied because the two vehicles pass each other (see Figure 6).

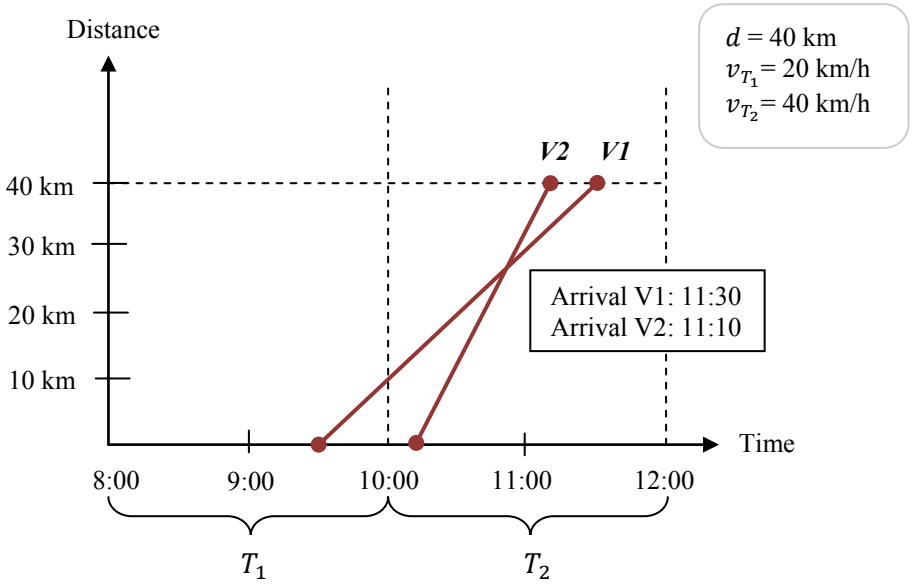


Figure 6: Travel speeds remain constant over the entire length of the link

According to Ichoua et al. (2003), passing situations occur when travel times are represented by step functions of the starting time at the origin node. Therefore travel times or travel speeds 'jump' at time interval boundaries, resulting in a possible passing situation if the travel time in the next interval decreases (or the travel speed increases).
For example, the model of Malandraki and Daskin (1992) does not satisfy the FIFO property because it uses step functions for the travel time distribution. The authors smooth the travel time function by allowing vehicles to wait at customer locations. This suggestion is criticized by Ichoua et al. (2003) who state that this results in useless waiting times which are not affordable under real conditions. To avoid this problem, Ichoua et al. (2003) model travel speed as a step function of the time of the day. They develop a simple travel time calculation procedure which adjusts the travel speed whenever a time interval boundary is crossed. They state that the resulting travel time function satisfies the FIFO property. Table 1 presents the travel time calculation procedure of Ichoua et al. (2003).

1. set t to t_0 $\quad t_0 \in T_k =]\underline{t_k}, \overline{t_k}]$

   set d to d_{ij}

   set t' to $t + (d/v_{T_k})$

2. **while** $(t' > \overline{t_k})$ **do**

   2.1. $d \leftarrow d - v_{T_k}(\overline{t_k} - t)$

   2.2. $t \leftarrow \overline{t_k}$

   2.3. $t' \leftarrow t + \left(d \,/\, v_{T_{k+1}}\right)$

   2.4. $k \leftarrow k + 1$

3. **return** $(t' - t_0)$

Table 1: Travel time calculation procedure of Ichoua et al. (2003)

In order to explain the calculation procedure, we continue Example 1. Now the new arrival time of $V1$ will be calculated:

Example 2: We recall that $V1$ departs from origin i at t_0 = 9:30; the projected arrival time at this moment is t' = 11:30 (see Step 1 in Table 2). $V1$ travels at a speed of v_{T_1} = 20 km/h until it reaches the time interval boundary at 10:00. At this point we have to adjust the travel speed. Now the remaining distance is 30 km and the new travel speed is v_{T_2}= 40 km/h (see Step 2 in Table 2). Therefore the new arrival time is t' = 10:45. In short, $V1$ departs from i at t_0 = 9:30 and arrives at j at t' = 10:45; the total travel time amounts to 75 minutes (see Step 3 in Table 2).

Step	**Formula**	**Initialization**	
0	$k = 1$		
1	$t = t_0$	t = 9:30	
	$d = d_{ij}$	d = 40 km	
	$t' = t + (d/v_{T_k})$	t' = 9:30 + (40 km / 20 km/h) t' = 11:30	
		Iteration 1	
2	while $t' > \overline{t_k}$ do	(11:30 > 10:00)	(10:45 < 12:00) STOP!
2.1	$d = d - v_{T_k}(\overline{t_k} - t)$	d = 40 km - 20 km/h (10:00 – 9:30) = 30 km	
2.2	$t \leftarrow \overline{t_k}$	t = 10:00	
2.3	$t' \leftarrow t + (d / v_{T_{k+1}})$	t' = 10:00 + (30 km / 40 km/h) t' = 10:45	
2.4	$k \leftarrow k + 1$	$k = 2$	
3	return $(t' - t_0)$		(10:45 – 9:30) = 75 min

Table 2: Travel time calculation for Vehicle 1

It is shown that $V1$ arrives earlier than $V2$. Thus the FIFO property is satisfied. Note that the arrival time of $V2$ requires no adjustment as both its departure time and arrival time belong to the same interval. Figure 7 illustrates Example 2.

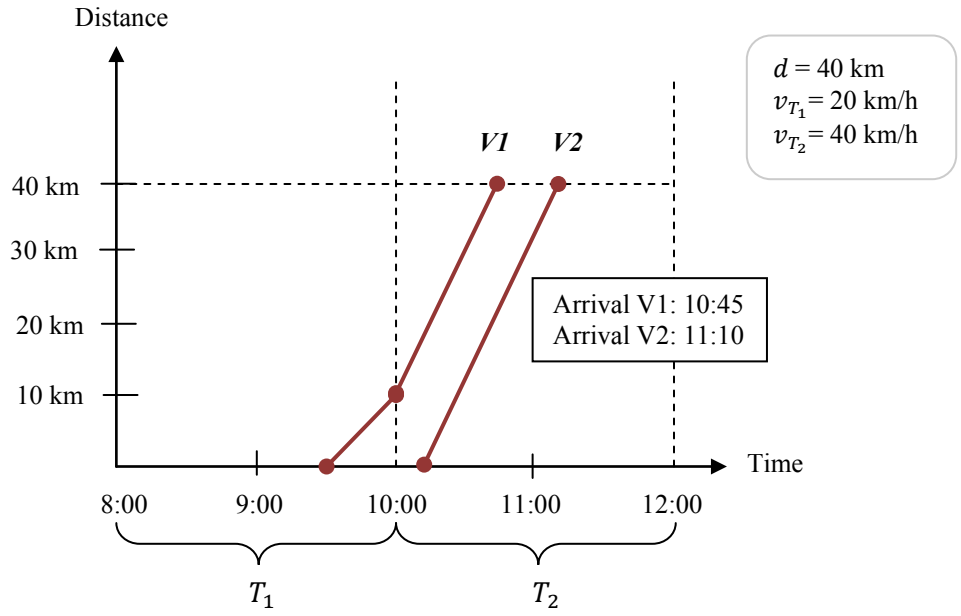


Figure 7: Travel speeds are adjusted when crossing the time interval boundary

4.6 Literature Review

VRP variants like the CVRP or the VRPTW have been widely studied during the last decades. In contrast to this, papers discussing the TDVRP were published not until the early 1990s. The late introduction can be related to three reasons:

The first reason concerns the fact that the TDVRP is much harder to model and solve than other VRPs (Ichoua et al. (2003)). When modeling a TDVRP, we need to determine a travel time function which complies with the FIFO property, i.e. the function must be capable of handling the transitions between the different time intervals. Furthermore, algorithms which are usually used to solve the VRP, require essential adaptations to handle time-dependency. For example, it becomes more complicated to evaluate local route changes. This is because a potential speed change might have an impact on all consecutive links (Fleischmann et al. (2004)).

The second reason is due to the fact that computers of the past were not able to store and process so much data. This problem is not relevant anymore as computer systems have made a lot of progress in the last decade.
The final reason relates to the lack of time-dependent data in the past. Nowadays, however, traffic monitoring systems and tracking devices provide a lot of data which can be used for vehicle routing.

The following sections provide a chronological overview of the TDVRP literature from 1991 until 2012.

4.6.1 Ahn and Shin (1991)

Ahn and Shin (1991) discuss a vehicle routing and scheduling problem with time windows and time-varying congestion. The aim of the problem is to minimize the total time traveled. The following paragraph briefly describes the model:

The service at customer *j* is restricted by the time window $[e_j, l_j]$ and starts at $t_j = max\{e_j, A_{ij}(t_i)\}$. $A_{ij}(y)$ stands for the arrival time at customer *j* and t_i denotes the service start time at customer *i*. Vehicles leave the depot at e_0 and have to return until l_0. A route is considered feasible if each customer is visited within his time window. Ahn and Shin (1991) define the arrival time at customer *j* as follows:

$$A_{ij}(y) = y + s_i + \tau_{ij}(y + s_i) \tag{6}$$

$A_{ij}(y)$ indicates the arrival time at node *j* via arc (i, j) if the service at node *i* starts at time instant *y*. The travel time on the shortest/fastest link at time x is represented by $\tau_{ij}(x)$ and the duration of service at customer *i* is denoted by s_i.
Next, Ahn and Shin (1991) formulate the non-passing property (= FIFO property): for any two nodes *i* and *j* and any two service start times *x* and *y,* the following condition must be satisfied:

$$x > y, A_{ij}(x) > A_{ij}(y) \tag{7}$$

The condition in (7) guarantees that departing earlier from node *i* leads to an earlier arrival at node *j*. Based on this property, $A_{ij}(\cdot)$ is a strictly increasing function of the start time.

In the next step, Ahn and Shin (1991) formulate the effective latest service start time. This is done with the inverse function $A_{ij}^{-1}(x)$ which represents the departure time at node *i* such that node *j* can be reached at time *x*. The effective latest service start time d_i at node *i* on a feasible route $(0, i, i+1, \dots, m, 0)$ is defined as follows:

$$d_i = min\{l_i, A_{i,i+1}^{-1}(d_{i+1})\}, \qquad 0 \leq i \leq m-1 \tag{8}$$

$$d_m = min\{l_m, A_{m,0}^{-1}(l_0)\} \tag{9}$$

The authors develop three feasibility conditions which are based on the effective latest service time. These conditions enable a faster feasibility check if time-varying travel times are assumed: The first feasibility condition deals with the insertion of unvisited customers into routes; the second feasibility condition refers to the combination of two different routes; and the third feasibility condition considers customer exchanges.

Finally, Ahn and Shin (1991) perform several experiments with problem sets consisting of 50 to 200 customers. The time-dependent congestion function $\tau_{ij}(\cdot)$ for each pair of nodes (i, j) is represented by a horizontal line with a single peak (see Figure 8). The peak times, congestion durations and the slopes of the congestion functions are based on random values. In order to satisfy the non-passing condition, the authors set the slopes between zero and one.

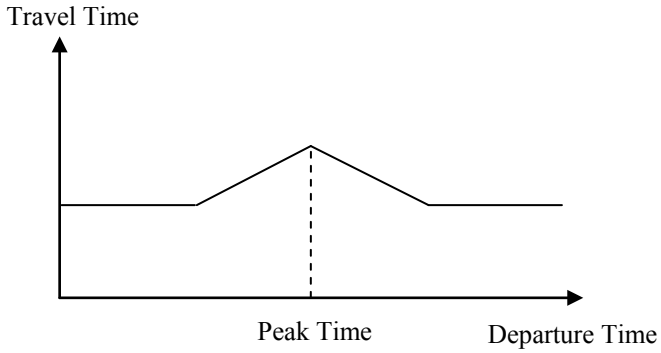


Figure 8: Travel time function with a single peak
(based on Ahn and Shin (1991))

Ahn and Shin (1991) investigate the effect of the first feasibility condition in an insertion heuristic and the effect of the third feasibility condition in the tour improvement heuristic of Or. They observe a substantial reduction in computation times compared to the case where no feasibility conditions are implemented.

