Modeling and solving the vehicle routing problem with stochastic travel times and hard time windows

Conijn, B.J.

Award date:
2013

Eindhoven University of Technology
Department of Mathematics and Computer Science

Modeling and Solving the Vehicle Routing Problem with Stochastic Travel Times and Hard Time Windows

B.J. Conijn
b.j.conijn@student.tue.nl

Supervisor:
Prof. dr. ir. W.P.M. Nuijten

July 10, 2013

Abstract

In practically all real-world planning and scheduling problems one is faced with uncertainty. Usually this uncertainty is abstracted away during the modeling process, such that standard combinatorial optimization algorithms can be used to solve the problem. This abstraction step is necessary, because these algorithms generally cannot deal with uncertainty.

In this thesis we study the Vehicle Routing Problem with Stochastic Travel Times and Hard Time Windows (SVRP-HTW), a vehicle routing problem where we focus on uncertainty related to travel times. If the travel times are stochastic and their distribution is known, then they can be modeled using random variables.

For solving the SVRP-HTW, an addition and a maximum operation for random variables is required. Therefore we developed a probabilistic arithmetic, which can be used for computing with arbitrary and independent random variables. It is sufficiently generic that it also can be used for solving other stochastic optimization problems. We show that this method is reasonably fast and is an improvement over Monte-Carlo simulation.

As no benchmark for the SVRP-HTW was available, we created one based on the Solomon benchmark. We solved the instances using an existing implementation of a VRPTW solver, which we adapted, using our probabilistic arithmetic, for solving the SVRP-HTW. Our results show that our probabilistic arithmetic is sufficiently accurate and that its use does not degrade the effectiveness of the original VRPTW solver. However, it does significantly increase computation time.

Contents

Chapter 1

Introduction

The Vehicle Routing Problem (VRP) [Dantzig and Ramser, 1959] is a thoroughly studied problem in which a number of vehicles have to deliver shipments to customers under various constraints. The VRP is a generalization of the traveling salesman problem and therefore is NP-hard [Lenstra and Rinnooy Kan, 1981].

There exist many variations of the VRP. In the Capacitated VRP (CVRP) the vehicles have limited cargo capacity, which limits the number of shipments that can be delivered by a single vehicle. In addition to this limited cargo capacity, in the VRP with Time Windows (VRPTW), each shipment has a time window in which the shipment must be delivered. A selection of exact methods, first solution heuristics, local search neighborhoods and meta-heuristics for solving the CVRP, VRPTW and various other variations is summarized in [Kumar and Panneerselvam, 2012]. Generic solutions that can solve multiple variations of the VRP have also been developed [Røpke and Pisinger, 2005].

In this thesis we will concentrate on the VRPTW. The above definition of the VRPTW assumes fixed travel times. However, due to traffic congestion, travel times are rarely fixed in practice. This means that a feasible solution for the VRPTW can be infeasible in practice, as there is a chance that the traffic congestion causes travel times to be larger than the fixed times used for generating the solution and as such can lead to missing the time windows in the VRPTW.

One common approach to reduce the chance of practical infeasibility, is to overestimate the travel times. However, if the travel time is shorter than the estimated time, the vehicle might have to wait, or the schedule might be using more vehicles than necessary. Hence assuming fixed travel times can lead to infeasible or suboptimal schedules.

Various variations of the VRPTW have been defined and analyzed, including some that take uncertainty of travel times into account.

- the VRPTW with Time Dependent travel times and hard time windows (TD-VRPTW);
- the VRPTW with soft time windows (VRP-STW);
- the VRPTW with Stochastic travel times and Soft Time Windows (SVRP-STW);
- the VRPTW with Stochastic travel times and Hard Time Windows (SVRP-HTW).

Methods for obtaining real life stochastic and time dependent travel time information have been described in [Ando and Taniguchi, 2006] and [Van Woensel et al., 2008].

TD-VRPTW An exact branch, cut and price method for solving the TD-VRPTW is described in [Dabia et al., 2011]. A heuristic for solving the TD-VRP, based on tabu search, is given in [Kuo et al., 2009]. They note that most papers on time dependent travel times fail to include the non-passing rule, which states that one cannot arrive earlier by departing later, causing these papers to

have little practical value. Optimization of departure times as a post processing step is analyzed in [Kok et al., 2011]. They achieve duty time reductions of up to 8 percent, while taking work hour regulations into account.

VRP-STW The VRP-STW is a multi-objective optimization problem in which travel distance and service quality are optimized. An exact method for solving the VRP-STW is described in [Qureshi et al., 2010]. A method based on local search neighborhoods is described in [Ibaraki et al., 2005]. In [Tang et al., 2009] the soft time windows have been modeled using fuzzy numbers.

Definition 1.1 (Fuzzy numbers[1])**.** *Fuzzy numbers are fuzzy sets over the real numbers. A fuzzy set differs from an ordinary set in that its elements can have fractional membership. As such, fuzzy numbers do not have a single value, however it has a unique value for which the membership function is 1.*

SVRP-STW The SVRP-STW has been studied in [Taş et al., 2013], which solves the problem using tabu search and assumes gamma distributed travel times. In [Taş et al., 2012] the problem is solved using column generation and a branch and price algorithm. The travel times are assumed to be log-normal distributed.

Both distributions have infinite support, hence arbitrarily large travel times can occur with non-zero probability. Therefore time window violations are inevitable. Both papers handle these violations by using *soft* time windows and incurring a penalty for arriving too early or too late.

Furthermore, the gamma distribution has the property that the sum of two gamma distributed random variables, with the same scale parameter, is again a gamma distributed random variable. The sum of two log-normal random variables can be approximated by another log-normal random variable. This allows both papers to keep their travel time computations simple. These computations will be explained in Section 4.1.

A similar problem, with uncertain (instead of stochastic) travel times, is solved in [Brito et al., 2008]. The uncertain travel times are modeled using fuzzy numbers (see Definition 1.1). The authors use approximations for calculating with fuzzy numbers instead of simulation (using the Monte-Carlo method) and thereby manage to reduce computation time by a factor 600, while also obtaining a better result. Fuzzy numbers are similar to random variables in that they can both be described as a mapping from the real numbers to the interval $[0, 1]$. In this thesis a method is presented for performing approximate calculations with arbitrary, real valued, random variables.

It is argued in [Lecluyse et al., 2009] that travel times and arrival times are approximately log-normal distributed. However [van der Hooft, 2013] concludes that velocities are gamma distributed and hence travel times are inverse gamma distributed. In this thesis we describe a method for computing with random variables regardless of distribution. Therefore we support log-normal, gamma, inverse-gamma and even empirically determined travel time distributions.

SVRP-HTW Little work has been done on the SVRP-HTW. In [Chang et al., 2009] the traveling salesman problem with stochastic travel times and hard time windows is solved. However they used normal distributed random variables for approximating arrival times, which we believe is a too crude approximation. They handle time window violations by allowing a small chance that a shipment's time window is violated. Each shipment has an upper bound on this chance. In this thesis we handle time window violations in the same way. The description of the SVRP-HTW is given in Chapter 2.

There is already quite some research performed on the SVRP-STW, which has soft time windows. However, we have real-life problems with *hard* time windows and therefore our research will focus on the SVRP-HTW.

Example 1.2 (Hard time windows in real-life)**.** *Consider a vehicle that has to deliver goods to a shop. Regulations could restrict usage of the road that is required to access the shop. If that road*

[1] We will not use fuzzy numbers in this thesis.

is only accessible for vehicles until 10.00 o'clock, but our vehicle arrives too late, the goods cannot be delivered. Furthermore, if the vehicle arrives before the shop is unlocked, it will have to wait before it can deliver the goods.

In this thesis we will model travel times using random variables. This implies that the arrival times will also be random variables. The delivery time will then be the maximum of the arrival time and the start of the time window. So, the delivery time will be a mixed random variable, which is partly discrete and partly continuous (see section 3.1). The arrival time at the next location is then the sum of the delivery time, service time, and travel time. Hence to be able to compute arrival times, the sum of a mixed random variable and a continuous random variable must be computed.

Various approaches for computing with random variables have been used in the past. These approaches are described in Chapter 4. However most of these methods do not support the minimum and maximum operations that generally occur in optimization problems such as the SVRP-HTW that we want to solve. The only method that does support these operations, the Monte-Carlo method, is too slow.

To support the random variable computations, necessary for solving the SVRP-HTW, we have developed our own probabilistic arithmetic method (described in Chapter 5). This method allows us to compute with arbitrary random variables and perform operations like addition and computing the maximum of two random variables. For understanding our probabilistic arithmetic, some basic mathematical knowledge is required. A recap of some of the mathematical theory used throughout this thesis is given in Chapter 3.

In Chapter 6 we use our probabilistic arithmetic to convert an existing VRPTW solver into a solver for the SVRP-HTW instances. As there are no known benchmarks for the SVRP-HTW, we have developed our own benchmark, based on the Solomon benchmark for the VRPTW [Solomon, 1987]. The construction of our benchmark is described in Chapter 7. The computational results of our solver are presented in Chapter 8.

Chapter 2

The Vehicle Routing Problem with Stochastic Travel Times and Hard Time Windows

In this chapter we describe the vehicle routing problem with stochastic travel times and hard time windows (SVRP-HTW). The primary objective is minimizing the number of vehicles used. The secondary objective is to minimize the total traveled distance.

An SVRP-HTW instance consists of one depot and N shipments. Each shipment has a demand, a time window, a service duration and a minimum required reliability. For the vehicles a capacity is given. The depot does not have a time window. The shipments must be delivered within their time window with a probability of at least the required reliability.

The reliability leaves a small probability that a truck arrives at a customer past the time window. If this happens, the shipment is too late, but will still be delivered, hence the service time will be included. If a vehicle arrives too early, it will wait.

Input variables An instance of the SVRP-HTW consists of a set of shipments S. Each shipment $i \in S$ demands $\textsc{Demand}[i]$ goods, its time window is from $\textsc{Ready Time}[i]$ (ready time) to $\textsc{Due Date}[i]$ (due date) and takes $\textsc{Service Time}[i]$ time to unload. The probability that shipment i is delivered within its time window must be at least $\textsc{Reliability}[i]$.

The depot, denoted by *depot*, and the shipments have a location. The distance between two locations a and b is denoted as $\textsc{Distance}[a, b]$. The travel time from a to b is denoted as $\textbf{Travel Time}[a, b]$, whose result is a random variable. The maximum amount of cargo a truck can carry is $\textsc{Capacity}$ goods.

Solution A solution to the SVRP-HTW consists of a set of routes R. Each route is a list of locations. Let $r \in R$ be a route. We use $|r|$ to denote the number of edges (a transition from one location to another) in that route. Furthermore, let $k \in \{0, \ldots, |r|\}$, then r_k is the k-th location of the route. A route starts and ends at the depot, hence $r_0 = r_{|r|} = \textit{depot}$. For $k \in \{1, \ldots, |r| - 1\}$, we will use the shorthand notation $k \in r$. Let $k \in r$, then r_k denotes a shipment.

