Branch-and-price approach to the vehicle routing problem with hard time windows and stochastic travel times

Fasting, L.A.

Award date:
2014

TECHNISCHE UNIVERSITEIT EINDHOVEN

Branch-and-Price Approach to the Vehicle Routing Problem with Hard Time Windows and Stochastic Travel Times

Lloyd A. Fasting

October 27, 2014

Supervisors:
dr. M. Firat
dr.ir. M.A.A. Boon
J. van Twist MSc. (Quintiq)

Studentnummer:
0678205

TU/e QUINTIQ

Abstract

In this thesis the well-known Vehicle Routing Problem with Hard Time Windows (called VRPhTW in this thesis) is extended with stochastic travel times. This problem is denoted by the VRPhTW-STT. The probability that service at a customer starts before its due date must be greater than a required value. The optimal solution of the VRPhTW-STT is a set of feasible routes which minimizes the number of vehicles and the travelled distance, hierarchically in the respected order.

A Branch-and-Price algorithm is described to solve the VRPhTW-STT to optimality. An efficient method for calculating with stochastic travel times is developed. Furthermore, to create instances based on real data, travel time distributions are constructed using empirical travel time data from TomTom.

Acknowledgment

This thesis is the result of a nine month graduation project at Quintiq in cooperation with Eindhoven University of Technology (TU/e). It was part of the DAIPEX project (Data and Algorithms for Integrated Transportation Planning and Execution), which is a collaboration between Quintiq, TU/e, TomTom, and four logistic companies.

I want to thank TomTom for allowing me to use their data on travel time distributions. Thanks to the people from the Optimization Center and the R&D department from Quintiq for their help and advice. Wim Nuijten, thanks for introducing me to Quintiq, and for your valuable advice. I owe great gratitude towards my supervisors: Joost van Twist, Murat Firat and Marko Boon. Joost, your pleasure in solving problems, and your realistic view on the subject, encouraged me a lot during my time at Quintiq. Murat, besides your great help with the mathematical and implementational side of the problem, and your exceptional intuition for solving problems, you showed me what it takes to do scientific research. Marko, thanks for your continuous curiosity, and your expertise, which helped me with the stochastic parts of the problem.

I want to thank my friends, Omar, Rick and Irene for their appreciated distractions. I want to thank my parents for their support at all times. Madelon, without you this would not have been possible.

Contents

Chapter 1

Introduction

One of the big challenges companies face today is improving efficiency. Costs should be reduced while the quality is maintained. Making planning decisions is often computationally hard. Therefore, advanced planning methods are needed to determine the best strategy for the near and distant future. One of those hard planning problems is known as the Vehicle Routing Problem with Time Windows (VRPTW).

A fleet of identical vehicles with a fixed capacity is located at a central depot. Furthermore, there is a set of customers. Every customer has to be visited exactly once within a predefined time window by one of the vehicles. A time window consists of a *release date* and a *due date*. In transportation problems two types of time window handling can occur. Time windows are called *soft* when service can start before the release date, or after the due date. However, this induces a violation cost, which penalizes early or late service. When service must start within the time windows, time windows are called *hard*. When a vehicle arrives at a customer before its release date it is required to wait until the release date before service can start. A vehicle is not allowed to arrive after the due date. Each vehicle starts and ends at the depot. A typical solution can be found in figure 1.1.

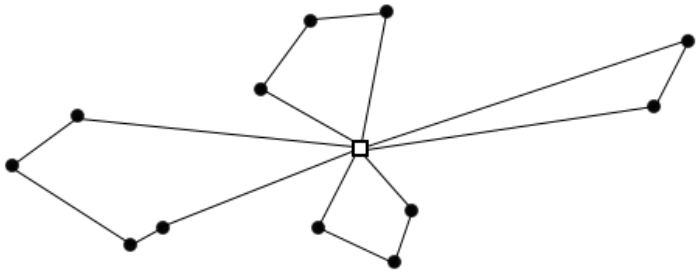


FIGURE 1.1: A typical solution of a VRP instance. The square denotes the depot, the dots represent the customers.

In the classical formulation of the VRPTW it is assumed that all elements are deterministic. But, uncertainty plays a crucial role in real life application. Therefore, stochastic versions of the VRPTW are studied to obtain more realistic and robust solutions. Various types of stochastic variations of the VRPTW are discussed in Chapter 2.

Generally the term VRPTW denotes the VRP with hard time windows. But to make a clear distinction between problems with hard time windows, and soft time windows we introduce the following notation:

- **VRPsTW**: VRP with soft time windows
- **VRPhTW**: VRP with hard time windows

In this thesis we assume time windows are hard, and travel times are stochastic. For example, variations in traffic congestion influence travel times of delivery trucks. This problem is known as the VRP with Hard Time Windows and Stochastic Travel Times (VRPhTW-STT). This means that the time to get from one customer to another is a random variable. As a consequence, the arrival time at a customer is also random. This highly complicates the problem, since time windows are hard. The event of arriving before the due date has no *TRUE / FALSE* answer anymore like in the deterministic case. Instead, there is a probability of arriving in time. Now a new concept of *reliability* arises. Service at a customer is said to be *reliable* if the probability that service starts before its due date is greater than its required reliability. Important customers could be given a higher reliability requirement to reflect business interests. The objective of the VRPhTW-STT is: Find a feasible solution that minimizes the number of vehicles and the total travelled distance, while arrival is reliable at every customer. A feasible solution satisfies the following: No vehicle can have more cargo assigned than its capacity, every vehicle starts and ends at the depot, all customers are visited exactly once, and service is reliable at all customers.

An application of the VRPhTW-STT can be found at package delivery companies. Suppose a package delivery company makes deliveries to customers in a city center. The city centre is not always accessible for vehicles due to traffic regulations. Therefore, every customer has a fixed time window in which deliveries can be made. However, travel times do not have a fixed value which is known on beforehand. A good planning should take this uncertainty into account.
For solving real life instances it is generally hard to find suitable travel time distributions between customers. The increase in data collection by navigation systems in cars offers numerous possibilities. In this thesis historical data collected by TomTom is used for constructing travel time distributions.

In Chapter 2 an overview of the current state of the literature is given. The problem is mathematically formulated in Chapter 3, and an exact solution method is described. A difficult part of the VRPhTW-STT is doing computations with travel time distributions. Therefore, in Chapter 4 we propose methods of representing distributions and how to compute with them. The creation of stochastic instances based on real travel time data from TomTom is discussed in Chapter 5. The algorithm is computationally analysed in Chapter 6. Finally, we derive our conclusions, and formulate our recommendations for further research.

Chapter 2

Literature study

The classical Vehicle Routing Problem (VRP) was introduced in 1959 by Dantzig and Ramser [5]. It is ever since widely studied, just as its numerous variations. Since the VRP is a generalisation of the Traveling Salesman Problem (TSP) which is NP-hard, the VRP is also NP-hard. Moreover, the VRP with time windows (VRPTW) is a generalisation of the classical VRP, so is NP-hard too. A description of the VRPTW was given by Cordeau et al. in "The Vehicle Routing Problem". An important distinction should be made between articles on *hard* an *soft* time windows. Soft time windows can be violated, and are handled by some authors in a way that affects the costs by penalizing time window violations. As a consequence, since lateness does not influence feasibility, the problem has the same feasibility set as the VRP, only a different cost function.
Hard time windows can not be violated. Therefore, hard time windows induce a smaller feasible set of solutions. Solution methods in the literature differ significantly for the two cases. Problems with hard time windows are more complex. The classical VRP problem is a special case of the VRPhTW when time windows are taken equal to the planning horizon.

To determine the quality and the speed of an algorithm benchmark instances are used. The first set containing 14 instances was called CMT and named after Christofides, Mingozzi and Toth (1979). The second set, named GWKC, contains 20 instances and was constructed by Golden et al. (1998). The sets of benchmark instances mostly used nowadays for the VRPhTW and the VRPsTW are the Solomon instances (100 customers per instance) (Solomon [22]) and the Homberger instances (1000 customers per instance) (Homberger et al. [13]). They provide us with a variety of problem types. Some having broad time windows, others tight; some having large capacities, and others small.

Heuristic methods for the VRPhTW

Because of the complexity of the problem most algorithms used in practice are heuristics. The paper of Dantzig and Ramser introduced the first heuristic for the VRP, alongside its first formulation. The heuristic constructs vehicles routes by solving a sequence of linear programs. Another example of a heuristic method is the well-known Savings Algorithm by Clarke and Wright [2]. Its popularity comes from the ease of implementation and the reasonable quality of its solutions. The algorithm starts with an initial solution, after which iteratively routes are merged. The merge resulting in the largest saving is chosen, considered the obtained solution is feasible.
The "Cluster-First, Route-Second" algorithm was introduced by Fisher and Jaikumar [11]. First, clusters of customers are determined satisfying the capacity constraint. Secondly, routes are constructed by solving TSP's. Finally, a large category of heuristics is based on a Local Search approach. Starting

with an initial solution the algorithm explores the solution space by moving from the current solution to another. Examples are simulated annealing and tabu search.

Exact algorithms for the VRPhTW

Besides heuristic methods, a large part of research is done on exact methods for the VRPTW. Exact methods are generally more time consuming. They need time to proof that no better solution exists. Exact methods use techniques to avoid an enumeration of all possible solutions.

The first paper proposing an exact method was published by Kolen et al. [16]. A Branch-and-Bound method was used to solve instances up to 15 customers to optimality. Even though computer power was considerably less at that time, Branch-and-Bound was not efficient enough for attacking larger problem instances. Therefore more advanced methods were needed such as Dynamic Programming (DP) and Column Generation (CG).

The solution method proposed in this thesis is based on CG. In the current literature CG based methods are the primal way of solving the VRPTW to optimality (Dabia [4],Kallehauge [14],Larsen [17],Taş [24]). One of the first articles addressing CG was published in 1992 by Desrochers et al. [7]. They proposed "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows" based on CG in combination with a Branch-and-Bound search. In CG the problem is spit into a Master Problem and a Sub Problem. A detailed description of these two problems is given in this thesis. Most computational challenges arise in the Sub Problem. Current research on solving the VRPTW with CG is mainly focussed on developing methods for accelerating the Sub Problem.

A Branch-and-Bound search is used for obtaining integer solutions. A Branch-and-Bound search in which lower bounds at the nodes are found using CG is called Branch-and-Price. The incorporation of CG in a Branch-and-Bound tree seems straightforward, but it has some challenges. For example, we must make sure the branching decisions do not change the structure of the Sub Problem. The ways to handle these pitfalls are described in Chapter 3.

Stochastic Travel Times in Vehicle Routing

Various stochastic versions of the VRP have been studied in the last two decades. Research has been done on stochastic service times (Errico [9],Sungur et al. [23]), stochastic demands (Bertsimas 1992, Laporte et al. 2002), stochastic travel times (Conijn [3],Samaranayake [21], Taş [24]), or a combination of them. In this thesis we focus on stochastic travel times.

An easy way of representing a distribution is by its expected value. When the expected value of a travel time distribution is used to represent it, one could still use deterministic solvers. But the expected values is not a sufficient statistic for representing the uncertainty in the problem [21]. Therefore, we must calculate with distributions to include variability.
The papers on stochastic travel times use various methods such as Dynamic Programming, Robust Optimization and chance-constrained programming. The complexity of the problem heavily depends on the choice of time window handling. Hard time windows induce a smaller feasible set of solutions, since

soft time windows allow late servicing with a penalty. Having hard time windows makes the problem extremely challenging [9].

VRPTW-STT with soft time windows

In (Laporte et al., 1992) a VRP with stochastic service and travel times is addressed with a soft constraint which bounds the route duration. A similar problem is described by Kenyon and Morton [15], but the objective has changed. Besides minimizing the expected route duration, also the probability of violating the completion deadlines is minimized. Both travel times and service times are stochastic. Sungur et al. [23] propose a solution method for the VRP with soft time windows, uncertainty in the customer presence and the service times. Their solution method is based on a combination of robust optimization and stochastic programming. Larger instances are solved using an insertion heuristic and tabu search. Another heuristic method for stochastic travel and service times with soft time windows is described by Lei et al. [19]. A penalty cost is given when service is not started within the time windows. Also D. Tas [24] uses this approach. Stochastic components are included in the objective function. Furthermore, travel times are assumed to be Gamma distributed. This assumption simplifies the addition of random variables since adding two Gamma distributions results in a Gamma distribution again. The problem is solved by both an heuristic method and an exact method based on Column Generation.

VRPTW-STT with hard time windows

This version of the VRPTW-STT has gained interest much later, so only little research has been done so far. There are even less studies on exact methods for the VRPhTW-STT. When time windows are hard uncertainty can not enter the model in the objective function. Furthermore, arrival distributions are truncated distributions, because of the release date which is a hard lower bound on the start of service time.

Chang et al. [1] published an article on heuristic methods for the TSP with hard time windows and stochastic travel times and service times. Travel times are assumed to be normally distributed, which may not be a very realistic assumption. In a typical travel time distribution one finds a steep rise and a long tail. There is a minimal time required to traverse a path. Furthermore, because of traffic congestion high travel times occur with non-zero probability. Erera et al. [9] considered the VRP with hard time windows and stochastic demands. Errico et al. [8] use a chance-constrained optimization model with a probabilistic constraint on the reliability of a vehicle route. This is different from our approach, since we have a reliability constraint at *every* customer. When the planned route appears to be infeasible during the operations, their algorithm takes into account the corrective actions that retrieve feasibility. Two recourse actions are considered upon late arrival at a customer: skipping service of the current customer, or skipping service of the next customer. Because of the recourse actions, an expression for the expected cost of a route is derived. Also an expression is derived for the probability that a route is feasible. The problem is formulated as a set partitioning problem, and solved using a Branch-Cut-and-Price algorithm. Service time distributions are modelled as discrete distributions with finite support.

Researchers have also been interested in the approximate shape of travel time distributions. In (Lecluyse et al. [18]) it is argued that travel times are approximately Log-normal distributed. Conijn [3] generates test instances based on the Solomon instances by assigning a shifted Log-normal distribution to every arc. The parameters of the distributions are based on the deterministic travel distances. Distributions are modelled by approximating the CDF with piecewise polynomial functions of third degree. An expression

Author(s)	Travel Time Det.	Travel Time Sto.	Service Time Det.	Service Time Sto.	Time Windows Soft	Time Windows Hard	Solution Method
M. Desrochers et al. (1992)	x		x			x	Branch-and-Price
J. Larsen (1999)	x		x			x	Branch-and-Price
P. Toth et al. (2001)	x		x			x	Overview of various solution methods
B. Kallehauge (2005)	x		x			x	Lagrangian Duality / Branch-and-Price
S. Dabia et al. (2012)	x		x			x	Branch-and-Price
A. Erera et al. (2009)	x		x			x	Planning methods for stochastic demands
Sungur et al. (2010)	x			x	x		Heuristic method
Lei et al. (2012)	x			x	x		Heuristic method
F. Errico (2014)	x			x		x	Branch-cut-and-Price
D. Tas (2013)		x	x		x		Tabu Search / Branch-and-Price
S. Samaranayake (2011)		x	x			x	Routing policies
B. Conijn (2013)		x	x			x	Nearest Neighbor heuristic
A. Kenyon et al. (2003)		x		x	x		Branch-and-cut with Monte Carlo simulation
T. Chang et al. (2009)		x		x		x	Heuristic method
Our problem (VRPhTW-STT)		x	x			x	Branch-and-Price

Table 2.1: Overview of the related literature

is found for the convolution of two piecewise polynomial functions. The problem is solved by an heuristic method.

An overview of the articles previously discussed can be found in Table 2.1.

In this thesis we will propose an exact method to solve the VRPhTW-STT. Similar to (Errico et al. [9]) distributions are assumed to be discrete. This method can be applied regardless of the chosen travel time distribution. The advantage of this method is that we do not make explicit assumptions about the travel time distributions, it is easy to deal with truncation, and it is extremely suitable in cases where actual travel time data is available. Furthermore, our algorithm can analogously be extended with stochastic service times, since computations will only change slightly.

Chapter 3

Set partitioning approach to the VRPhTW-STT

Problem statement
There is a directed graph $G = (N, A)$ that denotes the network, consisting of nodes $N = \{0, 1, ..., n\}$ and arcs $A = \{(i, j)|i, j \in N, i \neq j\}$. Nodes $\{1, ..., n\}$ represent customers and node 0 represents the depot. Every arc (i, j) has a cost c_{ij} which is equal to its distance. Every customer $i \in N$ has a demand of quantity $0 < q_i$. At every customer service must start within its hard time window $[r_i, d_i]$. In the case of early arrival a vehicle needs to wait until the time window opens at r_i. The time window of the depot, $[r_0, d_0]$, implies the planning horizon of the problem. Every customer has a fixed service time denoted with s_i. V indicates the set of homogeneous vehicles, each with capacity Q.

Let random variable T_{ij} represent the travel time from customer i to j, and S_j the start of service time at customer j. Since the service reliability must be satisfied at every customer, it is required that:

$$\mathbb{P}[S_j \leq d_j] \geq \gamma_j \quad \forall j \in N.$$

Our goal is to minimize the number of vehicles and the total travelled distance hierarchically in the respected order. We define our objective as the sum of the number of vehicles used and the total travelled distance. The number of vehicles is multiplied with a high coefficient M_0 to satisfy the hierarchy of the minimization objective. Practically this is done by modifying the costs of the arcs leaving the depot as follows:

$$c_{0i} := M^0 + c_{0i} \quad \forall i \in N \backslash \{0\}.$$

The structure of this chapter is as follows. First a MIP formulation of the deterministic VRPhTW is given in section 3.1. Subsequently, based on this formulation the problem is reformulated as a Set Partitioning Problem, which allows the VRPhTW-STT to be solved using Column Generation. Finally, our solution method is described in section 3.3.

