Utrecht University

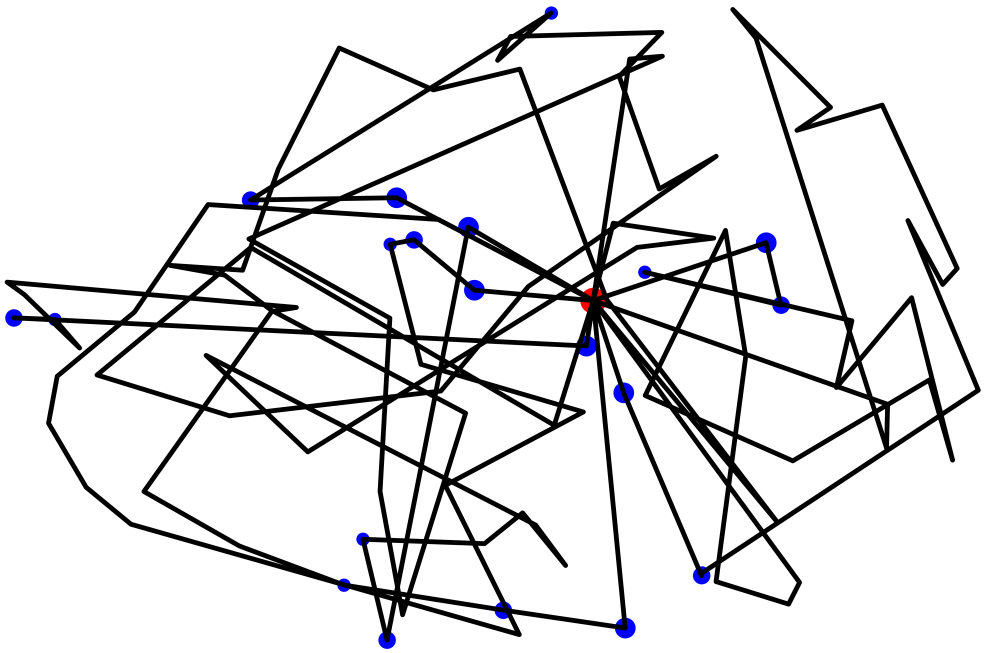
Faculty of Science

The Stochastic Vehicle Routing Problem

with Time Windows and Load and Wind Dependent Travel Times

Master's Thesis

Mathematics



Graphical representation of a VRP schedule with 7 vehicles.

Author

S.T. van der Wal

studentnr. 6148956

Supervisors

Dr. ir. J.M. van den Akker

Department of Informatics, UU

Dr. S. Dirksen

Department of Mathematics, UU

P. de Bruin, MSc

Department of Informatics, UU

May, 2024

Abstract

With increased e-commerce, the planning of last mile delivery has gained importance over the years. In light of dense urban areas and sustainability concerns, electric cargo bikes are well fit for this task. Due to constraints on the power output of the electro motor, the travel times are dependent, amongst others, on the load and the wind. In this thesis, a method is developed to compute optimal schedules for the stochastic vehicle routing problem (SVRP) for such cargo bikes. This is done using a combination of simulated annealing and a set cover formulation. The uncertainty in the weather forecast propagates into uncertainty in the travel times. The travel times are considered to be stochastic and strongly correlated. It is therefore non trivial to determine the arrival time distribution for each customer. In this thesis, the arrival time distribution is approximated using the non parametric technique of kernel density estimation (KDE). This technique can produce very robust schedules with only 5 or 20 samples. Average service levels of outputted schedules are often within one thousandth of a perfect score. With more samples, run time increases, without much benefit to average costs. Using the forecast wind as a deterministic given proved to be about as bad as ignoring the wind for the selected distribution of wind. Simply tightening the instance by shortening the time windows produces schedules with high service level and also high costs. Cheapest schedules are found approximating the arrival time distribution with only one sample that is exactly the weather forecast.

Contents

Introduction

In recent decades, the internet has made it possible for companies to build an online shop. The customer orders products online and these are then shipped in a multistage process to the customer. One part of this process is distribution through a local depot, from which the so-called last mile delivery is done. This last mile delivery can be done using e.g. a van or an electric cargo bike. In recent years, the e-commerce has grown significantly. In order to compete with physical shops, where the customer walks away with the product in hand, some companies promise next day, same day or even 2 hour delivery services (Savelsbergh M. 2016). Some companies also allow customers to select a time window for delivery. Keeping up with these promises is important to companies, since negative delivery experiences might influence the customers choice of retailers in the future (Vakulenko et al. 2019). The planning of such last mile deliveries is modeled as the Vehicle Routing Problem.

The Vehicle Routing Problem (VRP) is a generalization of the Traveling Salesman Problem (TSP) and is thus NP-hard. It was first introduced by Dantzig and Ramser 1959 as the Truck Dispatching Problem. The problem consists of one or more depots from which the vehicles follow a route visiting a subset of customers each. These customers each have an individual demand which must be satisfied by a vehicle. In a feasible solution, each customer is visited exactly once by a vehicle. The vehicles have a constraint capacity that is significantly lower than total customer demand. This increases the complexity of finding appropriate routes.

In the truck dispatching problem Dantzig and Ramser 1959 modeled the distribution of gasoline to service stations. They used a procedure based on a linear programming formulation to compute near optimal solutions. Solomon 1987 models the VRP with time window constraints (VRPTW) which stem from customers being able to set delivery deadlines and earliest deliver times. Due to the added structure of the time windows, an insertion heuristic proved very useful for finding good solutions. This heuristic is based on inserting an unscheduled customer in an existing route in the best possible place. Other variants include VRP with simultaneous pick-up and delivery (VRPSPD) (Koç, Laporte, and Tükenmez 2020). This variant is used to solve the routing of the distribution and collection of medicines and their containers. See e.g. (Chaieb and Sassi 2021) and references therein for many strategies and algorithms. In the past years, techniques varying from local search to population search have been proposed for finding good solutions to various variants, see e.g. Bräysy and Gendreau 2005 for an overview.

In the SVRPTW (stochastic vehicle routing problem with time windows), the travel times are modeled as random variables. Under this uncertainty, we would like to know what the probability of arriving within a time window is. The expected total travel time is relatively straight forward to calculate or approximate. A more difficult task is computing an approximation of the service level, often denoted by α. The service level is often defined as the probability of on-time delivery for individuals or as the expected fraction of on-time delivery for the entire schedule. An important factor for approximating the service level is approximating the arrival time distribution. This distribution depends directly on the travel times and on certain realisation specific factors, such as whether we have waited or not and in a time-dependent case even more complex relations exist. Zhang, Lam, and B. Y. Chen 2013 and Lent 2018 model the time windows as soft and thus a term in the objective functions incurs a penalty for tardiness and/or earliness. The former assign a specified service level to each customer. They propose a new method for computing the arrival time distribution based on discretizing the arrival time cumulative distribution function. The latter proposed a branching method that computes a probability for each leaf and the corresponding outcome. This method proved to perform better than using a parametric approximation of the sum and maximum of two normally or exponentially distributed random variables. Others, a.o. Li, Tian, and Leung 2010 Ehmke, Campbell, and Urban 2014 Gutierrez et al. 2018, modeled the time windows as hard and added a constraint regarding the overall service level. Generally the algorithms exhibit some kind of trade off between computational efficiency and accuracy of the arrival time distribution. They use a chance constraint formulation, in which the probability of arriving on time is bound by a number, also called the service level α. This can either include the start of the time window, i.e. $\mathbb{P}(a_i \leq AT_i \leq b_i) \geq \alpha_i$ or only a deadline $\mathbb{P}(AT_i \leq b_i) \geq \alpha_i$, while counting waiting time separately.

Many authors choose to use some form of a local search algorithm, such as a tabu search heuristic which has proven to perform well on deterministic instances, see e.g. Gendreau, Hertz, and Laporte 1994. All local search heuristics rely on a trade-off between exploration of the search space and locally optimizing a solution. In Hesselmans 2022, this is done with a simulated annealing (SA) approach, where potentially good solutions are then used in a set covering algorithm. This is similar to the approach of Dhahri, Zidi, and Ghedira 2014, while Chiang and Russell 1996 implements a combination of tabu search and simulated annealing. The combined effort of a local search approach and a secondary stage ILP optimization proved very effective in those studies.

The previously mentioned cargo bike is well fit for the last mile delivery. Especially in urban areas, this mode of transportation can be beneficial over trucks, as bicycles might contribute to better air quality (Savelsbergh M. 2016) and can be more cost effective (Sheth et al. 2019). Part of the effectiveness of the use of cargo bicycles comes from efficient planning. While the speed of trucks is largely unaffected by varying weights of cargo, bikes might suffer large losses in speed from transporting heavy goods. A new variant of the problem was therefor introduced with load dependent travel times (VRPLTT) by Fontaine 2021. In this study, it was shown that taking cargo weight into account is beneficial for finding feasible routes as ignoring the weight could lead to infeasible schedules. Expanding on this model, Hesselmans 2022 showed that assuming no wind can cause a considerable amount of deliveries to be too late. In that study, a fixed, known, wind was assumed, which leads to different routes. Sometimes even with better total travel times.

Besides the load, another important factor to cycling speed is the aerodynamic drag that acts on the bike, see e.g. Barry 2018. The force that acts on the bike depends on both the wind speed and the yaw angle, i.e. difference between travel direction and wind direction. Moreover, this force also depends on the speed of the bike itself via a third order polynomial. This causes small changes in speed, direction or both to change the force on the bike in a non straightforward way. Martin et al. 1998 showed that by using a complicated physical model and measuring the average wind direction, it is possible to account for most of the effects of wind. Since the wind is both volatile on the short term (Joffre and Laurila 1988), and changes on a day to day basis, it is impossible to make accurate exact predictions.

This thesis aims at expanding the knowledge on solving the vehicle routing problem with time window constraints and stochastic wind dependent travel times. It does so by considering a stochastic wind that affects the travel times. This results in different travel time distributions for each route that the vehicle can take. The travel times are neither independent nor identically distributed. Using a weather forecast and an assumed wind distribution, an approximation of the travel time distributions can be constructed that is used to compute a schedule to the SVRPTW. The two main aims of this thesis are using the KDE technique to compute a non parametric arrival time approximation, and using this approximation to compute robust and optimal schedules for the SVRPTW with non i.i.d. travel times.

The definition of the mathematical model is done in Section 1. It is followed by the physical setting that is used to compute travel times in Section 2. Section 3 describes how to construct the arrival time distributions and Section 4 describes how the solution algorithm works and Section 5 explains which settings are chosen. The thesis ends with numerical results for different instances in 6, tested against several heuristics and conclusions in 7.

1 Mathematical Model

This chapter serves to formalize the mathematical model underlying the stochastic vehicle routing problem with time windows that is considered in this thesis. This formulation is inspired by the works of Cordeau et al. 2007 and Fontaine 2021.

1.1 Graph Representation

The vehicle routing problem (VRP) can be modelled as a complete digraph $G = (N, E)$. The nodes $N \setminus \{0\} = \{1, ..., n\}$ represent the n customers, and 0 represents the depot. The edges $E = \{(i, j)|i, j \in N, i \neq j\}$ represent the roads between customers and the depot. To each customer $i \in N \setminus \{0\}$, we associate a demand q_i, a service time s_i and a time window $[a_i, b_i]$. Each edge (i, j) is associated with a travel time function $TT_{(i,j,l)}(\omega)$, which maps, for an edge and load level, the weather ω to a travel time; a load carried over that edge $f_{i,j}$, and decision variables $x_{i,j}$ indicating whether edge (i, j) is used and $z^l_{i,j}$ representing whether the edge is traversed with load level l. The computation of the travel time is further explained in Section 2.1. Furthermore we have V vehicles with uniform capacity Q. The vehicles have L load levels defined by intervals $l = [p_l, r_l]$, such that $r_{l-1} = p_l$ and $p_0 = 0$ and $r_L = Q$. In case of conflict i.e. $f_{i,j} = r_{l-1} = p_l$ we choose load level l, thus assuming the heavier load level.

1.2 MIP Formulation

The optimization problem that is considered is a mixed integer program (MIP). The program is written as a function to be minimized and a set of constraints.

$$\text{minimize} \quad \mathbb{E}_\omega \left[\sum_{(i,j)\in E, l\in L} \left(TT_{(i,j,l)}(\omega)\right) z^l_{i,j} + \gamma_1 \sum_{j\in N\setminus\{0\}} WT_j + \gamma_2 \sum_{j\in N\setminus\{0\}} U_j \right] \tag{1}$$

subject to

$$\sum_i x_{i,j} = 1 \quad \forall j \in N \setminus 0, \tag{2}$$

$$\sum_i x_{j,i} = 1 \quad \forall j \in N \setminus 0, \tag{3}$$

$$q_j x_{i,j} \leq f_{i,j} \leq (Q - q_i) x_{i,j} \quad \forall (i, j) \in E, \tag{4}$$

$$\sum_l z^l_{i,j} = x_{i,j} \quad \forall (i, j) \in E, \tag{5}$$

$$\sum_i f_{i,j} - \sum_k f_{j,k} = q_j \quad \forall j \in N, \tag{6}$$

$$\sum_l p_l z^l_{i,j} \leq f_{i,j} \leq \sum_l r_l z^l_{i,j} \quad \forall (i, j) \in E, \tag{7}$$

$$AT_j \geq AT_i + s_i + \sum_l z^l_{i,j}(TT^{l,\omega}_{i,j} + WT_j) - M_{i,j}(1 - x_{i,j}) \quad \forall j \in N \setminus 0, \tag{8}$$

$$a_i \leq AT_i + WT_i \quad \forall i \in N, \tag{9}$$

$$M * U_i < AT_i - b_i \quad \forall i \in N, \tag{10}$$

$$U_i \in \{0, 1\} \quad \forall i \in N, \tag{11}$$

$$x_{i,j} \in \{0, 1\} \quad \forall (i, j) \in E, \tag{12}$$

$$z^l_{i,j} \in \{0, 1\} \quad \forall l : \forall (i, j) \in E, \tag{13}$$

$$f_{i,j} \geq 0 \quad \forall (i, j) \in E. \tag{14}$$

The objective function minimizes a linear combination of the travel time, the waiting time and the tardiness. The first term refers to costs that are made by driving around and refers directly to the total travel time. The second term refers to costs associated with waiting. The last term refers to costs associated with not being able to deliver before the end of the time window. The factor U_j is the tardiness and is ensured by Constraints 10 and 11 to be one whenever the time window is not met. The factors $\gamma_{1,2}$ allow to assign different relative weight to waiting and not delivering within the time window. Constraints 2 and 3 make sure each customer is visited only once. Constraints 4,5, 6 and 7 restrict the allowed weight carried by the vehicle such that it always carries the demand it sets out to satisfy and simultaneously is never loaded heavier than the capacity Q. Constraints 8, 9 and 10 make sure vehicles arrive at the correct times at customers, where 8 is a linearized version of a nonlinear constraint, with $M_{i,j} = \max\{0, b_i + s_i + TT^{\omega}_{(i,j,l)} - a_j\}$ (see Fontaine 2021 and Cordeau et al. 2007). Constraints 11, 12 and 13 are integrality constraints, and 14 is a nonnegativity constraint on the carried load.

Every solution to this MIP is represented by a schedule. The solution is feasible if it meets all constraints. A schedule consists of routes, which consists of a list of customers. This determines the assignment of bicycles to customers and the order in which they are visited. The quality of such a schedule is determined by the expected costs and the expected service level. The expected costs and service level are entirely determined by the travel time distribution. It is thus important to correctly compute the travel time associated with any road.