Convenience functions For each route R, we recursively define the functions **Serve Time** and **Departure Time** on R. The random variable $\textbf{Serve Time}[r_k]$ is the time at which service starts at r_k and is defined by (2.3). The random variable $\textbf{Departure Time}[r_k]$ is the time at which the truck departs from location r_k, which is given by (2.2) for shipments and (2.1) for the depot.

$$\textbf{Departure Time}[0] = 0 \tag{2.1}$$

$$\textbf{Departure Time}[r_k] = \textbf{Serve Time}[r_k] + \text{Service Time}[r_k] \tag{2.2}$$

$$\textbf{Serve Time}[r_k] = \max \begin{pmatrix} \textbf{Departure Time}[r_{k-1}] + \\ \quad \textbf{Travel Time}[r_{k-1}, r_k], \\ \text{Ready Time}[r_k] \end{pmatrix} \tag{2.3}$$

Constraints The problem must be solved with respect to the following constraints.

Each vehicle can carry at most Capacity goods (2.4) and, as mentioned in the solution description, routes must start and end at the depot (2.5).

Furthermore, every shipment must be scheduled once in exactly one route (2.6). Finally, the probability that a shipment is delivered in time must be at least the required reliability (2.7). Remember that vehicles wait if they arrive too early.

$$\forall r \in R : \left(\sum_{k \in r} \text{Demand}[r_k] \right) \leq \text{Capacity} \tag{2.4}$$

$$\forall r \in R : r_0 = \mathit{depot} \wedge r_{|r|} = \mathit{depot} \tag{2.5}$$

$$\forall s \in S : \exists! r \in R, k \in r : r_k = s \tag{2.6}$$

$$\forall r \in R, k \in r : \mathbb{P}\big(\textbf{Serve Time}[r_k] < \text{Due Date}[r_k]\big) \geq \text{Reliability}[r_k] \tag{2.7}$$

If there is no schedule that satisfies all these constraints, the instance is said to be infeasible.

Objective The first objective is to minimize $|R|$, the number of routes. The second objective is to minimize (2.8), the total amount of distance traveled.

$$\sum_{r \in R} \sum_{k \in \{1, \ldots, |r|\}} \text{Distance}[r_{k-1}, r_k] \tag{2.8}$$

Part I

Computing with random variables

Chapter 3

Theory and Definitions

This chapter gives various definitions and some theory on random variables, convolution and polynomial fitting. These are referred to and used throughout this thesis.

3.1 Cumulative Distribution Functions and Random Variables

A random variable is a variable that does not have a fixed value. Instead a random variable has a probability that its value lies in a given range. Usually only two types of random variables are considered, discrete and continuous. Discrete random variables can assume a finite number of values, each with a certain probability. Continuous random variables can assume an infinite number of values, each non-trivial interval can contain the assumed value with a certain probability.

There are various functions that can describe which value a random variable can take. For discrete random variables, there is the probability distribution function. For continuous variables, there is a probability density function. However, the Cumulative Distribution Function (CDF) can describe both discrete and continuous random variables. A CDF is a function from the real numbers to the interval $[0, 1]$ and describes the probability that the random variable is at most the given value.

In this thesis, random variables and functions returning random variables are shown in bold. This distinguishes them from deterministic variables and functions.

Definition 3.1.1 (Cumulative Distribution Function)**.** *Let $\boldsymbol{a}$ be a random variable, then the CDF of $\boldsymbol{a}$, denoted $F_{\boldsymbol{a}}$, is the function $F_{\boldsymbol{a}} : \mathbb{R} \to [0, 1]$ given by:*

$$F_{\boldsymbol{a}}(x) = \mathbb{P}(\boldsymbol{a} \leq x).$$

Example 3.1.2 (Uniform distribution)**.** *Let random variable $\boldsymbol{a} \sim \text{Unif}(0, 1)$ be uniformly distributed between 0 and 1. Then its CDF $F_{\boldsymbol{a}}$ (shown in Figure 3.1) is given by:*

$$F_{\boldsymbol{a}}(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases}.$$

If $\boldsymbol{b}$ is a random variable that is neither discrete nor continuous, then $\boldsymbol{b}$ is called a *mixed random variable*. Such a mixed random variable can accurately be described by its CDF, which is why we use the CDF to describe random variables.

Example 3.1.3 (Mixed random variable)**.** *If $\boldsymbol{a} \sim \text{Unif}(0, 1)$ models the arrival time and 0.5 is the start of the time window, then random variable $\boldsymbol{b} = \max(\boldsymbol{a}, 0.5)$ models the time at which the customer will be served. The CDF of $\boldsymbol{b}$ is shown in Figure 3.2. This figure shows that the CDF can be discontinuous.*

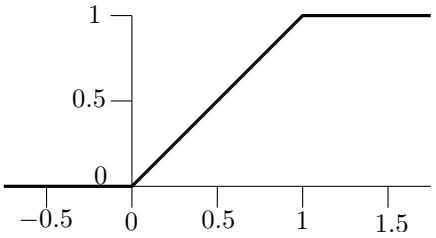

Figure 3.1: The Cumulative Distribution Function (CDF) of $\boldsymbol{a} \sim \text{Unif}(0, 1)$.

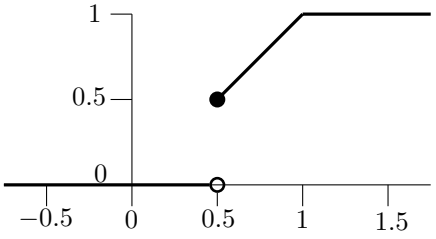

Figure 3.2: The CDF of $\boldsymbol{b} = \max(\boldsymbol{a}, 0.5)$, with $\boldsymbol{a} \sim \text{Unif}(0, 1)$.

There are a few properties that follow from the definition of a CDF. Let function $F_{\boldsymbol{a}}$ be a CDF, then $F_{\boldsymbol{a}}$ is non-decreasing (3.1), $F_{\boldsymbol{a}}$ ranges from 0 to 1 (3.2) and $F_{\boldsymbol{a}}$ is right continuous (3.3).

$$\forall x, y \in \mathbb{R} \wedge x \leq y : F_{\boldsymbol{a}}(x) \leq F_{\boldsymbol{a}}(y) \tag{3.1}$$

$$\lim_{x \to -\infty} F_{\boldsymbol{a}}(x) = 0 \wedge \lim_{x \to \infty} F_{\boldsymbol{a}}(x) = 1 \tag{3.2}$$

$$\forall x \in \mathbb{R} : F_{\boldsymbol{a}}(x) = \lim_{y \downarrow x} F_{\boldsymbol{a}}(y) \tag{3.3}$$

3.2 Convolution

Section 4.3 explains how the sum of two random variables can be computed using the convolution. A method for doing such computations is devised in Sections 5.2 and 5.3. These sections make extensive use of the theory of convolutions, which is described below.

Let F and G be two functions. If we want a function K, whose value $K(x)$ is somehow the sum of $F(a)G(b)$, for all $a + b = x$, then we use the convolution. We could write this as $K(x) = \int_{a+b=x} F(a)G(b)$, but the following equivalent definition is more common.

Definition 3.2.1 (Convolution)**.** *Convolution is a binary operator on* functions[1]*. Given two functions F and G, their convolution, denoted by $F * G$, is given by:*

$$(F * G)(x) = \int_{-\infty}^{\infty} F(z)G(x - z)\ dz.$$

From this definition it follows that the convolution operation is commutative (3.4), associative (3.5), scalar associative (3.6) and distributive (3.7).

$$F * G = G * F \tag{3.4}$$

$$F * (G * K) = (F * G) * K \tag{3.5}$$

$$s(F * G) = (sF) * G = F * (sG) \tag{3.6}$$

$$F * (G + K) = F * G + F * K \tag{3.7}$$

[1]There exists a different definition for the convolution of two cumulative distribution functions, whose result, $\int_{-\infty}^{\infty} F_{\boldsymbol{b}}(x - z)\ dF_{\boldsymbol{a}}(z)$, is again a CDF, but is the derivative of the definition that we use. Therefore we denote such a convolution as $(F_{\boldsymbol{a}} * F_{\boldsymbol{b}})'$.

Definition 3.2.2 (Translation)**.** *The translation $\mathcal{T}_s$ is a transformation function. Let $F : \mathbb{R} \to \mathbb{R}$ be a function, then $\mathcal{T}_s(F)$ is given by:*

$$(\mathcal{T}_s(F))(x) = F(x - s).$$

Definition 3.2.3 (Fourier transformation[2])**.** *The Fourier transformation $\mathcal{F}$ is a transformation function. Let $F : \mathbb{R} \to \mathbb{R}$ be a function, then $\mathcal{F}(F)$ is given by:*

$$(\mathcal{F}(F))(x) = \int_{-\infty}^{\infty} F(s)\ e^{-2\pi i x s}\, ds.$$

The convolution operation also has some properties with respect to differentiation (3.8), translation (3.9) and the Fourier transformation (3.10). Equation (3.10) is also known as the convolution theorem.

$$(F * G)' = F' * G = F * G' \tag{3.8}$$
$$\mathcal{T}_s(F * G) = \mathcal{T}_s(F) * G = F * \mathcal{T}_s(G) \tag{3.9}$$
$$F * G = \mathcal{F}^{-1}(\mathcal{F}(F) \cdot \mathcal{F}(G)) \tag{3.10}$$

3.3 Polynomial Fitting

In Section 5.1 we use polynomial fitting to approximate the CDF of a random variable.

Definition 3.3.1 (Polynomial)**.** *A polynomial $P : \mathbb{R} \to \mathbb{R}$ is a function of the form $P(x) = c_0x^0 + c_1x^1 + \ldots + c_dx^d$.*

We define $\deg(P) = d$ to be the degree of P and $\text{coef}_i(P) = c_i$ as the coefficient of the i-th term. Note that for $i > d$, we have that $\text{coef}_i(P) = 0$. The following equality combines these definitions:

$$P(x) = \sum_{i=0}^{\deg(P)} \text{coef}_i(P)x^i.$$

Fitting is the process of approximating a function f using another function of a given form. There are two fitting methods that we use: endpoint fitting and least-squares fitting. In both cases we use a third degree polynomial P for approximating f on a given interval $[x_a, x_b]$.

Endpoint fitting If we know the value and the derivative of f in the points x_a and x_b, we can approximate f using endpoint fitting. Let x_a, y_a, y'_a, x_b, y_b and y'_b be given, let P be a third-degree polynomial, such that $P(x_a) = y_a$, $P(x_b) = y_b$, $P'(x_a) = y'_a$ and $P'(x_b) = y'_b$, then P is unique and approximates f (the proof is left as an exercise for the reader). Expressions for the coefficients of P can be derived easily. For $x_a = 0$ and $x_b = 1$, the coefficients are:

$$\begin{aligned}
\text{coef}_0(P) &= y_a \\
\text{coef}_1(P) &= y'_a \\
\text{coef}_2(P) &= -3y_a - 2y'_a + 3y_b - y'_b \\
\text{coef}_3(P) &= 2y_a + y'_a - 2y_b + y'_b
\end{aligned}$$

[2]In this thesis, we only use the properties that follow from the given Fourier transformation. The definition itself is not used, but given for completeness and to show which variation of Fourier transformation we are using.