3.1 MIP formulation of the VRPhTW

We first formulate the deterministic VRPhTW as a Mixed Integer Program (MIP) model. A solution can be fully described by a subset of A, the set of arcs. The MIP searches for a subset of A minimizing the sum of arc costs. Notation, parameters and decision variables can be found in Table 3.1. Note that the

travel time t_{ij} is constant in this model, in contrast to the random travel time T_{ij} in the actual model.

TABLE 3.1: Sets, indices, parameters and variables of the MIP

Sets included in the model	
N	Set of customers (including the depot as customer 0)
A	Set of arcs
V	Set of vehicles
Indices	
i, j	Customer indices
Nonnegative Parameters	
$[r_i, d_i]$	Hard time window of customer i
s_i	Service time of customer i
q_i	Demand of customer i
Q	Capacity of a vehicle
c_{ij}	Travel distance from customer i to j
t_{ij}	Travel time from customer i to j. We assume that traveling one unit of distance takes one unit of time. So we have $t_{ij} = c_{ij} \;\; i \neq j \in N$
Variables	
x_{ij}	Equals 1 if arc (i, j) is used in the solution, and 0 otherwise
A_i	Arrival time at customer i, $i \in N\backslash\{0\}$
D_i	Departure time at customer i, $i \in N\backslash\{0\}$ By convention we have $A_0 = D_0 = 0$
L_i	Load of the vehicle when arriving at customer i, $0 < L_i \leq Q$

$$MIP: \quad \min \sum_{(i,j)\in A} c_{ij}x_{ij} \quad s.t. \tag{3.1}$$

$$\sum_{j\in N} x_{ij} = 1 \quad \forall i \in N\backslash\{0\} \tag{3.2}$$

$$\sum_{i\in N} x_{ik} - \sum_{j\in N} x_{kj} = 0 \quad k \in N \tag{3.3}$$

$$L_i \leq Q \quad \forall i \in N \tag{3.4}$$

$$A_i + s_i \leq D_i \quad \forall i \in N \tag{3.5}$$

$$r_i + s_i \leq D_i \quad \forall i \in N \tag{3.6}$$

$$A_i \leq d_i \quad \forall i \in N \tag{3.7}$$

$$L_i - q_i \geq L_j - d_0(1 - x_{ij}) \quad \forall i \in N\backslash\{0\}, \forall j \in N\backslash\{i\} \tag{3.8}$$

$$D_i + t_{ij} \leq A_j + d_0(1 - x_{ij}) \quad \forall i \in N, \forall j \in N\backslash\{i\} \tag{3.9}$$

$$\sum_{i\in N} x_{0i} \leq |V| \tag{3.10}$$

$$x_{ij} \in \{0, 1\} \quad i \in N, j \in N \tag{3.11}$$

$$0 \leq A_i \;\; 0 \leq D_i \;\; 0 \leq L_i \quad i \in N \tag{3.12}$$

Constraints (3.2) make sure one unit of flow is leaving every customer. Together with the flow conservation constraints (3.3) every customer is visited exactly once. (3.4) are the capacity constraints. This is

done by stating that the load of the vehicle when departing from the depot cannot exceed its capacity. When $x_{0i} = 0$ for some $i \in N\backslash\{0\}$ the constraint becomes redundant. Constraints (3.5), (3.6) and (3.7) make sure the time windows are satisfied. Constraints (3.8) are needed to make sure that the load in each vehicle is large enough to satisfy all the demands of the customers in its route. We make use of a large number to linearize the constraints. In this case we take it equal to the upper bound on the planning horizon, d_0. If $x_{ij} = 0$ the constraint is automatically satisfied. Constraints (3.9) are the time update constraints. Finally, constraint (3.10) bounds the number of vehicles used.

Solving the MIP can be a method to obtain optimal solutions for instances with up to 25 customers. A computational analysis can be found in Chapter 5 and in Appendix A.2. For larger instances one should use smart enumeration methods.

To solve the VRPhTW-STT a MIP can not be directly used, since travel times are given by distributions. However it is possible to incorporate uncertainty into a MIP (for example using robust optimization), these techniques are computationally not attractive when full travel time distributions are used. Therefore a reformulation of the problem is needed and a corresponding solutions method. This will be the topic of the next sections.

3.2 Reformulation with set partitioning structure

In the MIP formulation every arc has a corresponding integer valued variable. A solution was represented by a set of arcs. A popular and effective formulation contains route variables. This formulation allows us to include stochastic travel times into the model. Most articles on solving the VRPTW to optimality, such as [6, 9, 14, 17, 24, 25], use a formulation in terms of routes. This formulation will enable us to solve our problem when travel times are stochastic, since this changes the definition of feasible routes.

A route is a sequence of customers visited by one vehicle. A vehicle starts and ends at the depot. The number of customers in a route is limited because of the vehicle capacity, and because of the limited planning horizon. A customer can only be visited once in a route. For every customer in a route the probability of arriving before the deadline must be greater than the required reliability threshold of that customer. A route which satisfies these criteria is called *feasible*.

An optimal solution to the VRPhTW-STT is a set of feasible routes with minimal distance, such that every customer is visited exactly once. Mathematically this can be formulated as a problem with a set partitioning structure (3.13) – (3.16). Notation, parameters and variables of the set partitioning problem (SPP) can be found in Table 3.2.

$$SPP: \quad \min \sum_{r \in \mathcal{R}^S} c_r y_r \qquad s.t. \tag{3.13}$$

$$\sum_{r \in \mathcal{R}^S} \delta_r^i y_r = 1 \quad i \in N\backslash\{0\} \tag{3.14}$$

$$\sum_{r \in \mathcal{R}^S} y_r \leq |V| \tag{3.15}$$

$$y_r \in \{0,1\} \quad r \in \mathcal{R}^S \tag{3.16}$$

TABLE 3.2: Sets, indices, parameters and variables of the reformulation with set partitioning structure

Sets included in the model	
$\mathcal{R}^{\mathcal{S}}$	Set of all feasible vehicle routes for the VRPhTW-STT
Indices	
i	Customer index
r	Route index
Nonnegative Parameters	
δ_r^i	Equals 1 if customer i is visited by route r, and 0 otherwise
c_r	Cost of route r
$\|V\|$	Size of the fleet of vehicles
Variables	
y_r	Equals 1 if route r is used in the solution, and 0 otherwise

Constraint (3.14) makes sure every customer is visited once in the solution. Constraint (3.15) bounds the number of vehicles to the size of the fleet. (3.15) can be omitted if the size of the fleet is not limited. The binary requirement on the route decision variables is imposed by (3.16). The route set $\mathcal{R}^{\mathcal{S}}$ includes feasible route with respect to vehicle capacity, customer uniqueness, and service reliability.

Derivation: from arcs to routes

We will now give the mathematical derivation of the SPP (3.13) – (3.16).

Let a route be described by a sequence of customers:

$$r = \{\lambda_0, \lambda_1, \lambda_2, ..., \lambda_{n+1}\},$$

where $\lambda_1, ..., \lambda_n \in N$ and λ_0 and λ_{n+1} represent the depot. Binary parameters $\delta_i^r, \ \ i \in N$, take value 1 if customer i is visited by route r, and 0 otherwise. Furthermore, a cost a assigned to a route. The cost of a route is defined as follows:

$$c_r = \sum_{i=0}^{n} c_{\lambda_i, \lambda_{i+1}}, \tag{3.17}$$

where c_{ij} is the cost of arc (i, j). For feasibility of r, at least it must hold that:

- $\sum_{i=1}^{n} q_{\lambda_i} \leq Q$, i.e. the capacity is not exceeded (3.18)
- $\lambda_i \neq \lambda_j \ \ \forall i, j \in 1, ..., n$, i.e. the route contains no cycles (3.19)

Let the set $\mathcal{R}$ be defined as follows:

$$\mathcal{R} = \{ \ r \ \ | \ \ \text{(3.18) and (3.19) are satisfied by } r\}.$$

Furthermore, feasibility of a route depends on the start of service times at its customers. Therefore, we distinguish conditions **(D.1)** and **(S.1)**, for the VRPhTW with deterministic and stochastic travel

times, respectively.

D.1 : Service at λ_i starts within $[r_{\lambda_i}, d_{\lambda_i}]$ $\quad \forall i = 1, ..., n+1 \in r$ (3.20)

S.1 : The probability that service starts before d_{λ_i} is bigger than γ_{λ_i} $\quad \forall i = 1, ..., n+1 \in r$ (3.21)

Notice that the only difference between deterministic and stochastic feasibility is the time condition. Vehicle capacity 3.18 and customer uniqueness 3.19 are checked similarly. Let $\mathcal{R}^{\mathcal{D}}$ and $\mathcal{R}^{\mathcal{S}}$ be defined as:

$$\mathcal{R}^{\mathcal{D}} = \{\ r\ \in \mathcal{R}\ \ |\ \ (3.20) \text{ is satisfied by } r\}.$$

$$\mathcal{R}^{\mathcal{S}} = \{\ r\ \in \mathcal{R}\ \ |\ \ (3.21) \text{ is satisfied by } r\}.$$

So $\mathcal{R}^{\mathcal{D}}$ is the feasible route set for the deterministic VRPhTW, and $\mathcal{R}^{\mathcal{S}}$ for the VRPhTW-STT. If $\mathcal{R}^{\mathcal{S}}$ is replaced by $\mathcal{R}^{\mathcal{D}}$ in the reformulation, it solves the VRPhTW. Solving the VRPhTW-STT means finding a subset $R^{sol} \subset \mathcal{R}^{\mathcal{S}}$ which minimizes the sum of route costs.

Let the binary decision variable of route r be defined as follows:

$$y_r = \begin{cases} 1 & \text{if route } r \text{ is in the solution} \\ 0 & \text{otherwise.} \end{cases}$$

To change from a formulation of arcs to a formulation of routes, arcs and routes should be linked. Therefore we define the following variables:

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

Using this notation we can rewrite δ_i^r and x_{ij} as follows:

$$\delta_i^r = \sum_{j \in N} x_{ijr} \tag{3.22}$$

$$x_{ij} = \sum_{r \in \mathcal{R}^{\mathcal{S}}} x_{ijr} y_r. \tag{3.23}$$

In the arc based formulation, constraints (3.3) – (3.9) are feasibility constraints. They do not influence the selection of routes in the reformulation, since we assume all routes in $\mathcal{R}^{\mathcal{S}}$ are feasible. Thus, using the defined notation, we are going to reformulate the remaining parts, (3.1), (3.2), (3.10), and (3.11), as a formulation with route decision variables.

The objective (3.1) is rewritten as follows:

$$\begin{aligned} \min \sum_{(i,j) \in A} c_{ij} x_{ij} &= \min \sum_{(i,j) \in A} c_{ij} \Big(\sum_{r \in \mathcal{R}^l} x_{ijr} y_r \Big) \\ &= \min \sum_{r \in \mathcal{R}^{\mathcal{S}}} \sum_{(i,j) \in A} c_{ij} x_{ijr} y_r. \\ &= \min \sum_{r \in \mathcal{R}^{\mathcal{S}}} c_r y_r \end{aligned} \tag{3.24}$$

Furthermore, the customer constraint (3.2) changes to:

$$\begin{aligned}\sum_{j\in N} x_{ij} &= \sum_{j\in N}\sum_{r\in\mathcal{R}^{\mathcal{S}}} x_{ijr}y_r \\ &= \sum_{r\in\mathcal{R}^{\mathcal{S}}} \delta_i^r y_r = 1 \quad \forall i \in N\backslash\{0\}. \end{aligned} \tag{3.25}$$

Using the fact that $\delta_0^r = 1$, since every route starts and ends at the depot, the constraint on the number of vehicles (3.10) changes to:

$$\begin{aligned}\sum_{j\in N} x_{0j} &= \sum_{j\in N}\sum_{r\in\mathcal{R}^{\mathcal{S}}} x_{0jr}y_r \\ &= \sum_{r\in\mathcal{R}^{\mathcal{S}}} \delta_0^r y_r. \\ &= \sum_{r\in\mathcal{R}^{\mathcal{S}}} y_r \leq |V| \end{aligned} \tag{3.26}$$

Ultimately, the binary decision variable (3.11) becomes:

$$y_r \in \{0,1\} \tag{3.27}$$

In total, (3.24) – (3.27) form together the reformulated problem with route variables (the SPP), as described in (3.13) – (3.16).

Restricted Master Problem

Solving the SPP to optimality results in the optimal solution to the VRPhTW-STT. But the SPP is to complex to solve directly. The SPP does not posses the duality property since it is an Integer Program (IP). To accomplish this, variables y_r will be relaxed. The resulting problem is known as the *Master Problem* (MP).
Moreover, there is another complication in solving the SPP. The set $\mathcal{R}^{\mathcal{S}}$ grows exponentially in the number of customers. It is therefore not possible to use the whole set $\mathcal{R}^{\mathcal{S}}$ as basis for the optimization problem. To overcome this, we only use a subset of $\mathcal{R}^{\mathcal{S}}_{res} \subset \mathcal{R}^{\mathcal{S}}$ as a basis. Because of the restriction on the set of routes in the basis, the problem is called the *Restricted Master Problem* (RMP).

Additional notation for the RMP can be found in Table 3.3.

Table 3.3: Sets, indices, parameters and variables of the RMP

Sets included in the model	
$\mathcal{R}^{\mathcal{S}}_{res}$	Restricted set of feasible vehicle routes for the VRPhTW-STT
Variables	
y_r	Continuous route variable $\in (0,1)$ of route r

The mathematical formulation of the RMP is:

$$RMP: \quad \min \sum_{r \in \mathcal{R}^{\mathcal{S}}_{res}} c_r y_r \qquad s.t. \tag{3.28}$$

$$\sum_{r \in \mathcal{R}^{\mathcal{S}}_{res}} \delta^i_r y_r = 1 \quad i \in N \backslash \{0\} \tag{3.29}$$

$$\sum_{r \in \mathcal{R}^{\mathcal{S}}_{res}} y_r \leq |V| \tag{3.30}$$

$$0 \leq y_r \quad r \in \mathcal{R}^{\mathcal{S}}_{res}. \tag{3.31}$$

Constraints (3.29) make sure every customer is visited exactly once. Constraint (3.30) bounds the number of vehicles used by the size of the fleet V. Constraint (3.31) is written without an upper bound (≤ 1) since (3.29) already takes care of this.

3.3 Column Generation

The RMP delivers a partitioning of the routes in its base set of minimal cost. In general, the restricted set does not contain all routes of the optimal solution. Therefore, new routes should be added to satisfy convergence to the optimal solution.

In this section we will see that promising routes are found based on the dual values of the RMP. The method of adding routes to the basis of the RMP until it is solved optimally is called Column Generation (CG). CG is a method for solving large scale optimization problems. The fundamental idea of CG is to only work with a sufficiently meaningful subset of the variables [7]. The RMP solves a linear programming (LP) relaxation of the master problem using only this subset. Promising variables are found to enter the basis only when needed by solving a Sub Problem. More details about the Sub Problem will be given in de proceedings of this section.

One iteration of CG consists of:

- Optimizing the RMP and obtaining the current objective value $\bar{z}$ and dual values π^*
- Finding one or more variables to enter the basis of the RMP by solving a Sub Problem

A graphical overview of the CG procedure can be found in Figure 3.1.

The classic example of a problem which can be solved with CG is the Cutting Stock problem. Imagine a paper factory produces rolls of paper having a fixed width. Customers order rolls of different widths. What is the best way of cutting to satisfy the customer needs? An important part of solving the problem via CG is finding a suitable formulation. In this case a variable is not assigned to a customer order, but to a cutting pattern of a roll. Similarly, for the VRPTW a suitable formulation must be found.

The remainder of this section is dedicated to a detailed description of the CG method for the VRPhTW-STT.

After solving the RMP to optimality dual variables (π, ν) can be obtained, also called shadow prices. Every constraint in the primal problem (RMP) has a corresponding variable in its dual problem, and every

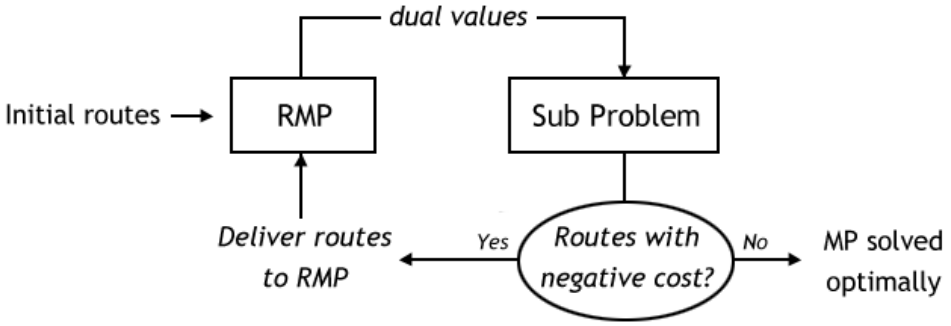


FIGURE 3.1: Schematic overview of the CG procedure

variable in the primal problem has a corresponding constraint in its dual problem. When the primal problem is solved, the dual values are available as a property of the optimal solution. $\pi \in \mathbb{R}^{|N\setminus\{0\}|}$ corresponds to constraints (3.29), and ν corresponds to constraint (3.30) in the primal problem.