2 Computing the Travel Time

In order to solve the SVRPTWLTT with wind dependent travel times, an accurate technique for computing the travel times is needed. The time it takes to traverse a road between to customers is a consequence of characteristics of the bike, such as the mass and shape, and of the environment, such as wind and road slope. The effect of wind on the bike is very nontrivial as small changes in true wind velocity result in nonlinear changes in the apparent (experienced) wind, see e.g. Martin et al. 1998 and Barry 2018. To correctly compute the travel time over an arc, a force balance equation is solved for the bike speed v_b for a given wind in Section 2.1. Then Section 2.2 describes how the wind distribution is defined. Finally, Section 2.3 explains how we obtain a set of samples from the travel time distribution.

2.1 Bike-Wind Interaction

The forces on a bike that we consider are the propulsion force ($\vec{F}_{\mathrm{p}}$), the gravitational force ($\vec{F}_{\mathrm{g}}$), the roll resistance force ($\vec{F}_{\mathrm{r}}$), and the air drag force ($\vec{F}_{\mathrm{d}}$). Similar to Fontaine 2021, we assume 5% of the propulsion force is lost as mechanic resistance. The individual forces can be computed via their standard definitions

$$F_g = m \cdot g \cdot \sin\left(\arctan\left(h\right)\right), \tag{15}$$

where m is the mass of the vehicle, g is the gravitational acceleration, generally set to $9.81ms^{-2}$ corresponding to sea level, and h the slope of the road in percentage. The roll resistance force

$$F_r = C_r \cdot N = C_r \cdot m \cdot g \cdot \cos\left(\arctan\left(h\right)\right), \tag{16}$$

where C_r is the coefficient of roll resistance, and N the size of the normal force at the road. The normal force is the force that prevents objects from falling through the road or ground, normal refers to the force acting perpendicular to the road. The air drag force

$$F_d = \frac{\rho C_d A}{2} \|\vec{v}_{wind} - \vec{v}_{bike}\|_2^2, \tag{17}$$

where C_d is the coefficient of drag, ρ the air density, and A the frontal area. For computational ease, we will assume the rider and bicycle, aerodynamically, form a standing cylinder together.

Throughout this paper, we assume the bicycle is at constant speed on each road segment. According to Newton's second law, this means the net forces on the bicycle are zero. This yields the equation

$$\sum \vec{F} = 0.95\vec{F}_p + \vec{F}_g + \vec{F}_r + \vec{F}_d = 0, \tag{18}$$

which is a vector sum, see Appendix A for more details. The air drag force does not necessarily work in the direction of travel. The component that is not in the travel direction is countered by the friction force between the tires and the road. To account for this, we take the inner product with the direction of travel. We now get the scalar equation

$$F_p = \frac{1}{0.95}\left(F_g + F_r + F_d(\vec{e}_p \cdot \vec{e}_d)\right), \tag{19}$$

where $\vec{e}_p$ and $\vec{e}_d$ are the vectors of unit length in the direction of the propulsion and drag force, respectively. In words this equation means that the propulsion force (F_p) must be as great as the sum of the other forces in order to maintain a constant speed. To obtain the power needed to provide the necessary propulsion force the force balance (Equation 19) is multiplied by the bike speed. This yields

$$P(v_b) = \left(\frac{1}{0.95}\left(F_g + F_r + F_d(\vec{e}_p \cdot \vec{e}_d)\right)\right) v_b, \tag{20}$$

where $P(v_b)$ denotes the power needed to drive at speed v_b.

2.1.1 Maximum Achievable Speed

According to EU law, an e-bike can have an electro motor with a nominal power output of at most 250W and cannot provide power whilst driving more than 25 km/h[1]. For this reason, we assume in this thesis that the speed of the e-bike is such that the maximum power output is used, with a maximum of 25 km/h. The driver is assumed to be able to consistently provide a power output of 100W. Thus the total available power is 350 W. See Fontaine 2021 for the effect of different driver output powers. Unfortunately, there is no method known to the author for directly computing the maximum achievable speed in this case. The speed of the bike, in km/h is thus

$$v_{max} = \min\left(25; \sup\{v_b | P(v_b) \leq 350\}\right). \tag{21}$$

This value will be computed using a simple binary search algorithm with a tolerance of 1×10^{-5} and a maximum number of iterations of 10000. All input parameters and constants are summarized in Table 1.

Table 1: Constants and their value.

Constant	Symbol	Value	Unit
Air density	ρ	1.18	kgm^{-3}
Coefficient of airdrag	C_d	1.18	-
Frontal Area	A	0.83	m^2
Coefficient of roll resistance	C_r	0.01	-
Gravitational acceleration	g	9.81	ms^{-1}
Maximum power	P_{max}	350	W
Maximum velocity	v_{max}	25	kmh^{-1}

2.2 Wind Distribution

Knowing the physics of a bike ride on a straight road, highlights the importance of the wind. Both the speed and direction are important. The modelling of these parameters is of great influence on the modelled travel times. In this thesis, the speed and direction are assumed to be uncorrelated.

For the wind speed, we need a distribution that only considers positive values as we do not allow negative wind speeds (which could also be modelled as a 180 degree wind direction shift). In Jung and Schindler 2019 an overview is given of many possible ways of modelling the wind speed. These are, amongst others, the log-normal distribution, the Weibull distribution and the Gamma distribution. The Gamma distribution is used here, because it takes two parameters that can be determined through the expected value and standard deviation of the wind speed. These are two parameters that we could get from a weather forecast.

The wind direction is modelled as a random variable, symmetrically distributed around the forecast value. The forecast value is thus also the expected value. In this thesis, the modelling of the wind direction is kept relatively simple and chosen to be a normal distribution. The standard deviation estimated by Joffre and Laurila 1988 proved to be small compared to measurement data from the Dutch meteorological institute (KNMI) [2]. In order to more properly test the effect of variable wind, the standard deviation for the wind direction is set to be somewhere in the middle at 0.1 radians.

Since the direction and speed are assumed to be independent, the two distributions for wind speed and direction can be multiplied to obtain the joint distribution. The random variable ω represents the wind. The entire space of possible winds is called $\Omega = \{(\omega_s, \omega_d) | \omega_s \in \mathbb{R}_{\geq 0}, \omega_d \in [0, 2\pi)\}$. The probability density functions of the speed and direction are respectively

$$\omega_s \sim Gamma\left(k, \theta\right) \tag{22}$$

$$\omega_d \sim N\left(\mu, \sigma\right), \tag{23}$$

[1] https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32013R0168, consulted on september 25th, 2023.

[2] accessible via `https://dataplatform.knmi.nl/dataset/actuele10mindataknmistations-2`

where k, θ, μ, σ are chosen such that the expected values of the distributions coincide with the weather forecast and the standard deviations coincide with a chosen value. This leads to the density function

$$pdf_\omega = f_{\omega_s}(x) * f_{\omega_d}(y) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2}, \tag{24}$$

Any sample that is used to construct the travel time distribution is based on a sample from this distribution. In general, a single sample from this distribution will be called ω, consisting of two components, the wind speed (ω_s) and wind direction (ω_d). An example of the wind distribution can be seen in Figure 1.

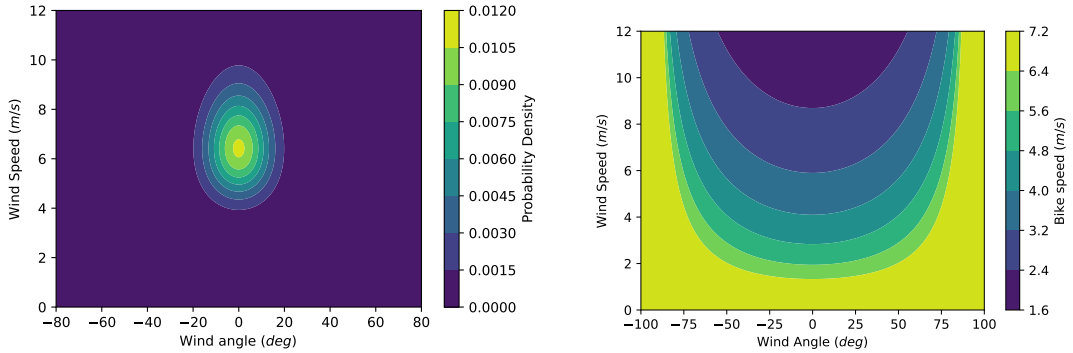


Figure 1: (left) Probability density function of wind, assuming wind speed is gamma distributed with $\mathbb{E}[\omega_s] = 6.75m/s$ and $\sigma_{\omega_s} = 1.5m/s$ and direction is normally distributed with $\mathbb{E}[\omega_d] = 0deg$ and $\sigma_{\omega_d} = 10deg$. (right) Maximum driving speed for different wind speeds and relative directions. Wind direction 0 corresponds with a perfect head wind.

2.3 Travel Time Samples

Once the wind distribution is fixed and the interaction of wind and bike is established, travel times can be computed for each circumstance. In Table 1 the constants and their values are summarized. These values will not be changed throughout this thesis. From the wind distribution, samples are drawn using Python. For this, the package 'random' is used with as seed 41. For each arc in the graph, i.e. road in the network, the set of travel times corresponding to the samples of the wind distribution are computed. These form a new set for each arc, which will also be called samples. The entire set of samples are used for an approximation of the travel time distribution. A graphical representation of the dataset can be seen in Table 2, each row refers to one road (i, j) with a certain load level (l), while the columns refer to the M wind samples. This will be done using a kernel density technique. This technique will be further explained in the Section 3.

Table 2: Data set containing M wind samples for each arc $(i, j) \in E$ and load level $l \in 1, ..., L$. Each sample from D_ω determines an entire column of this dataset.

arc	1	...	m	...	M
$(0,1,1)$	$TT_{(0,1,1)}(\omega_1)$		$TT_{(0,1,1)}(\omega_m)$		$TT_{(0,1,1)}(\omega_M)$
$\vdots$			$\vdots$		
(i,j,l)	$TT_{(i,j,l)}(\omega_1)$	...	$TT_{(i,j,l)}(\omega_m)$	...	$TT_{(i,j,l)}(\omega_M)$
$\vdots$			$\vdots$		
$(n,n-1,L)$	$TT_{(n,n-1,L)}(\omega_1)$		$TT_{(n,n-1,L)}(\omega_m)$		$TT_{(n,n-1,L)}(\omega_M)$

3 Approximating the Arrival Time Distribution

The samples that were obtained through the previously described method will be used to solve the SVRPTW. For this, it is needed to compute an approximation of the travel and arrival time distributions. These distributions will be estimated using the technique of kernel density estimation (KDE). In this chapter, it is first explained generally how this method works and what can be done to improve solution quality. Then the general form of the arrival time distribution is derived.

The technique of KDE was already mentioned by Parzen 1962 and is sometimes also referred to as Parzen's window. The method is very useful for visualizing data with more detail than using e.g. histograms, because it can give a smoother view of the approximated distribution. This provides more insight on the locations of modes of a density. This smoothing comes at the price of selecting the right smoothing parameter. If this is too large, the density will be over smoothed and all relevant details are lost. If the smoothing parameter is chosen too small, all individual samples will be recognizable, creating false modes. The technique of KDE, can also be used to obtain smooth approximations to unknown densities, without the prior assumption of some parametric distribution. In this thesis, KDE is used for non parametric approximation of the arrival time distribution. For further reading see e.g. Sheather 2004 and Scott 2015.

The KDE-method uses a sample from the unknown distribution to create an approximation of the real distribution. These samples are sometimes also called observations. Assigning a probability density to each 'observation' and summing them up creates a new density function. This can be thought of as 'dropping a pile of sand' at each observation. This has the effect that instead of assigning a probability density to the exact observation, we assign a probability density to a region around the observation. Thereby assuming the actual density is at least somewhat smooth. Intuitively, this makes sense as we might just as well sampled a value nearby the actually sampled value. After normalization, we then have a new probability density, which approximates the true density. The function that determines the shape of this 'pile of sand' is called the kernel function K. The properties of this function determine to a great extend the properties of the resulting distribution approximation. The kernel function, is generally defined such that it has the following characteristics

(K1) K is a probability density i.e. $\int_{-\infty}^{\infty} K(u)du = 1$ and $K(u) \geq 0 : \forall u$,

(K2) K is symmetric around zero i.e. $K(u) = K(-u)$,

(K3) K has compact support, or equivalently $K(u) = 0$ for $|u| > 1$.

Some examples of well-known kernels are triangular, uniform, Epanechnikov and normal kernel (although the latter does not satisfy (K3), it can still be used).

The general form of a kernel density estimate is

$$\tilde{f}_h(x) = \frac{1}{M} \sum_{m=1}^{M} K_h(x - X_m), \tag{25}$$

where $\{X_m\}_{m=(1,\dots,M)}$ is the set of samples and

$$K_h(u) = \begin{cases} 0 & \text{if } |u| > h \\ \frac{1}{h} K\left(\frac{u}{h}\right) & \text{otherwise} \end{cases} \tag{26}$$

is the h-scaled kernel function, defined such that it satisfies (K1)-(K3). The variable h, called the bandwidth or scaling parameter, determines the width of the 'pile of sand'. This determines the width of the kernel function. The resulting function then satisfies $\tilde{f}_h \geq 0$, because all kernels are always nonnegative, and since

$$\int_{-\infty}^{\infty} \tilde{f}_h(x)\, dx = \int_{-\infty}^{\infty} \frac{1}{M} \sum_{m=1}^{M} K_h(x - X_m)dx = \frac{1}{M} \sum_{m=1}^{M} \int_{-\infty}^{\infty} K_h(x - X_m)dx = \frac{1}{M} M * 1 = 1, \tag{27}$$

it is thus a well-defined probability density.

Depending on the chosen kernel, the corresponding cdf can be computed by

$$\tilde{F}_h(x) = \int_{-\infty}^{x} \frac{1}{M} \sum_{m=1}^{M} \frac{1}{h} K\left(\frac{y - X_m}{h}\right) dy. \tag{28}$$

Two questions that are now obvious are what kernel should be used and what should be chosen for the bandwidth h. Does it matter?

3.1 The Kernel

As the name suggest, the kernel plays a central role in kernel density estimation. A reasonable question to ask is what is the best choice for a kernel? A frequently chosen simple choice for a kernel is the uniform kernel, defined by

$$K_h(u) = \begin{cases} 1 & \text{if } |u| \leq h \\ 0 & \text{otherwise} \end{cases}. \tag{29}$$

This kernel essentially defines a region around the observation and gives all points in that region the same density, hence the name uniform. The benefit of using the kernel is that it is very easy to understand and easy to work with. A problem is that this density is not continuous over the entire domain and does not result in a very smooth curve. The resulting approximated density looks more like a version of a histogram.

Another very typical choice is the normal kernel, also called Gaussian kernel, which is familiar to many people because of its frequent use in statistics. It is given by

$$K_h(u) = \frac{1}{h\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{u}{h}\right)^2}. \tag{30}$$

It's familiar shape and properties can be a benefit. However, since the normal kernel does not satisfy (K3), it does not give a localized probability density, but actually assigns a density value to every point in $\mathbb{R}$. Moreover, no known closed form expression exists for the integral of this kernel.