Least-squares fitting If we know the value $y_i = f(x_i)$, for k distinct points $x_1, \ldots, x_k$ in the interval $[x_a, x_b]$, we can use least-squares fitting. We want to obtain P, such that $\sum_{i=1}^{k}(P(x_i)-y_i)^2$ is minimized. The coefficients of P can be determined by computing, assuming the matrix is not singular:

$$\left(\sum_{i=1}^{k}\begin{pmatrix} x_i^0 & x_i^1 & x_i^2 & x_i^3 \\ x_i^1 & x_i^2 & x_i^3 & x_i^4 \\ x_i^2 & x_i^3 & x_i^4 & x_i^5 \\ x_i^3 & x_i^4 & x_i^5 & x_i^6 \end{pmatrix}\right)^{-1} \sum_{i=1}^{k} y_i \begin{pmatrix} x_i^0 \\ x_i^1 \\ x_i^2 \\ x_i^3 \end{pmatrix}.$$

We implemented this using Gaussian-elimination. This allows us to detect singularity, in which case we can return the intermediary result, which fits the points using a polynomial of lower degree.

The above fitting methods are numerically unstable if $\max(|x_a|, |x_b|)/|x_b - x_a|)$ is large. Therefore we assume $x_a = 0$ in this thesis.

3.4 Piecewise Polynomial Functions

Definition 3.4.1 (Piecewise function)**.** *Let the functions $F_1, \ldots, F_{n-1}, F_n : \mathbb{R} \to \mathbb{R}$ and the values $-\infty = x_0 < x_1 < \cdots < x_{n-1} < x_n = \infty$ be given. The function $F : \mathbb{R} \to \mathbb{R}$, given by:*

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

is a piecewise function. An example of such function is given in Figure 3.3.

The values $x_1, \ldots, x_{n-1}$ are called the breakpoints of F. Let $i \in \{1, \ldots, n\}$, then $[x_{i-1}, x_i)$ is called the i-th segment of F and F_i is the i-th segment function. Furthermore, we define the breakpoint differences $f_1, \ldots, f_{n-1} : \mathbb{R} \to \mathbb{R}$ to be given by $f_i = F_{i+1} - F_i$.

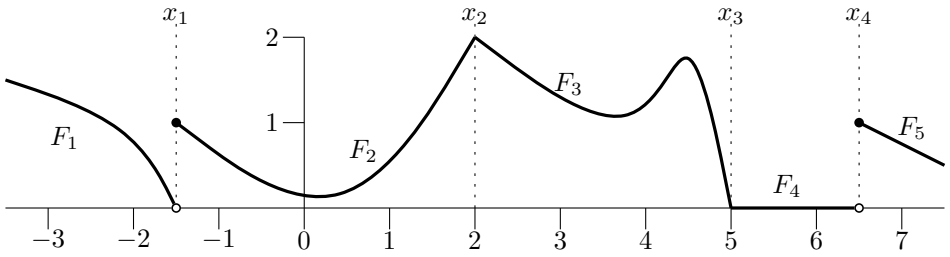

Figure 3.3: Example of a piecewise function whose segment functions are all continuous. The breakpoints are marked with dotted lines.

Definition 3.4.2 (Piecewise polynomial function)**.** *Let F be a piecewise function, whose segment functions are polynomials (see Definition 3.3.1), then F is a piecewise polynomial function.*

We define the degree of F as:

$$\deg(F) = \max\big(\deg(F_1), \ldots, \deg(F_n)\big).$$

Chapter 4

Approaches to Computing with Random Variables

There are various different approaches for computing with random variables. Those approaches are briefly discussed in this chapter.

4.1 Non-generic methods

If we did not require the maximum operation, we could stick to a specific class of distributions, that allows us to do exact or approximate computations, using an explicit formula. These methods can also be used when the actual distribution is approximated using such specific class of distributions. A list of distribution classes and how to do addition within them follows:

Normal distribution If $\boldsymbol{a} \sim \mathcal{N}(\mu_a, \sigma_a^2)$ and $\boldsymbol{b} \sim \mathcal{N}(\mu_b, \sigma_b^2)$ are independent random variables, then $\boldsymbol{a} + \boldsymbol{b} \sim \mathcal{N}(\mu_a + \mu_b, \sigma_a^2 + \sigma_b^2)$.

This is used as an approximation method for solving the Traveling Salesman problem with stochastic travel times and hard time windows in [Chang et al., 2009].

Gamma distribution Let scale parameter λ be fixed. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are independent random variables, with $\boldsymbol{a} \sim \text{Gamma}(\alpha_a, \lambda)$ and $\boldsymbol{b} \sim \text{Gamma}(\alpha_b, \lambda)$, then $\boldsymbol{a}+\boldsymbol{b} \sim \text{Gamma}(\alpha_a+\alpha_b, \lambda)$.

In [Taş et al., 2013] this is used to solve the vehicle routing problem with gamma distributed travel times and soft time windows.

Log-normal distribution If $\boldsymbol{a} \sim \log\text{-}\mathcal{N}(\mu_a, \sigma_a^2)$ and $\boldsymbol{b} \sim \log\text{-}\mathcal{N}(\mu_b, \sigma_b^2)$ are independent random variables, then $\boldsymbol{c} \sim \log\text{-}\mathcal{N}(\mu_c, \sigma_c^2)$, with

$$\sigma_c^2 = \log\left(\frac{e^{2\mu_a+\sigma_a^2}(e^{\sigma_a^2}-1) + e^{2\mu_b+\sigma_b^2}(e^{\sigma_b^2}-1)}{(e^{\mu_a+\frac{1}{2}\sigma_a^2} + e^{\mu_b+\frac{1}{2}\sigma_b^2})^2} + 1\right);$$

$$\mu_c = \log\left(e^{\mu_a+\frac{1}{2}\sigma_a^2} + e^{\mu_b+\frac{1}{2}\sigma_b^2}\right) - \frac{1}{2}\sigma_c^2,$$

is an approximation of $\boldsymbol{a} + \boldsymbol{b}$. [Dufresne, 2008]

This method is used in [Taş et al., 2012], for solving the vehicle routing problem with log-normal distributed travel times and soft time windows.

4.2 Monte-Carlo simulation

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two random variables and let $\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b}$ be their sum. We do not know the CDF of $\boldsymbol{c}$, but we can take samples from $\boldsymbol{c}$. A common used method to compute $\mathbb{P}(\boldsymbol{c} \leq x)$ for some given x is Monte-Carlo simulation.

Description 4.2.1 (Monte-Carlo simulation)**.** *Let $x \in \mathbb{R}$ and random variable $\boldsymbol{c}$ be given. Let $\boldsymbol{p}$ be the random variable that is 1 if $\boldsymbol{c} \leq x$ and 0 otherwise. Now if S is the sum of N samples of $\boldsymbol{p}$, then $\frac{S}{N}$ is an approximation of $\mathbb{P}(\boldsymbol{c} \leq x)$.*

Note that $\boldsymbol{p}$ is Bernoulli distributed[1] *with $p = \mathbb{P}(\boldsymbol{c} \leq x)$. This distribution has a standard deviation of $\sigma = \sqrt{p(1-p)}$. Let $\boldsymbol{S} = \sum_{i=1}^{N} \boldsymbol{p}_i$, with $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N$ independent and distributed as $\boldsymbol{p}$. The Central Limit Theorem states that, by approximation, $\frac{\boldsymbol{S}}{N} \sim \mathcal{N}(p, \frac{\sigma}{\sqrt{N}})$. Hence $\frac{S}{N}$, which is a sample of $\frac{\boldsymbol{S}}{N}$, has 0.95 chance that it differs at most $\frac{2\sigma}{\sqrt{N}}$ from p. For example, with $p = 0.5$ and $N = 10^6$, we have that $\sigma = 0.5$ and $\mathbb{P}(\frac{S}{N} \in [p-\varepsilon, p+\varepsilon]) \geq 0.95$, with $\varepsilon = 10^{-3}$. So in this case $\frac{S}{N}$ has an accuracy of 0.001.*

Monte-Carlo simulation can be generalized for computing the CDF of $\boldsymbol{c}$. However, the running time of Monte-Carlo simulation is $O(\frac{1}{\varepsilon^2})$ for an accuracy of ε. So this method is prohibitively slow, as we need to compute such CDF for every shipment in a route, each time that route changes.

Besides slow, Monte-Carlo simulation is also robust and a very simple method to implement. Therefore we use it in Chapter 7 for the creation of our problem instances and in Section 8.3 to verify the results we got using our probabilistic arithmetic implementation.

4.3 Convolution

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be independent random variables and let $\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b}$. We would like to express $F_{\boldsymbol{c}}$ in terms of $F_{\boldsymbol{a}}$ and $F_{\boldsymbol{b}}$.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are discrete, then $\mathbb{P}(\boldsymbol{a}+\boldsymbol{b} = x)$ can be computed using equation (4.1). If $\boldsymbol{a}$ and $\boldsymbol{b}$ are continuous, then the sum is replaced by an integral, the probabilities are replaced by probability densities and we get equation (4.2), which is a convolution. Some theory on convolutions is described in Section 3.2.

$$\mathbb{P}(\boldsymbol{a} + \boldsymbol{b} = x) = \sum_{a+b=x} \mathbb{P}(\boldsymbol{a} = a)\mathbb{P}(\boldsymbol{b} = b) \tag{4.1}$$

$$\begin{aligned} F'_{\boldsymbol{c}}(x) = \frac{d}{dx}\mathbb{P}(\boldsymbol{a} + \boldsymbol{b} \leq x) &= \int_{a+b=x} \frac{d}{da}\mathbb{P}(\boldsymbol{a} \leq a) \cdot \frac{d}{db}\mathbb{P}(\boldsymbol{b} \leq b) \\ &= \int_{a+b=x} F'_{\boldsymbol{a}}(a) \cdot F'_{\boldsymbol{b}}(b) \\ &= \int_{-\infty}^{\infty} F'_{\boldsymbol{a}}(y) \cdot F'_{\boldsymbol{b}}(x-y)\ dy \\ &= F'_{\boldsymbol{a}} * F'_{\boldsymbol{b}} \end{aligned} \tag{4.2}$$

However, in our case, $\boldsymbol{a}$ and $\boldsymbol{b}$ are mixed random variables, which means neither of the previous two equations can be used. The problem with the formula for addition of continuous random variables, is that it uses the derivatives of the CDFs of $\boldsymbol{a}$ and $\boldsymbol{b}$, which are not well-defined. We solve this by applying equation (3.8): $F_{\boldsymbol{c}} = \int F'_{\boldsymbol{c}} = \int (F'_{\boldsymbol{a}} * F'_{\boldsymbol{b}}) = \int (F_{\boldsymbol{a}} * F'_{\boldsymbol{b}})' = \int (F_{\boldsymbol{a}} * F_{\boldsymbol{b}})'' = (F_{\boldsymbol{a}} * F_{\boldsymbol{b}})'$. Hence we can express $F_{\boldsymbol{c}}$ in terms of $F_{\boldsymbol{a}}$ and $F_{\boldsymbol{b}}$ using $F_{\boldsymbol{c}} = (F_{\boldsymbol{a}} * F_{\boldsymbol{b}})'$, which is the derivative of the convolution[2] of the CDFs of $\boldsymbol{a}$ and $\boldsymbol{b}$. This equivalence allows us to rewrite the addition of random variables as a convolution.