4.6.2 Malandraki and Daskin (1992)

Malandraki and Daskin (1992) discuss a TDVRP with time windows. The objective is to minimize the total route time, consisting of the sum of all travel times, service times and waiting times. The authors develop a model based on a mixed integer linear programming (MILP) formulation. It is summarized in the following paragraph:
The travel time between two locations is represented by $c_{ij}(t_i)$, which is a deterministic function of the distance and the time of the day a vehicle departs from node *i*. The time horizon is partitioned into a number of time intervals each corresponding to a different travel time. As soon as the departure time of a vehicle becomes known, the travel time is derived from the associated time interval. The model does not guarantee the FIFO property as the travel time remains constant until the vehicle reaches its destination. Malandraki and Daskin (1992) smooth the travel time function by allowing vehicles to wait at customer locations if it leads to a decrease in the objective value.
The authors illustrate this with the following example (see Figure 9): If a vehicle is ready to leave the origin node *i* at any time t_2 such that $a < t_2 < b$, it is preferable to wait until time instant *b*. The travel time function is then given by the piecewise continuous function *P-A-C-Q*.

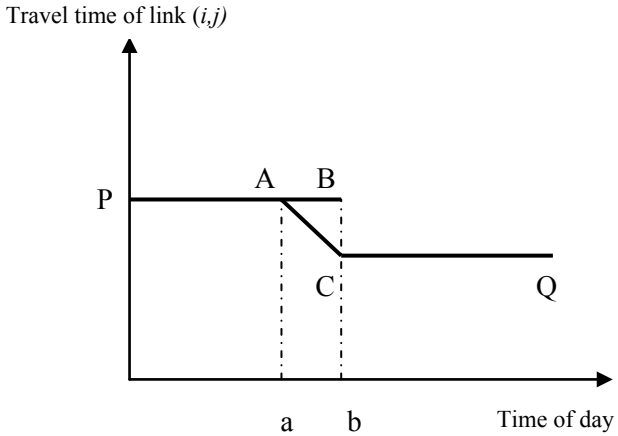


Figure 9: Travel time step function
(based on Malandraki and Daskin (1992))

Malandraki and Daskin (1992) solve the TDVRP without time windows with a nearest neighbor heuristic. They test a sequential and a simultaneous approach. The sequential approach exists in two versions: in the first version vehicles are filled from smallest to largest whereas in the second version vehicles are filled from largest to smallest. The simultaneous approach fills vehicles from smallest to largest.

The heuristics are tested on 32 randomly generated problems which consist of 10 to 25 customers. The travel time functions are represented by step functions with two or three time intervals per link. Malandraki and Daskin (1992) report that the sequential approach which fills vehicles from largest to smallest yields better results than the sequential approach which does the opposite. The authors conclude that the filling order considerably influences the solutions.

4.6.3 Hill and Benton (1992)

Hill and Benton (1992) propose a model for vehicle scheduling problems based on intra-city time-dependent travel speeds. The model consists of n nodes where the distance between two nodes is represented by d_{ij}. The time-dependent travel speeds are associated with the nodes of the network instead of the arcs. To be precise, the average speed associated with the area around node i in time interval T is denoted by r_{iT} whereas the average speed associated with the area around node j in time interval T is represented by r_{jT}. This implies that the first part of the journey is based on r_{iT} and the second part is based on r_{jT}. Consequently, the average travel speed r_{ijT} and the travel time c_{ijT} are defined as follows:

$$r_{ijT} = (r_{iT} + r_{jT})/2 \tag{10}$$

$$c_{ijT} = d_{ij}/r_{ijT} \tag{11}$$

According to Ichoua et al. (2003), this model does not satisfy the FIFO property either.

Finally, Hill and Benton (1992) present results on an example with a single vehicle and five locations. They use a small set of historical travel time data and assume 24 periods per day. First of all, a solution based on constant travel speeds is estimated. Afterwards, the constructed routes are simulated with time-dependent travel speeds. In this case, the

total route distance increases by 9% whereas the total travel time decreases by 22.4% when compared to the constant speed case.
Furthermore, the time-dependent travel speed model is implemented in a commercial courier vehicle scheduling package. The authors conclude that the results of the implementation prove the validity of their modeling approach.

4.6.4 Ichoua et al. (2003)

One of the most cited papers in the TDVRP literature is the work of Ichoua et al. (2003). They present a VRP based on time-dependent travel speeds. In this problem each customer i is associated with a soft time window $[e_i, l_i]$ and a service time. Earlier arrival results in a waiting time whereas late arrival incurs a penalty cost. The aim of the problem is to minimize the weighted sum of the total travel time and the total lateness over all customers.

Ichoua et al. (2003) divide the time horizon into three time intervals T_1, T_2 and T_3 where the first and the third one refer to lower travel speeds and the second one corresponds to a higher travel speed. The authors develop a symmetric distance matrix $D = (d_{ij})$ and the travel speed matrices $V_T = v_{ijT}, T \in \{T_1, T_2, T_3\}$ for all $i, j \in \{1, \dots, n\}$. The travel speed matrices are indexed by the corresponding arc and the corresponding time period. Furthermore, the set of arcs is divided into three subsets $(A_c)_{1 \le c \le C}$. It follows that the travel speed on link (i, j) that belongs to a certain arc subset A_c during period T becomes $v_{cT} = v_{ijT}$. This results in a 3x3 time-dependent travel speed matrix $(v_{cT})_{1 \le c \le 3, 1 \le T \le 3}$ where the values depend on the arc category and on the time interval.

The authors develop a parallel tabu search heuristic with an adaptive memory based on the work of Taillard et al. (1997). The neighborhood of the incumbent solution is generated with the cross exchange operator. Ichoua et al. (2003) explain how they evaluate the cross exchanges in the time-dependent context. They propose a procedure which evaluates each move approximately; then the M best moves are calculated exactly. Finally, the best one is implemented.

Ichoua et al. (2003) perform several experiments to evaluate the model in a static setting. They use three scenarios which represent different degrees of time-dependency: scenario 1 refers to the lowest degree of time-dependency and scenario 3 corresponds to the highest degree of time-dependency. They create the time-independent solutions by taking the average speed over the three time intervals. It is shown that a significant number of these solutions becomes infeasible in the time-dependent context. Moreover, this number increases with the degree of time-dependency.
Ichoua et al. (2003) report that the use of time-dependent travel speeds improves the objective values in almost all problems: in scenario 1 the objective value is improved by 1% to 5% except for one problem, in scenario 2 the improvement ranges between 2% and 12.5% and in scenario 3 it ranges between 9.2% and 18%.

Finally, the TDVRP is implemented in a dynamic setting where new customer demands occur during the day. To solve the dynamic version, Ichoua et al. (2003) use an algorithm based on the work of Gendreau et al. (1999) where vehicles are allowed to wait at their current location so that they can react to new requests in their vicinity. In this case, a departure time which prevents too early arrival at the next customer location needs to be determined. Because of the potential travel speed change when crossing time interval boundaries, the calculation of the departure time is quite complicated. To overcome this difficulty, Ichoua et al. (2003) propose a backward recursive procedure. Finally, the dynamic approach is tested with the same framework as before. Once again, the time-dependent results are better than the results of the time-independent case.

4.6.5 Fleischmann et al. (2004)

Fleischmann et al. (2004) discuss the implementation of time-varying travel times in various vehicle routing algorithms with time windows. The travel time data is provided by the traffic information system LISB (Berlin). The authors propose the following model:

The day is divided into K time intervals $Z_k = [z_{k-1}, z_k[$ for $(k = 1, \dots, K)$. The interval $[z_0, z_K]$ represents the whole day. The shortest travel time between node i and node j when the start time lies in time interval Z_k is denoted by τ_{ijk}. Fleischmann et al. (2004) state that the raw travel time data from the traffic monitoring system is not

suitable for vehicle routing because the travel times jump at the time interval boundaries z_k. This might result in a passing situation if the travel time decreases in the next interval. To overcome this problem, the authors propose a smoothed travel time function for every pair (i, j). In this case, the travel time between node i and node j when starting at time t is denoted by $\tau_{ij}(t)$. The interval $[z_k - \delta_{ijk}, z_k + \delta_{ijk}]$ linearizes the jumps at the interval boundaries z_k. It is based on the slope s_{ijk} and on parameter δ_{ijk}. The slope s_{ijk} is defined as follows:

$$s_{ijk} = \frac{\tau_{ij,k+1} - \tau_{ijk}}{2\delta_{ijk}} \tag{12}$$

In the next step, Fleischmann et al. (2004) formulate the travel time function:

$$\tau_{ij}(t) = \begin{cases} \tau_{ijk} & for \quad z_{k-1} + \delta_{ijk-1} \le t \le z_k - \delta_{ijk} \\ \tau_{ijk} + (t - z_k + \delta_{ijk}) s_{ijk} & for \quad z_k - \delta_{ijk} \le t \le z_k + \delta_{ijk} \end{cases} \tag{13}$$

$$for \quad k = 1, \dots, K \; and \; \delta_{ij0} = \delta_{ijK} = 0$$

Figure 10 illustrates the smoothed travel time function:

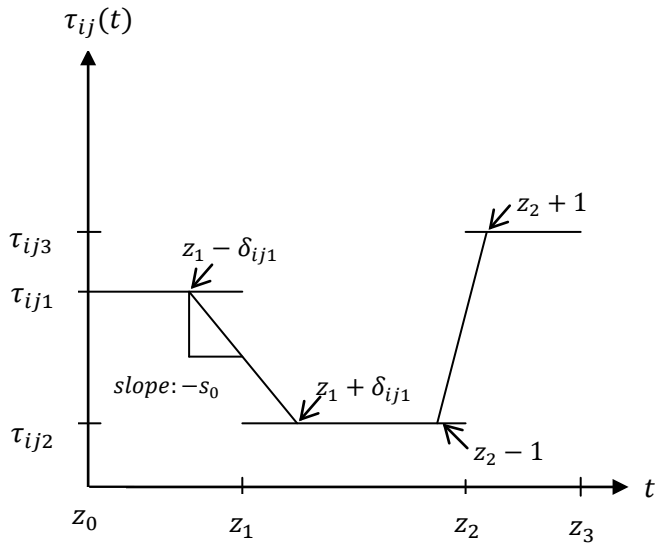


Figure 10: Smoothed travel time function
(based on Fleischmann et al. (2004))

If the departure time falls into the range $[z_k - \delta_{ijk}, z_k + \delta_{ijk}]$, we use the second formula in (13) to calculate the travel time, otherwise we assume the given travel time

τ_{ijk}. According to Fleischmann et al. (2004), this function guarantees the FIFO property. The authors define the following properties for the travel time function:

$$\delta_{ijk} > 0 \quad (k = 1, \dots, K-1) \ \textit{and} \ \ z_{k-1} + \delta_{ij,k-1} \leq \ \ z_k - \delta_{ijk} \ \ (k = 1, \dots, K) \qquad (14)$$

$$\textit{and} \quad s_{ijk} > -1 \qquad (k = 1, \dots, K-1) \qquad (15)$$

The authors state that the arrival time function is continuous and strictly monotonic in $[z_0, z_K]$ if conditions (14) and (15) are satisfied. Moreover, passing is excluded.
Next, they define the arrival time:

$$A_{ij}(t) = t + \tau_{ij}(t) \qquad (16)$$

The parameter δ_{ijk} and slope s_{ijk} are selected in such a way that conditions (12), (14) and (15) are satisfied:

$$\delta_{ijk} \leq \frac{1}{2}(z_k - z_{k-1}) \ \ \textit{and} \ \ \delta_{ijk} \leq \frac{1}{2}(z_{k+1} - z_k) \quad for \ \ (k = 1, \dots, K-1) \qquad (17)$$

If $\tau_{ijk} > \tau_{ij,k+1}$, this should be equivalent to:

$$s_{ijk} \leq \frac{\tau_{ij,k+1} - \tau_{ijk}}{z_k - z_{k-1}} \ \ \textit{and} \ \ s_{ijk} \leq \frac{\tau_{ij,k+1} - \tau_{ijk}}{z_{k+1} - z_k} \qquad (18)$$

According to Fleischmann et al. (2004), the terms in (18) are only compatible with condition (15) if the terms on the right hand side are greater than -1. That is, the decrease in the original travel times must be smaller than the length of the time intervals before and after. According to the authors, the slope can be arbitrarily steep if $\tau_{ijk} > \tau_{ij,k+1}$.