A lesser known kernel is the Epanechnikov kernel, defined by

$$K_h(u) = \begin{cases} \frac{3}{4}(1 - u^2) & \text{if } |u| \leq 1 \\ 0 & \text{otherwise} \end{cases}. \tag{31}$$

This kernel is often called the most efficient. It means that fewer observations are needed with respect to other kernels in order to obtain the same error bound. The error bound that is used for this is the asymptotic mean integrated squared error (AMISE), see Section 3.2.1 for a derivation.

Figure 2 shows the shape and integral of these kernel functions. It turns out that in practice the choice of kernel has relatively little impact on the error made in the approximation of the density, see e.g. Section 6.2.3 in Scott 2015. More important than the choice of kernel is the bandwidth, which is discussed in the next section.

3.2 The Bandwidth

In contrast to the kernel, the bandwidth has a much bigger influence on the theoretical convergence properties of the kernel density method. The visual influence is also very noticeable. When the bandwidth is too small, the function has many modes and general conclusions are hard to draw from the resulting density approximation. However, when the bandwidth is too large all relevant information is smoothed out into a situation where we can also not extract any useful knowledge. This idea is demonstrated in Figure 3. Most of the research on kernel density estimation is done on properly choosing the bandwidth. Several authors proposed rules of thumb based on the assumption that the

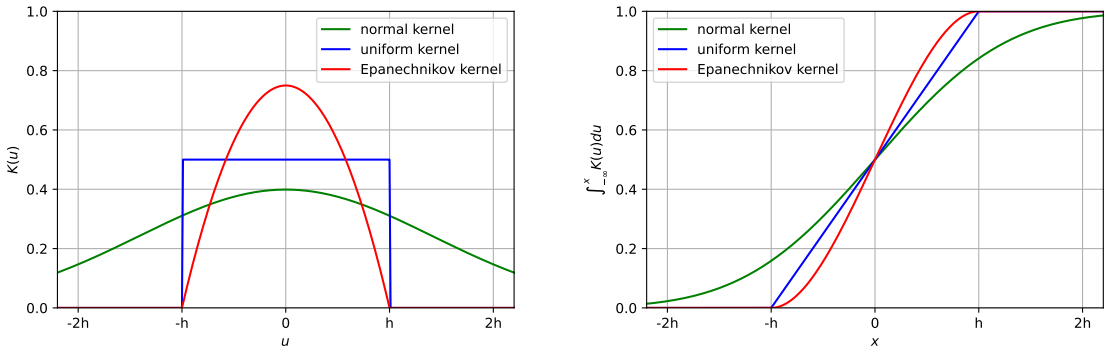


Figure 2: On the left the shape of different kernel functions is shown, and on the right the integral of different kernel functions is shown.

underlying density is a Gaussian. Silverman 1986 proposed the following bandwidth as a rule of thumb (Silvermans' rule of thumb)

$$h_{SROT} = 0.9An^{-1/5}, \tag{32}$$

where $A = \min \text{standard deviation}, \text{interquartile range}/1.34$. However, the author recognizes that although this works very well if the underlying density is indeed close to a Gaussian, it tends to oversmooth when the underlying density is multimodal. Another common way to choose the bandwidth is choosing that value that minimizes a certain error measure on the estimated density. Often, the error measure that is used for this is the asymptotic mean integrated square error (AMISE). First, it is shown how to write AMISE in terms of powers of the bandwidth h. Next, it is shown how this leads to the optimal value for h.

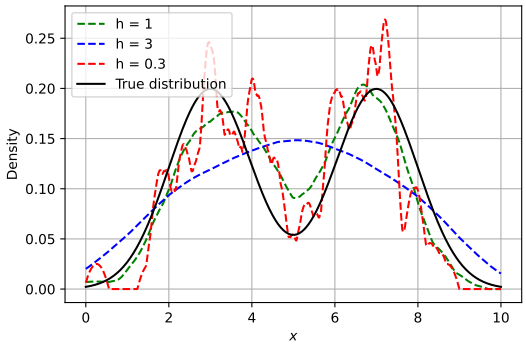


Figure 3: A plot of a bimodal normal density approximated by the KDE technique using a Epanechnikov kernel and three different bandwidths with 100 samples.

3.2.1 AMISE

In order to properly derive the AMISE (asymptotic mean integrated square error), the MSE (mean square error) is defined first. The MSE is defined for an underlying density $f(x)$ and an approximation to that density $\tilde{f}_h(x)$

$$MSE(\tilde{f}_h(x)) = \mathbb{E}\left[(\tilde{f}_h(x) - f(x))^2\right]. \tag{33}$$

The expected value here is taken over the samples that could have been drawn from the original distribution, assuming these samples are i.i.d.. This can be written as

$$MSE(\tilde{f}_h(x)) = \mathbb{E}\left[(\tilde{f}_h(x) - f(x))^2\right] \tag{34}$$

$$= \mathbb{E}\left[\tilde{f}_h(x)^2\right] - \mathbb{E}\left[\tilde{f}_h(x)\right]^2 + \mathbb{E}\left[\tilde{f}_h(x)\right]^2 - \mathbb{E}\left[2\tilde{f}_h(x)f(x))\right] + \mathbb{E}\left[f(x)^2\right] \tag{35}$$

$$= \left(\mathbb{E}\left[\tilde{f}_h(x)^2\right] - \mathbb{E}\left[\tilde{f}_h(x)\right]^2\right) + \left(\mathbb{E}\left[\tilde{f}_h(x)\right] - f(x)\right)^2, \tag{36}$$

where it is used that $\mathbb{E}\left[f(x)\right] = f(x)$. This is often called the variance bias trade off and is then written as

$$MSE(\tilde{f}_h(x)) = \mathrm{Var}\left\{\tilde{f}_h(x)\right\} + \mathrm{Bias}^2\left\{\tilde{f}_h(x)\right\}. \tag{37}$$

First the bias is derived. In order to do this, the linearity and integral definition of the expected value are used. For a set of samples ${X^m}_{m\in(1,\dots,M)}$ and KDE approximation $\tilde{f}_h(x) = \frac{1}{M}\sum_{m=1}^{M} K_h(x - X^m)$ this is given by

$$\mathbb{E}\left[\tilde{f}_h(x)\right] = \mathbb{E}\left[\frac{1}{M}\sum_{m=1}^{M} K_h(x - X_m)\right] \tag{38}$$

$$= \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}\left[K_h(x - X_m)\right] \tag{39}$$

$$= \mathbb{E}\left[K_h(x - X_i)\right] \tag{40}$$

$$= \int_{-\infty}^{\infty} K_h(x - y)f(y)dy. \tag{41}$$

By writing $K_h(x-y) = \frac{1}{h}K(\frac{x-y}{h})$, doing the coordinate transform $y = x - hz$ and $dy = -hdz$, and using a Taylor series this becomes

$$\mathbb{E}\left[\tilde{f}_h(x)\right] = \int K_h(x - y)f(y)dy \tag{42}$$

$$= \int \frac{1}{h}K(z)f(x - hz)hdz \tag{43}$$

$$= \int K(z)\left\{f(x) + \sum_{n=1}^{\infty}\frac{(-hz)^n}{n!}f^{(n)}(x)\right\}dz \tag{44}$$

$$= f(x)\int K(z)dz - hf'(x)\int zK(z)dz + \frac{h^2}{2}f''(x)\int z^2K(z)dz \tag{45}$$

$$- \frac{h^3}{6}f'''(x)\int z^3K(z)dz + \mathcal{O}(h^4) \tag{46}$$

Using that the kernel function is symmetric around $x = 0$ (characteristic (K2) from the definition). This means that the integral $\int z^nK(z)dz = 0$ whenever n is odd. Using this and characteristic (K1), i.e. $\int K(z)dz = 1$, this yields

$$\mathbb{E}\left[\tilde{f}_h(x)\right] = f(x) + \frac{h^2}{2}\int z^2K(z)dz + \mathcal{O}(h^4) \tag{47}$$

$$= f(x) + \frac{h^2}{2}\sigma_K^2 + \mathcal{O}(h^4), \tag{48}$$

with definition $\sigma_K^2 = \int z^2K(z)dz$ of the second moment of K to write the last line. This means that the squared bias takes the form

$$\mathrm{Bias}^2\left\{\tilde{f}_h(x)\right\} = \left(\mathbb{E}\left[\tilde{f}_h(x)\right] - f(x)\right)^2 = \left(\frac{h^2}{2}\int z^2K(z)dz + \mathcal{O}(h^4)\right)^2 = \frac{h^4}{4}f''(x)^2\sigma_K^4 + \mathcal{O}(h^6) \tag{49}$$

For the variance some basic manipulations must be done first, after which the full expression can be derived.

$$\text{var}\left\{\tilde{f}_h(x)\right\} = \text{var}\left\{\frac{1}{M}\sum_{i=1}^{M} K_h(x - X_i)\right\} \tag{50}$$

$$= \frac{1}{M^2}\text{var}\left\{\sum_{i=1}^{M} K_h(x - X_i)\right\}. \tag{51}$$

$$\tag{52}$$

Because the samples X^m are i.i.d, the $K_h(x - X_i)$ are as well such that the variance of their sum is the sum of their variance. This becomes

$$= \frac{1}{M^2} M\text{var}\left\{K_h(x - X_i)\right\} \tag{53}$$

$$= \frac{1}{M}\text{var}\left\{K_h(x - X_i)\right\}. \tag{54}$$

Using the definition of the variance, this is written

$$\text{var}\left\{K_h(x - X_i)\right\} = \mathbb{E}\left[K_h^2(x - X_i)\right] - \mathbb{E}\left[K_h(x - X_i)\right]^2. \tag{55}$$

Using the integral definition of the expected value and again a coordinate transformation and Taylor series, the first term can be written as

$$\mathbb{E}\left[K_h^2(x - X_i)\right] = \int K_h^2(x - y) f(y) dy \tag{56}$$

$$= \frac{1}{h}\int K^2(z) f(x - hz) dz \tag{57}$$

$$= \frac{1}{h}\int K^2(z)\left\{f(x) + \sum_{n=1}^{\infty}\frac{(-hz)^n}{n!} f^{(n)}(x)\right\} dz \tag{58}$$

$$= \frac{1}{h} f(x)\int K^2(z) dz - f'(x)\int z K^2(z) dz + \mathcal{O}(h). \tag{59}$$

The second term in Equation 55 can be obtained by following the derivation of the bias from line 40 and on. These can be combined, keeping only the leading order, to obtain the variance, yielding

$$\text{var}\left\{K_h(x - X_i)\right\} = \frac{1}{h} f(x)\int K^2(z) dz - f'(x)\int z K^2(z) dz - f(x)^2 + \mathcal{O}(h). \tag{60}$$

Equations 60 and 49 can be combined to get the final expression of the mean integrated square error

$$MSE(\tilde{f}_h(x)) = \text{Var}\left\{\tilde{f}_h(x)\right\} + \text{Bias}^2\left\{\tilde{f}_h(x)\right\} \tag{61}$$

$$= \frac{1}{Mh} f(x)\int K^2(z) dz - \frac{1}{M} f'(x)\int z K^2(z) dz \tag{62}$$

$$- \frac{f(x)^2}{M} + \frac{h^4}{4}\left(f''(x)\right)^2 \sigma_K^4 + \mathcal{O}(\frac{h}{M}) + \mathcal{O}(h^2) \tag{63}$$

The MSE, however, only gives a measure of the local error, namely at x. However in general the interest is in the error over the entire function. A way to obtain this global error is by taking the integral over the entire function. Writing $R(g) = \int_{-\infty}^{\infty} g^2 dx$, this yields the integrated mean squared

error (IMSE)

$$IMSE(\tilde{f}_h) = \int_{-\infty}^{\infty} MSE(\tilde{f}_h(x))dx \tag{64}$$

$$= \frac{1}{Mh}\int K^2(z)dz \int f(x)dx - \frac{1}{M}\int f'(x)dx \int zK^2(z)dz - \int \frac{f(x)^2}{M}dx \tag{65}$$

$$+ \frac{h^4\sigma_K^4}{4}\int f''(x)^2dx + \mathcal{O}(\frac{h}{M}) + \mathcal{O}(h^5) \tag{66}$$

$$= \frac{1}{Mh}R(K) - \frac{R(f)}{M} + \frac{h^4}{4}R(f'')\sigma_K^4 + \mathcal{O}(\frac{h}{M}) + \mathcal{O}(h^5). \tag{67}$$

Using Fubini's theorem the order of the integral and the expected value can be switched such that

$$IMSE(\tilde{f}_h) = \int_{-\infty}^{\infty} \mathbb{E}\left[(\tilde{f}_h(x) - f(x))^2\right]dx = \mathbb{E}\left[\int_{-\infty}^{\infty}(\tilde{f}_h(x) - f(x))^2dx\right] = MISE(\tilde{f}_h), \tag{68}$$

which is called the mean integrated square error (MISE). The last equality follows from applying Fubini's theorem. Fubini's theorem states under what conditions it is possible to switch the order of integration (see e.g. Section 4.3 in Weir 1973). Since we are interested in the effect of h on this error, we can neglect the second term in line 67 for the AMISE. Finally, in order to get the asymptotic mean integrated square error (AMISE), the final expression is written with leading terms, only. This yields

$$AMISE(\tilde{f}_h) = \frac{1}{Mh}R(K) + \frac{h^4}{4}R(f'')\sigma_K^4. \tag{69}$$

The optimal bandwidth is than that value for h such that the AMISE is minimized.

$$h^*_{AMISE} = \text{argmin}_h AMISE(h) = M^{-1/5}\left[\frac{R(K)}{\sigma_k^4 R(f'')}\right]^{1/5}. \tag{70}$$

Note that the fraction $\frac{R(K)}{\sigma_K^4}$ depends entirely on the kernel K. The choice of kernel thus influences the AMISE bandwidth. The Epanechnikov kernel minimizes this fraction compared to other known Kernels. Therefor it is called the most efficient kernel.

3.2.2 Approximating the Optimal Bandwidth

In order to compute h^*_{AMISE} it is needed to compute the components $R(K)$, σ_k^4, $R(f'')$. Since $R(f'')$ depends on the actual density, which is unknown, it only makes sense to try and approximate this value. Notice how this value depends solely on the second derivative of the density function. The second derivative thus needs to be bounded. It is assumed, however, that the travel times are smoothly distributed across a range of values, thus no blow ups are expected. Moreover, from a try and fit process in experiments, the value $R(f'') \approx \frac{1}{40}$ turned out to be a reasonable value in practice. It is possible, however, to determine the value for $R(K)$ and σ_K precisely. For the Epanechnikov kernel these are $\frac{3}{5}$ and $\frac{1}{5}$, respectively. This leads to the formula

$$h^*_{AMISE} = \left[\frac{120}{M}\right]^{1/5} \approx 2.6M^{-1/5}, \tag{71}$$

for the case where $R(f'') \approx \frac{1}{40}$. It is also possible to try to approximate $R(f'')$ by assuming a density for f. An educated guess is that f is actually the gamma density. The asymmetric shape of this density does visually look like many of the travel time distributions. However, depending on the relative angle between the travel direction and forecast wind, the shape parameter can be larger or smaller than 1. Resulting in different shapes. If we assume the true density to be $Gamma(x; 0.3, \frac{0.3}{\mathrm{E}[TT]})$, the value for $R(f'') \approx 2 \times 10^{-4}$ yields

$$h^*_{gamma} = \left[\frac{15000}{M}\right]^{1/5} \approx 6.8M^{-1/5}. \tag{72}$$

Note, however, that since every road has a different orientation with respect to the wind, it also has a unique probability density associated to the travel time over that road. This means that every triplet (i, j, l) we could define a different bandwidth. This would require an analysis of each road-load level combination. For simplicity, this is not done here.