In [Jones, 1977] a convolution based method for the addition of random variables is described. It allows the convolution of two probability distribution functions to be computed in $O(n^3)$ time and it is shown that this method is significantly faster and more accurate than the Monte Carlo method. The method requires that both probability distribution functions are approximated by piecewise constant functions. Each such function must have $O(n)$ segments of equal length. We do not use this method, because the equal length constraint is too restricting.

[1] The Bernoulli distribution is 1 with chance p and 0 otherwise.

[2] Convolution as defined in Definition 3.2.1.

Another method is described in [Plehn and Bruns, 2005]. This paper describes a method for approximating a probability distribution function with a non-continuous piecewise linear function, whose segments can have differing sizes. Its running time is 12 seconds for the convolution of two piecewise polynomials of degree two with 100 segments. A running time of 2 seconds is too slow for our purposes, as each addition of travel time requires the computation of such a convolution. Also, the method works on continuous random variables instead of the mixed random variables that we encounter.

4.3.1 Transformation Based Methods

A common method for computing convolutions uses the Fast Fourier Transform. This method is based on the convolution theorem (3.10). The Fast Fourier Transform is a fast algorithm for computing this transformation using a finite set of equally spaced samples. To be able to use this method for computing the convolution of two functions, these two functions must be approximated using a finite set of equally spaced samples and furthermore both these approximations must use the same sample spacing. In [Agarwal and Burrus, 1975] an algorithm is described that can compute the convolution of two such approximations with n samples in $O(n \log n)$ time, by using the Fast Fourier Transformation.

The requirement that each approximation uses the same sample spacing, means that we have to chose this spacing up front. For distributions with a small deviation, this results in a too coarse approximation, which will lead to computational errors. For distributions with a large deviation, it results in too many samples, causing the computations to become unnecessary slow. In [Hackbusch, 2008] an algorithm is described that alleviates these problems, by allowing a course sampling grid that is locally refined by repeatedly adding additional sample points in the middle of two consecutive and already existing sample points. The algorithm still runs in $O(n \log n)$ time. We believe that this is a promising method for computing with random variables, however we decided to take a different approach.

Another drawback of the Fourier Transform method is that it assumes that the function being transformed is continuous. This is mostly because the approximation cannot represent a discontinuity. Therefore we do not consider Fourier transform based methods a good candidate for working with mixed random variables.

Methods based on the Fourier Transform are also known as methods based on the characteristic function, because the characteristic function of a distribution is the complex conjugate of the fourier transform of the probability density function.

4.4 Other probabilistic arithmetic methods

In [Williamson and Downs, 1990] various methods for probabilistic arithmetic are discussed. Some of these are based on transformations different than Fourier Transformation. The paper focuses on computing error bounds for arithmetic with dependent random variables. Unfortunately, little attention is given to the speed and numerical accuracy of the methods. Also, the maximum and minimum operations are not mentioned and the described methods do not seem suitable for performing these operations. Furthermore, the paper assumes that the probability density function exists and therefore it is unclear which of the discussed methods are usable for computation with mixed random variables.

Chapter 5

The Developed Computation Method

As mentioned in the introduction, a generic method for computing with mixed random variables is needed. In this chapter we present our probabilistic arithmetic, a numerical method for computing with random variables. It uses the theory on random variables and convolution described in Chapter 3.

5.1 Approximating Distribution Functions

We must be able to work with mixed random variables, which, assuming the random variables are independent, can be fully expressed using the Cumulative Distribution Function. To be able to do computations numerically, we need to compute an approximation of the CDF. Furthermore, because the methods described in Sections 5.2, 5.3 and 5.4 increase the space complexity of the approximation, we need an approximation method to reduce this complexity.

We have chosen to approximate the random variables using piecewise polynomial functions with degree 3 and variable segment size. We have chosen for degree 3 so that the Probability Density Function (PDF), if it exists, can be smooth at the breakpoints. A degree 2 approximation of the CDF would yield a piecewise linear PDF. We have chosen to not use a higher degree, because we have to be able to compute the inverse of the polynomial and because higher degree polynomials are more susceptible to numerical errors.

As we will use a piecewise polynomial approximation of a random variable, the approximation will have finite support (5.1). As a result, the approximation will lack some of the properties of the original random variable. For example, the approximations of a heavy-tailed (5.2) random variable, such as a log-normal distributed random variable, will not be heavy-tailed.

$$\exists l, u \in \mathbb{R} : F(l) = 0 \wedge F(u) = 1 \tag{5.1}$$

$$\forall \lambda > 0 : \lim_{x \to \infty} e^{\lambda x} F(-x) = \infty \vee \lim_{x \to \infty} e^{\lambda x}(1 - F(x)) = \infty \tag{5.2}$$

We have developed two approximation methods. One for approximating an existing CDF and one for simplifying a CDF that is the result of a computation.

5.1.1 Approximation of an existing CDF

Given a CDF $F_{\boldsymbol{a}}$ we can compute a piecewise polynomial approximation using the procedure described below.

Let integers $N, M > 0$, random variable a and $\eta \in (0, \frac{1}{2})$ be given. We compute the quantiles q_1 and q_N of $F_{\boldsymbol{a}}$ at η and $1-\eta$. Hence $F_{\boldsymbol{a}}(q_1) = \eta$ and $F_{\boldsymbol{a}}(q_2) = 1-\eta$. We then create the arithmetic sequence $q_1, q_2, \ldots, q_N$. Subsequently the sequence $p_1, \ldots, p_N$ is computed with $p_i = F_{\boldsymbol{a}}(q_i)$.

Additionally we compute an error sequence $err_1, \ldots, err_N$, for which err_i is an approximation of $\int_{-\infty}^{q_i} \sqrt[4]{\frac{d^4}{dt^4} F_{\boldsymbol{a}}(t)}\, dt$. This error formula is based on the error formula in [Plehn and Bruns, 2005] for polynomials of degree 2. The value of $err_j - err_i$ is an estimate of the error made when fitting $[q_i, q_j)$ with a single third degree polynomial.

A partition P of $[q_1, \ldots, q_N)$ with size M is created, such that $q_1, \ldots q_N$ is a refinement of P and for each interval $[q_i, \ldots, q_j) \in P$ the value of $err_j - err_i$ is close to $\frac{1}{M}(err_N - err_1)$. Now a piecewise polynomial function is created by using least-squares fitting (see Section 3.3) on $p_i, \ldots, p_j$ for each interval $[q_i, \ldots, q_j) \in P$. The boundaries of the intervals are used as the breakpoints.

5.1.2 Approximation of a computed CDF

Given a piecewise polynomial function F of arbitrary degree, we can create an approximation of lower degree and with fewer breakpoints using the procedure described below.

Let the integer $N > 0$ be given. The non-decreasing sequence $q_1, \ldots, q_N$ is created by concatenating partitions of the segments of F, such that $\max_i(q_{i+1} - q_i)$ is minimized. Then we create the sequences $p_1, \ldots, p_N$ and $p'_1, \ldots, p'_N$, such that $p_i = F(q_i)$ and $p'_i = F'(q_i)$, unless $q_i = q_{i+1}$, in which case $p_i = \lim_{q \uparrow q_i} F(q)$ and $p'_i = \lim_{q \uparrow q_i} F'(q)$.

The approximation is now constructed recursively, starting with the interval $I = [q_1, q_N)$. Interval $I = [q_i, q_j)$ is fitted using endpoint fitting (see Section 3.3). Let $q_e \in I$ be the point that maximizes the absolute error of the fit. If the error is smaller than a predefined threshold, the fit is used for segment $[q_i, q_j)$. If the error is larger, then q_e is added as breakpoint and interval I is fitted recursively by fitting the intervals $[q_i, q_e)$ and $[q_e, q_j)$.

5.2 Addition of Discrete Random Variables

In this section we explain a method for computing the CDF of the addition of two discrete random variables. The method has a running time of $O(n^2)$, with n being the number of possible values that the random variables can take. In the next section this method is generalized to support mixed random variables.

Let $\boldsymbol{o}$ be the random variable that is always zero. The CDF of $\boldsymbol{o}$ is then equal to the unit step function H.

Definition 5.2.1 (Unit step)**.** *The unit step function $H : \mathbb{R} \to \{0, 1\}$ is given by:*

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0. \end{cases}$$

The convolution of the unit step with itself can easily be computed as is shown in (5.3) and shown graphically in Figure 5.1.

$$\begin{aligned} (H * H)(x) &= \int_{-\infty}^{\infty} H(z)H(x-z)\, dz \\ &= H(x) \int_{0}^{\infty} H(x-z)\, dz \\ &= H(x) \int_{0}^{x} 1\, dz \\ &= xH(x). \end{aligned} \tag{5.3}$$

If we want the CDF of $\boldsymbol{o} + \boldsymbol{o}$, then we still need to compute the derivative of $H * H$, which is $(H * H)'(x) = \frac{d}{dx}\, xH(x) = H(x)$. Hence $\boldsymbol{o} + \boldsymbol{o} = \boldsymbol{o}$.

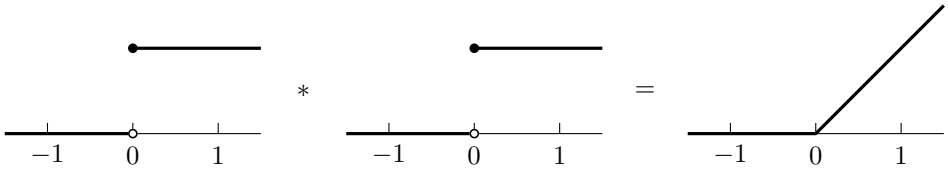

Figure 5.1: Convolution of two unit step functions $H(x)$ results in the function $xH(x)$.

Let x be a constant, the random variable that is always x is $\boldsymbol{o} + x$. The CDF of this random variable is $\mathcal{T}_x(H)$.

If $\boldsymbol{a}$ is a random variable, then we can write its CDF as:

$$F_{\boldsymbol{a}} = \sum_x \mathbb{P}(\boldsymbol{a} = x)\mathcal{T}_x(H), \tag{5.4}$$

which is a piecewise constant function. If we let $F = F_{\boldsymbol{a}}$, let $x_1, \ldots, x_n$ be the n breakpoints of F and let $f_1, \ldots, f_n$ be the breakpoint differences of F (see Definition 3.4.1), then we can rewrite the CDF as a linear combination of translated unit steps:

$$F = \sum_{i=1}^{n} \text{coef}_0(f_i) \cdot \mathcal{T}_{x_i}(H) \tag{5.5}$$

Note that all f_i are constant functions and that $\text{coef}_0(f_i) = \mathbb{P}(\boldsymbol{a} = x_i)$.

