# Solving the Stochastic Time-Dependent Orienteering Problem with Time Windows

C. Verbeeck[a,*], P. Vansteenwegen[b], E.-H. Aghezzaf[a]

[a] *Ghent University, Industrial Engineering Systems and Product Design*
[b] *KU Leuven, Leuven Mobility Research Centre, CIB*

**Abstract**

This paper introduces the stochastic time-dependent orienteering problem with time windows. The orienteering problem occurs in logistic situations where an optimal combination of locations must first be selected and then the routing between these selected locations must be optimized. In the stochastic time-dependent variant, the travel time between two locations is a stochastic function that depends on the departure time at the first location. The main contribution of this paper lies in the design of a fast and effective algorithm to solve this challenging problem. To validate the performance and the practical relevance of this proposed algorithm, several experiments were carried out on realistic benchmark instances of varying size and properties. These benchmark instances are constructed based on an actual large road network in Belgium with historic travel time profiles for every road segment.

*Keywords:* orienteering problem, stochastic time-dependent travel times, metaheuristics, time windows

## 1. Introduction

The orienteering problem (OP) is defined on a graph in which the vertices represent geographical locations at which a reward can be collected. Each arc in the graph represents a connection between two vertices and has a travel time. The goal of the OP is to determine which subset of vertices to visit and in which order so that the total collected reward is maximized and a given maximum total travel time is not exceeded. In addition, a feasible OP solution should start and end at a predetermined vertex. The OP includes the knapsack problem (KP) and the travelling salesperson problem (TSP) as subproblems. In contrast to the TSP, not all vertices can be visited in an OP due to the maximum total travel time constraint. However, determining the shortest path between the selected vertices will be helpful to visit as many vertices as possible within the available time.

The OP has many interesting applications in military surveillance, tourism and logistics. The military application of the OP considers Unmanned Aerial Vehicle (UAV) mission planning to collect intelligence information about different locations in the area of operations [22, 23, 25, 59, 67, 91]. The aim of these missions is to acquire as much information as possible during the flight, while the length of the flight is limited by the available fuel capacity of the UAV. To model personalized tourist trips, each vertex is a point of interest and the reward indicates the personal appreciation of the point of interest by the tourist. In this application, a tourist wants to maximise the enjoyment by visiting several different sightseeing locations, while the length of the tourist route is restricted by the total time the tourist can spend on sightseeing. These problems are called tourist trip design problems and research was presented in [33, 35, 70, 73, 74, 86–88, 91]. Gavalas et al. [34] present the most recent survey on algorithmic approaches for this class of problems. When applied in logistic planning tools, which is also the main focus of this paper, a vertex represents a customer and the reward reflects the profit margin when the customer is served. The aim is to select the combination of customers and sequence that maximizes the total reward [37, 43, 82]. Other applications of the OP include determining which suppliers to audit to maximize recovered claims [42], the planning of scouting tours to

*Corresponding author: Ghent University, Technologiepark 903, 9052 Zwijnaarde, Belgium, Tel.: +32 9 264 54 95, Fax.:+32 9 264 58 47
*Email addresses:* cedric.verbeeck@ugent.be (C. Verbeeck ), pieter.vansteenwegen@kuleuven.be (P. Vansteenwegen), elhoussaine.aghezzaf@ugent.be (E.-H. Aghezzaf)

different high schools to scout sport players [6, 10] and pharmaceutical representatives who visit doctors to build brand awareness [94]. Most of these applications are covered in more detail in the survey paper on the OP by Vansteenwegen et al. [84].

As a result of the traffic intensification and the limited capacity of a road network, traffic congestion has become a daily phenomenon. Congestion causes late arrivals at customers and increases transportation cost components. Therefore, accounting for and avoiding traffic congestion has a large potential for cost savings [46]. In general, congestion causes two undesirable effects to the transportation users: on the one hand it causes an increase of expected travel time due to a reduction of the mean speed and queuing, and on the other hand it increases the variability of travel times. Therefore, congestion is modelled by making various assumptions about the travel time required to traverse an arc. If the travel time is a fixed value it is defined as being deterministic. If this travel time is not a fixed value but the value is subject to variations, the travel time is said to be stochastic. On the other hand, a distinction should be made between time-independent and time-dependent travel times. If the time to traverse an arc is not dependent on the departure time during the day, it is considered to be time-independent and can be represented as a single value. When the travel time changes based on the departure time, the travel time is said to be time-dependent. Therefore, in a realistic road network, travel times are both time-dependent and stochastic. Due to possible traffic jams, the travel time in a road network is dependent on the hour of the day. Moreover, many factors (weather conditions, congestion, accidents, etc.) cause that one can never be sure when to arrive precisely at the destination.