3.2.3 Interpretation

The interpretation of the bandwidth in the context of this thesis is closely related to the margin that is taken around the end of the time window. From the definition in Equation 26, it can be seen that whenever a sample is farther away from the observation than a time h, it is not counted in exactly the same way as would be the case in the empirical distribution. The additional effect of using a KDE technique instead of the empirical pdf is thus smoothing out the effect of points close to the point of interest. The value of h can thus also be defined, not by virtue of mathematical optimality, but by preference of the one that operates the algorithm. Such that if more margin is required, a larger value for the bandwidth can be selected.

3.3 Arrival Time Distribution

The arrival time of the delivery bike at the customer is an important quantity. It determines whether the set time windows are met. For the very first trip of a route, the arrival time distribution is equal to the travel time distribution (possibly shifted with the departure time). An example of two travel time distributions is given in Figure 4. However, as we go further along the route, possible waiting times influence the arrival time at later customers.

Defining t_{min} as the earliest time we could arrive. This is the arrival time when driving v_{max} as defined in Equation 21. $f_{AT}(t)$ is the density function that describes the probability density of the arrival time. In general, the arrival time distribution is defined on the half line $\Omega = [t_{min}, \infty)$, with t_{min} equal to the earliest possible arrival time, i.e. if we would have ridden the entire route with maximum velocity and takes the form

$$\mathbb{P}(AT = t) = \begin{cases} p & \text{if } t = t_{min} \\ f_{TT}(t) & \text{if } t > t_{min} \\ 0 & \text{otherwise} \end{cases} \tag{73}$$

Previous authors have used different techniques for estimating the arrival times. A widely used technique is approximating the arrival time as the maximum of a density defined on $\mathbb{R}_{\geq 0}$ and the earliest arrival time t_{min}, see e.g. Gutierrez et al. 2018, Ehmke, Campbell, and Urban 2014,Lent 2018. Another technique is discretization as used by Zhang, Lam, and B. Y. Chen 2013. In this thesis, however, the aim is to approximate the arrival time distribution by the kernel density technique. For this a representative sample is needed from the real distribution. Section 2 explains how these samples can be obtained. Once a set of samples of the travel times is drawn, it is also possible to obtain a set of samples for the arrival times. The simplest way to do this is by going through a route for each wind sample and computing the corresponding arrival times. Then, the KDE technique can be used to construct a density from those samples. The arrival times are determined iteratively as follows for a route r with customers $N_r = \{r_0 = 0, r_1, ...r_k\}$

$$AT_{r_i}(\omega) = s_{r_{i-1}} + TT_{(r_{i-1}, r_i, l)}(\omega) + \max\{a_{r_{i-1}}, AT_{r_{i-1}}\} \quad \forall r_i \in r \setminus 0, \tag{74}$$

where s_{i-1} is the fixed service time, $TT_{(i-1,i,l)}(\omega)$ is the travel time that is associated with wind ω and arc $(i-1, i, l)$ and a_{i-1} is the start of the time window of the previous customer.

In the objective function, however, the interested is in the probability of arriving before the end of the time window. It is thus also useful to define the cumulative density function

$$\mathbb{P}(AT \leq t) = \begin{cases} \mathbb{P}(AT = t_{min}) + \int_{t_{min}}^{t} f_{AT}(s)ds & \text{if } t \geq t_m in \\ 0 & \text{otherwise} \end{cases} \tag{75}$$

By iterating through the entire route, all arrival times can be determined. By doing this for a set of winds ω drawn from the distribution D_ω, a set of samples is obtained of the arrival times for each customer i. Then, an approximation of the probability of arriving before the end of the time window b_i can be computing by computing the integral

$$\tilde{\mathbb{P}}(AT \leq b_i) = \int_{-\infty}^{b_i} \frac{1}{M} \sum_{m=1}^{M} K_h \left(y - AT_i(\omega_m)\right) dy. \tag{76}$$

This integral will be numerically solved by the solution algorithm that is explained in the next section. The computation of this integral gives an approximation of the real probability of arriving before the end of the time window. This is used in the objective function.

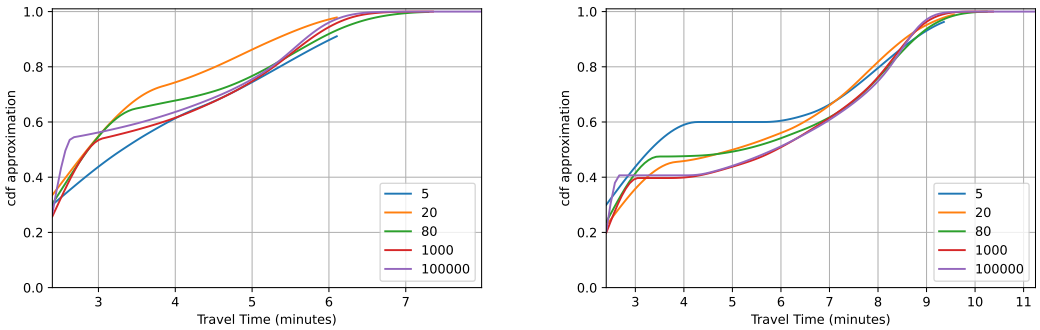


Figure 4: Two sets of travel time distributions computed for different numbers of samples from the unknown distribution, with the Epanechnikov kernel, bandwidth $h = 2.61n^{-1/5}$, and the same wind distribution as in Figure 1. On the left for load level 5, zero slope and a forecast head wind at 35 degrees and $6.75ms^{-1}$. On the right for load level 10, slope 0.03 and a forecast tailwind at 160 degrees and $6.75ms^{-1}$.

4 Solution Algorithm

The previously described technique for approximating a probability density can be utilized to obtain expected arrival times and probabilities of meeting the time windows. These approximations will be used in a solution algorithm in order to try and find a good solution to the VRP version that is considered here, i.e. with time windows and load and wind dependent travel times. As this problem is NP-hard, there is no guarantee that solutions are optimal. Moreover, although approximation algorithms for some versions of the VRP are known, there is no known bound on the solutions found to the SVRPTW using heuristics such as local search in combination with an ILP optimization. The method that is described here will be a local search heuristic. The quality of obtained schedules can thus only be measured against different heuristics.

In this thesis, an adaptation of the algorithm developed by Hesselmans 2022 is used. The algorithm consists of two phases. These are a local search phase and a linear programming phase. In the local search phase, the solution space is explored and a set of solutions is outputted. The solutions that are outputted are selected in a way such that it can be expected that these solutions are relatively good. In the linear programming phase, the solutions found in the local search phase are cut into separate routes, each with their own score. These routes are pieced together in the linear programming phase in a set cover formulation. In this phase, no new routes are explored, but only the combination of routes. The basic workings of the algorithm are explained here below, with references for further reading.

4.1 Local Search

The local search phase is based on simulated annealing. This method is based on the physical process of annealing, which is used to produce materials with fewer impurities by heating the material and slowly lower the temperature. By increasing the temperature, the material is allowed to take on less energy efficient states. This allows for overcoming local optima. By slowly lowering the temperature, the material in theory cools down into a more optimal state. By repeating the process several times, it is theorized that the material ends up in a (near) optimal state. See Brooks and Morgan 1995, Kirkpatrick, Gelatt Jr, and Vecchi 1983 and Dekkers and Aarts 1991 for further reading.

In the simulated annealing algorithm, the analogy for the physical state is a feasible solution. The analogy of the energy level of the state is the score that is associated to a schedule via the objective function. The algorithm makes changes on a random basis, by selecting a neighbourhood and making changes within that neighbourhood. This can be seen as an analogy to the random movements of particles in a material. If the algorithm finds a solution with lower score (lower energy in actual annealing), it always takes on that new solution (state). If no improvement is found, the temperature parameter determines the probability of the solution being accepted as new current solution via the Boltzmann function. The Boltzmann function is used in statistical physics to describes the probability of the material being in a certain energy state. As a higher temperature in annealing allows higher energy states, a higher temperature parameter increases the chance of accepting a worse solution. As the algorithm advances, the temperature parameter is slowly lowered, pushing the solutions score (energy) towards a minimum. By repeating the process, the algorithm explores the solution space and finds a collection of schedules that are relatively good. For these schedules, the routes that make up the schedule are individually stored, together with a recomputed objective value. A single iteration of the local search is as follows

1. Select a neighbourhood
2. Compute new objective value
3. Accept or discard new solution
4. Save improvements for ILP
5. Go back to 1

The pseudo code for computing the new objective value is as given in Algorithm 1. Every time the new objective value needs to be calculated, this algorithm is invoked. It starts by setting the initial value to zero. Then loops over the set of samples from the wind distribution. For each sample, the objective value for the current route and sample is computed. These are then averaged to compute the approximation of the expected value of the stochastic objective value.

As mentioned hereabove, every solution that improves the objective value is saved. This is done by recomputing the objective value and creating a copy of the customer id list. The recomputing of the objective value is necessary, because the algorithm might use the temperature parameter during the search to soften certain constraints. This allows for broader search through the solution space, but the effect is that the same route might have different scores for different temperatures. The score is therefor recomputed using the initial temperature. When the objective value found for a new schedule is not better than the current score, the Boltzmann distribution

$$\mathbb{P}(x = E) \propto e^{-\frac{E}{T}}, \tag{77}$$

with T the temperature, is used to determine whether or not the new schedule is accepted. See S. J. Blundell and K. M. Blundell 2006 for a derivation. The energy is replaced by difference between the current and the new solution. This results in a number between zero and one. Then, a random number in the interval is chosen. If it is lower, then the new solution is accepted, otherwise it is discarded.

Algorithm 1 Computing improvement of objective function for a route r.

```
NewScore = 0
for 1 ≤ m ≤ M do
    Compute F(r, ω_m)
    NewScore = NewScore + (1/M) F(r, ω_m)
end for
Improvement = PreviousScore - NewScore
```

4.2 Neighbourhoods

An important part of the local search algorithm is selecting a new schedule from the current one. This is done via the definition of neighbourhoods. A neighbourhood is defined by a simple action that can be applied to a solution and yields a new solution. The new solution needs to meet the hard constraints. These are the capacity constraints and the meeting of the demand of all customers. However, some soft constraints, such as the meeting of the time windows can be allowed to be violated at a cost in the objective function.

The previous work of Hesselmans 2022 selected a set of six neighbourhoods from a larger set. These neighbourhoods proved to work well for the deterministic vehicle routing problem with time windows and load dependent travel times (VRPTWLTT). For this reason, the same neighbourhoods are used for the stochastic version. These are the following

1. Move random customer to random customer;
   Selects a random customer and moves it to a random location in the schedule. This can be either the same route or any other route.

2. Swap random customers;
   Randomly selects two customers and swaps them. The rest of the routes remains the same.

3. Reverse a subroute;
   Selects a subroute from a route and reverses the order in which the customers are traversed.

4. Scramble a subroute;
   Selects a subroute from a route and scrambles the customers into a new random order.

5. Move random customer to best position in some different route;
   Selects a random customer and a random route that the customer is not currently in. The customer is inserted into this new route in its best possible position. The best location is determined by trying all locations and computing the corresponding objective value.

6. Move random customer to best position in any other route;
   Selects a random customer and move it to the best position in any other route than its currently in. Thus all locations outside the route it is currently in are checked for the new objective value. The position that results in the lowest objective function is chosen.

In each iteration, the new neighbourhood is selected from this list. This is done randomly, although operators can be assigned different weights, allowing some operators to be more likely than others. This way, the more time consuming operators can be reduced in frequency if these seriously harm the running time, without creating good improvements.

4.3 Computing the Objective Function

After selection of a neighbourhood, the decision needs to be made whether or not we keep that solution. In order to properly do this, the change in objective function is computed. Computing this value requires partially recalculating the objective function, i.e. Equation 1. The basic idea is to split the computation of the objective value into smaller parts. The objective function can first of all be split into a term for each route/bike. By assumption, the cargo bikes do not interfere with each other in any way. It thus suffices to only recalculate the objective value of the routes. That were altered by the application of the selected neighbourhood. The route-specific objective function can moreover be split in three parts that can be computed separately and added. For this purpose, define a route r with customers $N_r = (r_0 = 0, r_1, r_2, ..., r_k, r =_{k+1}= 0)$ and edges $E_r = \{(r_i, r_{i+1}) | i \in N_r\}$, consisting of k customers and starting and ending at the depot. The route specific objective value function can then be written

$$\mathbb{E}_{\omega,r}\left[\sum_{(i,j)\in E_r, l\in L} TT^l_{i,j} z^l_{i,j} + \gamma_1 \sum_{j\in N_r} WT_j + \gamma_2 \sum_{j\in N_r} U_j\right],$$

where the linearity of the expected value is used to switch the sum and expected value. The rest of this section explains how, given a set of samples, an approximation of the objective value can be computed.

The idea is to split the objective function for a route r into three parts, namely the expected total travel time (ETT_r), the expected total waiting time (EWT_r) and the expected service level (ESL_r). This yields

$$ETT_r = \mathbb{E}_\omega\left[\sum_{(i,j)\in E_r, l\in L} TT^l_{i,j} z^l_{i,j}\right], \tag{78}$$

$$EWT_r = \mathbb{E}_\omega\left[\sum_{j\in N_r} WT_j\right], \tag{79}$$

$$ESL_r = \mathbb{E}_\omega\left[\sum_{j\in N_r} U_j\right]. \tag{80}$$

These values are estimated through the travel time density that is approximated via the kernel density technique. Recall, that for a set of samples $\{X^m_{(i,j,l)}\}_{m\in\{1,...,M\}}$ of the travel time over an edge (i, j, l), the corresponding density is estimated by

$$\tilde{f}_{(i,j,l)}(x) = \frac{1}{M}\sum_{m=1}^{M} K_h(x - X^m_{(i,j,l)}). \tag{81}$$

The estimator based on the set of observations is denoted with a tilde ($\tilde{}$).

4.3.1 Expected Total Travel Time

The expected total travel time might be the easiest to compute. This is merely a linear combination of expected values of the travel times. The to be estimated term is

$$ETT_r = \mathbb{E}\left[\sum_{(i,j)\in E_r, l\in L} TT^l_{i,j} z^l_{i,j}\right] \tag{82}$$

$$= \sum_{(i,j)\in E_r, l\in L} \mathbb{E}\left[TT^l_{i,j}\right] z^l_{i,j} \tag{83}$$

By replacing the expected value of the travel time by the expected value of the density approximation, the expected total travel time can be approximated, this yields

$$E\tilde{T}T_r = \sum_{(i,j)\in E_r, l\in L} \mathbb{E}\left[\tilde{TT}_{(i,j,l)(x)}\right] z^l_{i,j} \tag{84}$$

$$= \sum_{(i,j)\in E_r, l\in L} \mathbb{E}\left[\tilde{f}_{(i,j,l)(x)}\right] z^l_{i,j} \tag{85}$$

$$= \sum_{(i,j)\in E_r, l\in L} \mathbb{E}\left[\frac{1}{M}\sum_{m=1}^{M} K_h(x - X^m_{(i,j,l)})\right] z^l_{i,j} \tag{86}$$

$$= \sum_{(i,j)\in E_r, l\in L} \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}\left[K_h(x - X^m_{(i,j,l)})\right] z^l_{i,j} \tag{87}$$

$$= \sum_{(i,j)\in E_r, l\in L} \frac{1}{M}\sum_{m=1}^{M} \left(\int_{-\infty}^{\infty} xK_h(x - X^m_{(i,j,l)})dx\right) z^l_{i,j} \tag{88}$$

$$= \sum_{(i,j)\in E_r, l\in L} \frac{1}{M}\sum_{m=1}^{M} \left[X^m_{(i,j,l)}\right] z^l_{i,j}, \tag{89}$$

where the last line follows from the definition of the kernel function. From this equation it can be concluded that whenever the travel times are time-independent, the total travel time is only dependent on the selection of arcs (i, j, l) and their travel times. Intuitively, this makes sense, since time-independence implies that the time it takes to traverse an arc only depends on the load and the road itself. The other two, EWT_r and ESL_r, however, do depend on the time windows as well and thus require a different treatment.