Let us now state the dual problem of the RMP (3.28) – (3.31):

$$\textit{Dual of RMP}: \quad \max \sum_{i \in N\setminus\{0\}} \pi_i - \nu \quad s.t. \tag{3.32}$$

$$\sum_{i \in N\setminus\{0\}} \delta_r^i \pi_i - \nu \leq c_r \quad r \in \mathcal{R}_{res}^{\mathcal{S}} \tag{3.33}$$

$$0 \leq \pi_i \quad i \in N\setminus\{0\}. \tag{3.34}$$

From constraint (3.33) it follows that the dual problem is infeasible if for at least one route $r \in \mathcal{R}_{res}^{\mathcal{S}}$ it holds that

$$\sum_{i \in N\setminus\{0\}} \delta_r^i \pi_i^* - \nu^* > c_r \tag{3.35}$$

$$\iff c_r - \sum_{i \in N\setminus\{0\}} \delta_r^i \pi_i^* + \nu^* < 0. \tag{3.36}$$

This value is called the *reduced cost* $\bar{c}_r$ of route r. So the reduced cost of route r is defined as follows:

$$\bar{c}_r = c_r - \sum_{i \in N} \pi_i^* \delta_r^i + \nu^* \tag{3.37}$$

Optimality property

From the strong duality theorem we know the following: If y^* is a feasible solution to the primal problem with objective value $c^T y^*$, and (π^*, ν^*) a feasible solution to the dual problem with objective value ${\pi^*}^T I - \nu^*$, and it holds that $c^T y^* = {\pi^*}^T I - \nu^*$, then y^* is the optimal solution to the primal problem, and (π^*, ν^*) is the optimal solution to the dual problem.

According to (3.37) finding the columns with negative reduced cost is equivalent to finding the columns which constraints in the dual problem are violated. Hence, adding columns with negative reduced cost to the RMP decreases the number of violated constraints in its dual problem. Via the optimality property we know that an infeasible dual problem can never have an optimal primal problem. So by adding columns with negative reduced cost to the RMP, we are moving in the direction of the optimal solution.

Sub Problem: Elementary Shortest Path Problem with Resource Constraints

In the previous section we have seen that routes with negative reduced cost have potential of reducing the objective value. The *Sub Problem* is responsible for finding those routes.

In the Sub Problem δ_r^i will be a decision variable: $z_i \in \{0, 1\}$ which is 1 if node i is visited by the route, and 0 otherwise. Note that in the classical route formulation we have x_{ij} indicating the selection of arc (i, j) for the route. Now z_i and x_{ij} can be related as follows:

$$z_i = \sum_{j \in N} x_{ij}, \forall i \in N \tag{3.38}$$

Using (3.38) the reduced cost can be rewritten as follows:

$$\overline{c}_r = c_r - \sum_{i \in N} \pi_i^* \delta_r^i + \nu^*$$

Using equation (3.17) and substituting x_{ij} we get:

$$\overline{c}_r = \sum_{(i,j) \in A} c_{ij} x_{ij} - \sum_{i \in N} \pi_i^* z_i + \nu^*$$

Now we use equation (3.38) to obtain:

$$\begin{aligned} \overline{c}_r &= \sum_{(i,j) \in A} c_{ij} x_{ij} - \sum_{(i,j) \in A} \pi_i^* x_{ij} + \nu^* \\ &= \sum_{(i,j) \in A} (c_{ij} - \pi_i^*) x_{ij} + \nu^* \end{aligned} \tag{3.39}$$

Since v^* is a constant the objective of the pricing problem becomes:

$$\min \sum_{(i,j) \in A} (c_{ij} - \pi_i^*) x_{ij}$$

In consequence, the Sub Problem is a minimization problem which can be formulated as follows: *Find a feasible route with minimal cost in the network with modified arc costs.* This is equivalent to solving an **Elementary Shortest Path Problem with Resource Constraints** (ESPPRC) in the modified network. The feasibility constraints of a route are represented by the resources. The first elementary version of the Shortest Path Problem with Resource Constraints was published by (Beasley and Christofides, 1989). In an elementary path every customer is visited exactly once such that no cycles occur.

The costs of the arcs are modified as follows:

$$c'_{ij} = c_{ij} - \pi_i^*.$$

Because of this modification of the arc costs the route with minimal cost is not necessarily equal to the route with no customers. An example of a modified network can be found in Figure 3.2. Assuming that all routes in network 3.2 are feasible there are two routes having negative cost (0-2-1-0 and 0-2-0).

In general, if the restricted master problem returns a large dual value π_i^* for customer i, all its outgoing arcs will have negative (or relatively small) modified cost $(c_{ij} - \pi_i^*) \quad \forall j \in N$. Because all arcs coming to i have costs which can highly decrease the route cost, customer i will probably be contained in a route with negative reduced cost. So the need for a customer to enter the RMP is reflected by a large dual value.

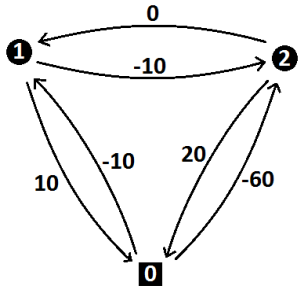


FIGURE 3.2: Network with modified arc costs

As long as there is at least one route with negative reduced cost we continue with CG. As soon as the minimal reduced cost becomes ≥ 0, we stop with the CG procedure since the optimal solution to the master problem has been found.

Now the structure the CG procedure and the formulations of the RMP and the Sub Problem are known, it is time to describe the solution methods to solve them. The RMP is an LP-problem and can easily be solved. Current state-of-the-art LP-solvers (such as IBM ILOG CPLEX or GUROBI) are very fast. In relation to solvable LP-problems our problems are by no means large [17].

Solving the Sub Problem is far more challenging. In literature various ways of solving the ESPPRC are proposed. All are roughly based on one of the following methods:

- Heuristic methods
- MIP formulations
- Dynamic Programming (DP)

We will shortly describe the first two methods. Subsequently, a solution method based on DP is described in full detail, since this is the method that is used in our algorithm.

Heuristic methods for the ESPPRC

Heuristic methods can be used for solving the ESPPRC. Examples are insertion heuristics and local search heuristics. The advantage of an heuristic method is its speeds. Initially, when the solution space contains many negative reduced cost routes, an heuristic method finds many routes in short time. But once the solution space becomes sparse computation times will increase when trying to find the remaining routes.

A drawback of an heuristic is that still an exact algorithm is needed. In the best case only the last iteration should be solved to optimality, to guarantee no negative reduced cost route exist. But there may be more iterations required of the exact algorithm.

MIP formulation of the ESPPRC

A MIP formulation of the ESPPRC can only be used for the deterministic VRPhTW. So we can not use this for our problem with stochastic travel times. For small deterministic instances of up to 20 customers a MIP can be practically solved to deliver the route with minimal cost. But once the instances become larger the computation times increase dramatically.

A full mathematical description of the MIP for solving the ESPPRC for the deterministic VRPhTW can be found in appendix A.1.

3.3.1 Solving the ESPPRC using Dynamic Programming

In current research methods based on Dynamic Programming are mostly used for solving the ESPPRC. Articles such as (Desrochers et al. [6]), (Larsen [17]), and (Feillet [10]) use this approach. Since DP uses a smart enumeration of all possible routes, it solves the problem to optimality within reasonable times. Furthermore, DP is able to handle various modifications of the problem while keeping its structure. Therefore, we will use DP to solve the ESPPRC.

DP algorithms search the solution space in a structured way. DP algorithms used for solving the ESPPRC are often called labeling algorithms. The name is derived from the basic elements of the algorithm, called labels. Desrochers [6] proposed an extension of the Ford-Bellman algorithm, resulting in a labeling algorithm for the VRPTW. The first labeling method for finding elementary routes was proposed by Beasley and Christofides (1996).

Let us define a *partial path* as a sequence of customer starting at depot. For partial paths the same feasibility requirements hold as for routes. A *label* corresponds to a partial path in the network of customers with modified arc costs. A label, l, is of the following form:

$$l = [C, \mathbf{R}, \mathbf{V}],$$

where C is the cost of the partial path, $\mathbf{R}$ the resource vector, and $\mathbf{V}$ the vector of visited customers.

$\mathbf{R}$ contains the following resources:

- Accumulated demand
- Distribution of the time service starts at the last customer

In literature labeling algorithms are mostly described for deterministic problems. Instead of a fixed time, we extended the concept by using a random variable instead. The exact way the random variables are represented in the labels is explained in Chapter 4.

Feillet et al. [10] describe an improved version of this algorithm for the deterministic ESPPRC in which unreachable customers are stored in a label. Naturally, visited customers are unreachable because of the elementary property of a path, but resource violations can also cause unreachability. The labels are now of the following form:

$$l = [C, \mathbf{R}, \mathbf{U}],$$

where **U** is the vector of unreachable customers. We will use this version since it allows us to improve the speed of the algorithm, as will be explained later.

Initially $|N|$ labels are constructed, representing the $|N|$ single arc paths starting at depot. Paths are iteratively extended to all possible customers, such that feasibility conditions are satisfied. In this way new labels are created. The search stops once no extensions are possible anymore. From the paths that reached the depot the path with minimal cost is returned to the RMP. Since this path starts and end at the depot it is equivalent to a route.
Moreover, to speed up convergence multiple routes with negative cost can be added to the RMP in one iteration of CG. Every route with negative cost has the potential of reducing the objective value. Experiments by (Kohl 1995) suggest that if more than one route with negative cost exist, they should be given to the RMP.

Domination

The number of labels grows extremely fast. To limit the proliferation of labels we include dominance relations. Domination allows us to only consider non-dominated paths, and still solve the ESPPRC to optimality.

Consider two feasible partial paths both ending at the same customer, represented by label $l = [C, \mathbf{R}, \mathbf{U}]$ and $l' = [C', \mathbf{R}', \mathbf{U}']$, respectively. Label l dominates l' if

1. any feasible extension of l' ending at customer j is also feasible for l,
2. for any of these extensions it holds that $C_j \leq C'_j$, where C_j is the cost of the extended path to customer j [9].

How this is practically checked is described in Proposition 3.2. In the proof of the Proposition 3.2 distributions need to be ordered. But comparing two distributions is not trivial. We need to use the concept of *stochastic ordering*, which allows us to compare two distributions. Multiple types of stochastic ordering are used in current research. It appears that stochastic ordering of the first order is suitable for label domination.

Lemma 3.1. (Stochastic dominance, Errico et al. 2013)
Let two partial paths ending at customer i have start of service times A and B at customer i with corresponding cumulative distribution functions (CDFs) $\mathfrak{D}_A$ and $\mathfrak{D}_B$. A is stochastically dominated by B in the first order, denoted by $A \preceq B$, if and only if

$$\mathfrak{D}_A(x) \geq \mathfrak{D}_B(x) \quad \forall x \in \mathbb{R}. \tag{3.40}$$

Moreover, assume $A \preceq B$, then for any identical feasible extension of the respective partial paths ending at j, also the extended labels satisfy:

$$\mathfrak{D}_{A_j}(x) \geq \mathfrak{D}_{B_j}(x) \quad \forall x \in \mathbb{R}$$

Proposition 3.2. (Dominance rule)
Practically, checking domination comes down to the following. Let labels l and l' be such that

(i) $C \leq C'$

(ii) $\boldsymbol{R} \leq \boldsymbol{R}'$

(iii) $\boldsymbol{U}_i \leq \boldsymbol{U}'_i$ *for all* $i \in N$,

where distributions are ordered using stochastic domination of the first order. Then l *dominates* l' *such as defined by conditions 1) and 2).*

Proof:
It needs to be shown that under the given assumptions (i) - (iii), conditions 1) and 2) are satisfied. Consider a feasible extension of label l' to customer j, denoted by l'_j. Since l'_j is feasible it holds that $\mathbb{P}[S'_j \leq d_j] \geq \gamma_j$. Given that $S \preceq S'$ holds for the start of service times and using lemma 3.1 we know

$$\begin{aligned} & & S \preceq S' \\ \text{(First-order st.dom.)} \quad & \Rightarrow & \mathfrak{D}_S(x) \geq \mathfrak{D}_{S'}(x) \quad \forall x \in \mathbb{R} \\ \text{(Lemma 3.1)} \quad & \Rightarrow & \mathfrak{D}_{S_j}(x) \geq \mathfrak{D}_{S'_j}(x) \quad \forall x \in \mathbb{R} \\ & \Rightarrow & S_j \preceq S'_j \\ & \Rightarrow & \mathbb{P}[S_j \leq d_j] \geq \mathbb{P}[S'_j \leq d_j] \geq \gamma_j \end{aligned}$$

So the service of label l_j at customer j is reliable. Furthermore, the unreachable customer vector $\mathbf{U}_i \;\; i \in N$ and the accumulated demand are nondecreasing functions. So given that l'_j is feasible and based on assumptions (i) - (iii), these properties will also be feasible after extension to j. So it follows that l_j is feasible, and condition 1) is satisfied.

Now we are left with checking condition 2). We can write:

$$\begin{aligned} C_j &= C + c_{ij} - \pi_i^* \\ &\leq C'_j + c_{ij} - \pi_i^* \\ &= C'_j \end{aligned}$$

So also condition 2) is satisfied, which proves the proposition. □

Because of Proposition 3.2 labels can be dominated as follows: If for two partial paths ending at the same customer (i) - (iii) are satisfied, the dominated label can be discarded. Note that if in further research one would consider time dependent travel times, these domination criteria will not hold in their existing form. Comparing start of service distributions would require a more advanced method.

3.3.2 Bi-directional handling of paths

Until this point partial paths have been extended from the depot in a forward direction. But the longer the partial paths become, the more labels there will be generated. To overcome this, partial paths can be extended both in a forward direction and in a backward direction. Merging them results in a full path. In this way paths are only extended until approximately half the length of the full paths. For

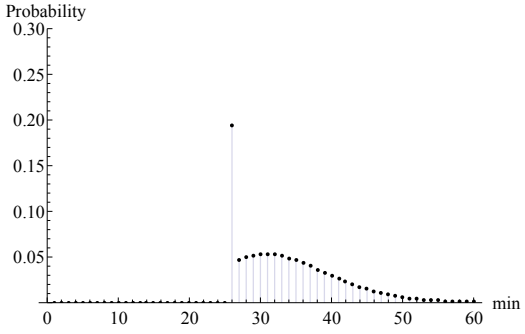
the deterministic VRPhTW this is described in (Desrochers et al. [7]). When our algorithm solves a deterministic problem, we use bi-directional extensions.

But in the VRPhTW-STT there is an additional difficulty. Service time distributions can only be calculated when a path is connected to the starting depot. When extensions are made backwards, this is not the case. Therefore, this method can not be used here.

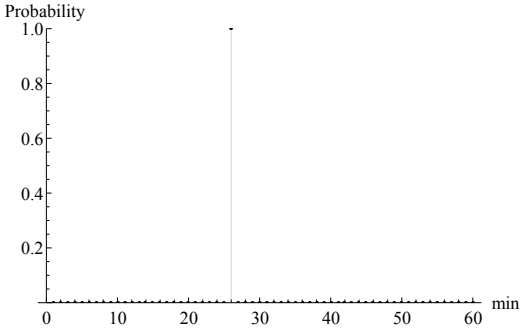
3.3.3 Finding infeasible partial paths

As a preprocessing step we can determine a list of infeasible partial paths. This can save time during the dynamic programming. Both in a forward direction and a backward direction partial paths can be extended from the depot. When partial paths are extended in a forward direction infeasibility can immediately be detected. But when partial paths are extended from the end, a more advanced method should be used to determine a list of infeasible partial paths. This is done by working with an upper bound on the service reliability, since the exact reliabilities can not be calculated. This method is only effective for problems which are constrained in terms of time windows. Instances with broad time windows will not benefit from this method.

Service reliabilities at customers are computed based on the random service start time at that customer. When a route is extended from the end, the distribution of the start of service time cannot be known. But we can use a lower bound to still be able to disallow an extension. This lower bound is given by the random variable which takes value 1 at the release date. Graphically this is shown in the figures below. Notice that a lower bound on the start of service gives an upper bound on the service reliability at that customer.

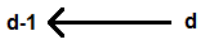


(a) A possible start of service distribution at a customer



(b) A lower bound on all possible start of service distributions at that customer

Let us check if extension to customer $d-1$ in the backward direction gives a feasible partial path, where d is the depot. In any case capacity and customer uniqueness must be satisfied.

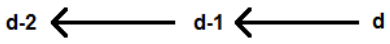


Furthermore, service reliability at d must be satisfied. But since we do not know the distribution of the start of service at $d-1$, we can only calculate an upper bound on the service reliability at d, denoted by $\overline{\gamma}_d$. This is done by assuming service at $d-1$ starts at r_{d-1}, which is a best case situation. It follows

that

$$\overline{\gamma}_d = \mathbb{P}[r_{d-1} + s_{d-1} + T_{d-1,d} \ \leq \ d_d].$$

The partial path is not feasible if $\overline{\gamma}_d < \gamma_d$, where γ_d is the required service reliability at d, since then the service reliability requirement at d would be violated in any case.