Theorem 5.2.2. *Let F and G be piecewise* constant *functions, let $x_1, \ldots, x_n$ be the n breakpoints of F, let $y_1, \ldots, y_m$ be the m breakpoints of G and let $f_1, \ldots, f_n$ and $g_1, \ldots, g_n$ be their breakpoint differences. Now we can write:*

$$(F * G)' = \sum_{i=1}^{n}\sum_{j=1}^{m} \text{coef}_0(f_i) \cdot \text{coef}_0(g_j) \cdot \mathcal{T}_{x_i+y_j}(H) \tag{5.6}$$

Proof. As shown in (5.5), we can write F as a linear combination of translated unit steps and we can do this with G as well. Using this formulation we can rewrite the convolution of F and G as (5.7).

$$\begin{aligned}
F * G &= \left(\sum_{i=1}^{n} \text{coef}_0(f_i) \cdot \mathcal{T}_{x_i}(H)\right) * \left(\sum_{j=1}^{m} \text{coef}_0(g_j) \cdot \mathcal{T}_{y_j}(H)\right) \\
&\overset{(3.7)}{=} \sum_{i=1}^{n}\sum_{j=1}^{m} \left(\text{coef}_0(f_i) \cdot \mathcal{T}_{x_i}(H)\right) * \left(\text{coef}_0(g_j) \cdot \mathcal{T}_{y_j}(H)\right) \\
&\overset{(3.6)}{=} \sum_{i=1}^{n}\sum_{j=1}^{m} \text{coef}_0(f_i) \cdot \text{coef}_0(g_j) \cdot \left(\mathcal{T}_{x_i}(H) * \mathcal{T}_{y_j}(H)\right) \\
&\overset{(3.9)}{=} \sum_{i=1}^{n}\sum_{j=1}^{m} \text{coef}_0(f_i) \cdot \text{coef}_0(g_j) \cdot \mathcal{T}_{x_i}\left(\mathcal{T}_{y_j}(H * H)\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \text{coef}_0(f_i) \cdot \text{coef}_0(g_j) \cdot \mathcal{T}_{x_i+y_j}(H * H)
\end{aligned} \tag{5.7}$$

Now we can differentiate (5.7) and obtain (5.6), which is a linear combination of translated unit steps and thus is a piecewise constant function. □

As a result of this theorem, we can now compute the CDF of the sum of two discrete and independent random variables. Note that the resulting function generally consists of $n \cdot m$ segments. This number can be reduced by computing an approximation as described in Section 5.1.2.

5.3 Addition of Mixed Random Variables

In this section we present the generalization that we have developed of the method described in the previous section. This generalization can compute the convolution of two piecewise polynomial functions in $O(n^2d^2\log(nd))$ time, where n is the number of segments and d is the degree of the piecewise polynomial functions. This running time will be derived at the end of this section.

In the previous section we decomposed our piecewise function into a linear combination of translated unit steps. For the piecewise polynomial functions we generalize the concept of the unit step function:

Definition 5.3.1 (Generalized unit step)**.** *The generalized unit step $H^a : \mathbb{R} \to \mathbb{R}$ is given by*

$$H^a(x) = x^a H(x).$$

Note that $H = H^0$. Figure 5.2 shows graphs of the generalized unit step H^a for various a. A formula for the convolution of two such generalized unit steps will be given by Lemma 5.3.3.

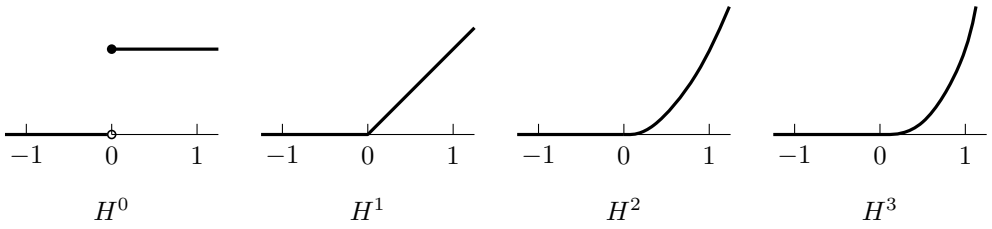

Figure 5.2: The generalized unit step H^a for $a \in \{0, 1, 2, 3\}$.

Lemma 5.3.2. *For non negative numbers a and b, the following equation holds:*

$$\int_0^x z^a \cdot (x-z)^b \, dz = \frac{a! \cdot b!}{(a+b+1)!} x^{a+b+1}. \tag{5.8}$$

Proof. We prove this lemma using induction on b. For our induction base $b = 0$ we have (5.9).

$$\int_0^x z^a \cdot (x-z)^0 \, dz = \int_0^x z^a \cdot 1 \, dz = \frac{1}{a+1} x^{a+1} = \frac{a! \cdot 0!}{(a+0+1)!} x^{a+0+1} \tag{5.9}$$

As induction step $b > 0$ we have (5.10), in which we use integration by parts (i.p.). We use $[f(z)]_a^b$ as a shorthand notation for $f(b) - f(a)$.

$$\begin{aligned}
& \int_0^x z^a \cdot (x-z)^b \, dz \\
\overset{i.p.}{=} & \left[\frac{1}{a+1} z^{a+1}(x-z)^b\right]_0^x - \int_0^x \frac{1}{a+1} z^{a+1} \cdot -b(x-z)^{b-1} \, dz \\
= & \ (0-0) - \frac{-b}{a+1} \int_0^x z^{a+1} \cdot (x-z)^{b-1} \, dz \\
\overset{(5.8)}{=} & \frac{b}{a+1} \cdot \frac{(a+1)! \cdot (b-1)!}{(a+b+1)!} x^{a+b+1} \\
= & \ \frac{a! \cdot b!}{(a+b+1)!} x^{a+b+1}
\end{aligned} \tag{5.10}$$

This finishes the induction and proves the lemma. □

Lemma 5.3.3. *If $q = H^a$ and $r = H^b$, with $a, b \in \mathbb{Z} \geq 0$, then equation (5.11) holds.*

$$q * r = \frac{a! \cdot b!}{(a+b+1)!} H^{a+b+1} \tag{5.11}$$

Proof. We now use Lemma 5.3.2 to rewrite the convolution of q and r into (5.12).

$$\begin{aligned}(q * r)(x) &= \int_{-\infty}^{\infty} z^a H(z) \cdot (x-z)^b H(x-z)\ dz \\ &= \int_0^x z^a \cdot (x-z)^b\ dz \cdot H(x) \\ &\overset{(5.8)}{=} \frac{a! \cdot b!}{(a+b+1)!} x^{a+b+1} H(x) \end{aligned} \tag{5.12}$$

This equation shows that (5.11) holds. □

Now we will use Lemma 5.3.3 to prove a generalization of Theorem 5.2.2. We can generalize equation (5.5) to (5.13), such that it also holds for F with $\deg(F) > 1$. With the generalized unit step, this can then be decomposed per term of each polynomial, giving (5.14).

$$F(x) = \sum_{i=1}^{n} f_i \cdot \mathcal{T}_{x_i}(H)(x) \tag{5.13}$$

$$= \sum_{i=1}^{n} \sum_{a=0}^{\deg(F)} \operatorname{coef}_a(f_i) x^a \cdot \mathcal{T}_{x_i}(H)(x)$$

$$= \sum_{i=1}^{n} \sum_{a=0}^{\deg(F)} \operatorname{coef}_a(\mathcal{T}_{-x_i}(f_i)) \cdot \mathcal{T}_{x_i}(H^a)(x) \tag{5.14}$$

Theorem 5.3.4. *Let F and G be piecewise* polynomial *functions, let $x_1, \ldots, x_n$ be the n breakpoints of F, let $y_1, \ldots, y_m$ be the m breakpoints of G and let $f_1, \ldots, f_n$ and $g_1, \ldots, g_n$ be their breakpoint differences. Now we can write:*

$$(F * G)' = \sum_{a=0}^{\deg(F)} \sum_{b=0}^{\deg(G)} \frac{a! \cdot b!}{(a+b)!} \sum_{i=1}^{n} \sum_{j=1}^{m} \operatorname{coef}_a(\mathcal{T}_{-x_i}(f_i)) \cdot \operatorname{coef}_b(\mathcal{T}_{-y_j}(g_j)) \cdot \mathcal{T}_{x_i+y_j}(H^{a+b}) \tag{5.15}$$

Proof. Using the decomposition (5.14) and applying a similar derivation as (5.7), we obtain (5.16) for the convolution of piecewise polynomial functions F and G.

$$F * G = \sum_{a=0}^{\deg(F)} \sum_{b=0}^{\deg(G)} \sum_{i=1}^{n} \sum_{j=1}^{m} \operatorname{coef}_a(\mathcal{T}_{-x_i}(f_i)) \cdot \operatorname{coef}_b(\mathcal{T}_{-y_j}(g_j)) \cdot \mathcal{T}_{x_i+y_j}(H^a * H^b) \tag{5.16}$$

With Lemma 5.3.3 we can eliminate the convolution in (5.16). This yields the function (5.17), which is a linear combination of translated generalized unit steps and therefore is piecewise polynomial.

$$\begin{aligned} F * G &= \sum_{i=1}^{n} \sum_{a=0}^{\deg(F)} \operatorname{coef}_a(\mathcal{T}_{-x_i}(f_i)) \mathcal{T}_{x_i}(H^a) * \sum_{j=1}^{m} \sum_{b=0}^{\deg(G)} \operatorname{coef}_b(\mathcal{T}_{-y_j}(g_j)) \mathcal{T}_{y_j}(H^b) \\ &= \sum_{i=1}^{n} \sum_{a=0}^{\deg(F)} \sum_{j=1}^{m} \sum_{b=0}^{\deg(G)} \operatorname{coef}_a(\mathcal{T}_{-x_i}(f_i)) \cdot \operatorname{coef}_b(\mathcal{T}_{-y_j}(g_j)) \cdot \mathcal{T}_{x_i+y_j}(H^a * H^b) \\ &= \sum_{a=0}^{\deg(F)} \sum_{b=0}^{\deg(G)} \frac{a! \cdot b!}{(a+b+1)!} \sum_{i=1}^{n} \sum_{j=1}^{m} \operatorname{coef}_a(\mathcal{T}_{-x_i}(f_i)) \cdot \operatorname{coef}_b(\mathcal{T}_{-y_j}(g_j)) \cdot \mathcal{T}_{x_i+y_j}(H^{a+b+1}) \end{aligned} \tag{5.17}$$

By differentiating (5.17) we obtain the piecewise polynomial function (5.15). □

As (5.15) is a sum of n^2d^2 terms and each term can be computed in constant time, we can compute $(F * G)'$ in $O(n^2d^2)$ time. However, to convert (5.15) into piecewise polynomial form, the terms must be sorted by their translation. This adds a $4\log(nd)$ factor to the running time, yielding an algorithm for the convolution of two piecewise polynomial functions in $O(n^2d^2\log(nd))$ time.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are independent random variables with a piecewise polynomial CDF, then we can use Theorem 5.3.4 to compute the CDF of $\boldsymbol{a} + \boldsymbol{b}$.