Congestion itself is typically a local matter, it can be observed and confined within certain parts of the network and affects a limited number of arcs. However, the consequences of congestion can propagate through a much larger part of the network. Local arc travel time variation is experienced by various road network users travelling on different routes, whose reactions causes changes in the traffic patterns elsewhere. For that, even if congestion effects are observable and quantifiable at the arc level, it is important to extrapolate their effects to the route level. This concept is more thoroughly discussed by Boyce et al. [9]. Since the decisions of road users rely on accurate and reliable travel time estimation and prediction, knowing the variability of travel time might be as important as knowing its expected value.

Research on travel time variability and route choice models for point-to-point navigation already exists and therefore we refer to Corthout et al. [15] for more information. However, the mainstream vehicle routing research incorporates as travel time estimation, the mean route travel time or some measures of travel time reliability (i.e., 10th and 90th percentile). As a consequence, route travel time variability is rarely considered in vehicle routing applications. The research described in this paper aims to show that, given the growing availability and detail of travel time measurements, it is also possible and worthwhile to estimate route travel time distributions and link it directly to the likelihood of collecting a reward or penalty at the customer. Furthermore, we will measure the impact of route variability on the objective function.

Based on the two already mentioned travel time classifiers, four different variants of the orienteering problem can be devised and they are listed in Table 1.

Table 1: Overview on orienteering problems with different travel time properties

| | time-independent | time-dependent |
|---|---|---|
| deterministic | OP | TD-OP |
| stochastic | OPSW | TD-OPSW |

In the classic OP the travel time is deterministic and time-independent. This means that in the problem formulation the travel time is represented as a single constant value. Secondly, travel time information can have an amount of uncertainty attached to it but still be independent of the departure time. The orienteering problem with stochastic travel time information is called the orienteering problem with stochastic weights (OPSW). The travel time is represented by a single probability distribution. Thirdly, the orienteering problem with time-dependent and deterministic travel time information is called the time-dependent orienteering problem (TD-OP). Recall that in this case, the travel time is dependent on the hour of the day the specific arc is taken. Therefore, it can no longer be seen as a single value but as a function of values dependent on the departure time. As thoroughly discussed by Verbeeck et al. [89], the time-dependent problem formulation allows to model the deterministic congestion related issues in routing problems and multi-modal applications for logistic or tourist trip planners. In the fourth variant of the orienteering problem the

travel time is both stochastic and time-dependent. Instead of being represented by a single value, the travel time at a given hour of the day is now represented by a probability distribution. So the travel time on each arc is represented by different probability distributions, each dependent on the departure time. This leads to the time-dependent orienteering problem with stochastic weights (TD-OPSW). In this paper, we investigate the orienteering problem with both time-dependent and stochastic travel times.

In addition to the modelling of the travel time on the arc, the problem considered in this paper, assumes that each vertex has a time window (opening time and closing time) and a service time. Considering time windows in the time-dependent orienteering problem makes this latter more realistic since both tourist attractions and customers have opening hours in real-life situations. It is therefore an obvious step in the direction of modelling and solving realistic routing problems [2]. On the other hand, by incorporating time windows the resulting waiting times, which postpone the departure time at a vertex, directly affect the travel times in a time-dependent problem context. In order to develop accurate travel time predictions it is necessary to take into account time windows. For example, departing earlier at a vertex to avoid congestion might turn out to be not useful if you can not serve this subsequent vertex at the scheduled arrival time because of the opening time of the time window. This makes the problem significantly more difficult than the regular time-dependent orienteering problem. Furthermore, due to the stochastic nature of the travel times, waiting and arriving late at vertices also has an influence on the variability of departure times and therefore also on the variability of the total time of the route (See Section 4.1). Therefore, the problem discussed in this paper is called the time-dependent orienteering problem with stochastic weight and time windows (TD-OPSWTW).

This paper makes several contributions to the state of the art. Firstly, the TD-OPSWTW is introduced and the impact of time windows on the complexity of the time-dependent stochastic travel time calculation is discussed. Secondly, two metaheuristic methods for the TD-OPSWTW are compared and the results are analysed. Important construction concepts and practical insights are presented to stimulate further research. Thirdly, a set of problem instances based on a large and real road network are made available together with an efficient pre-processing method which allows to speed up significantly any solution method for the orienteering problem, or related problems such as the TSP and VRP, with time-dependent and/or stochastic travel times. Finally, the impact of (not) accounting for time-dependent stochastic travel times for the orienteering problem is investigated and stresses the importance of taking into account a more realistic representation of travel times.

Following an extensive literature review in Section 2, the TD-OPSWTW is further described in Section 3. Then, the solution method and instance generation procedure are proposed respectively in Section 4 and Section 5. The most important part of the proposed solution method is the estimation algorithm that provides the departure time distribution when travelling from one vertex to another. This algorithm can also be applied to other vehicle routing problems with time-dependent stochastic travel times and time windows. The solution method is experimentally evaluated in Section 6. Section 7 concludes this paper and discusses possible future work.