For the second extension 2 checks can be performed: reliability at $d-1$ and at d. Upper bounds on the service reliabilities are given by:

$$\overline{\gamma}_{d-1} = \mathbb{P}[r_{d-2} + s_{d-2} + T_{d-2,d-1} \ \leq \ d_{d-1}],$$

and

$$\overline{\gamma}_d = \mathbb{P}[\max(r_{d-2} + s_{d-2} + T_{d-2,d-1}, r_{d-1}) + s_{d-1} + T_{d-1,d} \ \leq \ d_d].$$

If $\overline{\gamma}_{d-1} < \gamma_{d-1}$ or $\overline{\gamma}_d < \gamma_d$ holds, the partial path is not feasible. Calculations made for determining $\overline{\gamma}_{d-1}$ can be reused to determine $\overline{\gamma}_d$, which speeds up the computations.

In general the backward check for feasibility can be formulated as follows. Let us pretend that the customer we extend to completes the route. So the route starts at this customer, instead of at the depot, at a time which is equal to its release date. Accordingly, we calculate the service reliabilities at the customers of this route as we are used to. These are in fact upper bounds on the service reliabilities. Unreliable service at one of the customers shows infeasibility of the partial path.

3.4 Branch-and-Bound search

Once CG has solved the Master Problem the solution is fractional in general. In a fractional solution of the MP every customer is visited exactly once but via a combination of fractional route variables. Figure 3.3 shows a schematic and simplified representation of the solution space. The constraints induce a fractional solution of the MP since it does not lie on the integer grid.

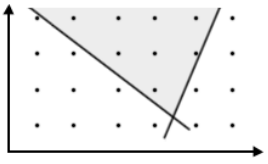


FIGURE 3.3: Solution space inducing a fractional solution

An example of an optimal solution of the MP could look like Table 3.4. This solution is valid because every customer is either in a route with solution value 1, or in three routes with solution value 0.333. In order to obtain an integer solution we perform a Branch-and-Bound search. A reliable and well defined branching methodology is important for a quick convergence to the optimal solution. The Branch-and-Bound tree is explored by finding upper bounds and lower bounds, during which additional routes may be generated at each node. The branching rules and bounding strategies we choose are explained in the following subsections, as well as the considerations that led to them.

Solution value	Route
1.0	0 2 5 7 6 8 3 1 4 0
1.0	0 19 18 20 0
0.333	0 14 15 11 12 9 10 13 17 0
0.333	0 14 12 15 16 9 10 0
0.333	0 11 15 16 17 13 0
0.333	0 14 12 11 16 9 10 13 17 0

TABLE 3.4: Solution of the RMP obtained by Column Generation

Using CG within a Branch-and-Bound search is called *Branch-and-Price*.

3.4.1 Branching

In literature one encounters a variety of branching methods. Branching splits the solution space into two or more disjoint parts. The most common branching strategies are branching on arcs, and branching on vehicles. Moreover, many articles describe strategies based on a combination of branching rules. For example, M. Desrochers et al. [6] combine branching on the number of vehicles with branching on arcs.

The branching strategy we will use is based on this two level strategy of Desrochers et al., and will be described in the next subsection. However, we will extend it with another branching strategy which branches on routes. So in total we use the following three branching strategies:

(a) Branching on the number of vehicles

(b) Branching on arcs

(c) Branching on routes

Since a solution is given in terms of route values, we use the following method to derive the solution values of the arcs from the solution values of the routes. Let y_r^* be the solution value of route r. We define the solution value of arc (i, j) as

$$x_{ij}^* = \sum_{r \in \mathcal{R}^S} y_r^*.$$

(a) Branching on the number of vehicles

Branching on the number of vehicles was first introduced by Desrochers et al. [6]. Once we obtain the solution after CG at a node the solution values of routes are fractional in general. In the solution of Table 3.4 this is the case. The sum of route solution values is denoted with η.

$$\eta = \sum_{r \in R} y_r^*.$$

Any feasible solution to our problem must have an integer number of routes. Therefore we branch on the number of vehicles used in the solution. One child node has a constraint in the RMP which enforces an upper bound on the number of routes used in the solution:

$$\eta \leq \lfloor \eta^* \rfloor.$$

The other child nodes enforces a lower bound:

$$\eta \geq \lceil \eta^* \rceil.$$

Branching on the number of vehicles does not change the Sub Problem. Let us denote the dual variable due to one of the constraints on the number of vehicles by ρ. Then the reduced cost of a route becomes:

$$\sum_{(i,j)\in A} (c_{ij} - \pi_i^*)x_{ij} + \nu^* - \rho^*, \tag{3.41}$$

where ρ^* is constant in the reduced cost expression, hence it is not included in the objective of the Sub Problem.

(b) Branching on arcs

The main question with arc branching is what the best arc is to branch on. D. Tas [24] branches on the arc with solution value closest to 0.5. The reason is to force difficult decisions to be made in the beginning. This is a well known branching strategy, but one thing is not taken into account. During CG we spend a considerable amount of time to obtain a good solution to the MP. The solution provides us with substantial information about the optimal solution. Rather than just a lower bound, it can be seen as an oracle. This brings us to the choice of branching on the arcs with highest solution value. With high probability they lead the process in the direction of the optimal solution.

A structured growth of the routes is preferred when choosing the branching arc. Therefore only arcs that leave the depot or connect to any already fixed arc are considered. Arcs are fixed in such a way that routes will emerge from the depot. These routes are called partial routes. Furthermore, since stochastic travel times are considered this partial route structure is favoured, because arrival distributions are only meaningful if a route is connected to the depot.

The downside is that fixing an arc in the beginning that is not in the optimal solution may lead to exploring an enormous part of the solution space before returning to the point the, apparently wrong, decision was made.

Tiebreak decisions

If the maximal solution value appears at multiple arcs, a tiebreak decision must be made. Naturally, in the end only one arc should be returned for branching. For example, an arc with solution value 0.5 can be used in one route with solution value 0.5, or in two routes with solution value 0.25. We favour the arc which is used in the least number of routes. The idea is that this arc is favoured the most by one specific route, and thus is more likely to occur in the optimal solution.

Table 3.5 shows an example of this tiebreak decision.

Arc	Scenario	Branch arc?
$0 \rightarrow 5$	0.25 - 0 5 7 6 8 3 4 0 0.25 - 0 5 8 9 2 0	
$0 \rightarrow 6$	0.5 - 0 6 4 1 7 0	✓

TABLE 3.5: Tie break decision (level 1)

This tiebreak decision is made as follows:

$$\text{Choose arc } (i,j) \text{ that maximizes } \frac{x^*_{ij}}{R_{ij}},$$

where R_{ij} is the number of routes that use arc (i,j).

It is possible that this maximum is attained by multiple arcs. In particular, this happens every time a solution contains multiple routes with solution value 1. Therefore, a tiebreak decision at a second level should be defined, which exists mainly for making sure a single arc is chosen.

We choose to branch on the arc that is used in the longest route. This is because we prefer to visit as many customers in one route as possible. An example can be found in Table 3.6.

Arc	Scenario	Branch arc?
$0 \rightarrow 4$	1.0 - 0 4 13 6 0	
$0 \rightarrow 5$	1.0 - 0 5 28 19 2 12 7 0	✓

TABLE 3.6: Tie break decision (level 2)

Post-optimizing the branch arc

When the chosen branch arc connects a partial route the the depot, post-optimization can delay the connection to the depot. The idea is that extending a route is always better then going back to depot. Suppose the branch arc is of the form $(c,0)$. If customer c has other outgoing arcs with positive solution value, we choose the one with the highest value to branch on, instead of arc $(c,0)$.

(c) Branching on routes

In Toth and Vigo [25] a suggestion is made that it should be possible to branch on route variables. However they did not describe a workable method. As mentioned is several papers, it is difficult to exclude a route. One must ensure the excluded route is not generated again in the Sub Problem.

Branching on routes is an extension of branching on arcs. In the solution of the MP routes may come with high solution values. We would probably branch on all arcs of that route sequentially. Computation time can be saved by branching on all arcs of the route at once. We call this branching on a route. If the solution value of a route equals 1, and after branching on the first arc it stays 1, we choose to branch on the route.

Let r_1 be a route on which we decided route branching. In the child node that includes r_1 its corresponding variable is fixed to 1 in the RMP. In the child node that excludes r_1 we set its variable to 0, but more importantly, we must make sure r_1 is not generated again in the Sub Problem, the ESPPRC. This can not be accomplished by forbidding all its arcs. That is because only the specific combination of arcs that forms route r_1 is forbidden, not the arcs by themselves. A possibility is to check every time a route is given to the RMP whether it is equal to a forbidden route. This method works, but is not computationally attractive.
Another possibility is to add a resource to the partial paths for every forbidden path. This resource equals the number of consecutive arcs similar to the respective forbidden path. If the sequence of arcs is not exactly the same, the resource is 0. In the mono-directional labeling algorithm a path is feasible if

all forbidden path resources equal 0. In the bi-directional labeling algorithm a forward and a backward path are merged only if at least one of the two resources representing the same forbidden path equals 0.

3.4.2 Bounding

At every node of the search tree we use CG to find a lower bound. The CG procedure provides us with a lower bound for the optimal solution value. A node of the branching tree is called *integral* if all route variables of its solution have integral values. Let the *incumbent node* be defined as the integral node which has currently the best MP value.

When CG has solved the MP at a node two cases can be distinguished:

- If the solution is integral, the node is pruned. We compare its value with the incumbent node, and update it when the new solution has a lower objective value.
- If the solution is not integral, the node is only pruned if its objective value is higher than the value of the incumbent node.

If a node is not pruned we branch on its solution. The two resulting nodes are added to the queue of nodes that need to be explored. We continue finding lower bounds while the queue is non-empty. At that point the incumbent node contains the optimal solution to the problem.

To speed up the search of the branching tree we also determine an upper bound at each node. Using these upper bounds we try to lower the value of the incumbent node, which generally results in the pruning more nodes.
Upper bounds are found by extending the partial paths, resulting from branching decisions, to the depot using an extension heuristic. This results in an integral solution which is an upper bound for the optimal solution.

3.4.3 Acceleration strategies

To speed up the algorithm acceleration strategies are applied. At first, we do not need to solve the Sub Problem to optimality in every iteration of CG. We can stop when at least 1 feasible route with negative reduced cost is found. In practice, we stop once we have found 5000 of those routes, and return all of them to the RMP. Only in the last iteration of CG, we need to solve the Sub Problem to optimality, to guarantee no feasible routes with negative reduced cost exist.

Furthermore, once the RMP gets filled more and more it is not solved quickly anymore. Most of the routes in the RMP will never be part of a solution. Therefore, we use *column management* to reduce the number of routes in the RMP. Routes which are not part of a solution for more than a certain number of iterations, are removed from the RMP. There is a risk of removing routes which will be in the solution later on, but they will be simply generated again by the Sub Problem. There is a trade-off between the computation time needed for column management and the time that is saved in solving the RMP. During the first iterations of CG the effect of column management will be neglectable, but once the number of routes becomes larger, column management contributes to decreasing the running time.

Chapter 4

Modeling Stochastic Travel Times

To handle uncertainty in travel times, a new approach is needed, incorporating random variables. A random variable can be described by its probabilistic density function (PDF), or its cumulative distribution function (CDF). Most random variables are either from the discrete or the continuous type. However, random variables having both discrete and continuous parts do exist. As we will see later in this chapter, this type of random variable arises in our case because of the hard time windows. In this chapter we provide the details, and we will explain our method of calculating with stochasticity.

As discussed in Chapter 3, uncertainty is part of the algorithm via the travel time distributions, and start of service distributions. Start of service distributions are part of the labels which are used to solve the Sub Problem. Every label represents a partial path with a corresponding start of service distribution at the last customer of its customer sequence. Therefore, a method of representing probability distributions is needed. Furthermore, we need to determine the start of service distributions.

4.1 Stochastic computations for the VRPhTW-STT

For a given a sequence of customers we should determine the service reliability at every customer. Therefore, the start time of service at every customer must be calculated. Note that this is a random variable, due to the fact that we consider random travel times T_{ij}. Let the random variable of the start of service time at customer i be denoted with S_i. Assume customer j is visited just after customer i. Based on this information we will find an expression for S_j, written in terms of S_i and the travel time T_{ij}.

The departure time at customer i satisfies: $D_i = S_i + s_i$, where s_i is the (deterministic) service time at customer i. The arrival time at j satisfies $A_j = D_i + T_{ij}$. The reliability requirement bounds the probability that service starts after the due date.

Since time windows are hard, service at j starts at r_j if $A_j \leq r_j$, and at A_j otherwise. So it holds that $S_j = \max\{A_j, r_j\}$.

In conclusion, a recursive expression for the start time of service can be formulated:

$$S_j = \max\{S_i + s_i + T_{ij}, r_j\}.$$

When the distribution of S_j is known, the service reliability at customer j can be checked as follows:

$$\mathbb{P}[S_j \leq d_j] \geq \gamma_j. \tag{4.1}$$

The computation of the distribution of S_j requires the following:

- Addition of random variables. This requires the computation of convolutions of their probability distribution functions.
- Computing the distribution of the maximum of a random variable and a constant.

Moreover, for the algorithm described in Chapter 3 we need the following:

- Computing the service reliability at a customer,
- Ordering two distributions using stochastic domination.

In Section 4.2 these operations will be explained in full detail for our method of choice. But first alternative ways of representing distributions are addressed which are found in literature.

Sample Average Approximation

A well-known method for including stochastic elements in a model is Sample Average Approximation (SAA). When travel times are deterministic every arc has a fixed travel time. Using SAA every arc can attain multiple values which are contained in a pre-computed sample.
A solution can be feasible for one value from the sample, and infeasible for another. The arrival reliability at a customer can be approximated by the fraction of samples that induce a solution which is reliable at that customer.
The disadvantage is that the approximation only gets better once the sample size increases. But increasing the sample size increases the computation time. L. Evers (2013) used relatively small samples size while guaranteeing a certain accuracy.

Fitted distributions

Based on available data, or existing knowledge about the system, a specific distribution can be used to describe the random travel times. For example, in [24] travel times are assumed to be Gamma distributed. In this way the addition of two distributions again returns a Gamma distribution. Moreover, they assume time windows are soft, so there is no maximum operator required. As a result, every distribution can be represented by the two parameters of a Gamma distribution.
But as in our case, hard time windows result in computationally hard mathematical operations, like the convolutions and (in particular) maximum operations. For example, Gamma distributed travel times do not result in Gamma distributed start of service times. The maximum operator, in combination with a convolution, changes the shape of distributions drastically.
We have studied the possibilities of Laplace-Stieltjes transforms (LSTs) for computing with distributions. A nice property of the LSTs is that a convolution is equivalent to a multiplication in the Laplace domain. However, nesting multiple max operations makes also this technique less efficient to implement. Moreover, computing the actual service reliability at customers of a route would involve inversion of the LST, which is also computationally too involved for our purposes.

Approximating distributions

As we have seen, theoretical distributions can not be used directly, but they can be approximated. Conijn [3] uses a piecewise polynomial approximation of CDF functions. In this way a distribution function is described by a set of breakpoints and corresponding polynomial coefficients. Furthermore, Conijn describes calculation methods for this representation.
This method is not the method of our choice, since it is computationally demanding, especially for computing convolutions. Every extension check in the labeling algorithm requires many convolutions. Since also the number of generated labels in one iteration of the Sub Problem is extremely large, a method which is computationally less demanding is preferred.
However, in Chapter 5 a method is constructed for generating travel time distributions based on real data. This method uses a similar approach as described in [3].

4.2 Efficient implementation

In this section we will explain in detail our implementation. Travel times are assumed to be *discrete* random variables. A similar assumption is made by Errico et al. [9]. This method can handle distributions of any shape, and computations have a low complexity. The computational tools we need are also discussed in this section.

Let X be a discrete random variable with possible outcomes $(t_0, t_0 + 1, ..., \tau)$, called its *support*. The support is finite since $0 \leq t_0$ and $\tau \leq d_0$, where d_0 is the upper bound of the planning horizon. In our model X may represent one of the travel times T_{ij}, or a start of service time S_i. In our implementation we represent the PDF of X by a vector $\boldsymbol{x}$ with elements $x(t) = \mathbb{P}[X = t]$ for all t in the support of X. Figure 4.1 shows this representation graphically.

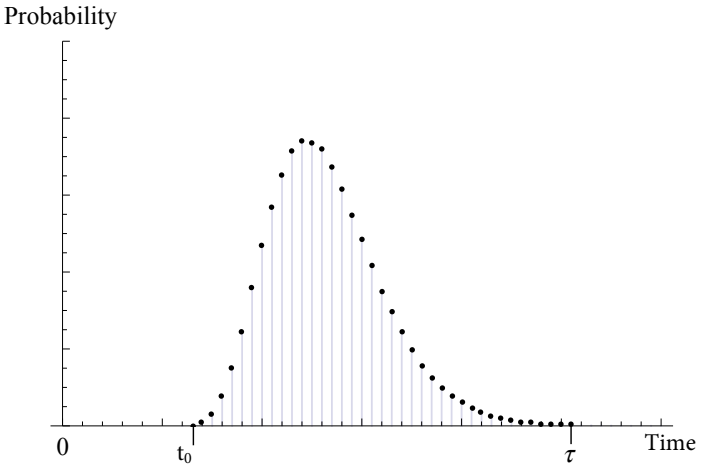


FIGURE 4.1: Representation of a discrete distribution.