In the following part, the smoothed travel time function will be illustrated with an example:
Example 3: The time horizon is divided into two time intervals, each corresponding to a length of 20 minutes. The travel time in the first time interval τ_{ij1} is 40 minutes whereas

the travel time in the second interval τ_{ij2} is 30 minutes. As we can see the travel time decreases in the second time interval (see Figure 11).

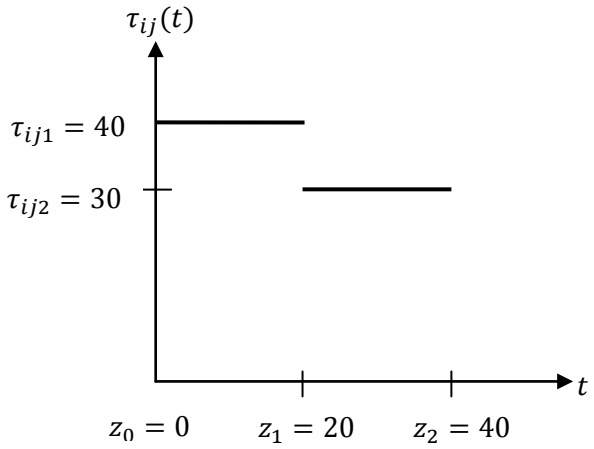


Figure 11: Example 3

Table 3 presents four different start times t and the corresponding arrival times $A_{ij}(t)$. Departing at t = 19 results in an arrival time $A_{ij}(t)$ = 59, whereas departing later at t = 20 leads to an earlier arrival time $A_{ij}(t)$ = 50. Therefore the FIFO property is not satisfied.

$\boldsymbol{t}$	$\boldsymbol{\tau_{ijk}}$	$\boldsymbol{A_{ij}(t)}$	
13	40	53	
19	40	59	The FIFO property
20	30	50	is not satisfied
27	30	57	

Table 3: Arrival Times in Example 3

In the next step, we will calculate the smoothed travel time in order to satisfy the FIFO property. First of all, the parameters s_{ij1} and δ_{ij1} will be determined:
As τ_{ij1} is greater than $\tau_{ij,2}$, slope s_{ij1} must satisfy the following conditions:

$$s_{ij1} \leq \frac{\tau_{ij,2}-\tau_{ij1}}{z_1-z_0} \; and \; s_{ij1} \leq \frac{\tau_{ij,2}-\tau_{ij1}}{z_2-z_1} \rightarrow s_{ij1} \leq \frac{30-40}{20-0} \; and \; s_{ij1} \leq \frac{30-40}{40-20} \qquad (21)$$

The conditions in (21) imply that s_{ij1} must be smaller or equal to -0.5. For this example a value of s_{ij1} = -0.8 will be chosen. In the next step, we will determine the parameter δ_{ij1}. It must satisfy the following conditions:

$$\delta_{ij1} \leq \frac{1}{2}(z_1 - z_0) \; and \; \delta_{ij1} \leq \frac{1}{2}(z_2 - z_1) \rightarrow \delta_{ij1} \leq \frac{1}{2}(20\text{-}0) \; and \; \delta_{ij1} \leq \frac{1}{2}(40\text{-}20) \quad (22)$$

According to the conditions in (22), δ_{ij1} must be smaller or equal to 10. Next, parameter δ_{ij1} is calculated:

$$s_{ij1} = \frac{\tau_{ij,2} - \tau_{ij1}}{2\delta_{ij1}} \qquad \text{-0.8} = \frac{30min - 40min}{2\delta_{ij1}} \quad (23)$$

It follows that δ_{ij1} = 6.25. Figure 12 illustrates the new travel time function:

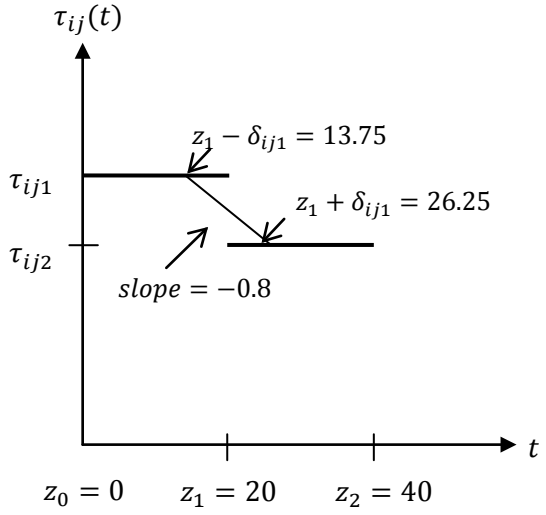


Figure 12: Smoothed travel time function in Example 3

In the last step, the new travel times and corresponding arrival times will be calculated with the following function:

$$\tau_{ij}(t) = \begin{cases} \tau_{ij1} & for \quad z_0 + \delta_{ij0} \leq t \leq z_1 - \delta_{ij1} \\ \tau_{ij1} + (t - z_1 + \delta_{ij1})s_{ij1} & for \quad z_1 - \delta_{ij0} \leq t \leq z_1 + \delta_{ij1} \end{cases} \quad (24)$$

$$For \; k = 1, \ldots, K \; and \; \delta_{ij0} = \delta_{ijK} = 0$$

The range $[z_1 - \delta_{ij1}, z_1 + \delta_{ij1}]$ covers the period from t = 13.75 until t = 26.25. If the start time falls into this range, the second formula in (24) will be used for calculating the travel time. Otherwise, the travel time is defined by τ_{ij1} if the start time belongs to the first interval or it is defined by τ_{ij2} if the start time belongs to the second interval.
Table 4 presents the new arrival times calculated with the formulations of Fleischmann et al. (2004). It is shown that an earlier start time results in an earlier arrival time, i.e. the FIFO property is satisfied.

t	*Travel Time Function*	$\tau_{ij}(t)$	$A_{ij}(t)$
13	τ_{ij1}	40	53
19	$\tau_{ij1} + (t - z_1 + \delta_{ij1})s_{ij1}$	35.8	54.8
20	$\tau_{ij1} + (t - z_1 + \delta_{ij1})s_{ij1}$	35	55
27	τ_{ij2}	30	57

Table 4: New arrival times in Example 3

Now we will return to the paper of Fleischmann et al. (2004) again:
Besides modeling the travel time function, the authors also discuss the start time calculation for a given arrival time. This is done by applying the inverse function $A_{ij}^{-1}(t)$. Furthermore, they explain how they model the time feasibility check in the time-dependent context. The authors also discuss possible difficulties when considering time-varying travel times in the objective function.

Finally, Fleischmann et al. (2004) test the savings algorithm, the savings algorithm with insertion and the sequential insertion algorithm with constant average travel times and with time-varying travel times from the vehicle monitoring system. They create seven test problems based on real data from logistics service providers in Berlin. They report that the computational effort of the time-dependent savings algorithm increases only modestly if compared to the case with constant average travel times. CPU times are doubled when performing the savings algorithm with insertion and the sequential insertion algorithm with time-varying travel times. According to the authors, the increase in CPU times is influenced by additional calculations concerning $\tau_{ij}(t)$ and $A_{ij}^{-1}(t)$. The authors conclude that using constant average travel times results in an

underestimation of the true total travel times by approximately 10%. Furthermore, several time windows are violated if time-dependency is not taken into account.

4.6.6. Haghani and Jung (2005)

Haghani and Jung (2005) examine the dynamic VRP with time-dependent travel times, soft time windows and real-time vehicle control. In the problem routes are adjusted at different times of the day. During each adjustment, new information about vehicle locations, travel times and demands is integrated into the model. The authors use a continuous travel time function where the slope is set to less than one in case of a travel time decrease (see Figure 13).

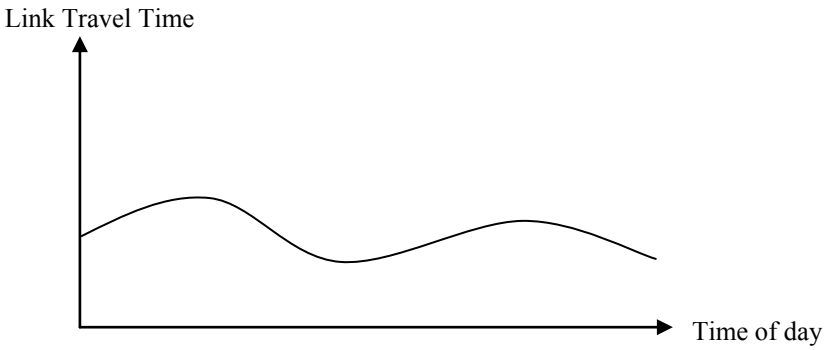


Figure 13: Continuous travel time function
(based on Haghani and Jung (2005))

Haghani and Jung (2005) define the VRP as a MILP problem. The objective is to minimize the total cost, consisting of the fixed vehicle costs, the routing costs and the penalties for time window violations.

Haghani and Jung (2005) develop a GA solution, a lower bound (LB) solution and an exact solution. They are based on a set of randomly generated test problems. The authors obtain exact solutions for up to ten customers and LB solutions for 15 to 30 customers. For problems with up to ten customers, they implement 10 to 15 time intervals whereas for problems with 30 customers they use 30 time intervals.

Haghani and Jung (2005) report that the GA results based on ten customers are similar to the exact results. The gaps between the GA results and the lower bounds are below 5% for all problems where the number of customers ranges between 5 and 25 (10 time intervals). Given the problem with 30 customers (30 time intervals), the gap amounts to 7.9%.

Finally, Haghani and Jung (2005) test the GA in a simulated network where accidents cause significant congestion in certain parts of the road network. They compare a static and a dynamic approach. The dynamic approach allows the re-planning of routes based on real-time travel time information at regular intervals during the day. In this case, the travel speed of a link can be calculated at any time during the day. The authors conclude that the dynamic routing plan leads to better results than the static one especially if the traffic situation is very unstable.

4.6.7 Eglese et al. (2006)

Eglese et al. (2006) use time-dependent travel time information to develop a road timetable for a road network. This timetable provides data on the shortest-time paths and minimum travel times between the nodes at different times of the day, different days of the week and different seasons of the year. The travel time information is based on historical data collected by the ITIS Floating Vehicle Data monitoring system which updates travel times based on current road conditions. The data for each arc in the network is summarized into 15-minutes time intervals throughout each day and is then related to the corresponding day of the week. Eglese et al. (2006) use the travel time calculation procedure of Ichoua et al. (2003) for adjusting travel times. Eglese et al. (2006) report that the adjustments are only performed in rare cases because the arc lengths and travel times are short when compared to the width of the time intervals. The minimum time and shortest-time paths between the nodes of the time-dependent network are established with Dijkstra's algorithm. This algorithm finds the shortest path from a source node to every other node in a graph.

In the next step, Eglese et al. (2006) perform a test to determine how effective the road timetable works with real world data. They design a scenario in which a logistics service provider delivers goods to a set of 18 customers. The problem is restricted by capacity limits, customer time windows and tour length constraints. The aim is to minimize the total driving time. To solve the problem, Eglese et al. (2006) use a tabu search algorithm which incorporates the road timetable data.
First, they create an initial solution based on the data of the least congested period. In the next step, the solutions are re-evaluated with time-dependent travel times. In this

case, the travel time increases by 7.3% when compared to the initial solution. Furthermore, several time windows and tour length constraints are violated. Finally, Eglese et al. (2006) develop a distribution plan based on time-dependent travel times. This plan leads to a minimization of the total travel time and driver overtimes. Moreover, all time windows are satisfied.

4.6.8 Donati et al. (2008)

Donati et al. (2008) develop a Multi Ant Colony System (MACS) for the TDVRP with time windows. The problem includes hard customer time windows, a depot time window and a limit on vehicle capacity. The authors use a hierarchical objective: the primary objective is to minimize the number of routes while the secondary objective is to minimize the total travel time. Donati et al. (2008) base their algorithm on the MACS of Gambardella et al. (1999).

Gambardella et al. (1999) describe the MACS as follows:
The algorithm consists of two colonies of artificial ants. The first colony, *ACS-VEI*, aims at reducing the number of vehicles whereas the second colony, *ACS-TIME*, aims at reducing the total travel time. Each colony has its own pheromone trail. In case of an improvement, the best solution and the pheromones are updated globally. Then the procedure is restarted with two new colonies and a reduced number of vehicles.
At the beginning of the algorithm, each ant builds a single route. It tries to find an unvisited customer whose insertion ensures route feasibility. Moreover, the customer choice is based on a probability distribution which describes the attractiveness of the customer. The route construction is continued until there are no more feasible customers left. At the end of the algorithm, an insertion heuristic tries to insert unvisited customers into the solution. If the resulting solution is feasible, a local search procedure is applied for further improvement.