## 2. Literature Review

The name of the orienteering problem derives from the sport game of orienteering [12, 82]. In this game, individual competitors start at a specified control point and try to maximize their reward by visiting checkpoints and returning to the control point within a given time frame. Each checkpoint has a known reward and the objective is to maximize the total collected reward. A comprehensive survey on the OP and its extensions can be found in Vansteenwegen et al. [84].

Apart from our own research [89] and [90], time-dependent extensions of the OP(TW), called time-dependent orienteering problem (with time windows) or TD-OP(TW), have, to the best of our knowledge, only been studied by [1, 29, 33, 54]. We refer to Verbeeck et al. [89] and Verbeeck et al. [90] for a summary of the research on the TD-OP(TW) and we will focus here on the stochastic aspects of the problem under consideration.

There exists three different types of stochastic orienteering problems: orienteering problems with stochastic weights (OPSW) or travel times as called in this paper, orienteering problems with stochastic profits (OPSP) and orienteering problems with both stochastic profits and weights called stochastic scouting problems (SSP). The TD-OPSWTW discussed in this paper belongs to the stochastic weight type. The OPSP was introduced by Ilhan et al. [42] and the SSP was presented by Barros & Evers [6].

Next, solution methods for the OPSW were presented in [11, 24, 25, 63, 64, 79, 81]. Teng et al. [81] study the time constrained TSP where both the travel and service time are independent discrete random variables. The

objective is to maximize the total profit collected on a route while a limit is set to the total travel and service time of a route. A penalty is taken proportional to the amount in excess of this time limit. This problem is formulated as a two-stage stochastic problem with recourse, where in the first stage a subset of customers are sequenced before the travel and service time are realized and in the second stage the expected penalty is imposed on the objective function. Unlike the TD-OPSWTW, the problem discussed by Teng et al. [81] is limited to discrete travel and service time distributions. The problem only uses a single penalty parameter as a mechanism for maintaining the feasibility of the solution. Tang & Miller-Hooks [79] propose a selective TSP with discrete stochastic service times and allow readily extension for discrete stochastic travel times as well. They formulate the problem as a chance-constrained stochastic integer program where the objective is to maximize the total profit collected minus the travel cost from a route while restricting the probability that the route length exceeds a certain threshold. The problem introduced by Tang & Miller-Hooks [79] incorporates deadlines via a chance constraint rather than modelling the economic cost of their violation using a penalty as in the case of the TD-OPSWTW.

Campbell et al. [11] propose an orienteering problem with stochastic travel and service times (OPSTS) where a customer specific penalty is incurred for each scheduled customer not reached before a given known general deadline. However, contrary to the regular orienteering problem, the routes do not need to end at an end vertex. They investigate special versions of the problem that can be solved optimally and present variable neighbourhood search heuristics to solve the general versions of the problem. In comparison with the TD-OPSWTW discussed in this paper, time-dependent travel times, time windows and the constraint that routes need to start and end at the predefined vertex were added to the problem formulation of Campbell et al. [11] and the service times are deterministic. Apart from these additions the objective function of the TD-OPSWTW is very similar to the one proposed by Campbell et al. [11]. Evers et al. [24] compare a robust optimisation to the two-stage orienteering problem with stochastic weights using a case study. Contrary to the two-stage approach of Teng et al. [81], during the second stage no penalties are imposed but the recourse action of returning to the end vertex is made when the remaining capacity equals the expected weight required to return from the current location to the end depot. A certain amount of extra capacity is assumed to be available to cover for possible late arrivals at the end vertex. In Papapanagiotou et al. [64] and its extension Papapanagiotou et al. [63], the objective function of the OPSTS proposed by Campbell et al. [11] is further investigated. The authors compare the effectiveness of an exact objective evaluation versus a Monte Carlo sampling method and a hybrid method which combines aspects of the latter two.

Evers et al. [25] study an orienteering problem with a stochastic weight on each arc and a hard constraint on the total weight of a route. They introduce a two-stage recourse model where a recourse action of aborting the route (returning to the depot) is taken in the second stage. This action is executed when the remaining capacity equals the expected weight required to return from the current location to the end depot. They propose a sample average approximation procedure to solve the problem, in which the objective function is approximated using Monte Carlo sampling. Lau et al. [51] present the time-dependent orienteering problem with stochastic weights and call it the dynamic orienteering problem with (discrete) stochastic travel times. The authors propose, apart from the time-dependent stochastic travel times, a risk-sensitive criterion to allow for different risk preferences and develop a branch-and-bound algorithm and a local search algorithm to solve the DSOP. They also provide two approaches to calculate the travel time distributions: a sampling approach and a matrix-based approach. Unlike Campbell et al. [11], there is no customer specific penalty, only a general penalty for a route with a length that exceeds the maximum allowed travel time.