To compute efficiently with random variables we also store its CDF. In this way cumulative probabilities are at hand without additional calculations. The CDF has the same support as the PDF and is

represented by the vector $\boldsymbol{X}$. The values of the PDF and the CDF are related as follows:

$$X(z) = \sum_{t=t_0}^{z} x(t). \tag{4.2}$$

In our implementation a random variable is stored as a triplet $(t_0, \boldsymbol{x}, \boldsymbol{X})$. Note that the tuples $(t_0, \boldsymbol{x})$ or $(t_0, \boldsymbol{X})$ already fully define the PDF, but as stated before, storing both $\boldsymbol{x}$ and $\boldsymbol{X}$ increases the efficiency.

Naturally, $t_0 = r_i$ when X is the start of service time at customer i, since service cannot start earlier than the release data. When X is a travel time, t_0 represents the minimal time required to traverse the path.

The following subsections describe how the required operations are performed in our implementation using triplets $(t_0, \boldsymbol{x}, \boldsymbol{X})$.

(i) Adding two random variables (Convolution)

Let X and Y be two independent discrete random variables with corresponding PDF functions $x(t)$ and $y(t)$ having the same support $(t_0, t_0 + 1.., \tau)$. Then the PDF of $C = X + Y$ is calculated using a convolution:

$$C(2t_0 + z) = \sum_{m=0}^{z} x(t_0 + m)y(t_0 + z - m) \quad \forall\, z \in (0, 1, ..., 2\tau).$$

But in general the supports of two distributions are not the same.

Now, let X and Y be two arbitrary independent discrete random variables. In our implementation they are represented by the triplets $(t_0^X, \boldsymbol{x}, \boldsymbol{X})$, and $(t_0^Y, \boldsymbol{y}, \boldsymbol{Y})$. The support of X is given by $(t_0^X, t_0^X + 1, ..., \tau^X)$. The support of Y is given by $(t_0^Y, t_0^Y + 1, ..., \tau^Y)$. Assume $\tau^X < \tau^Y$, without loss of generality. Then the triplet $(t_0^C, \boldsymbol{c}, \boldsymbol{C})$ which represents the distribution of $C = X + Y$ is computed as follows:

Step 1. $t_0^C = t_0^X + t_0^Y$.

Step 2. The PDF of C is computed as follows:

$$c(t_0^C + z) = \sum_{m=\max\{0, z-\tau^Y\}}^{\min\{\tau^X, z\}} x(t_0^X + m)y(t_0^Y + z - m) \quad \forall\, z \in (0, ..., \tau^X + \tau^Y).$$

So C has support $(t_0^C, ..., t_0^C + \tau^X + \tau^Y)$.

Step 3. Once the PDF of the convolution is determined the CDF can be obtained using Equation (4.2).

(ii) Adding a constant

Since we assume service times are deterministic, adding a service time to a distribution means adding a constant. In our implementation this results in a translation of the PDF and the CDF. A constant s is added to a discrete random variable X having triplet $(t_0, \boldsymbol{x}, \boldsymbol{X})$ by changing t_0 to $t_0 + s$. The support of $X + s$ becomes $(t_0 + s, ..., \tau + s)$.

Adding a constant to a distribution can also be seen as a special case of a convolution. In this case one of the distributions has only one value with probability 1.

(iii) Maximum operator

Arriving before the release date induces a waiting time until the window opens. This means service starts at the maximum of the arrival time and the release date. Since the arrival time is a random variable, the maximum of a distribution and a constant needs to be calculated.

Let X be a discrete random variable. Its distribution is represented by the triplet $(t_0, \boldsymbol{x}, \boldsymbol{X})$, and $a > 0$ is a constant. In our implementation, the triplet $(t_0^M, \boldsymbol{m}, \boldsymbol{M})$ which represents the distribution of $M = \max(X, a)$ is calculated as follows:

Step 1. If $a \geq t_0^X$, we set $t_0^M = a$ and continue with step 2. If $a < t_0^X$ we stop and do not continue with step 2 and 3. The distribution of M is equal to the distribution of X.

Step 2. The values of $\boldsymbol{m}$ are computed as follows: $\forall t \in (t_0^M, , ..., \tau_i)$

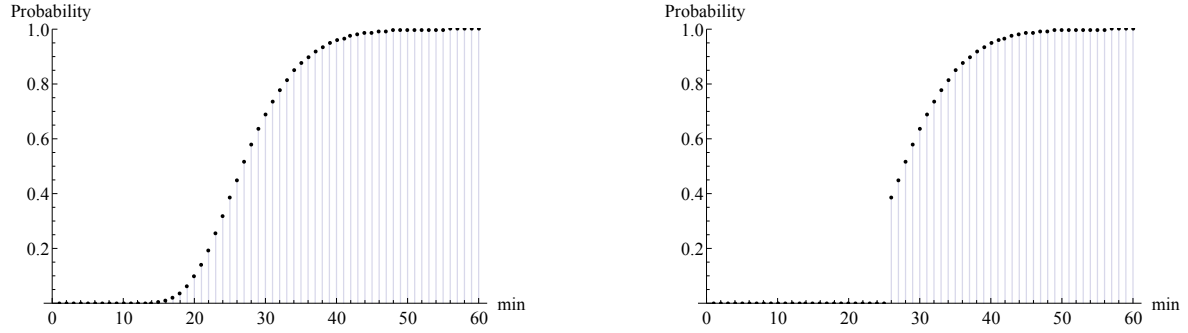
$$m(t) = \begin{cases} X(a) & \text{if } t = a \\ x(t) & \text{if } t > a. \end{cases} \tag{4.3}$$

Note that the support has been reduced to $(t_0^M, , ..., \tau_i)$, since $t_0^M = a \geq t_0^X$.

Step 3. The values of $\boldsymbol{M}$ are given by: $\forall t \in (t_0^M, , ..., \tau_i)$

$$M(t) = X(t)$$

In our implementation we compute the maximum of a arrival time and a release date. An example of the CDF of an arrival time and a corresponding service start time can be found in Figure 4.2, where the release date equals 26.



(a) (b)

FIGURE 4.2: CDF of an arrival time distribution (a), and a corresponding start of service distribution (b).

(iv) Checking reliability

To calculate a service reliability, we need to determine the probability that a random variable is smaller than a value. When a CDF is available this can be easily done.
Let random variable X be represented by the triplet $(t_0, \boldsymbol{x}, \boldsymbol{X})$. The probability that X is smaller than a is given by $X(a)$. According to (4.1), reliability is satisfied if $X(a) \geq \gamma$, for some reliability threshold γ, where X represents a start of service time.

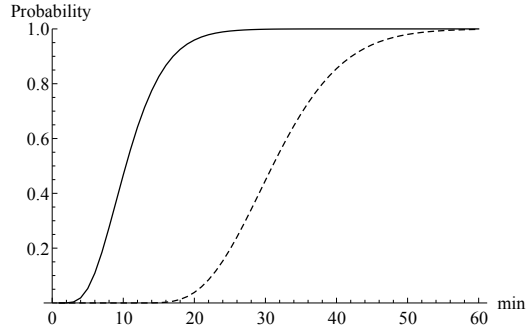
(v) Stochastic ordering of distributions

For determining if a label dominates another label when solving the ESPPRC, start of service distributions need to be stochastically ordered. As we have seen in Theorem 3.2 stochastic domination of the first order gives the desired ordering.

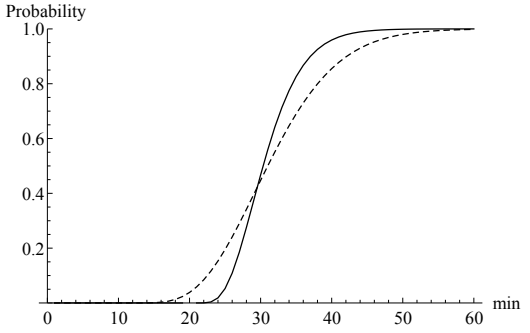
Let distributions X and X' have triplets $(t_0, \boldsymbol{x}, \boldsymbol{X})$ and $(t_0', \boldsymbol{x}', \boldsymbol{X}')$. X is stochastically dominated by X' in the first order, denoted by $X \preceq X'$, if

$$X(t) \geq X'(t) \quad \forall\, t \in (\min(t_0, t_0'), ..., \max(\tau, \tau')).$$

Graphically this is shown in Figure (a) below. For clarity we show continuous graphs, however start of service distributions are discrete.



(a) The dashed distribution dominates the solid distribution in the first order



(b) No decision on the ordering of the distributions can be made

Analogously, $X \succeq X'$ if $X(t) \leq X'(t) \quad \forall\, t \in (\min(t_0, t_0'), ..., \max(\tau, \tau'))$. But one must be careful. If $X \preceq X'$ does not hold, this does not imply $X \succeq X'$, because there is a third possibility in which one cannot decide an ordering. This is the case when $X(t) \geq X'(t)$ for some t, and $X(t) \leq X'(t)$ for some others. An example of this situation can be found in Figure (b). If this situation occurs during a domination check, we can not dominate either of the labels.

The computational tools for distributions which the algorithm requires have now been described. For a given sequence of customers we are able to compute the start of service distributions to determine the service reliability at every customer. Furthermore, distributions can be stochastically ordered which is needed for domination of labels in the ESPPRC.

Chapter 5

Using real travel time data

In many applications the construction of networks containing properties of real life road networks is desired. Companies solving real life transportation problems, such as Quintiq, benefit from an increase in representative data. The vast increase in data collection of the last few years offers numerous possibilities. In this section we will describe methods for using real life data to obtain travel time distributions to use in stochastic instances. In this project data collected from car navigation systems is made available by TomTom (TomTom International BV). The data is gathered by GPS devices in cars. Travel time distributions between any two coordinate points could be queried. Travel time data is provided in terms of quantiles $Q = (q_{\alpha_1}, ..., q_{\alpha_n})$ defined by $q_{\alpha_i} = \inf(t \mid \mathbb{P}[T \leq t] \geq \alpha_i)$, where T is a random variable representing a travel time distribution.

In this chapter two methods are described for generating probability distribution functions based on the data. First, a method is described in which theoretical distributions are fitted. Secondly, we propose an interpolation method for the data points. Based on the continuous interpolation function a discretization can be made, which is used as input for our algorithm. Finally, an instance based on real data is constructed, and solved.

5.1 Fitting theoretical distributions

In analytical research oftentimes theoretical distributions are used with well-guessed parameter configurations. Tas [24] assumes travel times are Gamma distributed. Conijn [3] uses a shifted Log-normal distribution. Shifting makes sure speed limits are reflected by the distribution. In Lecluyse et al. [18] a queueing approach to traffic flows was used.

In this section we find theoretical distributions which reflect the TomTom travel time data as good as possible. An a-priori distribution type is chosen, followed by a parametric optimization method. Since the resulting function belongs to a class of theoretical distributions we are certain it satisfies all properties of a distribution.

Vector Q provides $|Q|$ measured points $\{q_{\alpha_i}, \alpha_i\}$ of the CDF function. Distributions are fitted using an a-priori choice of shape, and an optimized parameter configuration, which includes a shifting constant ρ.

We define the following error function: $e_Q = \sum_{i=0}^{|Q|} w_i(q_{\alpha_i} - \mathcal{D}(\xi) - \rho)^2$, with w a weight vector, $\mathcal{D}$ a CDF function of a chosen type, ξ a parameter vector, and ρ a shifting constant.

The shape of the a-priori distribution must be capable of resembling the general shape of a travel time distribution. In an article by Lecluyse et al. [18] it is argued that travel times are approximately Log-normal distributed. Also Conijn [3] uses Log-normal distributions for the creation of stochastic instances. Therefore, we will use Log-normal distributions to fit the data, but our method is not restricted to any specific type of distribution.

Variables ξ and ρ are determined such that e_Q is minimized. For the route between Den Bosch and Utrecht the measured quantiles are shown in Figure 5.1. The parameters of a shifted Log-normal distribution were determined such that e_Q was minimized. The result can also be found in Figure 5.1. The PDF of this distribution is shown in Figure 5.2.

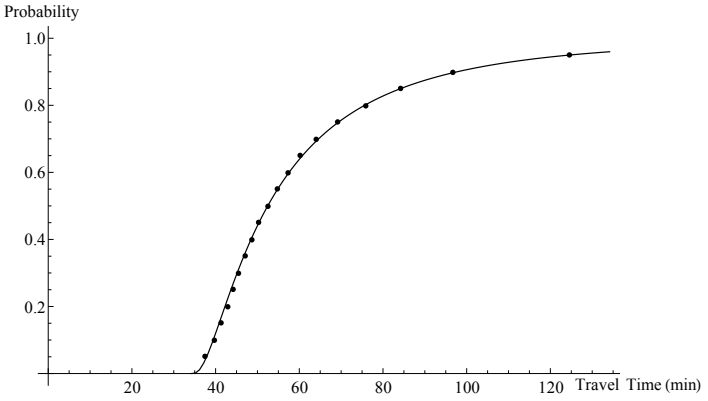


FIGURE 5.1: Observed quantiles of the Den Bosch - Utrecht travel time distribution between 16:00 and 20:00 over the year 2009. A shifted log-N$(2.91, 0.97)$ distribution with shifting constant 34.16.

We see that a shift of the distribution is essential. This shift reflects the minimum time required to travel the path. The PDF in Figure 5.2 shows this minimal travel time clearly, followed by a steep rise of the function.

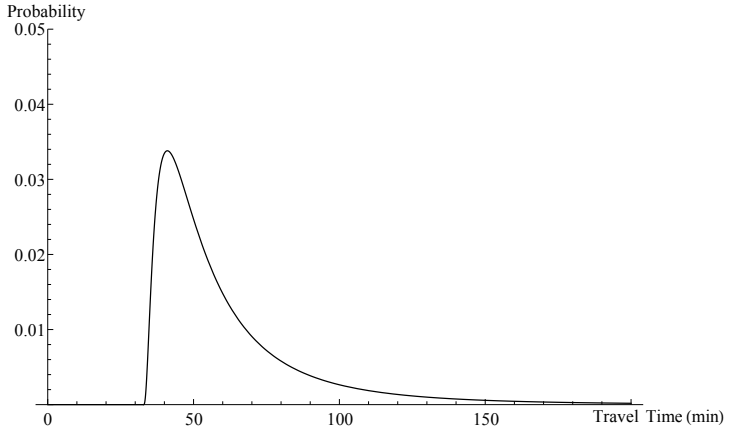


FIGURE 5.2: PDF of the travel time distribution from Figure 5.1.

We see that cars mainly traverse the path in a time just above the minimal time. These are the cars which are not slowed down by traffic congestion. On the other side, there is a probability that traffic congestion increases the travel time, which results in a long tail of the PDF.

To use the constructed distribution in our algorithm, we need to discretize it, since our algorithm assumes travel times are discrete random variables.

5.2 Polynomial Interpolation

Another way of constructing distributions for our algorithm based on the data is described in this section. Since discrete distributions have a function value for every $t_0, t_0 + 1, ..., \tau$ of their support, interpolation between quantiles is necessary. Other then when fitting a theoretical distribution, we must make sure the resulting function satisfies the properties of a distribution function. This section describes our developed interpolation method. First a continuous CDF will be found which interpolates the data points. Subsequently, the CDF can be discretized to fit into our algorithm. This method is used in Section 5.4 to generate a real life instance based on TomTom data.

The continuous CDF is represented by a *piecewise function*.

Definition 5.1. (Piecewise function)
Values $-\infty < x_1 < x_2 \cdots < x_{n-1} < \infty$ and continuous functions $F_1, F_2, ..., F_n : \mathbb{R} \to \mathbb{R}$ are given. Function $F : \mathbb{R} \to \mathbb{R}$ is a piecewise function given by:

$$x_i \leq x \leq x_{i+1} \Longrightarrow F(x) = F_i(x)$$

Where $x_1, ..., x_{n-1}$ are called breakpoints of F.

The easiest interpolation method connects the quantiles linearly. However, in general this will not give a suitable CDF function. The resulting function is not smooth at the breakpoints, and its PDF is a piecewise constant function. So a more advanced interpolation method should be defined, which also takes into account the properties of a proper CDF. This section described the interpolation method we have developed.

Given a set of data points $(x_i, y_i), i = 1, ..., n$ where we assume monotonicity, i.e. $y_i \leq y_{i+1}, i = 1, ..., n-1$. To approximate values between data points we will construct an interpolating function. We choose to interpolate with polynomials of degree 3 (cubic), because it allows the PDF to be smooth at the breakpoints. In each interval x_i, x_{i+1} a cubic polynomial can be constructed which can be represented by:

$$p(x) = y_i H_1(x) + y_{i+1} H_2(x) + d_i H_3(x) + d_{i+1} H_4(x),$$

where $d_i = p'(x_i)$ and $H_k(x)$ are the Hermite basis functions. $H_1(x) = 2x^3 - 3x^2 + 1$, $H_2(x) = -2x^3 + 3x^2$, $H_3(x) = x^3 - 2x^2 + x$, $H_4(x) = x^3 - x^2$. The piecewise cubic interpolating function satisfies:

$$p(x_i) = y_i \quad i = 1, ..., n.$$

An algorithm is needed for computing the derivatives $d_1, ..., d_n$. In Conijn [3] endpoint fitting and least-squares fitting are used. Famous methods such as the Three Point Difference formula, and the least-squares procedure described by Ellis and McLain do not necessarily preserve monotonicity of the data. Since a CDF needs to be monotonically increasing we need to find a satisfactory way to generate the derivatives.