Donati et al. (2008) adapt the MACS algorithm of Gambardella et al. (1999) in order to integrate time-dependency. They assume that the speed distributions are known for each arc of the network. They divide the time horizon into a number of time intervals in which the speed is considered to be constant. The speed distribution is represented by

$v_{ij}(t)$ which describes the speed on arc (i,j) when the trip is initiated at time t. The time-dependent distribution of an arc (i,j) in time interval k is denoted as τ_{ijk}; it represents the benefit of traversing the arc (i,j) when departing from i at time t during time interval S_k. It is derived from the speed distribution.

Donati et al. (2008) apply several neighborhood operators in their local search procedure: relocation, exchange, 2-opt, branch relocation and branch exchange. The route changes are first evaluated locally, i.e. only directly affected arcs are taken into account. If this local evaluation shows an improvement, the entire impact of the change is computed. The feasibility check is done in constant time by back propagating the so-called slack time which provides information on how much a delivery can be delayed without violating the time windows of the current customer and all consecutive customers. Donati et al. (2008) formulate the slack time so that the FIFO property is ensured.

In the final part of their work, Donati et al. (2008) perform several tests with Solomon's instances. They calculate the constant-speed solutions and evaluate them in a time-dependent context. It is shown that the feasibility of solutions decreases if there is an increase in the variability of traffic conditions. Moreover, in some cases time windows are missed.
In the next step, Donati et al. (2008) test the time-dependent MACS for the VRPTW. They state that the algorithm performs efficiently in terms of computation time and solution quality, especially if traffic conditions are highly variable.

Finally, Donati et al. (2008) test their algorithm with data from the road network of Padua, Italy. The travel time distributions for each arc and for each hour of the day are provided by the traffic control system Cartesio. They choose a set of customers from the nodes in the network. Then the shortest paths among all customer pairs are computed with the robust shortest path algorithm of Montemanni et al.. As soon as the departure time is known, a suitable pre-calculated path is chosen and the corresponding travel time distribution is determined. Donati et al. (2008) conclude that in the real road network the level of sub-optimality for the constant time case amounts to 7.58% if compared to the time-dependent case.

4.6.9 Woensel et al. (2008)

Woensel et al. (2008) present a dynamic VRP with time-dependent travel times due to traffic congestion. Their model is based on queuing models which capture the stochastic and dynamic aspects of travel time.

Woensel et al. (2008) compare three different speed approaches:
In the first case, the authors assume constant travel speeds. In the second case, they model the travel time function by dividing the day into three time intervals, each corresponding to a different travel speed. In the third case, the day is divided into 10-minutes intervals where the associated travel speeds are based on queuing models.

Woensel et al. (2008) use TS to solve the problem. For comparison purposes, they recalculate the resulting solutions with a different validation dataset for a different day.
Finally, the authors perform several tests based on the instances of Augerat et al. (1998). They report that the time-dependent case with three time zones performs considerably better than the time-independent case. When comparing the time-dependent case corresponding to three time intervals to the case where speeds are based on queuing models, the first one is less effective in terms of total travel time.
Furthermore, it is shown that the solution quality improves significantly when using travel speeds based on queuing theory. However, an increase in computation times can be observed. The authors conclude that taking time-dependent travel times into account is especially efficient when there is a high variability of travel speeds. Moreover, the solution quality increases the more time intervals and road types are considered.

4.6.10 Maden et al. (2010)

Maden et al. (2010) discuss vehicle routing and scheduling with time-varying data. The authors develop an algorithm named LANTIME which integrates data from a road timetable developed by Eglese et al. (2006) (for further details see Section 4.6.7). The problem under consideration aims at minimizing the total travel time. The authors impose capacity constraints, a limit on driver times and time windows.

Maden et al. (2010) compute the initial solution with the parallel insertion algorithm of Potvin and Rousseau (1993). Afterwards, the LANTIME algorithm is applied for further improvement. LANTIME is based on tabu search and applies four different neighborhood operations: cross exchange based on Taillard et al. (1997), insertion/removal, one exchange and swap. The neighborhood operation is randomly selected based on pre-defined probabilities.

Finally, Maden et al. (2010) perform several experiments on the VRPTW test problems of Solomon (1987). Furthermore, the algorithm is tested with real data from a vehicle fleet in the South West of the UK. The authors compare the total distances travelled, the route travel times and the amount of CO_2 emissions.
Maden et al. (2010) perform three runs:
The first one is based on time-independent travel speeds. During the second run, the solutions of the first run are recalculated with time-varying travel speeds. It is shown that several routes become infeasible. The third run tests the LANTIME algorithm with time-varying travel speeds. It is shown that the algorithm constructs feasible routes. The authors conclude that the LANTIME algorithm avoids routes with high congestion, low travel speeds and relatively high CO_2 emissions.

4.6.11 Balseiro et al. (2011)

Balseiro et al. (2011) examine the TDVRP with time windows. For this purpose, they develop a MACS algorithm hybridized with insertion heuristics (MACS-IH) based on the minimum delay technique (MDL). The aim is to create an insertion heuristic which leads to a higher number of feasible solutions.

Balseiro et al. (2011) formulate the problem as a MILP problem based on the formulation of Malandraki and Daskin (1992). Vehicles are allowed to wait at customer locations if they arrive too early. The time horizon is divided into M time intervals where the end of an interval is denoted T_m. The travel times are piecewise linear functions of the departure time. Given that a vehicle departs at time instant t during time interval m, the travel time of an arc (i, j) is defined as follows:

$$\alpha_{ij}^{m} + \beta_{ij}^{m} t \tag{25}$$

The two coefficients α_{ij}^{m} and β_{ij}^{m} are selected in such a way that the function remains continuous over all time intervals and β_{ij}^{m} is greater or equal to -1. According to the authors, this travel time function satisfies the FIFO property.
A hierarchical objective is proposed: the primary objective minimizes the number of vehicles whereas the secondary objective minimizes the total time of all routes, consisting of all travel times, service times and waiting times.

The MACS-IH is based on the MACS framework of Dorigo and Gambardella (1997):
During the update phase of the *ACS-TIME* colony, unrouted customers are tried to be inserted. Next, a local search is performed to improve the total time of the solution. Balseiro et al. (2011) use the following neighborhood operators: relocate, exchange, 2-opt, Or-opt, 2-opt* and cross exchange. The evaluation of the moves is facilitated by restricting the neighborhood to solutions in which newly linked nodes employ a nearest neighbor relationship.

Balseiro et al. (2011) combine three procedures in their insertion heuristic:
The *"local search + insertion"* procedure generates all neighboring solutions and inserts unrouted customers. The *"local search + MDL"* procedure applies the MDL metric which measures the hardness of inserting a customer. This hardness describes the degree to which the imposed constraints are violated if an unrouted customer is inserted. The goal of the procedure is to find the neighboring solution with the smallest total MDL. If this procedure is not successful in decreasing the MDL, the *"local search + maximal free time"* procedure is applied. It tries to increase the maximal free time which describes the maximum contiguous waiting time within a route.

Finally, Balseiro et al. (2011) test the three procedures with the VRPTW instances of Solomon. The authors report that the *"local search + insertion"* and the *"local search + MDL"* procedures are successful in maximizing the number of visited customers. The introduction of the maximal free time does not lead to better results. When incorporating the local search and insertion procedure into the MACS algorithm, the number of routes is reduced by 2.4% on average compared to the case where this has not been done.

In addition, the authors perform tests with the time-dependent instances of Ichoua et al. (2003). However, Balseiro et al. (2011) can only compare five sets with the same number of vehicles. It is shown that the MACS-IH minimizes the average travel times in each set when compared to the results of Ichoua et al. (2003).

4.6.12 Ehmke et al. (2012)

Ehmke et al. (2012) discuss the TDTSP and the TDVRP. They use time-dependent travel times which are based on Floating Car Data (FCD) from Stuttgart, Germany. These travel times are transformed into planning data sets through Data Mining procedures. Ehmke et al. (2012) use two types of travel time planning sets: The first planning set is based on FCD averages which represent day-specific travel times derived from aggregating historical FCD to one average measure per network segment and day of the week. The second planning set relies on FCD hourly averages which represent travel times derived from aggregating historical FCD to 24 averages per network segment, dependent on the day of the week and the time of the day.

The authors examine two routing approaches: the static routing approach is based on the first travel time planning set whereas the time-dependent routing approach relies on the second travel time planning set.
Ehmke et al. (2012) use time-dependent distance matrices which are determined with shortest path algorithms. These algorithms are based on a time-dependent digital road map of the city road network which satisfies the FIFO property. The authors use the approach of Fleischmann et al. (2004) for smoothing the jumps between the different time intervals.
In the final part of their paper, Ehmke et al. (2012) present a city logistics case study to examine the impact of time-dependent travel times. They divide the road network of Stuttgart into an inner and an outer city scenario.
The TDVRP is solved with an adapted version of the LANTIME algorithm of Maden et al. (2010) (see Section 4.6.10). The objective is to minimize the number of vehicles as well as the total travel time. A temporal constraint is imposed too.

Ehmke et al. (2012) compute three different route plans: the first one involves 60 inner city customers whereas the second one involves 60 outer city customers; the third plan represents a combined scenario (30 inner and 30 outer customers).
The authors report that the static routing approach leads to driver overtimes, infeasible schedules, etc. When comparing the tour durations of the static and the time-dependent routing approach, the differences range between 16% and 20%. According to the authors, the time-dependent routing approach determines more reliable and more efficient routes than the static one.

Concluding Remark

The comparison of different solution approaches plays an important role in the vehicle routing literature. However, the TDVRP literature lacks this possibility as the models differ too much from each other:
For example, Donati et al. (2008) use a hierarchical objective function where 1. the number of vehicles and 2. the total travel time is minimized whereas Ichoua et al. (2003) focus on the minimization of the total travel time and the total lateness. When considering constraints, we have seen that Donati et al. (2008) impose hard time windows and capacity restrictions whereas Ichoua et al. (2003) assume soft time windows and unlimited capacity. The model of Ichoua et al. (2003) satisfies the FIFO property whereas the model of Malandraki and Daskin (1992) does not. When considering the other authors in the literature review, this list could be continued forever.
In short, the authors i) define different objective functions, ii) impose different constraints and iii) use different travel time (or travel speed) functions. In this way it is not possible to make a comparison.

After this theoretical overview, we will continue with the practical part of this diploma thesis. The next chapter discusses the experimental evaluation of the CVRP and the TD-CVRP.

5. Implementation

5.1 Introduction

This chapter deals with the implementation of a CVRP and a TD-CVRP in C++. Both problems are tested with eight instances of Christofides et al. (1979). The instances consist of 50, 75 or 100 customers. Table 5 gives an overview of the data.

Instance	*number of customers*	*Max Capacity*	*Max Total Length*	*Service Time*
C01	50	160	999999	0
C02	75	140	999999	0
C03	100	200	999999	0
C06	50	160	200	10
C07	75	140	160	10
C08	100	200	230	10
C12	100	200	999999	0
C14	100	200	1040	90

Table 5: Instances of Christofides et al. (1979)

Both in the CVRP and the TD-CVRP a fleet of *m* identical vehicles delivers goods from a single depot to a set of *n* customers. Each vehicle has a limited capacity *C* and each customer is associated with a demand d_i. The travel times and distances are assumed to be the same. The following variables will be used:

routeCost	in the CVRP: sum of all distances in a route
	in the TD-CVRP: sum of all travel times in a route
routeTotalTime	sum of all travel times plus all service times in a route
totalCost	sum of all *routeCosts*
totalTime	sum of all *routeTotalTimes*

Instances 6, 7, 8 and 14 include a service time *s* which defines the duration of service and a tour length constraint which limits the *routeCost* and the *routeTotalTime*.
The objective of both problems is to minimize the total cost, which is defined by the sum of all *routeCosts*.

The following constraints must be satisfied:

i) each customer is visited exactly once;
ii) each vehicle route starts and ends at the depot;
iii) the demand of each customer is fulfilled;
iv) the capacity limit is not exceeded and
v) the tour length is not exceeded.