The orienteering problem with stochastic weights and time windows is discussed by [22, 94]. Firstly, Evers et al. [22] extend the orienteering problem with uncertain travel and service times, time windows and the addition of new vertices (targets) during the flight, for the case of online stochastic UAV mission planning. They call this problem the maximum coverage stochastic orienteering problem with time windows (MCS-OPTW), which generates a route with a maximum expected profit of targets that are known in advance. During the execution of the route, a heuristic similar to Vansteenwegen et al. [85] is used to re-plan the route each time a target is visited. Apart for the time-independent stochastic travel times, the difference with the TD-OPSWTW is that there is no penalty received for scheduled vertices that are not visited and late arrivals at the depot are also allowed. Furthermore, for the TD-OPSWTW all vertices are known in advance and a route needs to be found. Secondly, Zhang et al. [94] propose a stochastic orienteering problem with time windows where the travel time is deterministic but the waiting time at a customer is a discrete stochastic variable dependent on the arrival time at a customer. When serving a customer within its time window a reward is gained, otherwise a penalty is taken. The authors propose an analytical formula to compute the expected reward from

a route that takes into account the recourse actions of not travelling to a scheduled customer and deciding not to wait at a customer. Furthermore, they propose a VNS heuristic similar to Campbell et al. [11] to solve this problem. The difference with the TD-OPSWTW proposed in this paper resides in the fact that the travel times are time-dependent stochastic variables and the waiting time originates from arriving too early at a customer. Furthermore, no recourse actions are modelled or taken into account during the evaluation of possible solutions.

In general, these solution methods do not take into account the impact of time windows on the convolution of the departure time and travel time distribution. The proposed sampling approaches only work because, when no time windows are considered, independent distributions of the same type (discrete and gamma distributions are used in most cases) can be added together. However, in case of time windows, the resulting waiting times or late arrivals significantly complicate the calculation of the departure time distribution. In this paper, the aim is to find a route that takes into account stochastic travel times and time windows during the optimization process.

Existing solution methods for the related time-dependent vehicle routing problem (TD-VRP) described in [21, 28, 38, 41, 46, 71, 83, 95] and the time-dependent vehicle routing problem with time windows (TD-VRPTW) which can be found in [5, 13, 16, 19, 27, 39, 48, 57, 58, 60, 62, 65, 72, 92] provide inspiration on how to efficiently deal with time-dependency and time windows. Most of these works are also mentioned by the survey paper about time-dependent routing problems [36]. This paper also provides an overview on the various speed models that are developed by Ahn & Shin [3], Fleischmann et al. [28], Ichoua et al. [41], Malandraki & Daskin [58] to model more realistic congestion behaviour between vertices.

As far as solution methods are concerned, tabu search is the most commonly applied metaheuristic for the TD-VRP(TW) [21, 41, 57, 60, 62, 71, 83, 92, 95]. Other algorithms that have been developed for the TD-VRP(TW) include a genetic algorithm [38], a heuristic combining route construction and route improvement [13, 27, 65], a variable neighbourhood search approach [48], an iterated local search [39] and three ant colony systems [5, 19, 95]. Osvald & Stirn [62] present a distribution problem of fresh vegetables in which the perishability represents a critical factor. They apply tabu search to this problem and test their algorithm on data of the Slovenian food market. They achieved improvements of up to 47% reduction in perished goods. Kok et al. [47] presented a heuristic based on dynamic programming for TD-VRP while considering driving hour regulations. Kuo [49] studied a variant of TD-VRP aiming to minimize the total fuel consumption. The authors proposed a simulated annealing procedure for finding the routing plan with the least total fuel consumption. Experiments showed that the proposed method provided a 24.6% improvement in fuel consumption over the method based on minimizing transportation time and a 22.7% improvement over the method based on minimizing transportation distances. Maden et al. [57] proposed and applied a tabu search heuristic for the TD-VRPTW on real data from an electrical wholesale company in the SouthWest of the UK. The proposed approach led to savings in $CO_2$ emissions of about 7%. Based on the heuristic of Maden et al. [57], Ehmke et al. [21] test a planning system for city logistics service providers efficiency and adaptability to time-dependent planning data sets and the quality and structure of resulting delivery tours. Soler et al. [72] transform the TD-VRPTW into an asymmetric capacitated vehicle routing problem in order to solve it with a state-of-the-art exact and heuristic solution method. Kok et al. [46] use a time-dependent shortest path algorithm together with a restricted dynamic programming heuristic to tackle real-life TD-VRP test instances. Dabia et al. [16] developed a branch-and-price algorithm for the TD-VRPTW. They used new dominance criteria which enabled them to solve some small and medium sized instances. Nguyen et al. [60] introduce a tabu search meta-heuristic for the time-dependent multi-zone multi-trip vehicle routing problem with time windows. A diversification strategy, guided by an elite solution set and a frequency-based memory, is used to search for unexplored good regions. Zhang et al. [95] developed a hybrid algorithm that integrates both ant colony system and tabu search algorithms for the time-dependent vehicle routing problem with simultaneous pickup and delivery. Wen & Eglese [92] present a new algorithm for TD-VRPTW inspired by tabu search and shows how costs for freight distribution may be influenced by traffic conditions using a case study in the London area. Setak et al. [71] present the time-dependent vehicle routing problem in a multigraph where one vertex can be accessible from another with more than one arc and proposed a tabu search solution method for this problem.