The method we use is described by F. Fritsch and R. Carlson [12]. They propose a shape preserving interpolation method which is characterized by its efficiency and the minimal storage it requires. The

Fritsch and Carlson algorithm calculates for every breakpoint a value m_i, satisfying $d_i = (x_{i+1} - x_i)m_i$. The interpolated value of a point $z \in [x_i, x_{i+1}]$ is obtained by evaluating $p(\frac{z-x_i}{x_{i+1}-x_i})$.

Expressions can be derived for the coefficients of the cubic polynomials: Let $\theta = \frac{z-x_i}{x_{i+1}-x_i}$.

$$\begin{aligned} p(\theta) &= y_i H_1(\theta) + y_{i+1} H_2(\theta) + d_i H_3(\theta) + d_{i+1} H_4(\theta) \\ &= y_i H_1(\theta) + y_{i+1} H_2(\theta) + (x_{i+1} - x_i) m_i H_3(\theta) + (x_{i+1} - x_i) m_{i+1} H_4(\theta) \\ &= C_0 + C_1 z + C_2 z^2 + C_3 z^3, \end{aligned}$$

where $C_0, ..., C_3$ are functions of $x_i, x_{i+1}, y_i, y_{i+1}, m_i, m_{i+1}$. They are given by:

$$\begin{aligned} C_0 &= (x_{i+1}(x_i(-x_i + x_{i+1})(m_{i+1}x_i + m_i x_{i+1}) - x_{i+1}(-3x_i + x_{i+1})y_i) + x_i^2(x_i - 3x_{i+1}2)y_2)(x_i - x_{i+1})^{-3} \\ C_1 &= (m_{i+1}x_i(x_i - x_{i+1})(x_i + 2x_{i+1}) - x_{i+1}(m_i(-2x_1^2 + x_i x_{i+1} + x_{i+1}^2) + 6x_i(y_i - y_{i+1})))(x_i - x_{i+1})^{-3} \\ C_2 &= (-m_i(x_i - x_{i+1})(x_i + 2x_{i+1}) + m_{i+1}(-2x_1^2 + x_i x_{i+1} + x_{i+1}^2) + 3(x_i + x_{i+1})(y_i - y_{i+1}))(x_i - x_{i+1})^{-3} \\ C_3 &= ((m_i + m_{i+1})(x_i - x_{i+1}) - 2y_i + 2y_{i+1})(x_i - x_{i+1})^{-3} \end{aligned}$$

The function which interpolates the data points can now be fully described by its piecewise cubic polynomials.

In Figure 5.3 a set of quantiles is shown and a function which is the result of our interpolation method. We can see that the function follows the data points exactly, and that connections between polynomials are smooth at the breakpoints.

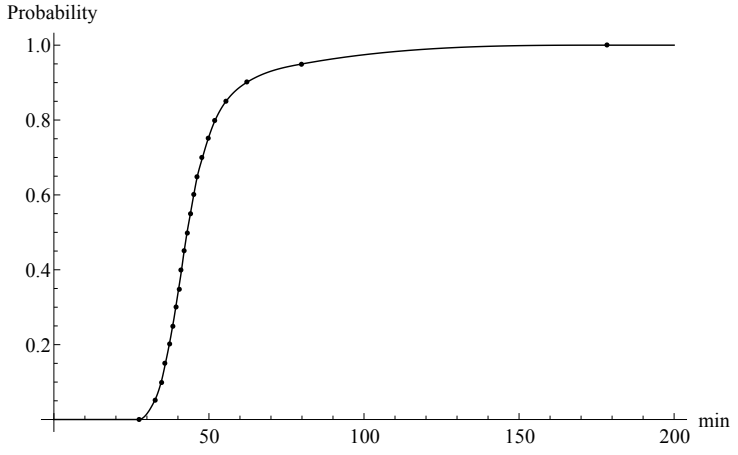


FIGURE 5.3: Quantiles from Udenhout - Waalwijk travel time distribution between 16:00 and 20:00 over the year 2009, and the function resulting from our interpolation method.

Our method constructs a function $p(x)$ which satisfies all conditions for CDF functions, namely

- $p(x)$ is non-decreasing (because our method preserves monotonicity)
- $p(x)$ is right-continuous (because polynomials are connected smoothly at breakpoints)
- $\lim_{x \to -\infty} p(x) = 0$
- $\lim_{x \to \infty} p(x) = 1$

Since our method constructs valid CDFs, all features of distribution functions are now available to us. One practically attractive possibility is to draw samples from the observed travel time distributions. This can be useful in a variety of applications.

5.3 Change of travel time distributions during the day

An extension to the problem, which is not considered in our implementation, uses different travel time distributions during the day for the same arc. However this is not included in our model, the data from TomTom offers us the possibility to analyse the change of distributions during the day. In Section 7 suggestions for further research on this topic are given.

For five different time periods data of the travel time distribution is provided by TomTom. The time periods can be found in the legend of Figure 5.4. We expect to find differences in the shape of travel time distributions at different time periods. For example, during rush hours travel times are expected to be pushed towards higher values. Also we expect rush hours increase the variation of the travel time distributions. Figure 5.4 shows the CDF of the travel time distributions between Den Bosch and Utrecht at 5 time periods during the day. A Log-normal distribution was used as an a-priori distribution. During

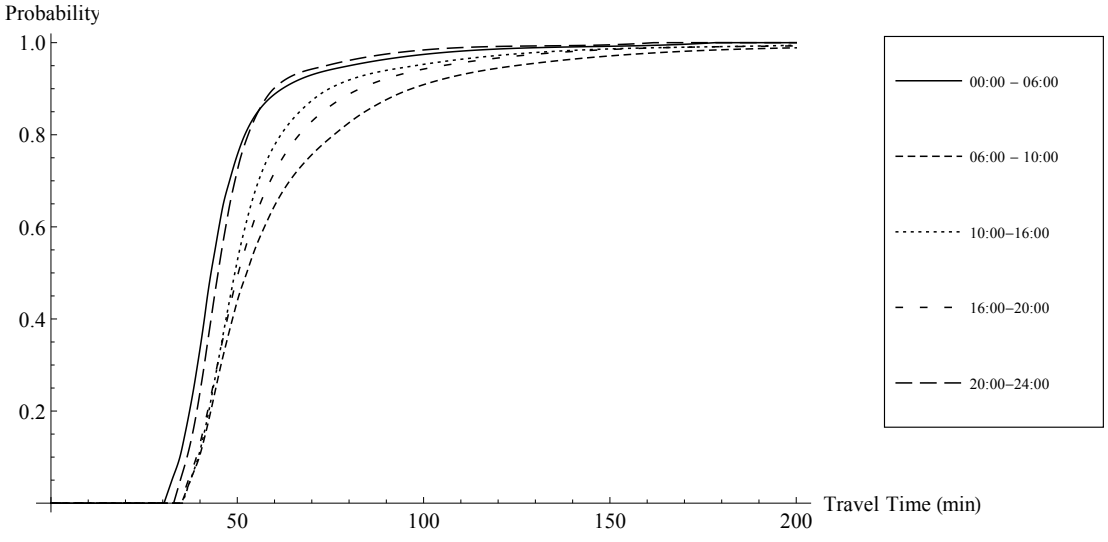


FIGURE 5.4: Travel time distribution of Den Bosch - Utrecht at 5 time periods during the day

morning rush hour (06:00 - 10:00) and evening rush hour (16:00 - 20:00) the distributions have longer tails, which means there is a higher probability of longer travel times. Also the standard deviation is higher. Further, the minimal travel times slightly shift to the right at rush hours. This can also be seen in Table 5.1 where the shifting value and the parameters of the distributions from Figure 5.4 are shown.

Since we see a clear difference between travel time distributions during the day, time dependance could give a better approximation of real life situations. Time dependance could be incorporated in an algorithm by storing multiple travel time distributions for an arc. Based on the time of the day a different distribution should be used. Moreover, a method should be developed for determining the travel time

	Log-normal				
Time period	Shift	α	β	Mean	Standard deviation
00:00 - 06:00	24.73	2.92	0.54	46.18	12.48
06:00 - 10:00	34.16	2.91	0.97	63.54	36.73
10:00 - 16:00	33.84	2.71	0.78	54.21	18.64
16:00 - 20:00	33.58	2.80	0.86	57.38	24.91
20:00 - 24:00	25.82	2.94	0.46	46.85	10.21

TABLE 5.1: Parameters of the Den Bosch - Utrecht travel time distribution during time periods of the day

distribution if the vehicle starts in one time period, and a new period is entered during the travel. This is a topic for further research.

5.4 An instance based on real data

Using TomTom data, the interpolation method described in Section 5.2, and our Branch-and-Price algorithm, we can solve instances based on real data. In this section we will consider such an instance and solve it.

The input of our algorithm consists of the standard instance data, which includes vehicle capacity, customer demands, service times, required reliabilities, arc distances, and time windows. Furthermore, for every arc a vector of quantiles is part of the input. The quantiles are interpolated using the interpolation method which is described in Section 5.2, and subsequently discretized to fit into the algorithm.

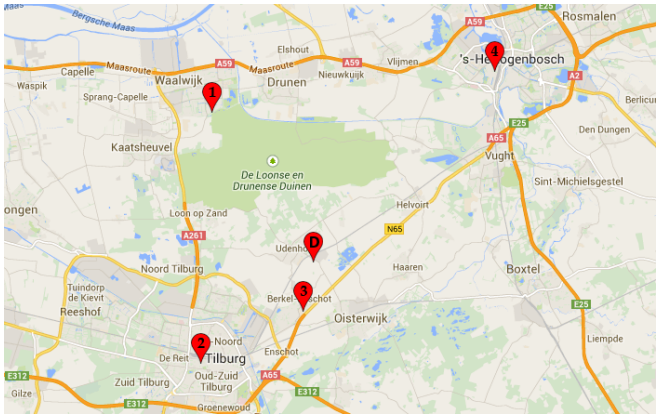


FIGURE 5.5: Map of the instance

We are going to solve a VRPhTW-STT instance with one depot and 4 customers. So the set of arcs has $n * (n-1) = 5 * 4 = 20$ elements, each with a different travel time distribution. At first we assign the same reliability requirement of $\gamma = 0.95$ to every customer. In Table 5.2 the parameters of this problem are shown. The customers are shown on a map in Figure 5.5.

Solving the problem to optimality using our algorithm results in the solution shown in Figure 5.6. Also the service reliabilities of the customers in this solution are shown. Clearly, the reliability requirement of $\gamma = 0.95$ is satisfied at every customer. All customers can be visited with one vehicle following route $D-2-1-4-3-D$ which has minimal distance. The total travel distance is 69.7 km.

Customer	Demand	Release date	Due date	Service time (min)	Required reliability
D (depot)	0	9:00	12:30	20	0.95
1	10	10:00	10:30	20	0.95
2	10	9:00	10:30	20	0.95
3	10	11:00	12:00	20	0.95
4	10	10:00	10:50	20	0.95

TABLE 5.2: Parameters of the VRPhTW-STT instance, travel time distributions come from TomTom data

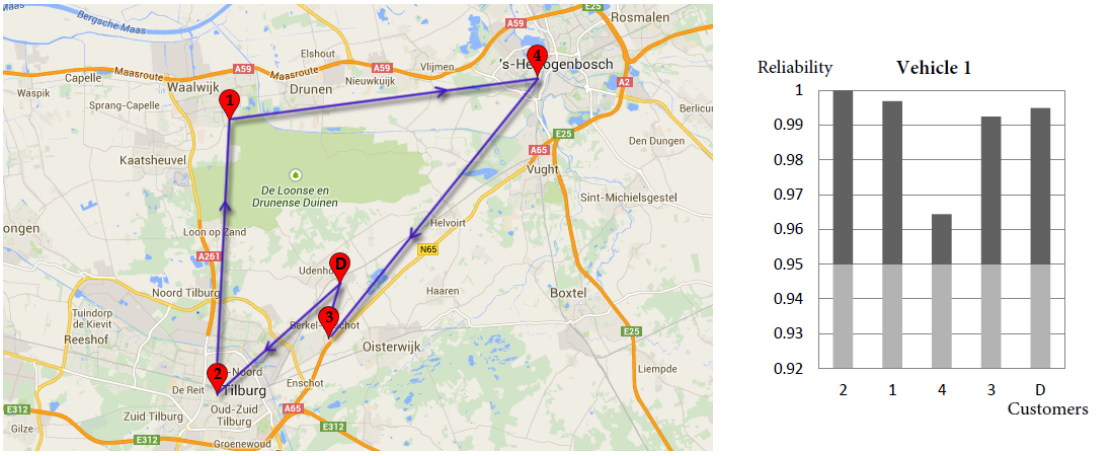


FIGURE 5.6: Solution of the real life instance, and the service reliabilities.

Increasing the required service reliability

Now let us assume that customer 4 should be served more reliable. For example, because of business requirements. Its reliability requirement is increased to $\gamma_4 = 0.98$. The solution can be found in Figure 5.7. All service reliability requirements are satisfied, in particular the increased requirement at customer 4. We can see that one additional vehicle is needed to serve all customers reliably. Furthermore, the distance of these two routes together is 74.9 km. So an increment in the reliability requirements resulted in a more expensive solution.

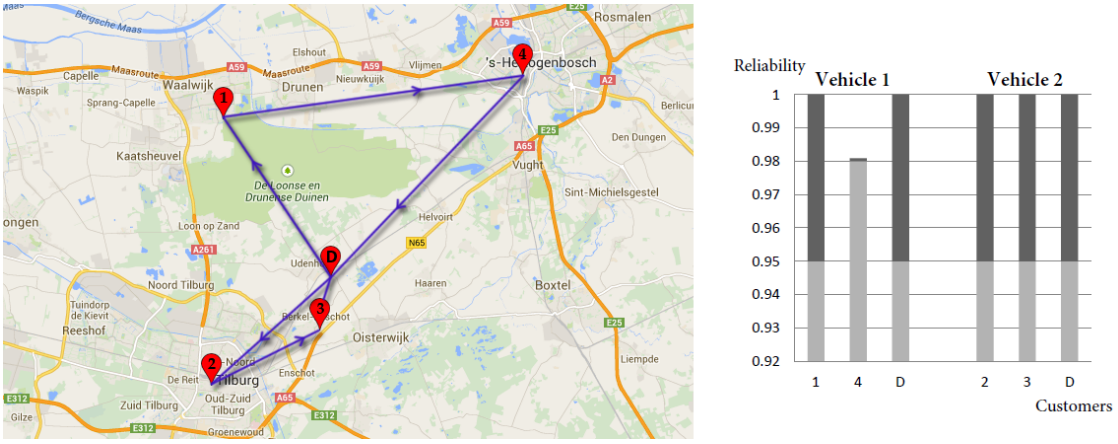


FIGURE 5.7: Solution satisfying an increased reliability requirement at customer 4, and the corresponding service reliabilities.

Chapter 6

Computational results

To analyze our algorithm the well-known set of Solomon instances for the VRPhTW is used [22]. This set contains 6 types of instances. In C instances the customers are clustered, in R instances customers are randomly spread. RC instances contain both clustered and random customers. Moreover, instances can have short routes (denoted with 1), or long routes (denoted with 2). Based on the Solomon instances, which contain 100 customers each, smaller instances are created. In research on the VRPhTW this is always done by taking the first n customers from the instances.

We will analyse our Branch-and-Price algorithm for two cases: First when the MP uses routes from $\mathcal{R}^{\mathcal{D}}$. This means it solves the deterministic VRPhTW. Secondly, when the MP uses routes from $\mathcal{R}^{\mathcal{S}}$, we solve the actual VRPhTW-STT problem.
To compare the results for the deterministic case, we start by solving the Solomon instances using the MIP defined in Section 3.1.

6.1 Results of the MIP

The deterministic VRPTW formulated as a MIP is practically solved using CPLEX. Computations are made on a computer with an Intel(R) Core(TM) i5 CPU (2.40 GHz) and 8.00 GB RAM. Table 6.1 shows the number of Solomon instances solved within 20 minutes (1200 seconds). The instances are grouped by instance type. A full enumeration of computation times can be found in Table A.1 of the Appendix.

		Nr. of customers									
Instance	#	10	20	30	40	50	60	70	80	90	100
C1	10	10	6	5	4	4	4	4	4	4	3
C2	8	8	8	6	2	1	1	1	1	1	1
R1	12	12	3	3	2	2	1	1	1	1	1
R2	11	11	3	1	1	0	0	0	0	0	0
RC1	8	8	4	1	1	1	0	0	0	0	0
RC2	8	8	2	0	0	0	0	0	0	0	0

TABLE 6.1: Number of Solomon instances solved within 20 minutes, grouped by instance type.

Table A.1 shows that instances up to 20/30 customers can be solved within reasonable times. However, a clear dependency on the instance type can be observed. Some instances are solved up to 100 customers, others not even to 20. This MIP based method could be further developed, and probably computation times would decrease. But since there is no guarantee of convergence, advanced solution methods provide a better prospect of solving larger instances. The following section will show the results of our Branch-and-Price method when it solves the same problems.