The time horizon of the TD-CVRP is divided into three time intervals T_1, T_2 and T_3. Each interval is associated with a fixed speed v_{T_k} where k denotes the number of time interval T_k.

After this general information, Chapter 5 continues with the following points:
First of all, the algorithm for solving the CVRP will be explained. The next section deals with the evaluation of the best CVRP solutions with five time-dependent scenarios. In the third section, the original algorithm is adapted so that time-dependency is taken into account. The adapted algorithm solves the TD-CVRP for all instances. The last section of this chapter presents the computational results of the implementation.

5.2 The Algorithm

First of all, the Euclidean distances between each pair of nodes (customers and depot) are determined and stored in a distance matrix. The Euclidean distance between node i and node j is defined as follows:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{26}$$

In the next step, an initial solution based on the savings algorithm will be computed.

5.2.1 Initial Solution

As we remember from Chapter 3, the savings heuristic merges two routes if it leads to a cost saving. In the first step, we will calculate the saving s_{ij} between each customer pair:

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \quad \text{for} \quad i, j = 1, \dots, n \ \ and \ i \neq j \tag{27}$$

Afterwards, the savings are sorted in a *savingsList* in decreasing order. Then each customer *i* is assigned to a separate vehicle route $(0, i, 0)$.

In the next step, we will check which customer pairs should be merged. We start with the first savings pair *(i,j)* from the *savingsList*. A merge is only performed if the following conditions are satisfied:

i) nodes i and j belong to two different routes;

ii) nodes i and j are in the front or end position of their routes and

iii) the potential route ensures the capacity limit and the tour length constraint.

If all conditions are satisfied, the routes are merged in such a way that node j is immediately visited after node i. Afterwards, the next savings pair from the *savingsList* is selected and the same procedure is applied. This is done until no more feasible savings are left. The routes are saved as *initialSolution*.

5.2.2 Tabu Search Algorithm

The TS algorithm starts with the initialization of several variables. First of all, *currentSolution* and *bestSolution* are initialized: *currentSolution* consists of the routes which were created by the savings algorithm; its total cost is defined by *currentCost*. *BestSolution*, on the other hand, stores the solution with the smallest total cost (*bestCost*) achieved during one run.

In the next step, *iteration*, *runtime* and *no_improvement* are initialized:
Iteration represents the number of the current iteration and *runtime* indicates the current runtime. *No_improvement* counts the number of iterations in which *bestSolution* cannot be improved. Every time *bestSolution* is improved, *no_improvement* is set to zero again. A run can be terminated in two ways: either if *no_improvement* reaches *max* or runtime reaches *maxRuntime* - whichever happens earlier.

The next paragraph deals with the structure of the tabu search algorithm:
First of all, we have to define a tabu restriction. The tabu restriction in the current algorithm forbids the reversal of a node into its original route. For example, if node *i* is moved from route *r* to route *s* then the reversal of node *i* to its original route *r* will be forbidden for *tabuLength* iterations. In order to save the tabu information, a *tabuList* is created (see Figure 14) where the rows refer to routes and the columns refer to nodes. The number of routes is based on the initial solution. Before the algorithm starts all *tabuList* matrices are set to zero. The *tabuLength* is randomly chosen from an interval and remains constant for the entire run.

	r1	**r2**	**…**
n1	0	0	0
n2	0	0	0
…	0	0	0

Figure 14: TabuList for *n* nodes and *r* routes

While one of the stopping conditions is not met, the following steps will be performed in each iteration:

First of all, the solution neighborhood of *currentSolution* is created. This is done with the *Relocate/Exchange* algorithm which will be explained in Section 5.2.2.1. The best solution of the neighborhood becomes the new *currentSolution*. In the next step, the *IntraRoute_Improvement* algorithm aims at improving the new solution (see Section 5.2.2.2). Afterwards, *currentCost* is compared to *bestCost*. In the case that *currentCost* is smaller than *bestCost*, *bestSolution* is replaced with the new *currentSolution*. If *bestCost* cannot be improved, *no_improvement* is incremented, otherwise it is set to zero again. In the next step, all values in the *tabuList* are decremented by one. Finally, the current *runtime* is updated. After that, a new iteration starts. At the end of a run, *bestSolution* is checked for feasibility (*Check_Feasibility*). If it satisfies all constraints it becomes the final solution. Algorithm 1 summarizes the whole procedure:

Algorithm 1: Tabu Search

```
1       runtime = 0
2       create intialSolution
3       currentSolution = initialSolution
4       bestSolution = initialSolution
5       iteration = 0
6       no_improvement = 0
7       choose tabuLength
8       while  runtime <  maxRuntime
9               RELOCATE / EXCHANGE
10              INTRAROUTE_IMPROVEMENT
11              if    currentCost < bestCost  then bestSolution = currentSolution
12                    no_improvement = 0
13              end if
14              else no_improvement ++
15              end else
16              if  no_improvement = max  then  terminate run
17              end if
18              iteration++
19              update runtime
20              update tabuList
21      end while
22      CHECK_FEASIBILITY
23      final solution = bestSolution
```

5.2.2.1 The *Relocate/Exchange* Algorithm

In this algorithm each customer node is selected once being:

1.) relocated to all positions of the other routes (except for the front and end positions) and

2.) exchanged with all customer nodes of the other routes.

Figure 15 illustrates the relocation of a node and the exchange of two nodes: the left picture shows the relocation of node *A2* from route *B* to route *A;* the right one depicts the exchange of *A2* and *B2*.

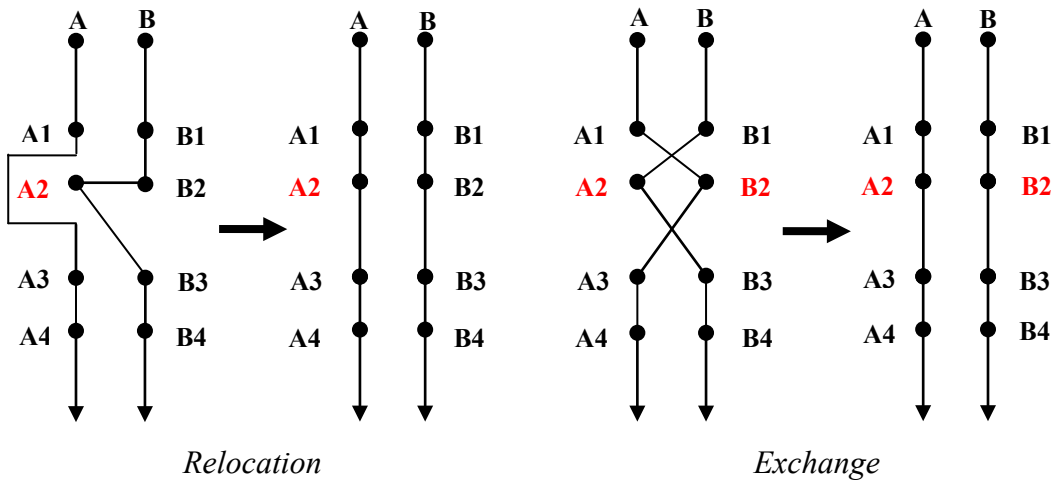


Figure 15: Relocation and Exchange

Every time a node is moved, we have to calculate the *changeCost*. This cost takes only those links into account which are directly affected by a relocation or an exchange, i.e. only the *local* change is considered. This is explained with the following example:
Let us consider the relocation of node *A2* to the position before node *A3* (see Figure 16). In this case, the links affected are those between *A1* and *A3* and between *B2* and *B3*. The other links remain the same and therefore they do not have to be considered for evaluation. The *changeCost* is defined as follows:

$$changeCost = \underbrace{-d_{A1,A3}+d_{A1,A2}+d_{A2,A3}}_{\text{Change of cost in route A}}\underbrace{-d_{B2,A2}-d_{A2,B3}+d_{B2,B3}}_{\text{Change of cost in route B}} \quad (28)$$

The same principle applies to the exchange of nodes *A2* and *B2*:

$$changeCost = \underbrace{-d_{A1,B2}-d_{B2,A3}+d_{A1,A2}+d_{A2,A3}}_{\text{Change of cost in route A}}\underbrace{-d_{B1,A2}-d_{A2,B3}+d_{B1,B2}+d_{B2,B3}}_{\text{Change of cost in route B}} \quad (29)$$

The red colored links in Figure 16 show which parts of the routes are evaluated when relocating node *A2* to route *A* and exchanging nodes *A2* and *B2*:

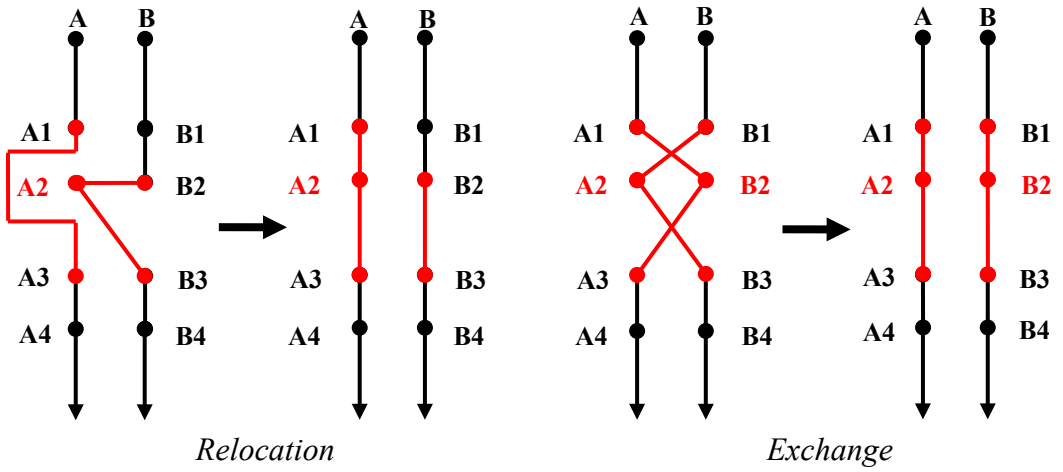


Figure 16: Evaluation of local changes

After having calculated the *changeCost*, we have to check if the constraints and the tabu restriction are still satisfied when performing the route change. If this is the case, *changeCost* is compared to *minCost*. *minCost* is set to a high value in the beginning of the *Relocate/Exchange* algorithm. As soon as a modification is found which satisfies *changeCost* < *minCost*, *minCost* is updated. At the end, the algorithm will perform the relocation or exchange which leads to the smallest *minCost* and satisfies all constraints. The tabu status of a relocation or an exchange can be overridden if it leads to a decrease in *bestCost*. Finally, the new solution is saved as *currentSolution*.

The *Relocate/Exchange* algorithm is based on a first improvement approach meaning that as soon as an improving modification is found the algorithm is stopped and the next step is performed (*IntraRoute_Improvement* algorithm). A modification is improving if it leads to a decrease in *currentCost*. Algorithm 2 summarizes the *Relocate/Exchange* algorithm.