4.3.2 Expected Waiting Time

The waiting time depends on both the arrival time and the start of the time window and is described by the waiting time relation $WT_j = \max\{0, a_j - AT_j\}$. The time windows are considered fixed in this version of the problem, so only the arrival times are stochastic. There are two main strategies to compute this. The first is to determine the expected waiting time for each individual customer and adding these up to obtain the expected total waiting time. The second strategy is to compute the total waiting time along a route given a set of samples and averaging this across all sets of samples.

The to be estimated term is

$$EWT_r = \mathbb{E}_\omega\left[\sum_{j\in N_r} WT_j\right], \tag{90}$$

$$= \mathbb{E}_\omega\left[\sum_{j\in N_r} \max\{0, a_j - AT_j\}\right]. \tag{91}$$

The first strategy is to first compute the expected waiting time at each customer. In order to do thus, a set of samples for the individual waiting time is constructed. As the weather is assumed to be constant throughout the day, this can be done as follows. Assume a route $r = (r_0 = 0, r_1, r_2, ..., r_k, r_{k+1} = 0)$. Using a set of samples $\{X^m_{(i,j,l)}\}_{m\in(1,...,M)}$ for the travel time over the arc (i, j, l), the waiting time $Y^m_{r_j}$ at the j^{th} customer given a weather ω_m, is given by

$$\begin{aligned} Y^m_{r_j} &= \max\{0, a_{r_j} - AT_{r_j}(\omega_m)\}, && (92)\\ &= \max\{0, a_{r_j} - X^m_{(r_{j-1},r_j,l)} - s_{r_{j-1}} - \max\{0, a_{r_{j-1}} - AT_{r_{j-1}}(\omega_m)\}\}, && (93)\\ &= \max\{0, a_{r_j} - X^m_{(r_{j-1},r_j,l)} - s_{r_{j-1}} - \max\{0, a_{rj-1} - X^m_{(r_{j-1},r_j,l)} - s_{r_{j-1}} - \max\{0, ...\}\}\}. && (94)\end{aligned}$$

This recursive relation allows for computing the individual expected waiting time at each customer. Using the tilde to denote the sample approximated value, the approximated expected total waiting time is then given by the equation

$$E\tilde{W}T_r = \sum_{r_j\in N_r} \frac{1}{M} \sum_{m=1}^{M} Y^m_{r_j} \tag{95}$$

The second way of computing this value does not involve computing the expected waiting time at each customer. Instead, the method makes use of the fact that only the expected value of the <u>total</u> waiting time is of interest. The waiting time at each individual, thus, does not matter. This can be exploited and for the total waiting time (TWT^m_r) for a route $r = (0, r_1, r_2, ..., r_k, 0)$ under weather ω_m this yields

$$TWT^m_r = \sum_{(r_i,r_j)\in E_r, l\in L} WT_{r_j} z^l_{r_i,r_j} = \max_{1\le j'\le k} \left\{ 0, a_{j'} - \sum_{i=1}^{j'-1} s_{r_i} - \sum_{i=0}^{j'-1} TT^m_{i,i+1,l} \right\}. \tag{96}$$

The intuition is given here, for the proof see Appendix B. The intuition of this formula is the following. At any given moment since our departure, the driver has done one of three things; either they were driving, they were delivering service, or they were waiting. The total waiting time at any moment is thus the past time, minus the past service time, minus the past travel time. Generally it is not known how long the driver was driving exactly, as at any given time the driver might be traversing an arc. However, upon arrival, when all previous arcs have been traversed and the service has not yet started, the past travel time and past service time are known. The evaluation is thus done at the start of the time windows. Whenever the driver has waited at that customer, they have already arrived, but not yet carried out the delivery. This allows for computing the total waiting time up to this point. The expected total waiting time is the average of these TWT^m_r, i.e.

$$E\tilde{W}T_r = \frac{1}{M} \sum_{m=1}^{M} TWT^m_r. \tag{97}$$

4.3.3 Expected Service Level

The expected service level is given by

$$\begin{aligned} ESL_r &= \mathbb{E}_\omega \left[\sum_{j\in N_r} U_j \right] && (98)\\ &= \sum_{j\in N_r} \mathbb{E}_\omega \left[U_j \right], && (99)\end{aligned}$$

where the linearity of the expected value is used. The value of $\mathbb{E}_\omega [U_j]$ can be interpreted as follows

$$\mathbb{E}_\omega [U_j] = \mathbb{E}_\omega \left[1 - \mathbb{1}_{AT_j\le b_j} \right] = \mathbb{E}_\omega \left[\mathbb{1}_{AT_j\ge b_j} \right] = 1*\mathbb{P}(AT_j \ge b_j) + 0*\mathbb{P}(AT_j < b_j) = \mathbb{P}(AT_j \ge b_j). \tag{100}$$

This probability will be approximated by computing an approximation to the cumulative arrival time distribution. To this end consider the approximated arrival time distribution $\tilde{AT}_j$, defined by

$$\tilde{AT}_j(x) = \frac{1}{M} \sum_{m=1}^{M} K_h \left(x - AT_j(\omega_m) \right), \tag{101}$$

where $\{AT_j(\omega_m)\}_{m \in (0,\ldots,M}$ is the set of arrival time samples corresponding to the M wind samples. These can be iteratively computed using Equation 74. The arrival time approximation based on the samples is denoted $\tilde{AT}_j(x)$. The probability of arriving late is computed by

$$\mathbb{P}(AT_j \geq b_j) = 1 - \mathbb{P}(AT_j \leq b_j) = 1 - \int_{-\infty}^{b_j} \tilde{AT}_j(x) dx, \tag{102}$$

where b_j is the end of the time window of the customer at whom the driver arrives. The integral can be rewritten to

$$\int_{-\infty}^{b_j} \frac{1}{M} \sum_{m=1}^{M} K_h \left(x - AT_j(\omega_m) \right) dx = \frac{1}{M} \sum_{m=1}^{M} \int_{-\infty}^{b_j} K_h \left(x - AT_j(\omega_m) \right) dx. \tag{103}$$

It thus suffices to find the general integral of K_h. In the case of the Epanechnikov kernel this yields

$$\int_{-\infty}^{b_j} K_h \left(x - AT_j(\omega_m) \right) dx = \begin{cases} 0 & \text{if } b - AT_j(\omega_m) < h \\ 1 & \text{if } b - AT_j(\omega_m) > h \\ \frac{1}{2} + \frac{3}{4}u - \frac{1}{4}u^3 \text{ where } u = \frac{b - AT_j(\omega_m)}{h} & \text{otherwise} \end{cases} \tag{104}$$

The final expression for the approximated expected service level for a route is thus

$$E\tilde{S}L_r = \frac{1}{M} \sum_{m=1}^{M} \min \left\{ 1,\ \max \left[0,\ \frac{1}{2} + \frac{3}{4} \left(\frac{b - AT_j(\omega_m)}{h} \right) - \frac{1}{4} \left(\frac{b - AT_j(\omega_m)}{h} \right)^3 \right] \right\} \tag{105}$$

4.4 Integer Linear Program

The routes that are selected during the local search phase of the algorithm are passed on to the integer linear program phase (ILP phase). The previously described method for evaluating the objective function is used to recompute a value for each route, to make sure mistakes during the local search phase are not propagated into the ILP phase.

The ILP phase of the algorithm is modeled a set covering formulation. This part of the algorithm is entirely based on the code of Hesselmans 2022 and uses a Gurobi solver [3]. Each set consists of one route that was found by the SA phase, together with its score. The ILP tries to find the minimum cost cover that covers all customers exactly once via a branch and cut method. Below, the mathematical formulation is given. Not all constraints that the final schedule must satisfy are present. This is because many constraints have already been checked in the local search phase and cannot be violated here such as the vehicle capacity. The set containing all routes passed on by the local search phase is denoted by R.

$$\text{minimize} \quad \sum_{r \in R} x_r c_r \tag{106}$$

$$\text{subject to}$$

$$\sum_{r \in R} x_r a_{r,i} = 1 \quad \forall i \in C \tag{107}$$

$$\sum_{r \in R} x_r \leq m \tag{108}$$

$$x_r \in \{0, 1\} \quad \forall i \in C, r \in R, \tag{109}$$

[3] This is a commercial solver, thus a licence is needed to run this part of the code.

where x_r is a decision variable that is equal to one if route r is used in the final solution, c_r is the cost of taking route r, and $a_{i,r} = \mathbb{1}_{\{i \in r\}}$ is the indicator function indicating whether customer i is in route r. Constraint 107 ensures all customers are visited exactly once, Constraint 108 ensures the maximum number of vehicles, if specified, otherwise this can be set to the number of customers and will not further influence the ILP solver.

4.5 Simulation

In order to test the quality of the schedules, the objective values computed during the local search do not suffice. These scores are entirely based on the input parameters and are thus an approximation of the actual values. Moreover, the decisions on what routes are good and what not were made based on these values. The scores thus tell us that that solution is very good, that is why it was outputted as the best solution. To thoroughly test the robustness and costs of the schedules, a simulator is used. The simulator draws new samples from the wind distribution and computes for each of these samples the total working time, the service level, and the objective value. The total working time is defined by the sum of the travel time, the service time and the waiting time.

The average of these values gives a measure of the actual quality of the proposed schedules. The robustness is measured by also measuring the average score of the 95^{th} percentile.

The simulation is repeated a thousand times per outputted schedule. For the simulation, the 'random' package of python is used. The seed is set to 42. All code is available through Wal 2024.

In order to test the quality of routes, we cannot plainly use the score given by the algorithm, as this algorithm bases its score on the constructed approximation to the travel time distribution. To thoroughly test the quality of the schedule, a simulator samples new realisations from the wind distribution, and computes the total working time (i.e. travel time plus waiting time plus service time) and service level corresponding to that realisation. The simulation is repeated a thousand times. In order to make the results reproducible, the simulator gets a seed, that is different from the seed used to create the input for the algorithm. The seed for the simulator is 42. The code of the simulator is available through GitHub via `https://github.com/STvdWalUU/Simulator`

5 Experimental Setup

This section explains the instances and algorithms that are used to test the quality of the KDE based algorithm. First, the instances are introduced. These consist of sets of customers with individual demands and time windows. Then, different parameter settings for the KDE method are discussed. Finally, a description is given of the final experiments.

5.1 Instances

For the testing of the algorithm, six instances are used. The first five are constructed by Fontaine 2021, who first introduced the load levels to the vehicle routing problem. These five instances are based on five different cities, Fukuoka, Madrid, Pittsburgh, Seattle and Sydney. These cities were chosen for their height differences. These emphasize the effect of the load. For each city a depot location and 100 customer locations are chosen. The distance matrix is created using google maps and elevation data is used to compute height differences and thus the average slope of the road. The planning horizon for these instances is 120 minutes. Time windows are randomly chosen throughout the planning horizon and have an average of around 52 minutes. The demand of customers range from 5 to 15, with an average of around 10.

The sixth instances was created by Hesselmans 2022. This instance represents Utrecht and was created in a similar manner. The city is relatively flat, but slopes are calculated nonetheless. This city is much more flat than the other cities and thus might give different insights, especially in the presence of wind. Moreover, the planning horizon for this instance is 180 minutes. The time windows have the same average length as the other instances.

5.2 KDE Parameter Settings

The kernel density technique is defined by a choice of kernel, a bandwidth and a number of observations. The choice of kernel is considered in the literature to make little impact (Y.-C. Chen 2017) and is thus fixed to the Epanechnikov kernel. This is considered to be the most efficient, meaning that when using other kernels more samples are needed to obtain the same bound on the asymptotic mean integrated square error (AMISE). The different parameter settings for the KDE algorithm are tested on the Utrecht instance. Since this is the flattest instance, the effect of the wind and arrival time distribution approximation is thought to be the biggest.

5.2.1 General Parameters

The parameter setup that is used is similar to the setup of Hesselmans 2022 with some changes. First of all, the temperature is set higher. This is because of the altered shape of the penalty function, which also awards a penalty for being just in time for the end of the time window. Moreover the objective function incorporates the waiting time and the probability of being late with different weights. The altered parameters are

- Starting temperature $T = 10$,
- relative weight for waiting $\gamma_1 = 1$,
- relative weight for lateness $\gamma_2 = 5$,
- the number of multistarts is 30.
- the maximum number of iterations is set to 9×10^6

The rest of the parameters remain unchanged.

5.2.2 Number of Samples

One important trade-off in the KDE-technique is the trade-off between computational accuracy and speed as a consequence of the number of samples that are used to create the approximation of the arrival time distribution. More samples means smaller AMISE, and thus is considered to be better. Under the assumption that all samples are i.i.d. from the original distribution we have $\lim_{M\to\infty} \tilde{f}_{h_{AMISE}}(x) = f(x)$. However, since the approximation is non-parametric, all samples need to be stored in order to store the density approximation. Moreover, to compute the value of the distribution at any point, a contribution of every sample must be computed. This heavily affects the computation time of the algorithm, hence the trade-off. Due to overhead, a double samples size leads to slightly more than double the computation time on average. Since the local search phase is based on exploring the solution space, it is important to do many iterations. In order to properly test this, first a very large set of samples is created, corresponding to the largest sample size. Then every smaller set of samples is a subset of the largest set such that if M_k is the size of the kth smallest set. The set obey

$$\forall k < l : M_k < M_l \text{ and } \{X_i\}_{i\in(1,\dots,M_k)} \subsetneq \{X_i\}_{i\in(1,\dots,M_l)}\,. \tag{110}$$

For testing the effect of different number of samples, five different sample sizes are created. The largest being 80 samples. From pretesting, it was clear that using more than this number of samples would significantly slow down the algorithm and would not yield useful solution within a reasonable time. From this pretesting, the sample sizes 5,10,20,50 and 80 were selected as those sizes show potentially interesting differences. A sixth 'sample' was created, called 'FC'. This stands for forecast and consists of only one 'sample' per travel time. The sample is not drawn from the distribution, but rather selected to correspond to the forecast wind. Each set of samples was given to KDE based algorithm and run with a maximum number of iterations of 9000000 for the local search and a time limit of 3600 seconds for both the local search and the ilp phase. The algorithm was run five times for each sample set.

The results from these tests are summarized in Table 3. Each outputted schedule was simulated 1000 times. Each row corresponds to the simulated results of all five runs of that setting. The average is thus an average of 5000 values and the 'worst 5%' is the average of the 95^{th} percentile of those 5000 simulations.

From these results, it can be concluded that using only the forecast wind in this method yields the cheapest solutions in terms working time. However, these solutions almost always have a low service level. The runtime, as expected, is lowest for this setting. Already from using five observations, the service level increase dramatically to almost 1. This comes at the price of increased working time. Still, overall, the objective value is best on average for using five samples. Moreover, also the 95^{th} percentile yields the best objective value. This comes at the cost of increased computation time. The sets with 10,20,and 50 samples scored very similarly. All three had a near perfect score for the service level and the total working times lie close to each other. Despite using the full time limit in the local search, the set of 20 samples produces the best objective value out of these three. The largest set, using 80 samples, also used the full time limit in the local search phase. This did not lead to a very good result. Even though the service level score is very good, the total working time scored significantly less. This resulted in the worst score in the objective value. Both on average as in the 95^{th} percentile.