6.2 Branch-and-Price results for the VRPhTW

Using the Branch-and-Price method described in Section 3 the Solomon instances are solved. We solve the instances for $10, 20, 30, 40$, and 50 customers. We can solve some instances such as c101, c105, c106, c107, and r101 up to 100 customers. But since in general this is not the case, we take 50 as an upper bound. The RMP is solved with CPLEX, and computations are made on a computer with an Intel(R) Core(TM) i5 CPU (2.40 GHz) and 8.00 GB RAM. The details of the solutions and the corresponding computation times can be found in appendix A.3.

Figures (6.1) – (6.5) show the computation times graphically. We use a time limit of 2500 seconds. Instances are grouped by instance type. For every instance type also the number of solutions is shown which is solved within the time limit. All solutions of the solved instances can be found in Appendix A.3.

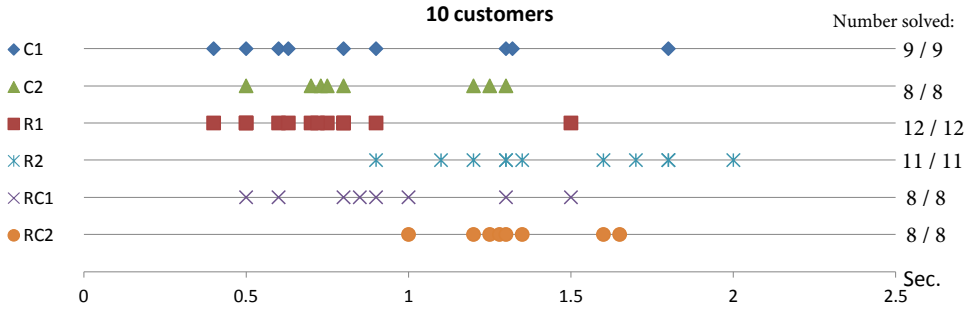


FIGURE 6.1: Solution time of Solomon instances restricted to the first 10 customers

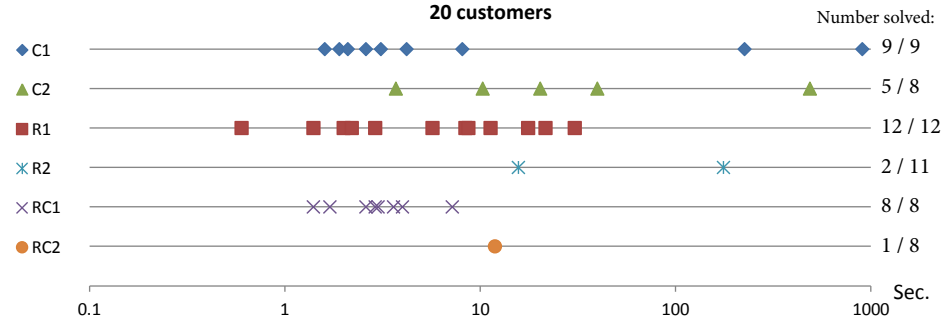


FIGURE 6.2: Solution time of Solomon instances restricted to the first 20 customers

Our algorithm solves the benchmark instances to optimality. The solutions are similar to the best known solution. As Figures 6.1 – 6.5 show, the instance type highly influences the computation time. Naturally, the more customers an instance has, the more difficult it is to solve, and the more time it takes to find the optimal solution. Furthermore, the 2 instances are clearly more complex. This is because their routes contain more customers than in the 1 instances. So the enumeration of all possible routes in the Sub Problem is more time consuming.

Let us compare the results of our MIP and this Branch-and-Price method. Compared to the MIP (results can be found in Table A.1), we generally solve the same instances, and we can often solve them with more customers. For example, the MIP can only solve c109 with 10 customers within 20 minutes. The Branch-and-Price method solves the same instance with up to 50 customers within 20 minutes.

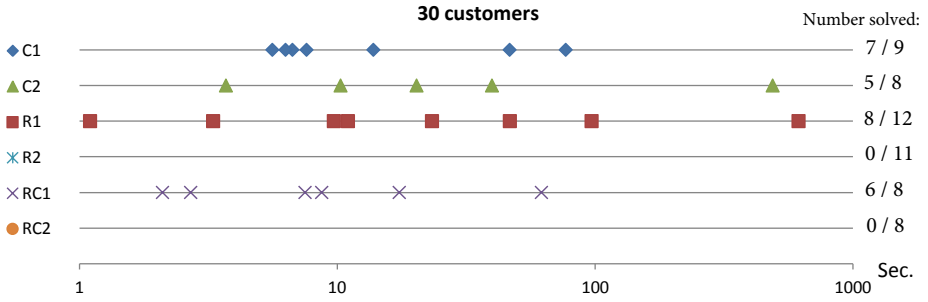


FIGURE 6.3: Solution time of Solomon instances restricted to the first 30 customers

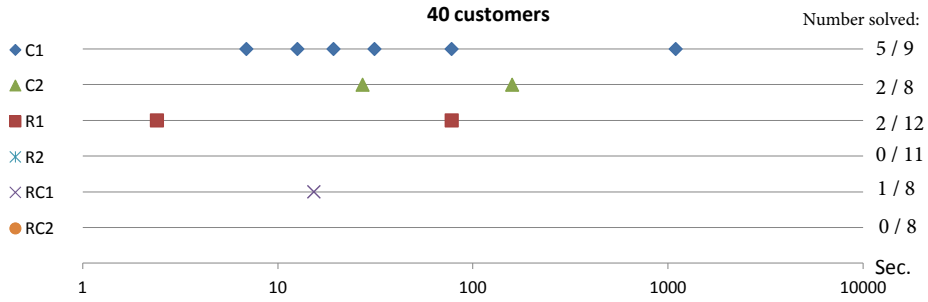


FIGURE 6.4: Solution time of Solomon instances restricted to the first 40 customers

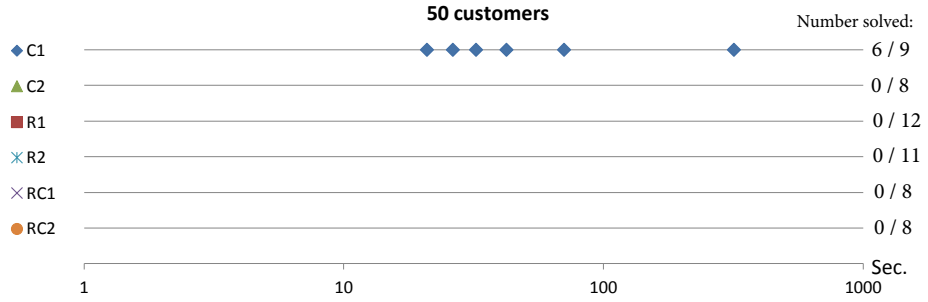


FIGURE 6.5: Solution time of Solomon instances restricted to the first 50 customers

For highly constrained instances which are clustered, such as c101, the MIP finds the optimal solution quick. But in general the MIP cannot solve instances with more than 20/30 within reasonable times. The power of the Branch-and-Price method lies in its flexibility to solve different kinds of instance types.

Furthermore, by improving the algorithm for the ESPPRC, the efficiency of the Branch-and-Price can be improved. In Chapter 7 we will make some suggestions for further research on this matter.

6.3 Branch-and-Price results for the VRPhTW-STT

In this section we are going to analyze our algorithm for solving the VRPhTW-STT. For the VRPhTW-STT no set of benchmark instances is available. Therefore, we add stochastic travel times to the deterministic Solomon instances. In the Solomon instances distance and time are equivalent. So based on the deterministic travel distance a travel time distribution is assigned to every arc. The distribution of our

choice is a shifted Log-normal distribution, since Chapter 5 shows it can resemble the properties of travel time distributions. Conijn [3] uses a similar approach.

In our analysis of the algorithm we vary two parameters in the instances:

- The variance of the travel time distribution,
- The required service reliability.

Two cases for the variance are considered: One in which the variance of the travel time distributions is low, and one in which the variance is higher. This is graphically shown in Figure 6.6. The Log-normal distribution with low variance has parameters $(3, 0.15)$ and is shifted according to the deterministic travel time. The Log-normal distribution with high variance has parameters $(3, 0.25)$ and is shifted similarly.

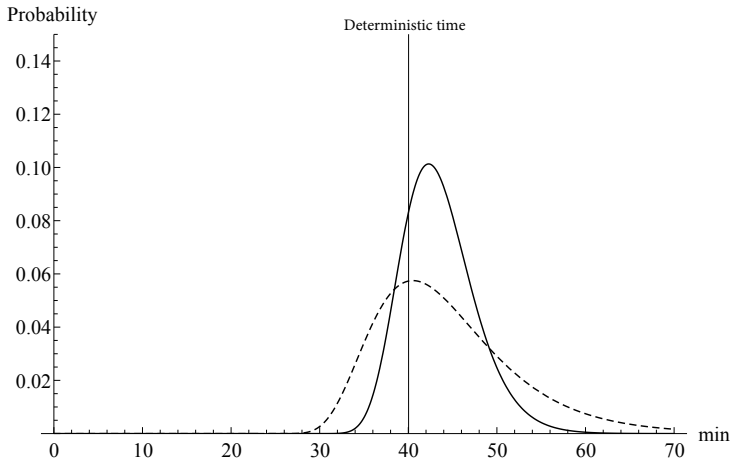


FIGURE 6.6: Transformation of a deterministic time to a travel time distribution. Two shifted Log-normal distributions are considered. One with low variance, and one with high variance.

Furthermore, we will consider two values of the reliability threshold, $\gamma = 0.95$ and $\gamma = 0.99$. Without loss of generality, it is assumed that all customers have the same reliability threshold.

For doing computational tests, we select a set of instances with a variety of properties. The set contains instances c101, r105, c201, and rc107. We compute the solution with our Branch-and-Price algorithm and compare it with the deterministic solution. We use a time limit of 2500 seconds. If the algorithm does not finish within the time limit, we denote this with a "-" for the corresponding instance. The results for the 10 customer instances can be found in Table 6.2. The results for the 25 customer instances can be found in Table 6.3. Computation times smaller than 10^{-2} are denoted with $*$.

All found solutions are feasible, and satisfy the service reliability requirements at the customers. We checked this by calculating the start of service distribution at all customers to verify the service requirements.

Instance	Variance	Reliability	Vehicles	Distance	Time (s)	RMP (s)	Pricing (s)	Branchings
c101 10	Deterministic		1	58.3259	0.6	∗	0.1	0
	Low	0.95	2	88.4504	1.3	∗	0.7	2
		0.99	2	88.4504	1.2	∗	0.7	2
	High	0.95	2	88.4504	1.8	∗	1.3	2
		0.99	2	92.4505	3.7	∗	2.4	13
r105 10	Deterministic		3	253.0714	0.5	∗	∗	1
	Low	0.95	4	287.3363	0.6	∗	0.3	0
		0.99	4	287.3363	0.7	∗	0.3	0
	High	0.95	4	287.3363	0.7	∗	0.3	0
		0.99	5	314.2545	0.7	∗	0.2	0
c201 10	Deterministic		1	194.6648	0.5	∗	∗	0
	Low	0.95	1	194.6648	0.6	∗	0.3	0
		0.99	1	194.6648	0.7	∗	0.3	0
	High	0.95	1	194.6648	0.9	∗	0.5	0
		0.99	1	194.6648	1.0	∗	0.5	0
rc107 10	Deterministic		2	168.1672	1.1	∗	0.2	0
	Low	0.95	2	207.8415	2.6	∗	2.1	0
		0.99	3	233.9216	2.8	∗	2.3	1
	High	0.95	3	233.9216	5.0	∗	4.4	1
		0.99	3	236.1552	9.1	∗	7.9	14

TABLE 6.2: Solutions to the selected instances restricted to 10 customers. Computation times smaller than 10^{-2} are denoted with ∗.

The results show that the solutions change both when the travel time variance is increased, and when the reliability requirement is increased. Computation times are reasonable, and generally increase once the variance of the travel time distribution increases, or the reliability requirement increases.
The computation times of the instances with 10 customers are of the same order as in the deterministic case. However, for the 25 customer instances, computation times have increased. Here the additional distribution calculations in every extension check of the Labeling Algorithm increase the computation times of the Sub Problem.

Many instances having more 25 customers can also be solved. But the computation times highly depends on the instance type. For example, instance r105 with 50 customers and low variance, is solved in 95 seconds. But instance rc107 with 50 customers and the same configuration can not be solved within 2 hours.

Instance	Variance	Reliability	Vehicles	Distance	Time (s)	RMP (s)	Pricing (s)	Branchings
c101 25	Deterministic		3	191.8136	2.8	0.1	0.6	0
	Low	0.95	-	-	-	-	-	-
		0.99	-	-	-	-	-	-
	High	0.95	5	302.5077	3164.2	49.5	2992.5	525
		0.99	6	309.7725	237.2	0.6	231.3	41
r105 25	Deterministic		5	556.7161	7.0	0.2	2.6	18
	Low	0.95	8	632.8664	1.7	∗	0.9	0
		0.99	8	642.6928	1.5	∗	0.7	0
	High	0.95	8	685.0695	2.9	∗	2.1	0
		0.99	11	797.7589	2.2	∗	1.3	0
c201 25	Deterministic		2	215.5426	7.7	0.8	2.6	1
	Low	0.95	2	215.5426	174.2	2.0	167.5	1
		0.99	2	227.2936	754.2	30.5	659.4	44
	High	0.95	2	227.2937	2681.3	40.8	2562.5	51
		0.99	2	228.1855	2105.5	45.5	1966.4	50
rc107 25	Deterministic		3	298.9498	10.6	0.4	7.1	0
	Low	0.95	-	-	-	-	-	-
		0.99	6	509.7346	618.2	0.5	608.5	81
	High	0.95	6	516.4358	1993.4	1.9	1963.6	184
		0.99	6	526.2544	248.8	0.1	243.1	27

TABLE 6.3: Solutions to the selected instances restricted to 25 customers. Computation times smaller than 10^{-2} are denoted with ∗.

The average computation times for solving the instances from Table 6.2 and Table 6.3 can be found in Table 6.4. We separate the four different configurations.

Variance	Low		High	
Reliability requirement	0.95	0.99	0.95	0.99
10 customers	1.3	1.4	2.1	3.6
25 customers	87.9	1100.3	1960.5	648.8

TABLE 6.4: Average computation time in seconds for the four different configurations

We can see that computation times generally increase when the variance becomes higher, or when the reliability requirement becomes higher. However, this is not always the case, as is shown by the 25 customer instances with high variance and a 0.99 reliability requirement. Here the reliability requirement restricts the number of feasible routes, such that the Sub Problem is solved quicker.

Comparing our results to results found in literature would be desirable. However, in recent articles mainly heuristic methods are used [3, 21]. This allows for solving instances with 100 customers and even more. Since we have an exact method which does not solve instances of those sizes, comparing the results can not be directly done.

Chapter 7

Concluding remarks

In conclusion, we have constructed an algorithm which solves the VRPhTW, and the VRPhTW-STT. The algorithm uses a Branch-and-Price approach. We have compared the results from our deterministic Branch-and-Price algorithm with the results from a MIP. The Branch-and-Price method proved to be more flexible, since it could solve instances from various instance types within reasonable times. The MIP was only suitable for solving highly constrained instances. Furthermore, our algorithm can be improvements while maintaining its structure.

We solved VRPhTW-STT instances based on Solomon instances. A representative set of instances was used to analyse the algorithm. Computation times are reasonable for instances up to 25 customers. Instances with a higher number of customers can be solved in some cases. This highly depends on the instance type. Instances with longer routes are more time consuming, since the enumeration of paths in the Sub Problem takes more time.
Furthermore, by varying parameters of the instances we have seen the following: An increment in the required service reliability, or in the variation of the travel time distributions, will lead to solutions having more vehicles and / or distance.

Travel time distributions were assumed to be discrete random variables, and offered us all necessary tools for computing with distributions efficiently. Efficiency was one of the main targets in computing with distributions, since solving the ESPPRC requires a number of extension checks, which grows exponentially in the number of customers.

Finally, we were able to solve an instance based on real travel time data from TomTom.

7.1 Further research

General extensions of the problem
A straightforward extension incorporates time dependent travel times into the problem. This will increase the correspondence with real life situations, since the effects of rush hours will be included in the solutions. Research could be done on the change of the shape of the travel time distributions during the day. In terms of known distributions, a function of the parameters could define the travel time distribution during the day. Furthermore, a method should be developed for determining the travel time distribution if a vehicle starts in one time period, and a new period is entered during the travel.

Improving the algorithm for the ESPPRC
The labeling algorithm could be improved by adding dominance rules. Furthermore, a combination

of a heuristic method, and an exact method could improve the speed. The heuristic method, which is generally quicker, could be used in the beginning. The exact method could be used in the end, to guarantee no routes with negative reduced cost exist.

In our labeling algorithm paths are extended only from the front. A suggestion for a bi-directional method was proposed in Section 3.3.2, but still forward checks were required. A full bi-directional method, which does not rely on forward checks anymore, could speed up the algorithm.

Representing distributions

We assumed that travel time distributions are discrete random variables. Using continuous distributions requires a different representation, and a different implementation. Once such a representation is found, e.g. using piecewise functions, and sufficiently efficient calculation methods are developed, it could be used in a Branch-and-Price framework.