Algorithm 2: Relocate/Exchange

```
1   minCost =1000000
2   for every node in route i in currentSolution (≠ depots) do
3     1.) RELOCATE
4     for every route j in currentSolution (≠ route i) do
5           relocate node into every position in route j
6           if relocation satisfies capacity constraint
7             calculate changeCost
8             if relocation satisfies tour length constraint AND changeCost < minCost
9               if (tabuList [node][route j] = 0) OR (currentCost+changeCost < bestCost)
10                  minCost = changeCost
11              end if
12            end if
13          end if
14    end for
15    2.)  EXCHANGE
16    for every route j in currentSolution (≠ route i) do
17        exchange node with all customer nodes in route j
18        if exchange satisfies capacity constraint then
19          calculate changeCost
20          if exchange satisfies tour length constraint AND changeCost < minCost
21            if (tabuList [node] [route j] = 0   AND  tabuList [node2] [route i] = 0)
22                OR (currentCost + changeCost < bestCost)
23                minCost = changeCost
24            end if
25          end if
26        end if
27     end for
28  end for
29  perform the modification which leads to the smallest minCost
30  set tabu reversal of the relocated/exchanged nodes
31  currentCost = calculate totalCost of currentSolution
```

5.2.2.2 The *IntraRoute_Improvement* Algorithm

The *IntraRoute_Improvement* algorithm is based on the following principle: all customer nodes are relocated into all positions of their own routes (except for the depot positions). If the cost of *currentSolution* can be decreased, the relocation is performed. Algorithm 3 provides a summary of the whole procedure:

Algorithm 3: IntraRoute_Improvement

```
1   minCost = currentCost
2   improvement = false
3   for every node in route i in currentSolution  do
4       relocate node into every position of its own route
5       calculate changeCost
6       if  relocation satisfies tour length constraint
7       AND currentCost +  changeCost < minCost
8               minCost = currentCost + changeCost
9               improvement = true
10      end if
11  end for
12  if improvement = true
13      perform the relocation which leads to the best improvement in minCost
14      currentCost = calculate total cost of currentSolution
15  end if
```

5.3 Evaluation of the CVRP solutions with time-dependent scenarios

Before the evaluation is performed, we need to determine an appropriate time horizon and the number of time intervals:

The time horizon covers a period of twelve hours. It is divided into three time intervals. The time intervals T_1 and T_3 have a length of three hours whereas time interval T_2 corresponds to six hours. The whole time horizon is based on the given tour length constraint. If the tour length constraint is 200 (as for example in instance 6), the time horizon is divided in the following way:

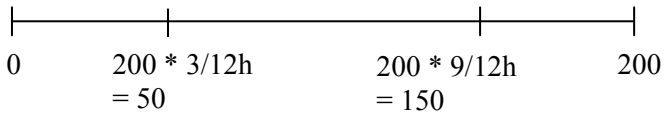


Figure 17: Division of the time horizon

Instances (1, 2, 3, 12) do not have a given tour length constraint, therefore several tests were performed to find an accurate one. Table 6 gives an overview of the chosen tour length constraints and service times.

Instance	*Max Tour Length*	*Service Time*
C01	470	10
C02	380	10
C03	500	10
C12	400	10

Table 6: Tour length constraints

Five time-dependent scenarios will be tested: In each scenario the first and the third time interval correspond to lower travel speeds whereas the second interval corresponds to higher travel speeds. The average speed for each scenario is one. The travel speeds are presented in Table 7.

Scenario	*Time Interval 1*	*Time Interval 2*	*Time Interval 3*
S1	1	1	1
S2	0.8	1.2	0.8
S3	0.6	1.4	0.6
S4	0.4	1.6	0.4
S5	0.2	1.8	0.2

Table 7: Travel speed in each time interval

Each scenario represents a different degree of time-dependency. The time-independent case is represented by S1 where the speed in each time interval is one. The highest degree of time-dependency can be found in S5. Figure 18 illustrates the different scenarios.

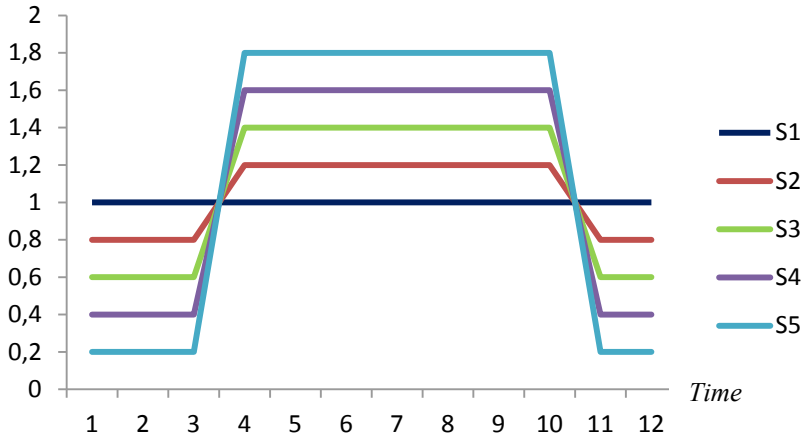


Figure 18: Time-dependent scenarios

The *bestSolution* of each instance is recalculated with the travel time calculation procedure of Ichoua et al. (2003) (see Algorithm 4). The current time instant is represented by t, the arrival time is denoted t' and the speed in time interval k is represented by v_{T_k}. Each time interval k is restricted by a lower and an upper bound: $T_k =]\underline{t_k}, \overline{t_k}]$. The number of time intervals is $K = 3$.

Algorithm 4: Evaluation of the best CVRP solutions with time-dependent scenarios

```
1   for every route i in bestSolution do
2       set  routeCost_i and routeTotalTime_i to 0
3       for every node ii of route i do
4           find out in which  T_k  the current routeTotalTime_i lies
5           t = routeTotalTime_i
6           distance = distance [node ii] [node ii+1]
7           t' = t + (distance / v_{T_k})
8           while  t' > t̄_k
9               distance = distance - v_{T_k} * (t̄_k - t)
10              t = t̄_k
11              t'= t + (distance / v_{T_{k+1}})
12              k ++
13          end while
14          distance = t' + routeTotalTime_i
15          routeCost_i = routeCost_i  + distance
```

```
16          routeTotalTime_i =  routeTotalTime_i + distance
17          if node ii+1 != depot  then  routeTotalTime_i + service time
18      end for
19  end for
```

5.4 Adaptation of the Algorithm

In order to take time-dependency into account, we have to adapt the original algorithm. From now on, *routeCost* defines the sum of all travel times of a route. The objective is to minimize the total cost where the total cost represents the sum of all *routeCosts*. The algorithm requires several changes in order to solve the TD-CVRP: The first major change concerns the evaluation of all cost variables (*routeCosts, routeTotalTimes*). From now on, they will be computed with the travel time calculation procedure of Ichoua et al. (2003) (see Algorithm 5).

Algorithm 5: Calculation of *routeCost* and *routeTotalTime*

```
1   set  routeCost_i and routeTotalTime_i to 0
2   for every node ii of route i do
3       find out in which  T_k  the current routeTotalTime_i lies
4       t = routeTotalTime_i
5       distance = distance [node ii] [node ii+1]
6       t' = t + (distance / v_{T_k})
7       while   t' > t̄_k
8                   distance = distance - v_{T_k} * (t̄_k - t)
9                   t = t̄_k
10                  t'= t + (distance / v_{T_{k+1}})
11                  k ++
12      end while
13      distance = t' + routeTotalTime_i
14       routeCost_i = routeCost_i  + distance
15       routeTotalTime_i =  routeTotalTime_i + distance
16      if node ii+1 != depot  then  routeTotalTime_i + service time
17  end for
```

The next change concerns the savings algorithm. In the TDVRP we need to determine the *time-dependent* savings:

$$s_{ij} = TT_{0i0} + TT_{0j0} - TT_{0ij0} \tag{30}$$

In expression (30), TT_{0i0} represents the *routeTotalTime* of route *(0, i ,0)*, TT_{0j0} indicates the *routeTotalTime* of route *(0, j, 0)* and TT_{0ij0} represents the *routeTotalTime* of the merged route. All total times are calculated with the calculation procedure of Ichoua et al. (2003) (see Algorithm 5). Moreover, we have to compare the *time-dependent routeTotalTime* of *(0, i, j, 0)* to the tour length constraint.

The next adaptation concerns the calculation of the *changeCost* (*Relocate/Exchange* and *IntraRoute_Improvement* algorithm). In the time-dependent case it is no longer sufficient to simply subtract the cost of the links removed and add the cost of the new links. For example, if node *A2* is relocated to the position before *A3*, all links following *A3* might be affected because of a potential speed change when crossing the boundaries (see Figure 19). It is no longer sufficient to only calculate the local change in the areas *A1-A3* and *B2-B3* – from now on the *global* change needs to be taken into account. The *changeCost* will be computed in the following way:

$$newTT_{Route\_A} + newTT_{Route\_B} - origTT_{Route\_A} - origTT_{Route\_B} \tag{31}$$

In expression (31), $origTT_{Route}$ defines the original *routeTotalTime* of a route whereas $newTT_{Route}$ describes the new *routeTotalTime* of a changed route. The calculation of the new *routeTotalTime* from the beginning to the end of a route every time a relocation or exchange is checked, is very time-consuming. In order to solve this problem, the following procedure is implemented:
After the initial solution has been calculated, we save the start time at each node. The start times of those nodes which are affected by a change are updated in each iteration. Knowing the start times alleviates the calculation of the new *routeTotalTimes*: instead of taking into account the whole route, we start the total time calculation at the node before the change. In this way the same results are achieved in a faster way.

The red colored links in Figure 19 show which parts of *A* and *B* are considered when calculating the new *routeTotalTimes*: in route *A* the calculation starts at *A1*; in route *B* at node *B2*. The *routeTotalTimes* of the original routes are known and therefore do not need to be computed. They are only updated if a change is performed.

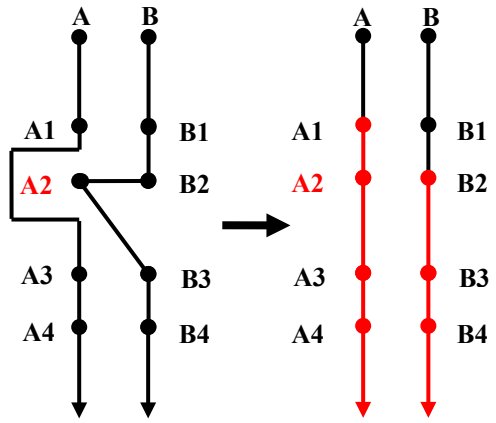


Figure 19: Total time calculation when relocating a node

5.5 Computational Results

This section starts with the presentation of the CVRP results. Next, the evaluation of the CVRP results with time-dependent scenarios is discussed. Finally, the TD-CVRP results are presented and compared to the CVRP results evaluated with time-dependent scenarios.

All experiments were performed on a laptop with 2.1 GHz and 4 GB RAM. Each algorithm - the original algorithm and the adapted algorithm - was run five times. In each run a *tabuLength* was randomly chosen from the interval [10,11,…,18]. The limit on *no_improvement* was set to 20.000 iterations and *maxRuntime* was set to 360 seconds.

First of all, the results of the CVRP are presented in Table 8. The *Best Known Results* are based on Cordeau and Laporte (2002). *Best Solution* shows the best solution over five runs and *Avg Solution* denotes the average over all best solutions. *Best Gap* describes the gap of the best solution compared to the best known solution and *Avg Gap*

shows the deviation of the average solution compared to the best known solution. *Avg best runtime* presents the average runtime to reach the best solution for each instance whereas *Avg total runtime* describes the average total runtime for each instance. The last row *Avg_Inst* presents the averages over all eight instances. The best known results are reached for all instances except for instances C03 and C07.

Instance	*Best Solution*	*Avg Solution*	*Best Known Solution*	*Best Gap*	*Avg Gap*	*Avg best runtime*	*Avg total runtime*
C01	524.61	524.69	524.61	0.00%	0.02%	2	5
C02	835.26	838.43	835.26	0.00%	0.38%	8	14
C03	834.78	834.87	826.14	1.05%	1.06%	2	15
C06	555.43	555.93	555.43	0.00%	0.09%	2	7
C07	912.34	913.10	909.68	0.30%	0.38%	10	18
C08	865.95	880.14	865.94	0.00%	1.64%	14	33
C12	819.56	819.56	819.56	0.00%	0.00%	0	15
C14	866.37	866.74	866.37	0.00%	0.04%	1	19
Avg_Inst	776.79	779.18	775.37	0.17%	0.45%	5	16

Table 8: CVRP results

In the next step, the best results of the CVRP are evaluated with time-dependent scenarios. Table 9 shows the total costs for each instance and each scenario. The last row *Avg_Inst* represents the average total cost over all instances.

Instance	*S1*	*S2*	*S3*	*S4*	*S5*
C01	524.61	547.01	590.44	656.90	753.77
C02	835.26	875.63	944.54	1051.11	1228.48
C03	834.78	859.21	923.66	1026.29	1227.70
C06	555.43	578.50	645.86	782.47	1244.27
C07	912.34	946.76	1058.55	1331.95	2074.85
C08	865.95	890.91	994.55	1224.04	1950.82
C12	819.56	856.96	944.37	1098.44	1331.07
C14	866.37	987.99	1222.70	1727.90	3217.10
Avg_Inst	776.79	817.87	915.58	1112.39	1628.51

Table 9: Evaluation of the CVRP solutions with time-dependent scenarios

Figure 20 compares the time-independent scenario S1 to the time-dependent scenarios S2, S3, S4 and S5. The comparison is based on the average total costs over all instances (see *Avg_Inst* in Table 9).