Solution methods for the vehicle routing problem with stochastic travel times and time windows are presented in [4, 20, 55, 68, 76, 78]. Ando & Taniguchi [4] propose a genetic algorithm and focus on small-sized problem instances (up to 25 vertices) and Russell & Urban [68] develop a tabu search method where the performance of the solution approach is evaluated on well-known problem instances with 100 vertices. Li et al. [55] proposed a heuristic algorithm based on a tabu search method and tested their procedure on problem instances with up to 100 vertices. In

the problem proposed by Taş et al. [76] and Taş et al. [78], the vertices have soft time windows. In [76] the authors propose a tabu search method together with a post-optimization method, while in [78] a column generation procedure and a branch-and-price solution approach are applied. Ehmke et al. [20] developed a chance-constrained approach where restrictions are placed on the probability that individual time window constraints are violated. They propose a way to estimate this probability for all customers. The approach considers how to compute the start-service time and arrival time distributions for each customer. The computational experiments show how the solutions change for well-known data sets across different levels of customer service and two types of travel time distributions. This approach is similar to the estimation algorithm proposed in this paper as the arrival and departure time are approximated by a normal distribution with estimated mean and variance. However, apart from the addition of time-dependency in this paper, in the case of a late arrival a customer can still be serviced in the paper of Ehmke et al. [20].

Research dedicated to the stochastic time-dependent vehicle routing problem with time windows or STD-VRPTW can be found in [53, 75, 77]. In [53] the time-dependent travel times are the result of a stochastic process due to traffic congestion. The proposed queueing models are solved for problem instances with up to 80 vertices by an algorithm based on a tabu search method. In the problem formulation discussed by Taş et al. [77] vertices have soft time windows. The authors propose tabu search and an adaptive large neighbourhood search to solve this problem. Sun et al. [75] propose a method to convert the STD-VRPTW into a TD-VRP problem and use a formerly presented route construction algorithm to solve the converted problem.

Time-dependent extensions for the dial-a-ride problem were investigated in [30, 31, 69]. Fu [30] and Fu [31] investigate dial-a-ride paratransit scheduling problems subject to tight service time constraints and time-varying, stochastic traffic congestion. Firstly, Fu [30] investigates the potential effects of these variations on the operational characteristics of a paratransit system such as vehicle productivity and schedule reliability are examined. A series of numerical experiments was performed on a practical problem from the city of Edmonton, Alberta, under hypothetical travel time variation patterns. A set of recursive relations is identified by Fu [31] to approximate the distribution parameters of arrival times at individual stops of a given route. This allows to extend conventional heuristic algorithms with only a marginal increase in computational complexity. Schilde et al. [69] adapted four existing metaheuristics for the stochastic time-dependent dynamic dial-a-ride problem using statistical information available about historical accidents.

The earliest studies on modelling the travel time and speed on arcs of the network were performed by Berry & Belmont [7]. The speed on an arc was found to be normally distributed. Kharoufeh & Gautam [45] showed that travel times were roughly normally distributed, although slightly skewed, indicating that a log-normal distribution might be interesting as an alternative. The results of Taniguchi et al. [80] and Kwon et al. [50] show that there is always a certain minimum time needed to cover the distance. After this minimum time, the probability corresponding to travel times higher than this minimum, increases rapidly to a maximum after which this probability slowly decreases with a long tail. Therefore, Taniguchi et al. [80] advise to use log-normal distributions instead of normal distributions. Furthermore, Noland & Polak [61] and Clark & Watling [14] also assume route travel times to be well approximated by normal or log-normal distributions, under the condition that arc travel times are (sufficiently) independent variables on a day-to-day time scale, or that the network is under user equilibrium constraints. Although assuming normality offers numerous analytical and computational advantages, it imposes some unrealistic restrictions such as symmetry and non-zero probability of negative travel times. Subsequently, the conventional (2-parameter) log-normal distribution is bounded below by zero leading to unreasonable free flow speeds.

The distributions of the travel times most commonly applied in the field of vehicle routing problems to solve road logistic problems are uniform, normal, log-normal, shifted gamma and gamma distributions [11, 26, 30, 31, 44, 53, 55, 63, 68, 76–79]. The gamma distribution is the most commonly used [11, 26, 53, 63, 68, 76–78]. These distributions can be estimated using different data sources such as floating car data, automatic vehicle identification and inductive loop detectors [56, 66, 93, 96]. Although the calculation of arc travel time distributions is possible through these data sources, the relationship between these distributions and travel time distributions at the route level is not straightforward [32, 40]. Because arc flows and travel times are often highly mutually correlated, the calculation of route travel time variability can not always be calculated by summing up the arc travel time variability. However, in this paper we will assume that the travel times on the arcs are uncorrelated. Correlation could, for instance, be incorporated by developing a network-consistent, time-dependent travel time layer like in the work of Schilde et al. [69] and Lecluyse et al. [52].