5.4 Maximum and Minimum

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two independent random variables and let $\boldsymbol{c} = \max(\boldsymbol{a}, \boldsymbol{b})$. We can compute the CDF of $\boldsymbol{c}$ as follows:

$$\begin{aligned} F_{\boldsymbol{c}}(x) &= \mathbb{P}(c \leq x) \\ &= \mathbb{P}(\max(\boldsymbol{a}, \boldsymbol{b}) \leq x) \\ &= \mathbb{P}(\boldsymbol{a} \leq x \wedge \boldsymbol{b} \leq x) \\ &= \mathbb{P}(\boldsymbol{a} \leq x) \cdot \mathbb{P}(\boldsymbol{b} \leq x) \\ &= F_{\boldsymbol{a}}(x) \cdot F_{\boldsymbol{b}}(x) \end{aligned}$$

Hence the CDF of $\boldsymbol{c}$ is the pointwise multiplication of the CDFs of $\boldsymbol{a}$ and $\boldsymbol{b}$. For computing the CDF of the minimum of two random variables, say $\boldsymbol{d} = \min(\boldsymbol{a}, \boldsymbol{b})$, we can use a similar derivation and obtain:

$$F_{\boldsymbol{d}} = 1 - (1 - F_{\boldsymbol{a}}) \cdot (1 - F_{\boldsymbol{b}}).$$

In our case, $\boldsymbol{b}$ will be deterministic, hence $F_{\boldsymbol{b}} = \mathcal{T}_x(H)$ for some value of x. In this case $F_{\boldsymbol{c}}$ will be a truncated version of $F_{\boldsymbol{a}}$, where:

$$F_{\boldsymbol{c}}(q) = \begin{cases} 0 & q < x; \\ F_{\boldsymbol{a}}(q) & q \geq x. \end{cases}$$

An example of such a CDF is given in Figure 3.2.

5.5 Remarks

There are a few things that change when replacing deterministic variables with random variables. In this section we list some of these differences.

Dependency Random variables can be dependent, which means that the observed value of one random variable provides information on the value of the other random variable. Such dependencies can arise during computations. If two random variables are dependent, then the methods described in this chapter cannot be used. With deterministic variables this problem does not occur.

For example: let a_1, a_2, b_1 and b_2, be operations with stochastic processing times with the requirement that b_1 and a_2 start as soon as a_1 ends and b_2 starts when a_2 and b_1 end. In this case all the end times are dependent. This is a problem when computing the CDF of the end time of b_2, as we now have to take the dependency of the end times of a_2 and b_1 into account.

Fortunately, for the vehicle routing problem, the arguments of each addition are independent random variables, as long as we assume that the travel times are independent.

Inverse operation With deterministic variables, we can undo operations such as addition. For example, if $c = a + b$, then we can compute a using $a = c - b$. For random variables, this is not possible, because, for given b and c, there generally exist multiple a such that $c = a + b$.

For the deterministic vehicle routing problem we compute a latest arrival time for each shipment, such that, if we would arrive later, we would violate a time window of one of the shipments in the remainder of the schedule. In the stochastic case, we cannot compute subtractions and therefore cannot compute such a latest arrival time. Therefore the arrival time of each shipment in the remainder of the route has to be recomputed to verify whether the route is feasible.

Comparison For two deterministic variables a and b, the expression $a < b$ is either true or false. However, for random variables $\boldsymbol{a}$ and $\boldsymbol{b}$, the expression $\boldsymbol{a} < \boldsymbol{b}$ has a certain probability. This is why we introduced a minimal required reliability parameter.

Complexity Random variables contain more information than deterministic variables. Therefore they use more computer memory and computations take more time. Hence replacing deterministic variables with random variables will increase the running time of the algorithm.

Part II

Solving the SVRP-HTW

Chapter 6

SVRP-HTW Solver

In this chapter we briefly describe the various methods used for solving and improving solutions of the SVRP-HTW as defined in Chapter 2. We reused the implementation we created for [Conijn, 2012] and adapted it such that it handles stochastic travel times and uses random variables in the arrival time computations.

Our search algorithms only work with feasible solutions, which, as mentioned in [Conijn, 2012], is a significant limitation. However, we believe that the algorithms are good enough to show that, with our probabilistic arithmetic implementation, existing VRPTW solvers can be used to solve SVRP-HTW instances.

6.1 First Solution Methods

We have two different classes of first solution methods: nearest neighbor heuristics and insertion heuristics. These heuristic are similar to two of the heuristics listed in [Solomon, 1987], but use entirely different metrics.

Nearest Neighbor We have three methods that use a nearest neighbor heuristic.

NN1 The first method creates routes by repeatedly picking the shipment that, when appended to the current route, has the lowest expected **Serve Time** and is feasible. If no such shipment exists, we start creating the next route. The heuristic assumes that the problem instance is feasible.

NN2 The second method initially creates n empty routes. Then it iterates through the shipments in increasing order of the due date. Each shipment is then appended to a route, such that the resulting schedule is feasible and minimizes the expected **Serve Time** of the shipment. If the method encounters a shipment that cannot be appended to any of the routes, then it increases n by one and restarts.

NN3 The third method is a variation on NN2, which creates multiple schedules by sorting the shipments using a linear combination of distance to depot, ready time and due date, with varying weights. The best found schedule is selected.

Insertion heuristic We have four methods that use the Insertion heuristic, which we define below.

Definition 6.1.1 (Insertion heuristic)**.** *The Insertion heuristic greedily inserts a shipment, such that the increase in travel distance is minimized. Hence placing the shipment somewhere else in the schedule cannot result in a solution with a lower travel time. If the shipment cannot be inserted in any of the existing routes, a new route is created.*

IH1 The first method traverses the shipments in the order given by the problem instance and inserts the shipments using the insertion heuristic one by one.

IH2 The second method inserts the shipments in decreasing order of the sum of the distance to the depot and the ready time. This sorting order was reported in [van Bemmelen, 2012] to work quite well on average. We observed this as well, but do not have a logical explanation as to why.

IH3 The third method creates multiple schedules using random permutations and selects the best.

IH4 The forth method selects the best of multiple created schedules that use different sorting criteria of the shipments. The criteria are linear combinations of distance to depot, ready time and due date, with varying weights.

6.2 Local Search Neighborhoods

As the above first solution heuristics in general do not find the optimal solution, local search is applied to further improve the found solutions. Each method changes the schedule in some way and if the resulting schedule is at least as good as the original one, it is kept. Otherwise, the changes are reverted.

LS1 The first local search neighborhood is n-insertion. This procedure repeatedly removes n shipments and reinserts them using the insertion heuristic (See definition 6.1.1). This neighborhood is applied with $n \in \{1, 2, 3, 4, 5, 6\}$. Note that for $n = 1$, the resulting solution is always as good as the original solution. For $n \geq 2$, this is not the case, as is shown in [Conijn, 2012].

LS2 The second neighborhood is elimination. This procedure removes an entire route and reinserts it using the insertion heuristic. This neighborhood is applied once for every route.

LS3 The third neighborhood, exchange, tries to improve the schedule by swapping two shipments. This is done for each unordered distinct pair of shipments.

LS4 The forth neighborhood is 2-opt, which tries to improve the solution by reversing the shipments in a subsequence of a route.

LS5 The fifth neighborhood is two-elimination, a variation on LS2, which removes two routes instead of one and then reinserts their shipments.

6.2.1 Subproblem generation

We have one local search neighborhood that is quite different than the other neighborhoods: the subproblem generation neighborhood (**SG**). This neighborhood randomly selects a few routes from the current solution and then creates a smaller version of the current problem instance, by only selecting the shipments that are in the selected routes. This smaller instance is then solved using the first solution heuristics. The selected routes are then replaced by the solution found for the smaller instance if this improves the original solution.

Chapter 7

Construction of Problem Instances

The Solomon instances have various properties. The shipments can be clustered (C), fully random (R), or partially clustered and partially random (RC). The instances either have a small scheduling horizon (1) or a large scheduling horizon (2). The last two digits of the instance name enumerate the instances within these categories. Different instances have different time window sizes. In some of the instances only some of the shipments have a time window. Properties of the selected instances are shown in Table 7.1.

As there are no known benchmark instances in literature for the SVRP-HTW, we generated these instances ourselves, based on the Solomon instances [Solomon, 1987]. For this we selected the following subset of the Solomon benchmark instances: C101, C107, C201, C208, R101, R209, RC105, and RC201, based on the following criteria:

- At least one instance for each combination of clusteredness and scheduling horizon.
- All shipments have time windows.
- The time window sizes vary between the instances.
- For half of the instances, our VRPTW solver finds the best known solution.

As our VRPTW solver was only capable of finding the best known solutions for the C instances, we selected 4 C instances and 4 non-C instances.

For each selected instance we changed the travel time into a random variable as follows: let d be the distance between two locations, then the travel time between these locations is $0.7d + 0.3\boldsymbol{r}$, with $\boldsymbol{r} \sim \log\text{-}\mathcal{N}(\log(d) - 2,\ 2)$. The distribution of r is chosen such that $\mathbb{P}(\boldsymbol{r}) = d$. The CDF of this distribution for $d = 100$ is shown in Figure 7.1.

Due to driving speed limits, there is a fixed lower bound on the travel time and this should be reflected by the distribution, such that the chance that the travel takes less than the lower bound is 0%. This is never the case with log-normal distributions, which have been used in [Lecluyse et al., 2009] and [Taş et al., 2012]. Therefore we decided to use a shifted log-normal distribution. As distance we still use the unrounded euclidean distance.

We removed the due date of the depot. For the shipments we computed the reliabilities, using the best known solutions for the VRPTW instances, as follows:

- replace the travel times with our stochastic travel times;
- determine the reliabilities of each shipment, when executing the best known solution, using Monte-Carlo simulation (see Section 4.2), with a sample size of 10^6, such that the computational error is at most 0.001;
- set the reliability of each shipment in the SVRP-HTW instance to the computed reliability minus 0.01 rounded to two decimals.

Problem name	Scheduling horizon	Time windows		Best known solution	
		Size μ	σ	Routes	Total distance
C101	1236	60.76	10.47	10	828.94
C107	1236	180.00	0.00	10	828.94
C201	3390	160.00	0.00	3	591.56
C208	3390	640.00	0.00	3	588.32
R101	230	10.00	0.00	19	1650.80[a]
R209	1000	349.50	163.02	3	909.16
RC105	240	54.33	41.60	13	1629.44
RC201	960	120.00	0.00	4	1406.94[b]

Table 7.1: The selected VRPTW problem instances and their properties, along with the scores of the best known solutions.

[a]The best known solution to instance R101 is often reported as having a total distance of 1645.79. The solution was published in [Homberger, 2000]. However the author confirms that this solution was found using different rounding rules [Jörg Homberger, personal communication, May 2, 2013]. By solving a Mixed Integer Program of R101, we were also able to determine that, when using unrounded euclidean distances, 1650.80 is the optimal solution.

[b]The best known solution to instance RC201 is often reported as having a total distance of 1406.91. The solution was published in [Mester et al., 2007]. However, the author confirms that their solutions were found using two digit accuracy [David Mester, personal communication, Juni 19, 2013].

As a result, the best known solution for the VRPTW instance is also a feasible solution of the constructed SVRP-HTW instance. Figure 7.2 shows boxplots of the resulting reliabilities in the selected instances. Note that the reliability of each shipment is at most 0.99.

The shipments in the C instances all got relatively high reliabilities, except for shipment 47 in the C1 instances and shipment 90 in the C2 instances. The instances that have more shipments per route have higher reliability on average than those instances that only allow short routes. The instances with small time windows (the 01 instances) have shipments with relatively lower reliabilities.