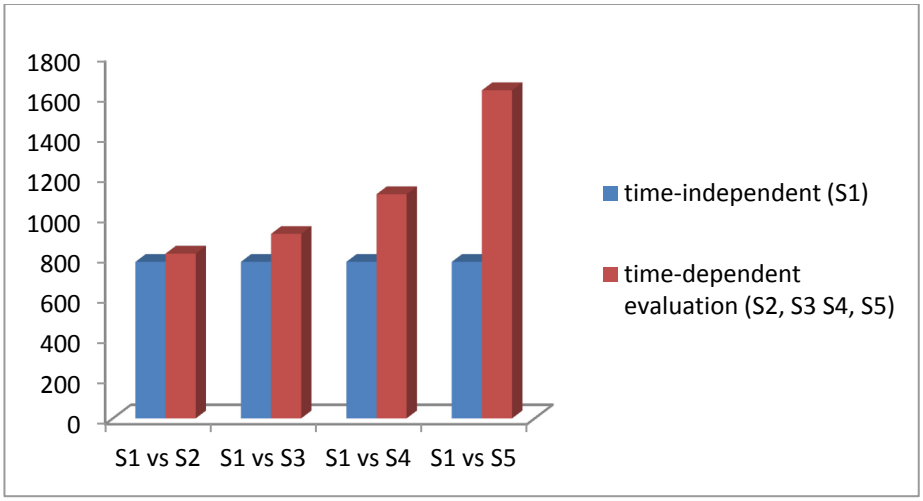


Figure 20: Comparison of average results

Table 10 shows in percentage terms the increase of the total costs in S2, S3, S4 and S5 compared to S1. In S2 the total costs increase by 2.88% to 14.04%, in S3 the increase ranges between 10.65% and 41.13%, in S4 it is between 22.94% and 99.44% and finally in S5 the increase goes from 43.68% until 271.33%. The last row *Avg_Inst* shows the average increase over all instances. It starts with 5.18% in S2 and goes until 106.04% in S5. It can be seen that the higher the degree of time-dependency, the higher is the increase in *totalCost*.

Instance	***S1 Result***	***S2 Increase***	***S3 Increase***	***S4 Increase***	***S5 Increase***
C01	524.61	4.27%	12.55%	25.22%	43.68%
C02	835.26	4.83%	13.08%	25.84%	47.08%
C03	834.78	2.93%	10.65%	22.94%	47.07%
C06	555.43	4.15%	16.28%	40.88%	124.02%
C07	912.34	3.77%	16.03%	45.99%	127.42%
C08	865.95	2.88%	14.85%	41.35%	125.28%
C12	819.56	4.56%	15.23%	34.03%	62.41%
C14	866.37	14.04%	41.13%	99.44%	271.33%
Avg_Inst	776.79	5.18%	17.47%	41.96%	106.04%

Table 10: Percentage increase in total costs when time-dependency is assumed

Several routes become infeasible in the time-dependent context. Table 11 and Table 12 show the percentage of routes which do not satisfy the tour length constraints anymore. Table 11 summarizes the percentage of routes which are infeasible when comparing their *routeCosts* to the corresponding tour length constraint. In this case, tour length violations only appear in S5 for instances 6, 7 and 8.

Table 12 illustrates the percentage of routes which become infeasible when taking their *routeTotalTimes* into account. In this case, the tour length constraints are already broken in S2 for instances 6, 7, 8 and 14. In S5 all routes of instances 7 and 8 become infeasible. The solutions of instances 1, 2, 3 and 12 remain feasible as the tour length constraint is set to a high value.

Instance	***Nr of Routes***	***S2***	***S3***	***S4***	***S5***
C01	5	0%	0%	0%	0%
C02	10	0%	0%	0%	0%
C03	8	0%	0%	0%	0%
C06	6	0%	0%	0%	83%
C07	11	0%	0%	0%	73%
C08	9	0%	0%	0%	33%
C12	10	0%	0%	0%	0%
C14	11	0%	0%	0%	0%
Avg_Inst	9	0%	0%	0%	24%

Table 11: Percentage of infeasible routes based on *routeCosts*

Instance	***Nr of Routes***	***S2***	***S3***	***S4***	***S5***
C01	5	0%	0%	0%	0%
C02	10	0%	0%	0%	0%
C03	8	0%	0%	0%	0%
C06	6	33%	83%	83%	83%
C07	11	45%	82%	82%	100%
C08	9	22%	33%	78%	100%
C12	10	0%	0%	0%	0%
C14	11	9%	18%	27%	91%
Avg_Inst	9	14%	27%	34%	47%

Table 12: Percentage of infeasible routes based on *routeTotaltimes*

Now we will take a closer look at the violation of the tour length constraints. Table 13 and 14 compare the *routeCosts* and the *routeTotalTimes* to the corresponding tour length limitation: Table 13 shows the average percentage increase of the *routeCosts* compared to the corresponding tour length constraints. Table 14 illustrates the average percentage increase of the *routeTotalTimes* compared to the corresponding tour length constraints. It is shown that for those solutions which become infeasible, the average increase becomes higher, the higher the degree of time-dependency (Table 14).

Instance	*S2 Average Increase*	*S3 Average Increase*	*S4 Average Increase*	*S5 Average Increase*
C01	0%	0%	0%	0%
C02	0%	0%	0%	0%
C03	0%	0%	0%	0%
C06	0%	0%	0%	14.69%
C07	0%	0%	0%	20.81%
C08	0%	0%	0%	6.53%
C12	0%	0%	0%	0%
C14	0%	0%	0%	0%
Avg_Inst	0%	0%	0%	5.25%

Table 13: Average increase in *routeCosts* compared to tour length constraint

Instance	*S2 Average Increase*	*S3 Average Increase*	*S4 Average Increase*	*S5 Average Increase*
C01	0%	0%	0%	0%
C02	0%	0%	0%	0%
C03	0%	0%	0%	0%
C06	0.69%	4.53%	15.50%	53.03%
C07	0.88%	6.02%	19.72%	60.50%
C08	0.30%	2.08%	8.90%	42.55%
C12	0%	0%	0%	0%
C14	0.11%	0.47%	1.78%	10.97%
Avg_Inst	0.25%	1.64%	5.74%	20.88%

Table 14: Average increase in *routeTotaltimes* compared to tour length constraint

In the next step, the algorithm is adapted to solve the TD-CVRP. Table 15 shows the average results (*Avg*) and the best results (*Best*) for each instance and each speed scenario based on five runs. The last row *Avg_Inst* represents the average total cost over all instances.

	S1		*S2*		*S3*		*S4*		*S5*	
Instance	***Avg***	***Best***	***Avg***	***Best***	***Avg***	***Best***	***Avg***	***Best***	***Avg***	***Best***
C01	524.74	524.61	546.16	544.40	580.65	577.87	637.45	631,88	726.66	723.23
C02	838.80	835.26	875.31	867.75	935.09	929.80	1031.66	1029.02	1177.76	1172.45
C03	835.35	834.78	851.80	848.87	896.34	886.27	980.56	978.53	1172.48	1169.97
C06	555.43	555.43	569.94	568.92	600.36	595.50	645.12	642.92	758.78	757.11
C07	916.15	912.34	960.04	958.26	1062.09	1057.32	1162.03	1150.84	1297.24	1270.66
C08	877.68	865.95	894.02	891.41	967.59	963.74	1056.23	1040.85	1297.78	1266.12
C12	819.56	819.56	847.88	847.35	915.34	913.69	1045.45	1044.74	1235.37	1235.37
C14	866.82	866.37	988.35	988.25	1201.81	1190.37	1659.10	1599.54	2478.28	2188.00
Avg_Inst	779.32	776.79	816.69	814.40	894.91	889.32	1027.20	1014.79	1268.04	1222.86

Table 15: TD-CVRP results

In Figure 21 the average total costs which were evaluated with time-dependent scenarios (see *Avg_Inst* in Table 9) are compared to the average total costs of the TD-CVRP (see *Avg_Inst* in Table 15). The average total costs are based on all eight instances.

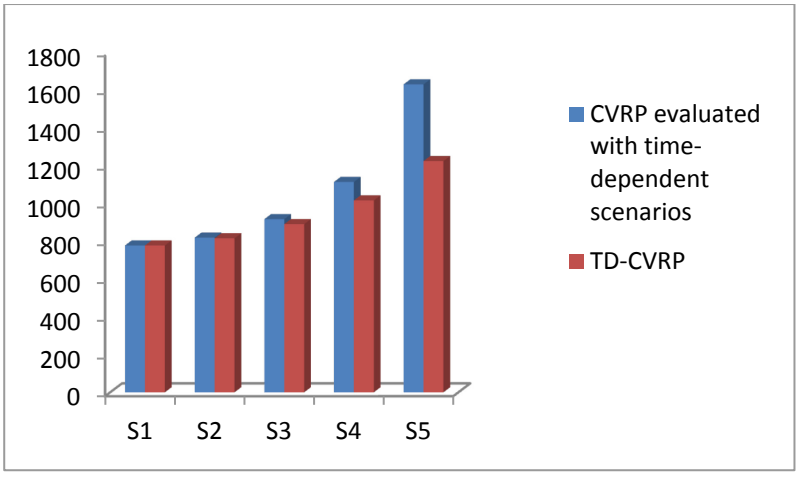


Figure 21: Comparison of CVRP and TD-CVRP results

In Table 16 the best TD-CVRP results (see Table 15) and the best CVRP results which were evaluated with time-dependent scenarios (see Table 9) are compared to each other.

In S2 improvements can be observed in all instances except for instances 7, 8 and 14; they range between 0.48% and 1.66%. In S3 the results of all instances become better: the improvements range between 0.12% and 7.8%. In S4 the improvements are between 2.1% and 17.83%. The highest improvements can be found in S5: they range from 4.05% to 39.15%. When looking at the average improvements over all instances, it can be seen that the improvement starts with 0.51% in S2 and goes until 20.69% in S5.
As we can see, the improvements are higher, the higher the degree of time-dependency. Moreover, the algorithm constructs routes which fulfill the tour length constraints in all instances and all scenarios.

Instance	*S1*	*S2*	*S3*	*S4*	*S5*
C01	524.61	-0.48%	-2.13%	-3.81%	-4.05%
C02	835.26	-0.90%	-1.56%	-2.10%	-4.56%
C03	834.78	-1.20%	-4.05%	-4.65%	-4.70%
C06	555.43	-1.66%	-7.80%	-17.83%	-39.15%
C07	909.68	1.21%	-0.12%	-13.60%	-38.76%
C08	865.95	0.06%	-3.10%	-14.97%	-35.10%
C12	819.56	-1.12%	-3.25%	-4.89%	-7.19%
C14	866.37	0.03%	-2.64%	-7.43%	-31.99%
Avg_Inst	776.79	0.51%	3.08%	8.66%	20.69%

Table 16: Comparison of the TD-CVRP and CVRP results

When comparing the improvements of the TD-CVRP results (see Table 16) and the deteriorations of the CVRP results evaluated with time-dependent scenarios (see Table 10), it can be observed that the improvements are not as impressive as the deteriorations. For example, the deterioration of the CVRP result in instance 6 for S5 is 124% whereas the improvement when solving the TD-CVRP is only 39.15 %.
Another observation can be made when comparing the TD-CVRP results of instances (1, 2, 3, 12) and the TD-CVRP results of instances (6, 7, 8, 14) (see Table 16). The improvements of the first group are not as good as for the second group. For example, the improvement in instance 1 for S5 amounts to 4.05% whereas the improvement in instance 6 for the same scenario amounts to 39.15 %. This difference arises from the tour length constraint. As already mentioned in Section 5.3, instances (1, 2, 3, 12) do not have a given tour length constraint. In this case it was required to determine an accurate one. However, it is very difficult to find a constraint which allows the TD-

CVRP results of (1, 2, 3, 12) to show similar characteristics as the TD-CVRP results of the other instances. A similarity is only achieved for S2 (see Table 16).

Table 17 summarizes the runtimes of the TD-CVRP for each instance and each scenario. *Avg Best* presents the average runtime to reach the best solution and whereas *Avg Total* describes the average total runtime. The averages are both based on five runs. The last row *Avg_Inst* shows the averages over all eight instances.