The average scores for the individual schedules are plotted in Figure 5. The basic conclusions here are the same as those from the table where the statistics of the five schedules was combined. Some more information is visible here. Namely that for 'FC' the points lie somewhat further from each other than the other cases. This can be explained by the fact that since the algorithm bases it's evaluation on a single value for the travel time, it might not be able to distinguish between two schedules as accurately. The points for 10 samples are so close to those of 20 and 5 samples, that they are barely visible.

The fact that using 5 samples produces the best average objective value motivates to select this sample size for the remainder of the experiments. The set with 20 samples is also selected. This is done because it scores the best service level, together with using 10 samples, but also scores a bit better on the total working time. For reference, the setting of using only the forecast wind as a single

sample (FC) is also selected. This allows for a better comparison to the heuristics as well. For further experiments, thus, the settings 'FC', 5, and 20 samples are used.

Table 3: Simulation results for the outputted schedules for the Utrecht instance. Each setting was run five times and simulated 1000 time per schedule. 'FC' refers to using only the forecast wind to approximate the arrival time distributions. Service level values are round to three decimal places, all other numbers are rounded to one decimal place. The best score in each column is in bold.

number of	average			worst 5%			average runtime (s)		
samples	α	TWT	obj.val	α	TWT	obj.val	ls time	ilp time	total
FC	0.84	**691.3**	771.3	0.83	**704.3**	786.4	**295.2**	**384.8**	**680.0**
5	0.996	719.8	**721.8**	0.99	733.9	**736.6**	2450.5	1280.3	3731.0
10	**1.0**	722.8	722.8	**1.0**	737.3	737.3	2563.9	468.4	3032.0
20	**1.0**	722.6	722.6	**1.0**	737.1	737.1	3601.1	537.5	4139.0
50	**1.0**	744.5	744.5	0.999	760.8	761.2	3600.2	1631.9	5232.0
80	**1.0**	834.9	835.0	0.997	855.7	857.4	3600.1	454.1	4054.0

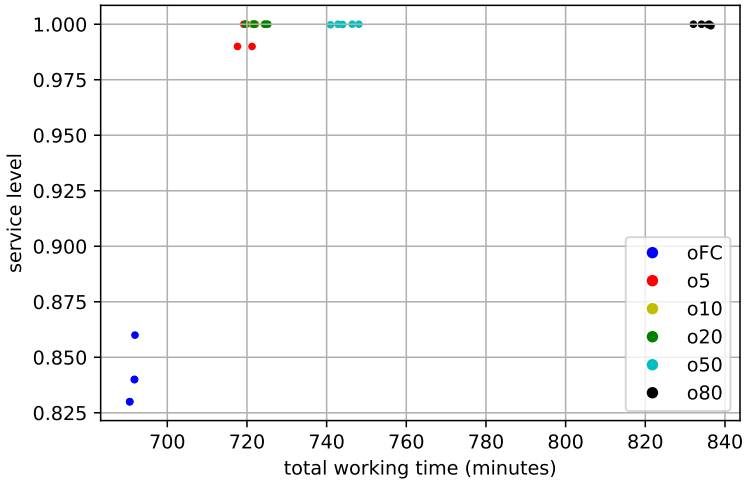


Figure 5: The simulated score for the Utrecht instance with wind speed 6.75 and wind direction 0 degrees (north wind). Results are shown plotting the average total travel time against the average service level over 1000 simulations.

5.2.3 Bandwidth

The bandwidth of the KDE method was already theoretically discussed in Section 3.2.2. As was stated, the bandwidth has a bigger influence on the quality of the resulting density approximation than the choice of kernel. Therefor a few options are tested. These options all fall within a range of bandwidths that proved potentially interesting in pretesting. These are a fixed bandwidth, the AMISE bandwidth with $R(f'') = 40$ based on experiments, the AMISE bandwidth with $R(f'') \approx 2 \times 10^{-4}$ based on assuming that the true density is $Gamma(x; 0.3, \frac{0.3}{\mathrm{E}[TT]})$. This leads to the bandwidths

- fixed: $h = 1$,
- AMISE: $h = 2.61n^{-1/5}$,
- AMISE(gamma): $h = 6.84n^{-1/5}$.

These three bandwidths lead to different arrival time distribution approximations and thus to different penalties corresponding to a particular arrival time. The effect of these bandwidths is that it alters how we measure arriving 'close to the deadline'. The larger the bandwidth, the earlier the driver has to arrive to not incur a penalty at all. Thus, the larger the bandwidth, the more margin is being built.

The results for the different bandwidths is summarized on Table 4, together with the results for using the empirical distribution, which will be discussed in Section 5.2.4. From the table, it can be seen that on average over all simulations, the fixed bandwidth is better than using the AMISE, based on either visual verification or assuming the gamma density. It scores better at the average, but also at the 95^{th} percentile. Even though differences are relatively small, it scores a bit better and in less time. The decrease in time is probably due to the fact that AMISE and gamma bandwidths are both computed each iteration, instead of saved as a global variable. This causes extra overhead that might have caused the algorithm to reach the time limit rather than the iteration limit. Only when using the forecast wind as a single 'sample' is using the AMISE bandwidth slightly better than the fixed bandwidth in the 95^{th} percentile. In Figure 6 the averages for each run are plotted. From this picture, it can be concluded that the gamma distribution based bandwidth is less suited than the other two, as all runs produced schedules that are worse than all runs from all bandwidths for that respective samples size. The fixed and AMISE bandwidth seem to perform very similarly width a similar spread in the averages of the outputted schedules.

From these results, the fixed bandwidth is chosen to further investigate in this thesis. However, let it be noted that only a small improvement is made from the AMISE to the fixed bandwidth. In the next section, these scores are also compared to the plain usage of the empirical distribution function instead of the kernel density estimate.

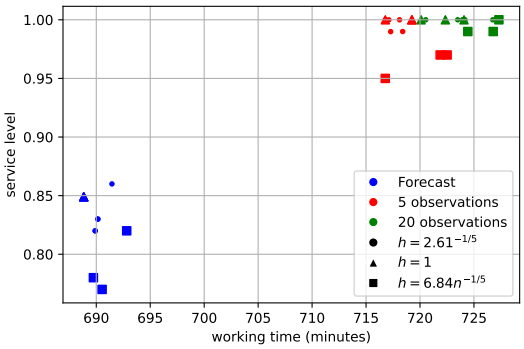
Table 4: Simulation results for the outputted schedules for the Utrecht instance. Each setting was run five times and simulated 1000 time per schedule. 'Empirical' refers to used the empirical distribution function instead of the KDE technique. Service level values are round to three decimal places, all other numbers are rounded to one decimal place. The best score in each column is in bold.

nr of		average			worst 5%			average runtime (s)		
samples	bandwidth	α	TWT	obj.val	α	TWT	obj.val	ls time	ilp time	total
FC	Empirical	0.847	688.9	765.4	0.81	701.8	791.1	635.3	730.3	1366.0
	fixed	0.845	**688.8**	766.4	0.823	**701.2**	787.2	**184.1**	360.4	**544.0**
	AMISE	0.84	691.3	771.3	0.83	704.3	786.4	604.1	760.8	1365.0
	gamma	0.793	692.0	795.6	0.77	704.8	813.5	324.6	348.5	673.0
5	Empirical	0.998	716.4	**717.3**	0.981	730.1	736.9	1292.2	277.1	1569.0
	fixed	**1.0**	720.5	720.6	0.997	734.5	**736.1**	754.0	913.5	1668.0
	AMISE	0.996	719.8	721.8	0.99	733.9	736.6	1296.5	602.9	1899.0
	gamma	0.958	721.4	742.4	0.95	734.8	756.9	1433.0	413.9	1847.0
20	Empirical	0.999	720.8	721.4	0.99	735.4	739.9	3601.2	274.7	3876.0
	fixed	**1.0**	721.9	721.9	**1.0**	736.2	736.3	2928.4	563.0	3491.0
	AMISE	**1.0**	722.6	722.6	**1.0**	737.1	737.1	3601.2	406.1	4007.0
	gamma	0.99	724.8	729.8	0.99	739.5	744.5	3600.9	**208.9**	3810.0

5.2.4 KDE vs Empirical

A question the reader might ask is whether it is at all necessary to go through the complications of constructing the non parametric arrival time distribution using the KDE technique. It is much easier to plainly use the empirical distribution function. it is defined

$$F_n(x) = \frac{\#\{x_i \leq x\}}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{(\infty,x]}(x_i), \tag{111}$$



e

Figure 6: The effect of using different bandwidths to approximate the arrival time distributions

where $\{x_i\}_{i\in(1,\dots,n)}$ are n random samples from the distribution $F(x) = \mathbb{P}(X \leq x)$ and $\mathbb{1}_A(x)$ is the indicator function such that

$$\mathbb{1}_A(x) \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}. \tag{112}$$

The empirical cumulative distribution function is unbiased, i.e.

$$\mathbb{E}[F_n(x)] = \mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{(\infty,x]}(x_i)] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\mathbb{1}_{(\infty,x]}(x_i)] = \mathbb{P}(X \in (\infty, x]) = F(x). \tag{113}$$

This, indeed, is an easy to use approximation of the arrival time distribution. The results for using the empirical distribution function instead of the KDE approximation method are summarized in Table 4 and in Figure 7. From this table, the conclusion can be drawn that it is on average about as good as using the fixed bandwidth in the KDE. However, it tends to be less good in the 95^{th} percentile. This can be explained by the fact that the empirical distribution can not distinguish between certain schedules, because the arrival times are similar and have the same number of on time arrivals. To the empirical distribution, these two schedules are then assigned the same score, while the KDE method might favor one of them, thereby improving just a bit. This effect might have caused the 95^{th} percentile of using either the AMISE or fixed bandwidth to be better.

The conclusion from this is that it is not certain whether or not using the KDE approximation is much better than the empirical distribution. However, on average, the KDE method with the fixed bandwidth finds better service levels and better 95^{th} percentiles. Moreover, it is faster on average. Therefor the fixed bandwidth will be used in the final experiments.

5.3 Heuristics

The KDE algorithm is tested against several heuristics. These are based on the deterministic algorithm as proposed by Hesselmans 2022. As was shown in their studies, for a deterministic wind an improvement can be made by incorporating this into the algorithm and altering the travel times as they would have been with that wind. This technique found solutions that were faster in total travel time while meeting all time windows. When these schedules that were produced with the wind are simulated without wind, many time windows were not met. Sometimes even up to 20% of the arrivals were late. This suggest that the algorithm was in a way over fitting to the new wind and used it on some roads to be faster than it could be without that wind. This begs the question whether these schedules would last in a stochastic setting. This algorithm is ready to use as it was already tested by Hesselmans 2022. This heuristic shall be named 'SAM'. Note that this algorithm used the same input information as the KDE algorithm with only the forecast wind (FC). The only difference between these two is the penalty for being late. For 'FC', this is based on the integral of the kernel function with

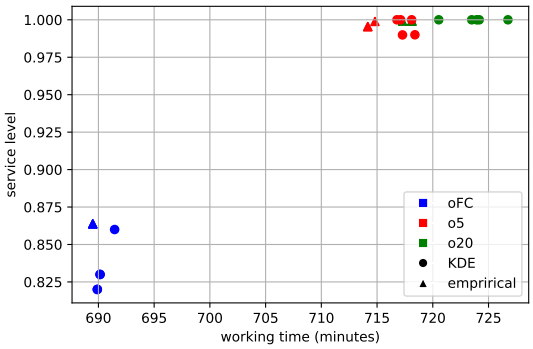


Figure 7: The effect of using either the empirical distribution function or the KDE estimated distribution function, computed for a single instance.

the end time window as midpoint. For 'SAM', it is a function that is zero before the end of the time window and increases linear with lateness.

A second heuristic that will be tested is an adjustment of the input instance. As travel times are uncertain because of wind, it might be wise to build in some extra safety margin. This is done by tightening the instance. The time windows are shortened by 10%. This means that all time windows are changed according to

$$[a_i, b_i] \to [a_i, b_i'] = [a_i, b_i - 0.1(b_i - a_i)]\,. \tag{114}$$

Since $b_i' = b_i - 0.1(b_i - a_i) = 0.9b_i + 0.1a_i \geq a_i$, this is a valid time window. The wind in this case is set to zero. This heuristic is expected to perform relatively good on service level. However, the total working time might blow up as it is a tighter instance and thus might need to choose less effient routes. This heuristc is named 'ATW' for adjusted time window.

The last heuristic is merely a control run of the original deterministic algorithm, but with the wind set to zero. This is expected to perform worst as it does not account for the wind in any way. This is called 'NoW'.

5.4 Final Experiments

Considering the previous sections, a selection of parameter setting for the KDE was chosen to be tested against the described heuristics. The creation of travel time samples and adjustments of instances were done using Python. The experiments were coded in C#. All relevant code can be found in Wal 2024. All experiments were run on an Intel Xeon(R) Gold 6130 CPU @ 2.10GHz x32. The simulation afterwards is again done using Python as described in Section 4.5. Table 5 gives an summary of the experiments that were run. Each experiment is repeated five times. The results are discussed in Section 6.

Table 5: Summary of the different algorithms that are used to compute a schedule to the vehicle routing problem with load and wind dependent travel times and time windows.

oFC	The KDE-based algorithm that uses only the forecast wind as a single sample.
o5	The KDE-based algorithm that uses a five samples to compute an approximate the arrival time distribution.
o20	The KDE-based algorithm that uses twenty samples to compute an approximate the arrival time distribution.
SAM	The deterministic algorithm built by Hesselmans 2022, with the forecast wind as input.
ATW	An adjustment of the time windows such that the deadlines are moved forward. Then, the deterministic algorithm is run without any input wind on this adjusted instance.
NoW	The deterministic algorithm is run without any knowledge of wind.

6 Numerical Results

This section presents and discusses the results of the experiments as summarized in Table 5. The results are presented both in the form of a table, where all data on one particle algorithm is summarized and in the form of a plot where the five individual runs of each algorithm can be seen separately.

Simulation results for the six different instances are given in Tables 6-11. Each algorithm was run five times and thus produced 5 schedules. Each schedules was simulated 1000 times. This yielded 5000 simulations per algorithm per instance. From these the average service level (α), average total working time (TWT) and average objective value (obj.val.) are computed. Moreover, from these, the five percent worst of each of these is computed. The column under 'worst 5%' and 'α' thus refers to the 250 simulations that scored worst on the service level and reports their average. This is also done for the total working time and the objective value. All reported values for service level are rounded to three decimals and all total working time and objective values are rounded to one decimal. Due to rounding, the reported average for all instances is 1.0 for Sydney with 'o5', while the worst 5% has an average of 0.999.

In these tables, the run time is also shown. The column 'ls time' refers to the time that the local search was running. The timing is started before the actual local search loop is being called and stopped when the set of routes for the ILP is stored. This results in some overhead and can cause the time to measure beyond the time limit of 3600s. Moreover, the algorithm might be doing an iteration with, e.g. Neighbourhood 6, which is relatively slow and can also cause the time limit to be passed. the column 'ilp time' refers to the time that the ILP phase of the algorithm was running. Again, due to overhead and pre solving, it is possible that the ILP solver passes the time limit of 3600s. The last column shows us the sum of these times and give a measure of the total runtime. All times are averages over the five runs that were done with that algorithm on that instance.