Next to this extensive literature survey, the main contribution of this paper is to show that it is possible to solve the

stochastic time-dependent version of the orienteering problem with (hard) time windows without using for example a 2-stage stochastic programming approach or computational expensive Monte Carlo simulations. Instead a meta-heuristic is proposed that takes the stochastic aspects of the problem into account during the optimization process. Furthermore, the significant impact of time windows on the variability of the solution is theoretically and empirically shown as time windows are often not included in the problem formulations of the orienteering problem with stochastic travel times. Therefore, in this paper, for the problem instances based on real-life data, time slots with a length of 15 minutes will be used and the stochastic travel times are assumed to be normally and independently distributed. Contrary to the commonly used gamma distribution, the normal distribution is chosen as it facilitates the travel time estimation process because of its analytical properties (convolution, truncation, symmetry). In future research the focus could be placed on the adaptation of the solution procedure to handle the more realistic probability distributions such as the gamma distribution.

## 3. Problem description

The TD-OPSWTW is defined based on the formulation proposed by Campbell et al. [11]. Let $V_c = \{1, ..., v\}$ be a set of serviceable vertices. Vertex 1 represents the start vertex and vertex $v$ represents the end depot. Furthermore, it is assumed that there is an arc $(i, j)$ between all $i$ and $j$ in $V_c$. Each vertex $i \in V_c$ has a deterministic non-negative reward $r_i$ as well as a deterministic non-negative penalty $e_i$. The reward is earned when visiting the vertex between its opening time $o_i$ and closing time $c_i$ and the penalty is incurred if the arrival occurs after its closing time which means the vertex is not visited. This penalty represents direct payment or a loss of goodwill. In reality, there is a cost associated with a failure to service a planned customer. Note that deterministic models generally do not (have to) consider this cost. The time window is defined for the start of the service and not the end. Let $T_{i,j,t_d}$ be a non-negative stochastic variable representing the time required to traverse arc $(i, j)$ when departing at time $t_d$. It is assumed that the distribution on $T_{i,j,t_d}$ is known for all $i$, $j$ and $t_d$. Next, let $s_i$ be the deterministic non-negative service time at vertex $i$.

Let the random variable $A_i$ be the planned arrival time at vertex $i$. For a realization (actually observed value ) $\overline{A_i}$ of $A_i$, we assume that a reward $r_i$ is earned for $\overline{A_i} \leq c_i$ and a penalty $e_i$ is incurred otherwise. Let $\omega$ be a route or sequence of the vertices in the selected set $M \subseteq V_c$ which begins at the vertex 1 at time $t_0$ and ends at end vertex $v$. The opening time of the start and end vertex is set equal to $t_0$ and their rewards are set to 0. Subsequently, the closing time of the end vertex is set equal to $t_0 + t^{max}$ and its incurred penalty is set equal to $e_v$. This way a penalty is also incurred if the route length exceeds the predetermined length, $t^{max}$. The expected reward of the route is equal to the following equation in which $\mathcal{P}$ stands for the probability:

$$\sum_{i \in \omega} [\mathcal{P}(A_i \leq c_i) \cdot r_i - (1 - \mathcal{P}(A_i \leq c_i)) \cdot e_i] \tag{1}$$

We seek a route $\omega^*$ such that the expected reward of $\omega^*$ is larger than or equal to the expected reward of $\omega$ for every $\omega$.

Furthermore, the time is discrete and divided into in $\kappa$ time slots. This means that for every vertex $i$ to $j$ and for every time slot $t \in \kappa$ there exists a normally distributed travel time $(T_{i,j,t})$. Finally, let $\underline{tp}_t$ and $\overline{tp}_t$ represent the moment at which time slot $t$ starts and ends respectively.

## 4. Solution method

The proposed solution method for the TD-OPSWTW is an ant colony optimization algorithm that takes into account the expected reward and the stochastic travel times using an estimation algorithm. Before this method is discussed in more detail, the Monte Carlo sampling and the estimation algorithm that provide a way of calculating the travel time is described in Section 4.1.

### *4.1. Arrival time and objective function calculation*

Given a sequence of vertices, an algorithm is required to calculate the arrival time distribution for every included vertex. This distribution allows to determine the probability that a reward or a penalty is collected and therefore

allows to calculate the expected profit of the vertex. The output of this algorithm is the expected profit when executing the proposed route in a stochastic context. In this research, two methods are proposed for this task: a Monte Carlo simulation and an estimation algorithm.

In general, three cases can occur when travelling from one vertex to another vertex:

1. The traveller arrives within the opening hours of the vertex which enables the traveller to collect the reward after servicing the vertex. A service time is added to the arrival time.
2. The traveller arrives before the opening time of the vertex which forces him to wait first and collect the reward after servicing the vertex. The departure time is equal to the opening time plus the service time.
3. The traveller arrives after the closing time of the vertex which forces him to collect the penalty without servicing the vertex. No service time is added in this case and the departure time is equal to the arrival time.