Because the average travel time in the SVRP-HTW is equal to the travel time in the VRPTW, we have that the reliabilities are all at least 0.5. Also, because the reliabilities are all at least 0.5, a feasible solution of the SVRP-HTW is also a feasible solution of the corresponding VRPTW instance. Therefore, if the solution used for constructing the SVRP-HTW is an optimal solution of the VRPTW instance, then it is also an optimal solution of the SVRP-HTW instance.

To see what happens when we increase the required reliability, we created 2 variations for each of the 8 instances. The first variation has all reliabilities increased by 2%. To ensure that the instance remains solvable, these reliabilities are then capped at the maximum obtainable reliability, obtained when the shipment is the first in the route, minus 0.01. In general the resulting required reliability will be at least 0.5% higher than what is obtained by the solution to construct the original SVRP-HTW instance. Hence, each route of that solution will, in general, be infeasible. Therefore we expect a significant increase in truck usage.

The second variation has all reliabilities increased by 4% and subsequently capped to the maximum obtainable reliability minus 0.01. This is a 2% increase of the first variation. While we do expect another increase in travel time and probably truck usage, we do not expect the increase to be as large as between the original and first variation. This is because the additional 2% increase likely does not invalidate all routes in the optimal solution of the first variation.

The 56 generated SVRP-HTW instances as well as their variations are available at: `www.student.tue.nl/S/b.j.conijn/svrp-htw.zip`.

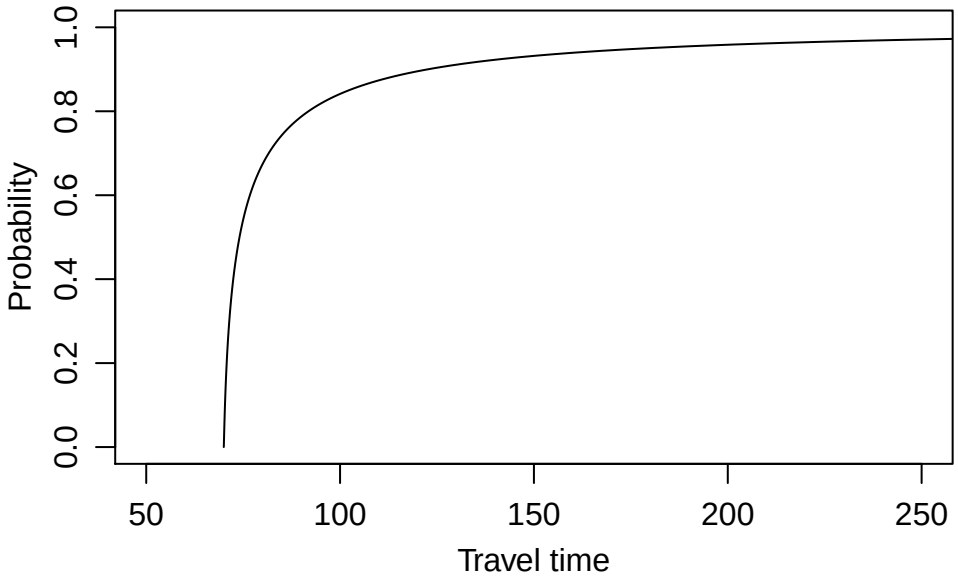

Figure 7.1: The cumulative distribution function of the travel time for an arc with distance 100.

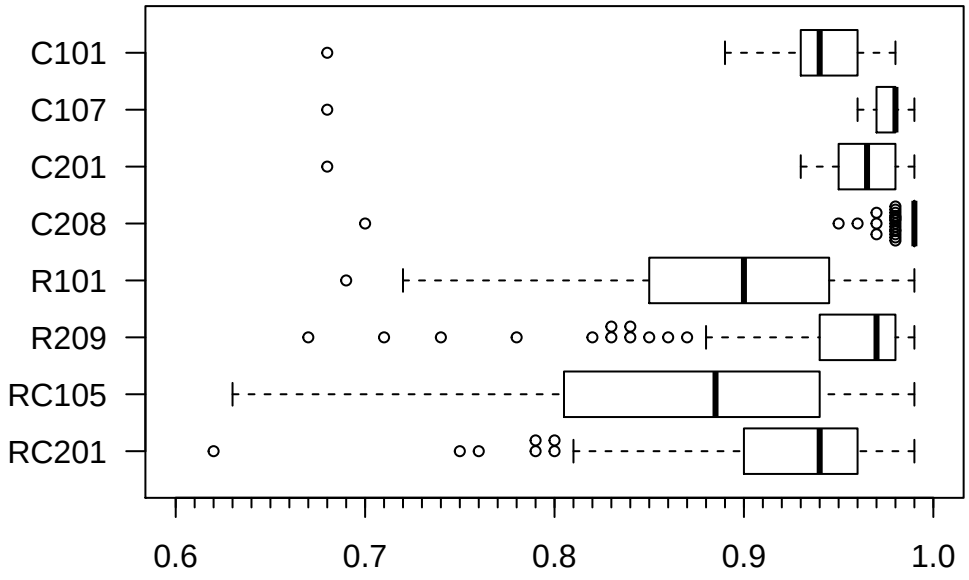

Figure 7.2: Boxplots of the reliabilities that were chosen for the selected instances. Instance C101 has a shipment with reliability 0.68, denoted by a small circle, the other reliabilities are within the range 0.89 and 0.98, denoted by the dotted line. Of the 100 shipments, 25 have a reliability of at most 0.93 (left side of the box). There are 50 shipments with a reliability of at most 0.94 (thick black line) and 75 shipments with a reliability of at most 0.96 (right side of the box).

Chapter 8

Computational Results

8.1 Running time

We executed the 7 first solution methods on the converted Solomon instances. We did this using both the deterministic and the stochastic solver. For each first solution method we computed the average[1] computation time (shown in Table 8.1).

	Deterministic time (ms)	**Stochastic time (ms)**	**Stochastic/ Deterministic**
NN1	0.11	154	1400
NN2	0.29	192	661
NN3	8.02	3538	441
IH1	0.19	231	1217
IH2	0.18	329	1826
IH3	18.41	22211	1206
IH4	10.01	14654	1464

Table 8.1: Average execution times of the 7 first solution heuristics on 8 instances. On average, executing the stochastic solver takes 1145 times as long.

8.2 Found solutions

We used our algorithm to solve the selected 8 VRPTW instances and the generated 24 SVRPHTW instances. Each instance was solved using the 7 first solution methods. Then, these 7 solutions were further improved by running the local search neighborhoods and repeating these as long as they find improvements. Of these 7 improved solutions the best one is selected. The results are presented in Table 8.2.

Our algorithm finds the best known solution for both the deterministic and original stochastic C instances. For the R instances, our deterministic algorithm does not find the best known solution. The set of feasible solutions for the original stochastic problem is a subset of the set of feasible solutions of the deterministic problem, therefore it is to be expected that our stochastic algorithm performs worse on these instances.

For most +2% instances, the solution found uses many more trucks than the solution for the original stochastic instance. This was expected, as the reliability constraints for the original instance were tight. For the +4% instances the increase in truck usage is a lot smaller and in some cases only the total travel distance increased.

[1] For computing average computation times, we use the more appropriate geometric mean. For a set of numbers A, this is computed as $\prod(A)^{\frac{1}{|A|}}$ or $\exp(\frac{1}{|A|}\sum_{a\in A}\log(a))$.

The first solution heuristic IH4 most often finds the best first solution. However, this does not guarantee that this solution will remain the best solution after local search is applied. So a better first solution heuristic does not necessarily result in a better final solution. For each first solution heuristic, there is an instance, whose best solution was found using that first solution heuristic. So none of the heuristics is strictly better than another.

Instance	Type	Time	Trucks	Distance	FSH	
		(s)	LB	Best found		FS
C101	deterministic	0.3		10	828.94	*	IH4
	stochastic	152.8	10	10	828.94	*	IH*
	stochastic +2%	180.2	20	20	1306.48	*	IH4
	stochastic +4%	257.4	20	20	1313.98	NN1	IH4
C107	deterministic	0.5		10	828.94	*	IH1
	stochastic	214.3	2	10	828.94	*	IH1
	stochastic +2%	235.8	3	15	1169.39	IH4	IH1
	stochastic +4%	315.8	3	15	1184.91	IH3	IH4
C201	deterministic	0.2		3	591.56	*	NN1
	stochastic	311.2	1	3	591.56	*	NN1
	stochastic +2%	423.8	1	4	654.55	NN2	IH3
	stochastic +4%	367.1	1	5	714.52	IH4	IH1
C208	deterministic	0.3		3	588.32	NN*	NN3
	stochastic	710.0	0	3	588.32	NN*	NN1
	stochastic +2%	763.1	0	4	616.76	IH4	NN1
	stochastic +4%	840.6	0	4	625.17	IH4	NN1
R101	deterministic	0.8		19	1687.47	NN2	IH4
	stochastic	192.9	19	20	1684.58	NN1	IH4
	stochastic +2%	159.3	26	30	2037.85	IH4	IH4
	stochastic +4%	155.8	30	33	2215.52	IH4	IH4
R209	deterministic	0.6		3	988.35	IH1	NN1
	stochastic	957.5	0	4	962.99	IH3	IH4
	stochastic +2%	836.1	0	5	1002.84	IH2	IH4
	stochastic +4%	797.2	0	6	965.33	NN3	IH4
RC105	deterministic	0.7		15	1571.13	IH4	IH4
	stochastic	346.1	6	15	1739.51	IH4	IH4
	stochastic +2%	351.4	8	19	1778.26	IH2	IH4
	stochastic +4%	458.3	11	21	1925.32	NN3	IH4
RC201	deterministic	0.5		4	1576.73	IH4	IH4
	stochastic	458.2	2	5	1528.74	IH3	IH4
	stochastic +2%	374.0	2	7	1439.80	IH4	IH4
	stochastic +4%	312.7	2	8	1472.34	IH1	IH4

Table 8.2: The solutions found by our algorithm for deterministic, stochastic and stochastic with increased reliabilities (+2% and +4%) are shown in the column 'best found'. The 'LB' column lists the number of shipments that cannot be planned after any other shipment. This is a lower bound for the required number of trucks. The 'FSH' column shows which first solution heuristic(s) found the best solution before local search ('FS') and after ('best solution'). If multiple heuristics found the best solution, this is shown with a '*'.

8.3 Accuracy of solutions

We used Monte-Carlo simulation (as explained in Section 4.2) to determine the accuracy of our solutions. The results of these simulations are summarized in Table 8.3.