6.1 Performance

The results of the runs and simulations are summarized in Tables 6-11. Similar to Sections 5.2.2 and 5.2.4, these show the average score over all simulations and the average of the 95^{th} percentile, together with the average run time. In Figure 8 the average score of each run is plotted. On the x-axis the total working time is shown. This is the sum of travel time, waiting time and service time. As deliverers costs are related to the working hours of delivers, it is of interest to find fast and thus cheap routes. On the y-axis the service level is shown, this is the fraction of deliveries that were made before the end of the time window. A fraction of 1 means that all deliveries were within the time window, 0 means no deliveries were within the time window. It is thus desirable to have a service level as high as possible.

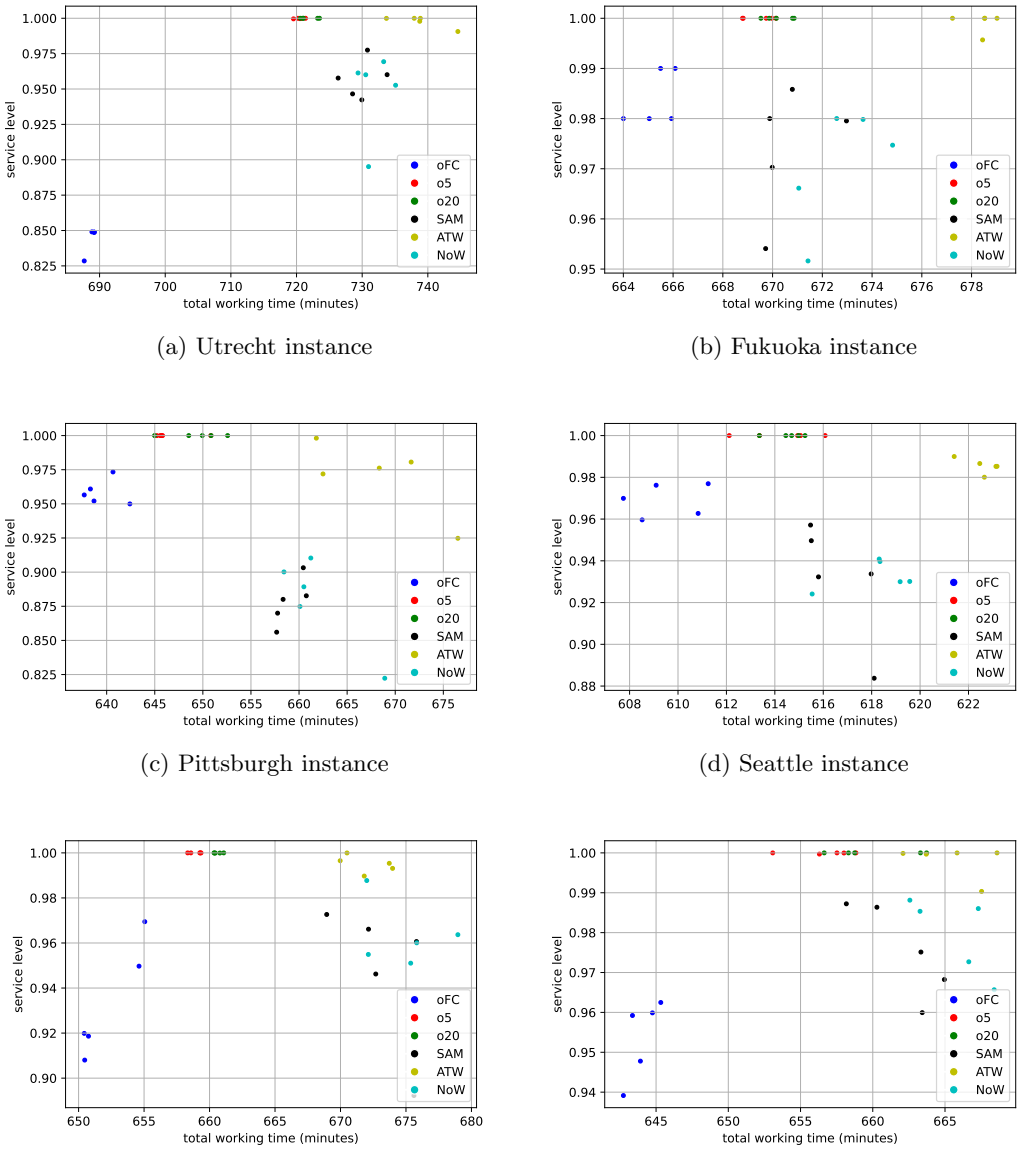
The first thing that the reader might notice, is that the best scores for service level, total work time, and objective value were always by the KDE based algorithms. The KDE algorithm with 5 and 20 samples performed very similar. The algorithm with 5 samples was on average slightly better. In the 95^{th} percentile it scored best almost always, except on the seattle instance. Both the algorithm with 5 and 20 samples always outperformed all heuristics and the KDE algorithm with only the forecast in terms of service level and objective value. The cheapest route was consequently found by the FC algorithm.

Overall the heuristics performed worse than the KDE algorithms with 5 and 20 samples. For the SAM algorithm, the service level varied significantly from 0.975 for the Sydney instance to 0.878 for the Pittsburgh instance on average. In the 95^{th} percentile, the differences are even greater. The ATW heuristic on the other hand performed generally very well on service level. With the exception of Pittsburgh it scored always higher than 0.98 for the service level on average and 0.97 in the 95^{th} percentile. However, the total work time was generally one of the worst out of all schedules that were produced. This can be seen in Figure 8, where except for Madrid, the worst working time was always achieved by the ATW heuristic.

An interesting observation is that although the SAM algorithm does take the wind into account and the NoW algorithm assumes no wind, the results tend to be very similar. This is surprising as previous research showed the benefit of considering the wind as part of the solution algorithm. The o5 and o20 algorithms also performed very similar.

Another interesting fact is that the FC algorithm and the SAM algorithm got the same information to start with. Both started with only the wind forecast and the same instance information. However, many times the FC schedules dominate the SAM schedules in terms of objective value. The working time was always lower for the schedules found by FC. For the Pittsburgh instance these schedules also had a higher service level.

Another point of interest is the spread in the points. From Figure 8 it is clear that the o5 and o20 algorithms have results that lie much closer together than the other algorithms. This is true for all instances. This is confirmed when the standard deviation of the objective value of the 5000 simulations is computed. Only FC in the Utrecht instance and ATW in the Sydney instance come close. This suggests that the o5 and o20 algorithms are more consequent in their output.



(a) Utrecht instance

(b) Fukuoka instance

(c) Pittsburgh instance

(d) Seattle instance

(e) Madrid instance

(f) Sydney instance

Figure 8: Average score for schedules produced by different methods. These schedules are made for $w_s = 6.75ms^{-1}$ and $w_d = 0deg$. Each schedule is simulated 1000 times.

6.2 Runtime

The run times can be read from Tables 6-11. In the last two columns. With respect to the run times, it is clear that the KDE method is slower than the heuristics. For all instances, the heuristics took less time in both the local search phase and the ilp phase. The former is to be expected, as a lot of work was done by Hesselmans 2022 to optimise that code. The code that acts as the neighbourhoods in the KDE algorithm have more overhead and loop over the samples. Thereby it is expected that the FC algorithm is somewhat slower, which is indeed the case. The o5 and o20 algorithm are much slower. In line with expectation that the computation time is linear in the number of samples M, the o5 algorithm is around 4 times slower than the FC algorithm and the o20 algorithm is around 16 times slower. The heuristcs rarely use more than three minutes in the local search phase.

The times in the ILP phase vary a lot more. The shortest being 4.2 seconds for the ATW and NoW algorithms on the Utrecht Instance, and the longest being 3602.4 for the FC algorithm on the Fukoka instance. Generally, the heuristics are faster in the ILP phase than the KDE algorithms. This is not readily explained. The runtime for the ILP phase turns out to be oddly specific to the instance. For example, for the Fukuoka instance all runs of all algorithms made it to the time limit, while for the Utrecht instance all algorithms were on average done within 20 minutes with this phase.

Table 6: Simulation results for the Utrecht instance. With wind direction $0deg$ and wind speed $6.75m^s{-}1$. The reported service levels are rounded to three decimals, the rest to 1 decimal.

	average			worst 5%			runtime (seconds)		
algorithm	α	TWT	obj.val	α	TWT	obj.val	ls time	ilp time	total
oFC	0.845	**688.8**	766.4	0.823	**701.2**	787.2	184.1	360.4	544.0
o5	**1.0**	720.5	**720.6**	0.997	734.5	**736.1**	754.0	913.5	1668.0
o20	**1.0**	721.9	721.9	**1.0**	736.2	736.3	2928.4	563.0	3491.0
SAM	0.957	729.9	751.5	0.927	746.0	778.4	**145.2**	6.2	**151.0**
ATW	0.998	738.8	739.9	0.976	756.0	767.1	152.7	**4.2**	157.0
NoW	0.948	731.8	758.0	0.884	747.5	796.5	153.5	**4.2**	158.0

Table 7: Simulation results for the Fukuoka instance. With wind direction $0deg$ and wind speed $6.75m^s{-}1$. The reported service levels are rounded to three decimals, the rest to 1 decimal.

	average			worst 5%			runtime (seconds)		
algorithm	α	TWT	obj.val	α	TWT	obj.val	ls time	ilp time	total
oFC	0.984	**665.3**	673.3	0.98	**677.6**	686.2	203.1	3602.4	3805.0
o5	**1.0**	669.5	**669.5**	**1.0**	682.0	**682.0**	787.5	3601.8	4389.0
o20	**1.0**	670.2	670.2	**1.0**	682.8	682.8	3012.8	3601.9	6615.0
SAM	0.974	670.7	683.7	0.949	684.0	703.3	148.0	3600.7	3749.0
ATW	0.999	678.4	678.8	0.99	692.1	693.9	**146.0**	3600.6	**3747.0**
NoW	0.97	672.7	687.5	0.945	686.1	707.3	148.6	**3600.4**	3749.0

Table 8: Simulation results for the Pittsburgh instance. With wind direction $0deg$ and wind speed $6.75m^s-1$. The reported service levels are rounded to three decimals, the rest to 1 decimal.

		average			worst 5%		runtime (seconds)		
algorithm	α	TWT	obj.val	α	TWT	obj.val	ls time	ilp time	total
oFC	0.959	**639.5**	660.3	0.891	**655.3**	708.4	209.3	3191.1	3400.0
o5	**1.0**	646.6	**646.6**	**1.0**	659.0	**659.0**	908.9	3181.9	4091.0
o20	**1.0**	649.4	649.4	**1.0**	659.0	**659.0**	3413.2	3452.9	6866.0
SAM	0.878	659.0	719.8	0.826	668.8	752.1	**154.2**	129.1	283.0
ATW	0.97	668.2	683.0	0.905	682.8	729.6	163.7	**75.0**	**239.0**
NoW	0.879	661.8	722.1	0.812	674.8	768.1	158.1	228.7	387.0

Table 9: Simulation results for the Seattle instance. With wind direction $0deg$ and wind speed $6.75m^s-1$. The reported service levels are rounded to three decimals, the rest to 1 decimal.

		average			worst 5%		runtime (seconds)		
algorithm	α	TWT	obj.val	α	TWT	obj.val	ls time	ilp time	total
oFC	0.969	**609.5**	624.9	0.949	**618.0**	642.3	227.8	2603.2	2831.0
o5	**1.0**	614.3	**614.3**	**1.0**	619.1	619.1	930.8	3056.3	3987.0
o20	**1.0**	614.5	614.5	**1.0**	618.4	**618.4**	3467.5	1965.9	5433.0
SAM	0.931	616.6	650.9	0.88	621.4	680.7	168.8	473.1	642.0
ATW	0.985	622.6	629.9	0.979	627.3	637.9	**161.5**	**437.9**	**599.0**
NoW	0.933	618.2	651.7	0.915	623.2	661.7	163.9	1349.9	1514.0

Table 10: Simulation results for the Madrid instance. With wind direction $0deg$ and wind speed $6.75m^s-1$. The reported service levels are rounded to three decimals, the rest to 1 decimal.

		average			worst 5%		runtime (seconds)		
algorithm	α	TWT	obj.val	α	TWT	obj.val	ls time	ilp time	total
oFC	0.933	**652.3**	685.7	0.9	**662.4**	705.6	180.3	1768.9	1949.0
o5	**1.0**	659.0	**659.0**	**1.0**	668.1	**668.1**	858.1	626.4	1484.0
o20	**1.0**	660.6	660.6	0.999	670.3	670.7	3301.5	667.5	3969.0
SAM	0.948	673.0	699.2	0.873	683.8	744.6	**138.7**	55.0	**194.0**
ATW	0.995	672.0	674.5	0.986	682.7	689.2	154.7	**39.8**	195.0
NoW	0.963	674.9	693.1	0.934	686.5	716.9	159.5	40.7	200.0

Table 11: Simulation results for the Sydney instance. With wind direction $0deg$ and wind speed $6.75m^s-1$. The reported service levels are rounded to three decimals, the rest to 1 decimal.

		average			worst 5%		runtime (seconds)		
algorithm	α	TWT	obj.val	α	TWT	obj.val	ls time	ilp time	total
oFC	0.954	**644.0**	667.2	0.936	**658.8**	687.9	180.4	1711.3	1892.0
o5	**1.0**	656.7	**656.8**	0.999	671.7	671.7	745.5	2837.9	3583.0
o20	**1.0**	660.1	660.1	**1.0**	671.5	**671.5**	2880.5	3128.7	6009.0
SAM	0.975	662.0	674.3	0.957	674.7	696.0	**121.3**	325.0	446.0
ATW	0.998	665.6	666.6	0.989	679.0	682.0	121.4	**197.9**	**319.0**
NoW	0.98	665.6	675.8	0.952	677.4	698.5	123.9	348.5	472.0

7 Conclusions

In this thesis, the aim was to use a non parametric approach to approximate the arrival time distribution for the stochastic vehicle routing problem with time windows (SVRPTW). Furthermore, the aim was to use the approximated arrival time distributions to compute schedules that are both robust to variations in wind and cheap in terms of travel and waiting time. A basic probability density distribution was defined to model the wind and samples were drawn that were used to compute a set of corresponding samples of travel times. These samples were used to construct the arrival time distribution for each customer. From the approximated arrival time distributions, the objective function was approximated, which is defined as the expected value of a score function that is a linear combination of travel time, waiting time and the number of late deliveries. This resulted in an algorithm that was tested against several heuristics based on previous work.

It was shown how the arrival time distributions of each customer can be determined using the non parametric technique of kernel density estimation (KDE). The technique can be implemented in any code and is relatively simple. The downside is that all samples need to be stored and that the computation time increases dramatically with an increased number of samples. The bandwidth for this method plays a key role in the convergence of the approximation. However, determining the optimal bandwidth depends either on the characteristics of the underlying density, which are assumed to be unknown, or on an analysis that would be needed for each approximation separately.

The results show that using non parametric approximation technique based on the kernel density estimation (KDE) method can produce schedules that have a relatively high service level in simulations, while also remaining cheap on average. A simple time window heuristic proved to also be able to produce very good service levels in much less time, but these schedules require more travel and waiting time on average.