Recall that the closing time of the end depot is set equal to $t_0 + t^{max}$ and the penalty, $e_v$ is defined. Therefore, the third case also penalizes for the fact of having a total time larger than $t^{max}$.

Given the fact that our travel times are assumed to be normally distributed, the resulting arrival distribution which is equal to the convolution of the departure time distribution and the corresponding travel time distribution will also be normally distributed if the departure time is also normally distributed. In most cases this departure time will be normally distributed. However, problems arise when waiting is necessary at a vertex, when the vertex can not be served or when the width of the departure time distribution spans multiple time slots. As will be discussed in the three subsequent subsections, in such cases the arrival and/or departure time distribution and the resulting distribution can no longer be categorized as normal. Therefore, executing the convolution of such an irregular departure time distribution with a subsequent normal travel time distribution becomes a complex task. Sampling these distributions during the execution of a metaheuristic is computationally expensive. Therefore, an estimation algorithm (Section 4.1.5) is used during the execution of the metaheuristic and its final solution sequence is validated using a Monte Carlo simulation (Section 4.1.4).

These three cases will now be explained, together with the way they are dealt with during the execution of the metaheuristic, i.e., by the estimation algorithm. Note that for a given random variable D, $\phi_D$ stands for its probability density function, $\Phi_D$ stands for its cumulative distribution function and $\Phi_D^{-1}$ represents its inverse cumulative distribution function. All equations derived in the subsequent sections were obtained and validated using the commercial software Mathematica 10.0 combined with the mathStatica 2.7 package.

### *4.1.1. Width of departure time distribution spanning multiple time slots*

When calculating the arrival time distribution, a time slot needs to be found which corresponds with a departure time distribution. Based on this time slot the correct travel time distribution can be obtained. Recall that each arc can have a different travel time distribution for each time slot. Therefore, it is important that the departure time distribution is convoluted with the travel time distribution that matches the time slot of that departure time distribution. However, the departure time distribution may span more than one time slot. If the mean departure time would be used to find an appropriate time slot, a part of the departure time distribution is convoluted with the wrong travel time distribution.

Let $D \sim \mathcal{N}(\mu_D, \sigma_D^2)$ represents the departure time and let $g > 0$ be the start time of a time slot. Furthermore, let $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ be the travel time corresponding to the first time slot and let $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ be the travel time corresponding to the second time slot. Then, the random variable $A$ representing the arrival time is equal to the following piecewise function:

$$A = \begin{cases} D + X_1 & \text{if } D \leq g \\ D + X_2 & \text{if } D > g. \end{cases}$$

Let $erf(x)$ be the error function:

$$\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}$$

The probability density function of A equals:

$$\phi_A(t) = \frac{\frac{e^{-\frac{(\mu_D+\mu_1-t)^2}{2\left(\sigma_D^2+\sigma_1^2\right)}}\left(\operatorname{erf}\left(\frac{\sigma_1^2(g-\mu_D)+\sigma_D^2(g+\mu_1-t)}{\sqrt{2}\sigma_D\sigma_1\sqrt{\sigma_D^2+\sigma_1^2}}\right)+1\right)}{\sqrt{\sigma_D^2+\sigma_1^2}} - \frac{e^{-\frac{(\mu_D+\mu_2-t)^2}{2\left(\sigma_D^2+\sigma_2^2\right)}}\left(\operatorname{erf}\left(\frac{\sigma_2^2(g-\mu_D)+\sigma_D^2(g+\mu_2-t)}{\sqrt{2}\sigma_D\sigma_2\sqrt{\sigma_D^2+\sigma_2^2}}\right)-1\right)}{\sqrt{\sigma_D^2+\sigma_2^2}}}{2\sqrt{2\pi}}. \quad (2)$$

The mean of the arrival time distribution is equal to:

$$\mu_A = \mu_D + \frac{1}{2}\mu_1\left(\text{erf}\left(\frac{g-\mu_D}{\sqrt{2}\sigma_D}\right)+1\right) + \frac{1}{2}\mu_2\left(\left(-\text{erf}\left(\frac{g-\mu_D}{\sqrt{2}\sigma_D}\right)-1\right)+1\right). \tag{3}$$

The variance is equal to:

$$\begin{aligned}\sigma_A^2 =& \frac{1}{4}e^{-\frac{(g-\mu_D)^2}{2\sigma_D^2}}\left(e^{\frac{(g-\mu_D)^2}{2\sigma_D^2}}\left((\mu_1-\mu_2)^2\left(-\left(\text{erf}\left(\frac{g-\mu_D}{\sqrt{2}\sigma_D}\right)^2-1\right)\right)\right.\right.\\ &+2\sigma_1^2\text{erf}\left(\frac{g-\mu_D}{\sqrt{2}\sigma_D}\right)+2\left(-\sigma_2^2\left(\text{erf}\left(\frac{g-\mu_D}{\sqrt{2}\sigma_D}\right)-1\right)+2\sigma_D^2+\sigma_1^2\right)\Big)\\ &\left.+4\sqrt{\frac{2}{\pi}}(\mu_2-\mu_1)\sigma_D\right).\end{aligned} \tag{4}$$

Details on the derivations of these equations can be found in Appendix A.