Both our computational method and the Monte-Carlo simulation use approximations, therefore it is to be expected that some of the reliability constraints are found to be violated. For only 16 of the 2400 shipments this is the case. Most violations are less than 0.1% and only two are larger

Instance	Type	Reliability error		Violations	
		Average ($\times 10^{-3}$)	Deviation ($\times 10^{-3}$)	Number	Largest ($\times 10^{-3}$)
C101	+0%	-1.269	0.898	0	
	+2%	-0.188	0.233	0	
	+4%	-0.128	0.254	1	0.093
C107	+0%	-0.306	0.650	0	
	+2%	0.074	0.231	1	0.206
	+4%	0.162	0.232	1	0.134
C201	+0%	-0.473	0.425	0	
	+2%	0.504	0.461	1	0.207
	+4%	0.592	0.447	0	
C208	+0%	0.666	0.606	0	
	+2%	0.719	0.488	0	
	+4%	0.537	0.454	0	
R101	+0%	-3.232	3.070	1	1.279
	+2%	-1.861	2.030	1	6.387
	+4%	-1.663	2.091	4	2.627
R209	+0%	0.228	0.847	0	
	+2%	0.321	0.873	0	
	+4%	0.339	0.519	0	
RC105	+0%	-3.035	2.745	1	1.220
	+2%	-1.823	2.053	2	0.442
	+4%	-1.673	2.270	0	
RC201	+0%	-0.706	1.582	0	
	+2%	-0.229	1.352	1	0.117
	+4%	0.005	1.236	2	5.019

Table 8.3: We used Monte-Carlo simulation, with 10^6 samples, to check the accuracy of the solutions we found. This table lists for each SVRP-HTW instance the error involved in our probabilistic arithmetic computations and how often those errors lead to a time window violation. The reliability error is the simulated reliability minus the computed reliability. The largest violation is the maximum over all shipments of the required reliability minus the simulated reliability. The simulation detects reliability violations in 11 of the instances, however the violations are very small. The average duration of a simulation is 18.7 seconds.

than 0.5%, but still below 1%. From the reliability errors, we can deduce that our method is within 0.37% of the actual reliability with a probability of at least 95%.

We listed detailed information on the 16 violations in Table 8.4. It shows that the 2 relatively large errors occur where the probability density function is large and thus where the CDF is steep. This suggests that these errors are caused by a CDF that is slightly shifted to the left.

The simulations took approximately 18.7 seconds to compute. With our probabilistic arithmetic implementation the reliabilities were computed in 12.5 milliseconds.

Instance	**Type**	**Shipment**	**Slack** ($\times 10^{-3}$)	**Violation** ($\times 10^{-3}$)	**Probability Density** ($\times 10^{-3}$)	**Reliability**
C101	+4%	83	0.096	-0.093	0.134	0.99
C107	+2%	98	0.077	-0.206	0.297	0.98
	+4%	98	0.077	-0.134	0.297	0.98
C201	+2%	64	0.317	-0.207	0.188	0.99
R101	+0%	10	0.738	-1.279	2.656	0.93
	+2%	47	4.778	-6.387	16.235	0.81
	+4%	73	0.463	-2.627	5.030	0.92
	+4%	90	2.079	-0.497	2.737	0.92
	+4%	75	0.234	-0.644	0.850	0.97
	+4%	35	0.781	-0.052	2.314	0.95
RC105	+0%	27	3.346	-1.220	3.392	0.89
	+2%	52	0.341	-0.442	0.936	0.96
	+2%	66	1.622	-0.050	1.785	0.94
RC201	+2%	40	0.490	-0.117	0.545	0.96
	+4%	68	1.958	-1.799	4.204	0.86
	+4%	62	3.219	-5.019	26.553	0.85

Table 8.4: The Monte-Carlo simulation found that in the solutios we found, there were 16 shipments that have the reliability requirement violated. This table lists these 16 shipments. 'Slack' is the computed reliability minus the required reliability. 'Violation' is the simulated reliability minus the required reliability.

Chapter 9

Conclusion

We adapted the Vehicle Routing Problem with Time Windows (VRPTW). The travel times were changed to be stochastic and we added minimal required reliabilities to the shipments, while retaining the hard time windows. This resulted in the Vehicle Routing Problem with Stochastic Travel Times and Hard Time Windows (SVRP-HTW).

By adapting our existing solver for the VRPTW, using our probabilistic arithmetic implementation, such that it can solve the SVRP-HTW, we have shown that an existing solver for a deterministic optimization problem can easily be adapted to a solver for a variation of that problem that includes uncertainty.

Computing with random variables is a lot slower than computing with deterministic variables. However, the developed probabilistic arithmetic method is still much faster than using Monte-Carlo simulation, while still being reasonably accurate.

Chapter 10

Further research

Time dependent travel times In this thesis, we assume that the travel times are uncertain, but constant throughout the day. In practice, travel times tend to be a lot larger and less predictable during rush hours. Further research has to be performed to add support for time dependency to the developed probabilistic arithmetic method.

Dependent random variables Traffic congestion generally has a local and a global component. We assumed that all uncertainty was local, thereby assuming that all travel times were independent. In other optimization problems, dependencies can also arise during the computations. Our probabilistic arithmetic assumes independence for each computation and therefore would introduce an error, which depends on the dependency.

Quasi Monte-Carlo method Other more advanced variations on the Monte-Carlo method exist. One variation uses low-discrepancy sequences rather than a sequence of random numbers, such that it has better convergence. This convergence might be sufficiently fast enough that it could be used as an alternative method for probabilistic arithmetic.

Other methods for probabilistic arithmetic There are many different ways to perform probabilistic arithmetic. An extensive list is given in [Williamson and Downs, 1990]. We chose our method, because it allows to easily compute the minimum or maximum of a random variable and a constant. However, there might exist other faster and more accurate probabilistic arithmetic methods that support addition and the computation of maximums and minimums. The Fast Fourier Transform based method in [Hackbusch, 2008], might also be a good candidate to use as the basis of a probabilistic arithmetic method.

Determine bounds on the computational errors For the Monte-Carlo method we could properly derive a theoretical bound on its accuracy. For the developed probabilistic arithmetic, such bound is much harder to obtain, but would increase confidence in the values obtained when using this method in computations.

Bibliography

[Agarwal and Burrus, 1975] Agarwal, R. and Burrus, C. (1975). Number theoretic transforms to implement fast digital convolution. *Proceedings of the IEEE*, 63(4):550–560.

[Ando and Taniguchi, 2006] Ando, N. and Taniguchi, E. (2006). Travel time reliability in vehicle routing and scheduling with time windows. *Networks and Spatial Economics*, 6(3):293–311.

[Brito et al., 2008] Brito, J., Campos, C., Castro, J., Martínez, F., Melián, B., Moreno, J., and Moreno, J. (2008). Fuzzy Vehicle Routing Problem with Time Windows. In *Proceedings of IPMU*, volume 8, page 1267.

[Chang et al., 2009] Chang, T.-S., Wan, Y.-w., and Ooi, W. T. (2009). A stochastic dynamic traveling salesman problem with hard time windows. *European Journal of Operational Research*, 198(3):748–759.

[Conijn, 2012] Conijn, B. (2012). Performance Analysis of Local Search Neighborhoods for the Vehicle Routing Problem with Time Windows. Available at `www.student.tue.nl/S/b.j.conijn/preparation-project.pdf`.

[Dabia et al., 2011] Dabia, S., Røpke, S., van Woensel, T., and de Kok, T. (2011). Branch and Cut and Price for the Time Dependent Vehicle Routing Problem with Time Windows. Technical Report 361, BETA.

[Dantzig and Ramser, 1959] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, (6):80–91.

[Dufresne, 2008] Dufresne, D. (2008). Sums of lognormals. In *Proceedings of the 43rd Actuarial Research Conference*.

[Hackbusch, 2008] Hackbusch, W. (2008). Fast projected convolution of piecewise linear functions on non-equidistant grids. *From Nano to Space*, pages 145–160.

[Homberger, 2000] Homberger, J. (2000). *Verteilt-parallele Metaheuristiken zur Tourenplanung*. PhD thesis, Deutscher Universitäts-Verlag.

[Ibaraki et al., 2005] Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., and Yagiura, M. (2005). Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science*, 39(2):206–232.

[Jones, 1977] Jones, T. (1977). A computer method to calculate the convolution of statistical distributions. *Mathematical Geology*, 9(6):635–647.

[Kok et al., 2011] Kok, A., Hans, E., and Schutten, J. (2011). Optimizing departure times in vehicle routes. *European Journal of Operational Research*, 210(3):579–587.

[Kumar and Panneerselvam, 2012] Kumar, S. and Panneerselvam, R. (2012). A Survey on the Vehicle Routing Problem and Its Variants. *Intelligent Information Management*, 4(3):66–74.

[Kuo et al., 2009] Kuo, Y., Wang, C., and Chuang, P. (2009). Optimizing goods assignment and the vehicle routing problem with time-dependent travel speeds. *Computers & Industrial Engineering*, 57(4):1385–1392.

[Lecluyse et al., 2009] Lecluyse, C., Woensel, T. V., and Peremans, H. (2009). Vehicle routing with stochastic time-dependent travel times. *4OR: A Quarterly Journal of Operations Research*, 7(4):363–377.

[Lenstra and Rinnooy Kan, 1981] Lenstra, J. K. and Rinnooy Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. *Transportation Science*, (11):221–227.

[Mester et al., 2007] Mester, D., Bräysy, O., and Dullaert, W. (2007). A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications*, 32(2):508–517.

[Plehn and Bruns, 2005] Plehn, I. and Bruns, I. (2005). Approximation and Computation of Random Variables using Finite Elements. *Logistics Journal*.

[Qureshi et al., 2010] Qureshi, A. G., Taniguchi, E., and Yamada, T. (2010). Exact solution for the vehicle routing problem with semi soft time windows and its application. *Procedia-Social and Behavioral Sciences*, 2(3):5931–5943.

[Røpke and Pisinger, 2005] Røpke, S. and Pisinger, D. (2005). A general heuristic for vehicle routing problems. Technical report, Technical report, DIKU-Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark., 25th February.

[Solomon, 1987] Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35:254–265.

[Taş et al., 2013] Taş, D., Dellaert, N., van Woensel, T., and de Kok, T. (2013). Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, (40):214–224.

[Taş et al., 2012] Taş, D., Gendreau, M., Dellaert, N., van Woensel, T., and de Kok, A. (2012). Vehicle Routing with Soft Time Windows and Stochastic Travel Times: A Column Generation and Branch-and-Price Solution Approach. *European Journal of Operational Research*.

[Tang et al., 2009] Tang, J., Pan, Z., Fung, R. Y., and Lau, H. (2009). Vehicle routing problem with fuzzy time windows. *Fuzzy sets and systems*, 160(5):683–695.

[van Bemmelen, 2012] van Bemmelen, M. (2012). Modeling and Solving a Real-life Load Building and Routing Problem in the Retail Industry. Master's thesis, Eindhoven University of Technology.

[van der Hooft, 2013] van der Hooft, E. J. (2013). Using Travel Time Predictions based on TomTom's Big Data in Logistical Models. Master's thesis, Eindhoven University of Technology.

[Van Woensel et al., 2008] Van Woensel, T., Kerbache, L., Peremans, H., and Vandaele, N. (2008). Vehicle routing with dynamic travel times: A queueing approach. *European Journal of Operational Research*, 186(3):990–1007.

[Williamson and Downs, 1990] Williamson, R. C. and Downs, T. (1990). Probabilistic arithmetic. i. numerical methods for calculating convolutions and dependency bounds. *International Journal of Approximate Reasoning*, 4(2):89–158.