The KDE based method achieves more accurate schedules but also becomes slower with each added sample to construct the arrival time distribution. From 20 samples and up, the algorithm took the full time limit. As was expected due to the time complexity being linear in the number of samples M, this caused fewer iterations to be performed. For 50 and 80 samples, the solutions were not as good as for smaller sample sizes. Varying the bandwidth of the kernel changed both the quality of the outputted schedules as the runtime of the algorithm. However, with some optimising of the code, the difference in run time could be decreased. On average, the smaller bandwidths performed better, however in the 95^{th} percentile, these difference were not as clear. The effect of bandwidth seems to be less important on performance and run time than the effect of the number of samples that are used. The use of the empirical distribution instead of the KDE method produced similar results, albeit slightly worse and a bit slower on average.

The schedules produced by the KDE method with 5 and 20 samples produced very similar results for each run. This suggests that these algorithms are very consistent in their output. This is confirmed by looking at the 95^{th} percentile of the simulations, which show that these algorithms had a much lower spread of outcomes than using only the forecast wind as input or the heuristic where the instance was edited. From this we conclude that the KDE algorithm produced schedules with 5 and 20 observations are much more robust to the variations in wind than the other algorithms.

For future research, it might be interesting to look into different distributions to model the wind. In combination with other stochastic delays, this could lead to a broader application of the non parametric distribution approach, as all deviations could be captured. A more practical route can also be taken by measuring travel times in practice and using those to construct an approximation to the arrival time distribution. It might also be worthwhile to look into using a weighted KDE method, where we choose the 'samples' that are used to construct the travel time distributions. These can then be chosen to represent certain weather events and by approximating the probability of those events happening, a weight can be assigned.

Acknowledgments

The author would like to thank Dr. ir. J.M. van den Akker, MSc P. de Bruin and Dr. S. Dirksen for the support and supervision during the research and writing of this thesis.

References

Barry, Nathan (2018). "A new method for analysing the effect of environmental wind on real world aerodynamic performance in cycling". In: *Proceedings*. Vol. 2. 6. MDPI, p. 211.

Blundell, Stephen J. and Katherine M. Blundell (2006). *Concepts in Thermal Physics*. Oxford University Press.

Bräysy, Olli and Michel Gendreau (2005). "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms". In: *Transportation science* 39.1, pp. 104–118.

Brooks, S. P. and B. J. T. Morgan (1995). "Optimization Using Simulated Annealing". In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 44.2, pp. 241–257. ISSN: 00390526, 14679884. URL: http://www.jstor.org/stable/2348448 (visited on 10/30/2023).

Chaieb, Marouene and Dhekra Ben Sassi (2021). "Measuring and evaluating the Home Health Care Scheduling Problem with Simultaneouw Pick-up and Delivery with Time Window using a Tabu Search meteheuristic solution". In: *Applied Soft Computing*. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2021.107957.

Chen, Yen-Chi (2017). "A tutorial on kernel density estimation and recent advances". In: *Biostatistics & Epidemiology* 1.1, pp. 161–187.

Chiang, Wen-Chyuan and Robert A Russell (1996). "Simulated annealing metaheuristics for the vehicle routing problem with time windows". In: *Annals of Operations Research* 63, pp. 3–27.

Cordeau, Jean-François et al. (2007). "Chapter 6 Vehicle Routing". In: *Transportation*. Ed. by Cynthia Barnhart and Gilbert Laporte. Vol. 14. Handbooks in Operations Research and Management Science. Elsevier, pp. 367–428. DOI: https://doi.org/10.1016/S0927-0507(06)14006-2. URL: https://www.sciencedirect.com/science/article/pii/S0927050706140062.

Dantzig, G. B. and J. H. Ramser (Oct. 1959). "The Truck Dispatching Problem". In: *Management Science* 6, pp. 80–91. URL: https://www.jstor.org/stable/2627477.

Dekkers, Anton and Emile Aarts (1991). "Global optimization and simulated annealing". In: *Mathematical programming* 50, pp. 367–393.

Dhahri, Amine, Kamel Zidi, and Khaled Ghedira (2014). "Variable Neighborhood Search based Set Covering ILP Model for the Vehicle Routing Problem with Time Windows". In: *Procedia Computer Science* 29. 2014 International Conference on Computational Science, pp. 844–854. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2014.05.076. URL: https://www.sciencedirect.com/science/article/pii/S1877050914002531.

Ehmke, Jan Fabian, Ann Melissa Campbell, and Timothy L Urban (2014). "Ensuring service levels in routing problems with time windows and stochastic travel times". In: *European Journal of Operational Research* 240.2, pp. 539–550.

Fontaine, P. (2021). "The vehicle routing problem with load-dependent travel times for cargo bicycles". In: *European Journal of Operational Research*. DOI: https://doi.org/10.1016/j.ejor.2021.09.009. URL: https://www.sciencedirect.com/science/article/pii/S0377221721007682.

Gendreau, Michel, Alain Hertz, and Gilbert Laporte (1994). "A tabu search heuristic for the vehicle routing problem". In: *Management science* 40.10, pp. 1276–1290.

Gutierrez, Andres et al. (2018). "A multi-population algorithm to solve the VRP with stochastic service and travel times". In: *Computers & Industrial Engineering* 125, pp. 144–156.

Hesselmans, S. (2022). "Routing electric cargo bikes: a hybrid solution approach". Master's thesis. Utrecht University. URL: https://studenttheses.uu.nl/handle/20.500.12932/42050.

Joffre, Sylvain M and Tuomas Laurila (1988). "Standard deviations of wind speed and direction from observations over a smooth surface". In: *Journal of Applied Meteorology and Climatology* 27.5, pp. 550–561.

Jung, Christopher and Dirk Schindler (2019). "Wind speed distribution selection–A review of recent development and progress". In: *Renewable and Sustainable Energy Reviews* 114, p. 109290.

Kirkpatrick, S., C. D. Gelatt Jr, and M. P. Vecchi (1983). "Optimization by simulated annealing". In: *science*, pp. 671–680. DOI: https://doi.org/10.1126/science.220.4598.671.

Koç, Çağrı, Gilbert Laporte, and İlknur Tükenmez (2020). "A review of vehicle routing with simultaneous pickup and delivery". In: *Computers & Operations Research* 122, p. 104987. ISSN: 0305-0548.

DOI: https://doi.org/10.1016/j.cor.2020.104987. URL: https://www.sciencedirect.com/science/article/pii/S0305054820301040.

Lent, G.P.T. van (2018). "Using column generation for the Time Dependent Vehicle Routing Problem with Soft Time Windows and Stochastic Travel Times". Master's thesis. Utrecht University. URL: https://studenttheses.uu.nl/bitstream/handle/20.500.12932/28683/thesis%20%2848%29.pdf?sequence=1&isAllowed=y.

Li, Xiangyong, Peng Tian, and Stephen C.H. Leung (2010). "Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm". In: *International Journal of Production Economics* 125.1, pp. 137–145. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2010.01.013. URL: https://www.sciencedirect.com/science/article/pii/S092552731000023X.

Martin, James C et al. (1998). "Validation of a mathematical model for road cycling power". In: *Journal of applied biomechanics* 14.3, pp. 276–291.

Parzen, Emanuel (1962). "On estimation of a probability density function and mode". In: *The annals of mathematical statistics* 33.3, pp. 1065–1076.

Savelsbergh M., Woensel T. Van (2016). "City Logistics: Challenges and Opportunities." In: *Transportation Science* 50, pp. 579–590. URL: https://doi.org/10.1287/trsc.2016.0675.

Scott, David W (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

Sheather, Simon J (2004). "Density estimation". In: *Statistical science*, pp. 588–597.

Sheth, M. et al. (2019). "Measuring delivery route cost trade-offs between electric-assist cargo bicycles and delivery trucks in dense urban areas." In: *Eur. Transp. Res. Rev.* 11. DOI: https://doi.org/10.1186/s12544-019-0349-5.

Silverman, B.W. (1986). *Density Estimation for statistics and data analaysis*. Chapman and Hall.

Solomon, M. M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: *Operations Research*. URL: https://doi.org/10.1287/opre.35.2.254.

Vakulenko, Yulia et al. (2019). "Online retail experience and customer satisfaction: the mediating role of last mile delivery". In: *The International Review of Retail, Distribution and Consumer Research* 29.3, pp. 306–320. DOI: 10.1080/09593969.2019.1598466. URL: https://doi.org/10.1080/09593969.2019.1598466.

Wal, S.T. van der (2024). *Github Repository for Masters Thesis*. URL: https://github.com/STvdWalUU/MasterThesisWal.

Weir, Alan J (1973). *Lebesgue integration and measure*. Vol. 1. Cambridge University Press.

Zhang, Junlong, William HK Lam, and Bi Yu Chen (2013). "A stochastic vehicle routing problem with travel time uncertainty: trade-off between cost and customer service". In: *Networks and Spatial Economics* 13, pp. 471–496.

Appendix A Bike-Wind Angles

In this appendix, an overview of the forces on the bike and their direction is given. Firstly, the the size and direction of the apparent wind is derived. Then, all forces are schematically drawn.

A.1 Apparent Wind

In wind still conditions, the speed of the bike will create the illusion of a wind in the direction opposite to the direction of travel (td). This wind is called the motion wind. Combining this effect with the true wind (v_w) provides us with the wind as experienced on the bike. This experienced wind is called the apparent wind (v_A). A schematic overview of these winds is given in Figure 9. It is the apparent wind speed and direction that needs to be taken into account when computing the air drag force.

The apparent wind is the vector sum of the motion wind and the true wind. The size can thus be computed using Pythagoras theorem,

$$\|v_A\|_2^2 = (\|v_B\|_2 + \|v_w\|_2 \cos(\alpha))^2 + (\|v_W\|_2 \sin(\alpha))^2. \tag{115}$$

We can also compute the direction of this force relative to the travel direction. This angle is called the yaw angle and is denoted by β and can be compute as follows

$$\beta = \arctan\left(\frac{\|v_W\|_2 \sin(\alpha)}{\|v_B\|_2 + \|v_W\|_2 \cos(\alpha)}\right). \tag{116}$$

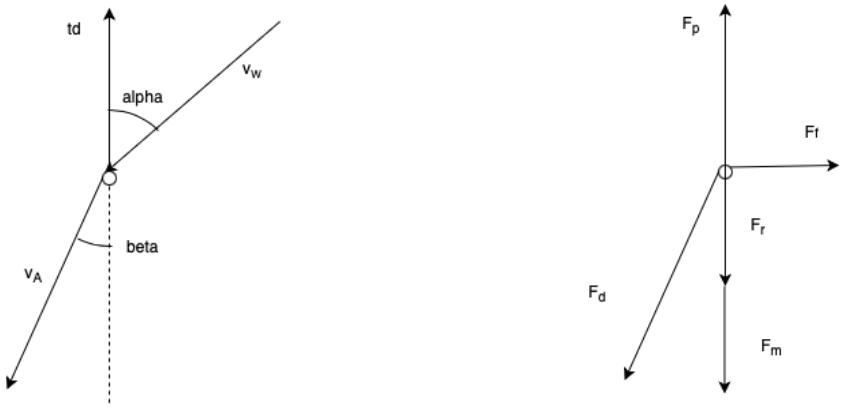


Figure 9: Direction of different winds while driving an e-bike (left). Direction of different forces on the bike while driving (right).

A.2 Forces

There are 5 main forces on the bike that are considered here. These are the propulsion force (F_p), the air drag force (Fd), the roll resistance force (F_r), the mechanical resistance force (F_m) and the friction force (F_f). Apart from the latter, these forces solely depend on the specifics of the bike and its surroundings. The friction force, however, depends on the air drag force. We assume that it always exactly compensates the vertical component of the air drag force.

In the previous section, we have computed the size of the apparent wind. We can now compute the size of the friction force via

$$F_f = F_d \sin(\beta) = \frac{\rho C_d A}{2} \|v_A\|_2^2 \sin(\beta). \tag{117}$$

The component of the air drag force in the direction of travel is

$$F_{d,//} = \frac{\rho C_d A}{2} \|v_A\|_2^2 \cos(\beta), \tag{118}$$

or equivalently, by writing $\|v_A\| \cos(\beta) = \|v_w\| \cos(\alpha) + \|v_B\|$, we get

$$F_{d,//} = \frac{\rho C_d A}{2} \left(\left(\|v_B\|_2 + \|v_w\|_2 \cos(\alpha) \right)^2 + \left(\|v_w\|_2 \sin(\alpha) \right)^2 \right)^{1/2} (\|v_w\| \cos(\alpha) + \|v_B\|), \tag{119}$$

in terms of parameters we can know or measure (i.e. v_B, v_w, α).

Appendix B Proof of Waiting Time Formula

Theorem 1. *For a route $R = \{0, r_1, r_2, ..., r_k\}$, with travel times $\{TT_{r_i,r_j,l}\}$, service times $\{\sigma_{r_i}\}$ and time windows $[a_{r_i}, b_{r_i}]$, that follows the constraints 3-14,assuming w.l.o.g. we leave at $t = 0$, the total waiting time along that route is equal to*

$$\max_{1\leq j'\leq k}\left\{0, a_{j'} - \sum_{i=1}^{j'-1}\sigma_{r_i} - \sum_{i=0}^{j'-1} TT_{i,i+1,l}\right\} \tag{120}$$

Proof. In order to prove this, we start by computing the arrival time at a customer, then using the definition for the waiting time to proof the theorem. The arrival time at a customer can be expressed in the arrival time at the previous customer, with the convention that $AT_0 = 0$ is the departure time at the depot and $\sigma_0 = 0$. We write

$$AT_i = \max\{a_{i-1}, AT_{i-1}\} + \sigma_{i-1} + TT_{i-1,i}. \tag{121}$$

The definition of the waiting time is

$$WT_i = \max\{0, a_i - AT_i\}. \tag{122}$$

Then by writing

$$\max\{a_i, AT_i\} = \max\{0, a_i - AT_i\} + AT_i = WT_i + AT_i, \tag{123}$$

we get

$$WT_i = \max\{0, a_i - (WT_{i-1} + AT_{i-1} + \sigma_{i-1} + TT_{i-1,i})\}. \tag{124}$$

By repeating this process we get

$$WT_i = \max\{0, a_i - \sum_{j=0}^{i-1}(WT_j + \sigma_j + TT_{j-1,j})\}. \tag{125}$$

This is equivalent to

$$\sum_{j=0}^{i} WT_j = \max\left\{\sum_{j=0}^{i-1} WT_j, a_i - \sum_{j=0}^{i-1}(\sigma_j + TT_{j-1,j})\right\}. \tag{126}$$

If we now consider $i = k$ is the last customer in the route we get

$$TWT = \sum_{j=0}^{k} WT_j \tag{127}$$

$$= \max\left\{\sum_{j=0}^{k-1} WT_j,\ a_k - \sum_{j=0}^{k-1}\sigma_j + \sum_{j=0}^{k-1} TT_{j-1,j}\right\}, \tag{128}$$

$$= \max\left\{\max\left\{\sum_{j=0}^{k-2} WT_j,\ a_{k-1} - \sum_{j=0}^{k-2}\sigma_j + \sum_{j=0}^{k-2} TT_{j-1,j}\right\},\ a_k - \sum_{j=0}^{k-1}\sigma_j + \sum_{j=0}^{k-1} TT_{j-1,j}\right\}, \tag{129}$$

$$= \max_{1\leq i\leq k}\left\{0,\ a_i - \sum_{j=0}^{i-1}\sigma_j + \sum_{j=0}^{i-1} TT_{j-1,j}\right\}. \tag{130}$$

□