	S1		***S2***		***S3***		***S4***		***S5***	
Instance	***Avg Best***	***Avg Total***	***Avg Best***	***Avg Total***	***Avg Best***	***Avg Total***	***Avg Best***	***Avg Total***	***Avg Best***	***Avg Total***
C01	11	41	17	51	35	60	31	64	29	61
C02	22	60	34	95	26	84	16	73	77	127
C03	26	209	95	288	187	327	74	295	132	334
C06	9	61	18	80	24	92	20	81	10	73
C07	130	201	111	238	56	182	58	204	73	221
C08	210	325	81	343	113	340	154	353	205	357
C12	0	174	46	254	57	255	42	266	29	195
C14	4	222	38	279	70	260	6	247	60	268
Avg_Inst	52	162	55	204	52	52	50	198	77	205

Table 17: TD-CVRP runtimes

The average total runtimes of the TD-CVRP (*Avg Total* in Table 17) are much higher than the average total runtimes of the CVRP. This is due to the fact that more calculations are performed in the adapted algorithm.

Even if the improvements of the TD-CVRP results cannot cover the deteriorations of the CVRP results, it can be concluded that taking time-dependency into account offers a more accurate picture of real-world conditions than assuming constant travel times throughout the day. It is shown that the results which were obtained with the adapted algorithm are better than those which were obtained with the original algorithm. Furthermore, the TD-CVRP solutions satisfy all imposed constraints.

6. Conclusion

In the first part of this diploma thesis a theoretical overview of the TDVRP is presented. The overview starts with the definition of the VRP and the main solution methods. Next, the TDVRP is introduced and discussed. Finally, a review covering the TDVRP literature from 1991 until 2012 is presented.

In the second part of this diploma thesis, an algorithm based on Tabu Search is developed to solve the CVRP. Eight instances of Christofides et al. (1979) are taken as benchmark. The algorithm reaches the best known results for six instances.

The best solutions of the CVRP are then evaluated with five time-dependent scenarios, each representing a different degree of time-dependency. The time horizon is divided into three time intervals, where the first and the third interval correspond to lower travel speeds. It is shown that the total costs increase if compared to the CVRP results which were evaluated with time-dependent scenarios. The higher the degree of time-dependency, the higher is the increase of the total costs. Furthermore, several routes do not satisfy the tour length constraints anymore. These findings seem to be consistent with other research, e.g. Ichoua et al. (2003) who report that several solutions based on constant speeds are infeasible in the time-dependent context and that this number increases with the degree of time-dependency.

In the next step, the original algorithm is adapted to take time-dependency into account. The speed adjustment procedure of Ichoua et al. (2003) is used for calculating the route costs and the total times. In this way, the FIFO property is satisfied. Finally the adapted algorithm is performed to solve the TD-CVRP. The algorithm improves the CVRP results which were evaluated with time-dependent scenarios. The improvement is higher, the higher the degree of time-dependency. Furthermore, the solutions of the TD-CVRP satisfy all tour length constraints.

In conclusion it can be said that taking time-dependency into account offers a more accurate picture of real-world conditions than assuming constant travel times throughout the whole planning horizon. It is shown that the results which were obtained with the adapted algorithm are better than those which were obtained with the original algorithm. Furthermore, the TD-CVRP solutions satisfy all imposed constraints.

Bibliography

Ahn, B. H., Shin, J. Y.: Vehicle-Routing with Time Windows and Time-Varying Congestion, Journal of the Operational Research Society 42 (5), 393-400, 1991.

Balseiro, S.R., Loiseau, I., Ramonet, J.: An Ant Colony Algorithm Hybridized with Insertion Heuristics for the Time Dependent Vehicle Routing Problem with Time Windows, Computers & Operations Research 38, 954-966, 2011.

Bräysy, O., Gendreau, M.: Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms, Transportation Science 39 (1), 104-118, 2005a.

Bräysy, O., Gendreau, M.: Vehicle Routing Rroblem with Time Windows, Part II: Metaheuristics, Transportation Science 39 (1), 119-139, 2005b.

Christofides, N., Mingozzi, A., Toth, P.: The Vehicle Routing Problem, in: N. Christofides, A. Mingozzi and P. Toth (Eds.), *Combinatorial Optimization,* Wiley, Chichester, 315-338, 1979.

Cordeau, J.F., Laporte, G., Savelsbergh, M.W.P., Vigo, D.: Vehicle Routing, in: C. Barnhart and G. Laporte (Eds.), *Handbook in OR & MS*, Vol. 14, pp. 367-428, 2007.

Donati, A.V., Montemanni, R., Casagrande, N., Rizzoli, A.E., Gambardella, L.M.: Time Dependent Vehicle Routing Problem with a Multi Ant Colony System, European Journal of Operational Research 185 (3), 1174-1191, 2008.

Eglese, R. W., Maden, W., Slater, A.: A Road Timetable™ to aid Vehicle Routing and Scheduling, Computers & Operations Research 33, 3508-3519, 2006.

Ehmke, J.F., Steinert, A., Mattfeld, D.C.: Advanced routing for city logistics service providers based on time-dependent travel times, Journal of Computational Science, doi: 10.1016/j.jocs.2012.01.006, 2012.

Fleischmann, B., Gietz, M., Gnutzmann, S.: Time-varying Travel Times in Vehicle Routing, Transportation Science 38, 160-173, 2004.

Ford Jr., L.R., Fulkerson, D.R.: Constructing maximal dynamic flows from static flows, Operations Research 6, 419-433, 1958.

Gambardella, L.M., Taillard, E., Agazzi, G.: MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, in: D. Corne, M. Dorigo and F. Glover (Eds.), *New Ideas in Optimization*, McGraw-Hill, London, 63-76, 1999.

Gendreau, M., Laporte, G., Potvin, J.-Y.: Metaheuristics for the capacitated VRP, in: P. Toth and D. Vigo (Eds.), *The Vehicle Routing Problem*, SIAM monographs on discrete mathematics and applications, Philadelphia, Chapter 6, 129-154, 2002.

Haghani, A., Jung , S.: A Dynamic Vehicle Routing Problem with Time-Dependent Travel Times, Computers & Operations Research 32 (11), 2959-2986, 2005.

Hansen, P., Mladenović, N.: Variable Neighborhood Search: Principles and Applications, European Journal of Operational Research 130, 449-467, 2001.

Hill, A.V., Benton, W.C.: Modeling Intra-City Time-Dependent Travel Speeds for Vehicle Scheduling Problems, Journal of the Operations Research Society 43 (4), 343-351, 1992.

Ichoua, S., Gendreau, M., Potvin, J. Y.: Vehicle Dispatching with Time-Dependent Travel Times, European Journal of Operational Research 144, 379-396, 2003.

Laporte, G., Gendreau, M., Potvin, J.Y., Semet, F.: Classical and Modern Heuristics for the Vehicle Routing Problem, International Transactions in Operational Research 7, 285-300, 2000.

Laporte, G., Semet, F.: Classical Heuristics for the Capacitated VRP, in: P. Toth and D. Vigo (Eds.), *The Vehicle Routing Problem*, SIAM monographs on discrete mathematics and applications, Philadelphia, Chapter 5, 109-128, 2002.

Lenstra, J.K., Rinnooy Kan, A.H.G.: Complexity of Vehicle Routing and Scheduling Problems, Networks 11, 221-227, 1981.

Maden, W., Eglese, R., Black, D.: Vehicle Routing and Scheduling with time-varying data: a case study, Journal of the Operational Research Society 61, 515-522, 2010.

Malandraki, C., Daskin, M.S.: Time Dependent Vehicle-Routing Problems - Formulations, Properties and Heuristic Algorithms, Transportation Science 14, 26 (3), 185-200, 1992.

Marguier, P.H.J., Ceder, A.: Passenger Waiting Strategies for Overlapping Bus Routes, Transportation Science 18 (3), 207-230, 1984.

Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer Programming Formulation on Traveling Salesman Problems, Journal of the Association for Computing Machinery 7, 326-329, 1960.

Psaraftis, H.N., Tsitsiklis, J.N.: Dynamic Shortest Paths in Acyclic Networks with Markovian Arc Costs, Operations Research 41 (1), 91-101, 1993.

Schmid, V., Dörner, K.F.: Ambulance Location and Relocation Problems with Time-Dependent Travel Times, European Journal of Operational Research 207 (3), 1293-1303, 2010.

Toth, P., Vigo D.: An overview of Vehicle Routing Problems, in: P. Toth and D. Vigo (Eds.), *The Vehicle Routing Problem*, SIAM monographs on discrete mathematics and applications, Philadelphia, Chapter 1, 1-26, 2002.

Woensel, T.V., Kerbache, L., Peremans, H., Vandaele, N.: Vehicle Routing with Dynamic Travel Times: A queueing approach, European Journal of Operational Research 186, 990-1007, 2008.

Abstract

Most vehicle routing models assume constant travel times throughout the whole planning horizon. In reality, however, travel times vary during the day. This is especially true for urban areas where daily traffic congestion leads to longer travel times. The time-dependent vehicle routing problem (TDVRP) takes this aspect into account by assuming that travel times depend on the time of the day.

This diploma thesis gives an overview of the TDVRP and presents the results of an experimental study.
The first part introduces the VRP and different solution methods. This is followed by a detailed description of the TDVRP.
The second part of the thesis presents an algorithm based on tabu search to solve the capacitated VRP (CVRP). Afterwards, the best solutions of the CVRP are evaluated with five time-dependent scenarios, each representing a different degree of time-dependency. Compared to the original CVRP results, the total costs increase significantly and several routes become infeasible. In the next step, the original algorithm is adapted to solve the TD-CVRP. It is shown that the total costs can be improved when assuming time-dependent travel times. The improvement is higher, the higher the degree of time-dependency. Furthermore, the new solutions satisfy all tour length constraints.

Zusammenfassung

In der Tourenplanung wird meistens angenommen, dass die Reisezeiten während des gesamten Planungshorizonts konstant sind. In der Realität ist es jedoch so, dass es während des Tages zu variablen Reisezeiten kommt. Vor allem im urbanen Bereich führen Staus zu längeren Reisezeiten. Im tageszeitabhängigen Tourenplanungsproblem wird dieser Aspekt berücksichtigt indem man annimmt, dass die Reisezeiten von der Tageszeit abhängen.

Die vorliegende Diplomarbeit gibt einen Überblick über die tageszeitabhängige Tourenplanung und präsentiert die Ergebnisse einer experimentellen Studie.

Im ersten Teil dieser Arbeit werden das klassische Tourenplanungsproblem und verschiedene Lösungsverfahren vorgestellt. Danach wird das tageszeitabhängige Tourenplanungsproblem beschrieben.

Im zweiten Teil wird zunächst ein Algorithmus basierend auf der Tabu Suche entwickelt um das kapazitierte Tourenplanungsproblem zu lösen. Die Lösungen werden dann mit tageszeitabhängigen Szenarien evaluiert, wobei jedes Szenario einen anderen Grad an Zeitabhängigkeit repräsentiert. Es wird gezeigt, dass die Gesamtkosten im Vergleich zu den ursprünglichen Kosten steigen. Desweiteren werden die Tourlängenbeschränkungen von vielen Touren nicht mehr erfüllt. Schließlich wird der ursprüngliche Algorithmus adaptiert um das tageszeitabhängige kapazitierte Tourenplanungsproblem zu lösen. Es wird gezeigt, dass die Gesamtkosten verbessert werden können wenn man tageszeitabhängige Reisezeiten einsetzt. Die Verbesserung ist umso stärker, je höher der Grad an Zeitabhängigkeit. Zusätzlich erfüllen die neuen Lösungen alle Tourlängenbeschränkungen.

Curriculum Vitae

PERSONAL DATA

Irena Ilić
born on 10. 11. 1982 in Mödling, Austria

EDUCATION

since 10/2002	University of Vienna, Austria International Business Administration Production Management and International Management Diploma Thesis: „The Time-Dependent Vehicle Routing Problem”
10/2001 - 06/2002	University of Vienna, Austria Journalism and Communication Studies
09/1993 - 06/2001	BRG Baden

PROFESSIONAL EXPERIENCE

since 03/2011	Quintiles Eastern Holdings GmbH Dep. Office Management

LANGUAGE SKILLS

German	native
Serbian	native
English	fluent
French	good
Italian	basic

COMPUTER SKILLS

MS Office, SAP R/3, C++