In the input of our algorithm, we do not check whether the triangular inequality is satisfied for the arcs. For example, it could be possible that the service reliability at customer c is higher in route $a - b - c$ than in route $a - c$. For practical purposes it could be necessary that this is avoided by preprocessing the travel time distributions of the arcs.

Travel time data

There is no limit on the information travel time data offers us. More research could be done on the shape of travel time distributions. In particular, in combination with a time dependent algorithm, instances which incorporate more properties of real life networks could be solved. A connection between an algorithm and a database of travel time data, such as from TomTom, could be made. This is something currently researched in the DAIPEX project.

Bibliography

[1] T. Chang, Y. Wan, W. Tsang, "Stochastic dynamic traveling salesman problem with hard time windows," 2009.

[2] G. Clarke, J.W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," 1962.

[3] B.J. Conijn, "Modeling and Solving the Vehicle Routing Problem with Stochastic Travel Times and Hard Time Windows," 2013.

[4] S. Dabia, S. Ropke, T. van Woensel, T. de Kok, "Branch and Price for the Time-Dependent Vehicle Routing Problem with Time Windows" 2013.

[5] G.B. Dantzig, J.H. Ramser, "The Truck Dispatching Problem," 1959.

[6] M. Desrochers, J. Desrosiers, M. Solomon, "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows," 1992.

[7] J. Desrosiers, M. Lubbecke, "A Primer in Column Generation," 2004.

[8] A. Erera, M. Savelsbergh, E. Uyar, "Fixed Routes with Backup Vehicles for Stochastic Vehicle Routing Problems with Time Constraints," 2009.

[9] F. Errico, G. Desaulniers, M. Gendreau, W. Rei, L. Rousseau, "A Priori Optimization with Recourse for the Vehicle Routing Problem with Hard Time Windows and Stochastic Service Times," 2014.

[10] D. Feillet, P. Dejax, M. Gendreau, C. Gueguen, "An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to Some Vehicle Routing Problems," 2004.

[11] M.L. Fisher, R. Jaikumar, "A Generalized Assignment Heuristic for Vehicle Routing," 1979.

[12] F.N. Fritsch, R.E. Carlson "Monotone Piecewise Cubic Interpolation," 1980.

[13] J. Homberger, H. Gehring "Two evolutionary metaheuristics for the vehicle routing problem with time windows," 1999.

[14] B. Kallehauge, "On the vehicle routing problem with time windows," 2005.

[15] A. Kenyon, D. Morton, "Stochastic Vehicle Routing with Random Travel Times," 2003.

[16] A.W.J. Kolen, A.H.G. Rinnooy Kan, H.W.J.M Trienekens, "Vehicle routing with time windows," 1987.

[17] J. Larsen, "Vehicle Routing with time windows - finding optimal solutions efficiently," 1999.

[18] C. Lecluyse, T. van Woensel, H. Peremans, "Vehicle routing with stochastic time-dependent travel times," 2009.

[19] H. Lei, G. Laporte, B. Guo, "A generalized variable neighborhood search heuristic for the capacitated vehicle routing problem with stochastic service times," 2011.

[20] J.K. Lenstra, A.H.G. Rinnooy Kan, "Complexity of vehicle routing and scheduling problems," 1981.

[21] S. Samaranayake, S. Blandin, A. Bayen, "A tractable class of algorithms for reliable routing in stochastic networks," 2011.

[22] M.M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," 1987.

[23] I. Sungur, Y. Ren, F. Ordonez, M. Dessouky, H. Zhong, "Model and Algorithm for the Courier Delivery Problem with Uncertainty," 2010.

[24] D. Tas, "Time and reliability in vehicle routing problems," 2013.

[25] P. Toth, D. Vigo, "The Vehicle Routing Problem," 2001.

Appendix A

Appendix

A.1 MIP formulation of the ESPPRC for the VRPhTW

This is a MIP formulation of the ESPPRC, the Sub Problem of the CG procedure for the deterministic VRPhTW. Practically solving this MIP results in the route with minimal reduced cost.

$$\min \sum_{(i,j)\in A} (c_{ij} - \pi_i^*)x_{ij} \quad s.t. \tag{A.1}$$

$$\sum_{i\in N} x_{ik} - \sum_{j\in N} x_{kj} = 0 \quad k \in N \tag{A.2}$$

$$\sum_{j\in N} x_{0j} = 1 \tag{A.3}$$

$$\sum_{i\in N} x_{i0} = 1 \tag{A.4}$$

$$L_i \leq Q + Q(1 - x_{0i}) \quad \forall i \in N \tag{A.5}$$

$$A_i + s_i \leq D_i \quad \forall i \in N \tag{A.6}$$

$$r_i + s_i \leq D_i \quad \forall i \in N \tag{A.7}$$

$$A_i \leq d_i \quad \forall i \in N \tag{A.8}$$

$$L_i - q_i \geq L_j - d_0(1 - x_{ij}) \quad \forall i \in N\backslash\{0\}, \forall j \in N\backslash\{i\} \tag{A.9}$$

$$D_i + t_{ij} \leq A_j + d_0(1 - x_{ij}) \quad \forall i \in N, \forall j \in N\backslash\{i\} \tag{A.10}$$

$$x_{ij} \in \{0, 1\} \quad i \in N, j \in N \tag{A.11}$$

Constraints (A.2) takes care of the flow conservation. Constraints (A.3) and (A.4) make the route start and end at the depot. (A.5) are the capacity constraints. This is done by stating that the load of the vehicle when departing from the depot can not exceed its capacity. If $x_{0i} = 0$ for some $i \in N\backslash\{0\}$ this constraint becomes redundant. Constraints (A.6), (A.7) and (A.8) are the timing constraints. Constraints (A.9) are needed to make sure that the load in the vehicle is large enough to satisfy all the demands of the customers in the route. A large coefficient u_0 is used to linearize the constraints. If arc x_{ij} is not used this constraint is automatically satisfied. Constraints (A.10) are the time update constraints.

A.2 Computation times for the VRPTW using a MIP

	10	20	30	40	50	60	70	80	90	100		10	20	30	40	50
c101	0.3	0.24	0.34	0.41	0.54	0.72	0.95	1	1.12	1.4	**r201**	0.38	3.1	0.79	550	
c102	0.88	0.41									**r202**	4.94				
c103	0.9										**r203**	5.13				
c104	8.2										**r204**	8.6				
c105	0.2	0.36	0.39	1.4	0.73	1.6	1.4	1.5	2	2.6	**r205**	0.21	9.45			
c106	0.2	0.3	0.39	0.5	0.67	2	3.2	3.9	50.1		**r206**	11.1				
c107	0.42	0.76	0.81	46.1	1.8	2.2	3.5	3.2	6.7	7.4	**r207**	11.8				
c108	0.25	51.4	741								**r208**	56.7				
c109	0.3										**r209**	0.93	435			
c201	0.4	0.5	0.9	0.48	0.83	5.3	2.84	1.2	1.6	1.7	**r210**	3.6				
c202	2.3	1.9	52.4								**r211**	25				
c203	1.8	17.1									**rc101**	0.2	0.48	0.83	3.9	59
c204	10.2	180									**rc102**	3.6	17.3			
c205	0.45	0.69	3.96	234							**rc103**	3.6				
c206	0.36	1.4	474.4								**rc104**	66				
c207	0.55	4	240								**rc105**	1.9	601			
c208	0.48	6									**rc106**	0.6	0.7			
r101	0.27	0.36	0.49	0.46	0.71	0.73	1.16	1.28	1.68	3.94	**rc107**	8.1				
r102	4.4	123.6	240								**rc108**	36.5				
r103	3.5										**rc201**	0.59	13.3			
r104	59										**rc202**	23.6				
r105	0.5	0.95	0.9	14	660						**rc203**	23.3				
r106	3.3										**rc204**	63.8				
r107	2.6										**rc205**	4.9				
r108	17.1										**rc206**	1.5	186			
r109	0.5										**rc207**	3.5				
r110	1.2										**rc208**	137.4				
r111	3.7															
r112	9.5															

Table A.1: Computation times in seconds for solving the VRPTW with the MIP defined in Section 3.1. An empty cell means the instance did not solve within 20 minutes.

A.3 Solving the deterministic VRPhTW using Branch-and-Price

Times smaller than 10^{-2} are denoted with $*$. We use a time limit of 2500 seconds. Instances for which the algorithm did not finish within 2500 seconds are denoted with "-".

Instance	Customers	Vehicles	Distance	Time (s)	RMP time (s)	Pricing time (s)	Branchings
c101	10	1	58.3259	0.6	*	0.1	0
	20	3	175.3729	1.7	*	0.3	0
	30	3	206.1765	6.8	0.2	4.0	0
	40	5	344.6347	6.8	0.3	2.7	0
	50	5	363.2468	20.7	2.6	10.3	0
c102	10	1	57.2500	1.4	*	*	0
	20	2	167.1649	8.3	0.8	4.2	0
	30	3	205.1005	101.0	10.6	77.0	0
	40	5	343.5588	81.8	7.3	60.0	0
	50	5	362.1708	634.9	64.4	529.1	0
c103	10	1	57.2499	1.4	*	*	0
	20	2	161.5734	225.8	4.2	212.3	0
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
c104	10	1	56.4075	1.9	*	0.8	0
	20	2	159.4762	905.1	2.3	861.7	0
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
c105	10	1	58.3260	0.6	*	*	0
	20	3	175.3729	2.1	0.0	0.5	1
	30	3	206.1765	6.0	0.2	2.9	0
	40	5	344.6347	18.9	2.6	10.0	1
	50	5	363.2468	31.8	5.5	15.1	0
c106	10	1	58.3260	0.6	*	*	0
	20	3	175.3729	1.8	*	0.4	0
	30	3	206.1765	5.4	0.4	2.2	0
	40	5	344.6347	12.0	1.3	5.3	1
	50	5	363.2468	25.4	3.3	12.8	0
c107	10	1	58.3259	0.8	*	*	0
	20	3	175.3729	2.0	*	0.4	1
	30	3	206.1765	7.1	0.7	2.6	0
	40	4	460.8425	30.8	5.2	16.4	0
	50	5	363.2468	40.9	7.1	21.0	0
c108	10	1	57.4975	0.9	*	0.1	0
	20	3	175.3729	2.9	0.1	1.1	1
	30	3	206.1765	13.7	1.7	7.2	0
	40	-	-	-	-	-	-
	50	5	363.2468	68.7	11.9	40.5	0
c109	10	1	57.4975	0.9	*	0.1	0
	20	2	160.8159	3.8	0.3	1.4	0
	30	3	206.1765	48.1	6.0	32.4	0
	40	4	336.8229	1096.1	104.8	353.1	11
	50	5	363.2468	318.2	26.4	265.0	0

TABLE A.2: Solutions of the $c1$ instances

Instance	Customers	Vehicles	Distance	Time (s)	RMP time (s)	Pricing time (s)	Branchings
c201	10	1	194.6648	0.5	*	*	0
	20	1	265.7299	3.6	0.1	1.4	0
	30	2	233.3983	27.3	6.8	13.1	1
	40	2	326.3196	29.1	5.6	9.9	1
	50	-	-	-	-	-	-
c202	10	1	144.9872	1.4	*	0.4	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
c203	10	1	144.9872	1.4	*	0.4	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
c204	10	1	133.1340	1.3	*	0.3	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
c205	10	1	192.7809	0.8	*	*	0
	20	1	265.5160	10.6	1.8	5.5	0
	30	2	233.3983	169.1	97.6	46.6	0
	40	2	326.3196	183.7	72.5	82.3	0
	50	-	-	-	-	-	-
c206	10	1	170.6844	1.4	*	0.1	0
	20	1	229.9636	20.9	6.5	8.9	0
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
c207	10	1	170.6844	0.9	*	0.1	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
c208	10	1	146.3695	0.8	*	*	0
	20	1	204.4865	41.2	14.8	18.2	0
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-

TABLE A.3: Solutions of the $c2$ instances

Instance	Customers	Vehicles	Distance	Time (s)	RMP time (s)	Pricing time (s)	Branchings
r101	10	4	269.5331	0.7	*	*	0
	20	6	516.6109	0.7	*	*	0
	30	10	682.0534	1.2	*	0.2	0
	40	11	887.9158	1.7	*	0.3	0
	50	-	-	-	-	-	-
r102	10	3	229.7685	0.7	*	*	0
	20	6	434.9094	1.5	*	0.3	1
	30	8	585.7980	3.4	0.1	0.9	1
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r103	10	3	229.7685	0.6	*	*	0
	20	4	379.9403	16.6	2.6	5.9	31
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r104	10	2	198.2123	1.0	*	0.1	0
	20	3	331.8813	6.0	0.3	3.2	0
	30	4	427.5688	615.1	11.2	584.7	7
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r105	10	3	253.0714	0.5	*	*	1
	20	4	460.7799	2.2	*	0.6	4
	30	6	612.7979	12.2	0.5	5.0	38
	40	7	850.0400	88.3	8.5	46.1	151
	50	-	-	-	-	-	-
r106	10	2	229.3870	0.7	*	0.1	0
	20	4	384.6513	3.2	0.1	1.0	4
	30	5	494.6138	9.6	0.9	4.6	1
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r107	10	2	229.3869	0.8	*	0.1	0
	20	3	372.0772	8.7	0.3	6.0	1
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r108	10	2	198.2123	0.9	*	0.1	0
	20	3	314.8980	33.9	1.3	28.5	1
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r109	10	2	251.3536	0.6	*	*	0
	20	4	373.5413	2.2	*	0.7	1
	30	5	511.1847	23.9	2.9	12.3	36
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r110	10	2	213.7465	0.9	*	0.2	1
	20	3	409.6713	9.2	0.3	6.3	7
	30	4	521.1077	48.3	3.3	38.4	8
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r111	10	2	223.3289	0.8	*	0.1	0
	20	3	372.4205	11.5	1.4	6.1	9
	30	4	467.2619	104.5	8.1	81.0	10
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r112	10	2	198.2123	1.5	*	0.4	1
	20	3	310.7046	23.6	0.6	20.2	1
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-

TABLE A.4: Solutions of the $r1$ instances

Instance	Customers	Vehicles	Distance	Time (s)	RMP time (s)	Pricing time (s)	Branchings
r201	10	1	253.8652	0.9	∗	0.1	0
	20	2	415.3270	1.6	2.6	9.7	1
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r202	10	1	226.2925	2.1	0.1	0.7	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r203	10	1	226.2926	2.0	0.1	0.6	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r204	10	1	182.0459	1.5	∗	0.4	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r205	10	1	209.8145	1.3	∗	0.2	1
	20	1	359.0294	168.6	34.0	116.4	0
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r206	10	1	173.0420	1.9	∗	0.2	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r207	10	1	173.0420	1.9	∗	0.7	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r208	10	1	173.0420	1.6	∗	0.4	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r209	10	1	196.4410	1.1	∗	0.2	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r210	10	1	202.0562	1.5	∗	0.3	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
r211	10	1	183.0391	1.5	∗	0.4	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-

TABLE A.5: Solutions of the $r2$ instances

Instance	Customers	Vehicles	Distance	Time (s)	RMP time (s)	Pricing time (s)	Branchings
rc101	10	2	185.9079	0.5	*	*	0
	20	4	383.2760	2.9	*	0.5	8
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc102	10	2	169.6832	0.9	*	0.1	0
	20	3	314.1720	2.9	*	1.4	0
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc103	10	2	169.6832	1.0	*	0.1	0
	20	3	306.2203	3.3	0.3	1.1	0
	30	4	460.5141	7.9	0.3	4.5	0
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc104	10	2	166.0485	1.4	*	0.4	0
	20	3	295.8761	3.3	0.2	1.1	0
	30	4	428.3405	16.6	0.6	11.9	0
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc105	10	2	179.3056	1.0	*	0.2	0
	20	4	377.0774	1.5	*	0.3	0
	30	5	545.0712	2.0	*	0.3	0
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc106	10	2	178.0879	0.7	*	0.1	0
	20	3	317.0178	2.0	*	0.5	0
	30	4	471.5379	2.7	*	0.9	0
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc107	10	2	168.1672	1.1	*	0.2	0
	20	3	285.8503	3.1	0.1	1.2	0
	30	4	420.1513	7.6	0.5	3.6	0
	40	5	532.4020	12.8	0.8	6.9	0
	50	-	-	-	-	-	-
rc108	10	2	166.0485	1.7	*	0.7	0
	20	3	283.7316	7.7	0.3	4.9	0
	30	4	416.1960	65.1	1.1	58.7	0
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-

TABLE A.6: Solutions of the $rc1$ instances

Instance	Customers	Vehicles	Distance	Time (s)	RMP time (s)	Pricing time (s)	Branchings
rc201	10	1	194.5662	1.1	*	0.2	0
	20	2	399.0126	12.2	1.8	6.5	1
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc202	10	1	179.9079	1.6	*	0.5	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc203	10	1	179.9079	1.6	*	0.5	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc204	10	1	137.7768	1.7	*	0.6	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc205	10	1	225.9774	1.4	*	0.4	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc206	10	1	186.0716	1.1	*	0.1	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc207	10	1	179.2982	1.0	*	0.2	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-
rc208	10	1	137.7768	1.5	*	0.4	0
	20	-	-	-	-	-	-
	30	-	-	-	-	-	-
	40	-	-	-	-	-	-
	50	-	-	-	-	-	-

TABLE A.7: Solutions of the *rc*2 instances