As an example, we choose the departure time distribution $D \sim \mathcal{N}(11, 1)$. The start of time slot 2 is located at 12 ($g$), the travel time associated with time slot 1 is $T_1 \sim \mathcal{N}(1, 0.01)$ and the travel time associated with time slot 2 is $T_2 \sim \mathcal{N}(1.2, 0.25)$. Note that at this point no information concerning the destination vertex is needed since the arrival time distribution is calculated. This example is visualised in Figure 1. On the left panel we can see the departure time distribution in black, as well as the start and end of time slot 1 indicated by a black vertical line. The part of the distribution that is located after the end of time slot 1 is shaded in orange. The part of the distribution that is located before the start of time slot 1 is very small and is therefore ignored. In this case the part of departure time distribution ranging from 7 to 12 needs to be convoluted with $T_1$ while the part of departure time distribution ranging from 12 to 15 needs to be convoluted with $T_2$. On the right panel the real arrival time distribution is drawn in black.

As can be seen in this specific case, the original arrival time distribution does not look normally distributed but is relatively well approximated using a normal distribution. During the execution of the metaheuristic the arrival time distribution is assumed to be normally distributed with a mean and variance given by equation (3) and (4) respectively. The approximation for this specific case is drawn in dashed orange.

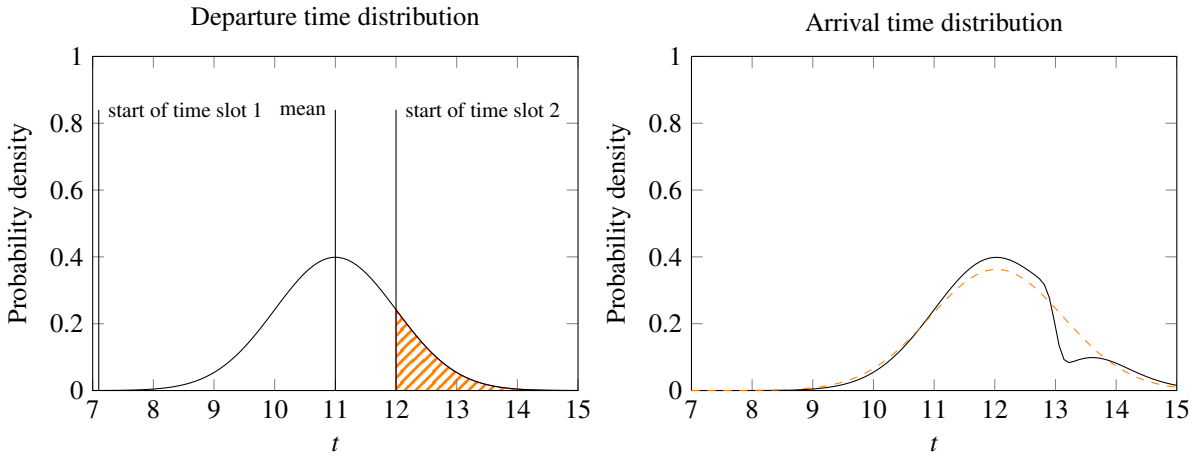



Figure 1: Effect of departure time spanning multiple time slots on arrival time distribution with $\mu_D = 11$, $\mu_1 = 1$, $\mu_2 = 1.2$, $\sigma_D^2 = 1$, $\sigma_1^2 = 0.01$, $\sigma_2^2 = 0.25$ and $g = 12$

#### *4.1.2. Arriving early at a vertex*

When a traveller arrives before the opening time ($o$) of a vertex, a waiting time and service time ($s$) are added to the arrival time $A \sim \mathcal{N}(\mu_A, \sigma_A^2)$. In a stochastic case, waiting at a vertex converts the normally distributed arrival time distribution into a departure time which is a combination of a mass point and a truncated normal distribution.

The random variable $D$ represents the departure time and equals the following piecewise function:

$$D = \begin{cases} o + s & \text{if } A \leq o \\ A + s & \text{if } A > o. \end{cases}$$

Its probability density function therefore equals

$$\phi_D(t) = \begin{cases} \frac{e^{-\frac{(\mu_A+s-t)^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A} & o + s - t < 0 \\ \frac{1}{2}\left(\text{erf}\left(\frac{-\mu_A - s + t}{\sqrt{2}\sigma_A}\right) + 1\right) & t = o + s. \end{cases} \tag{5}$$

The mean and variance ($\mu_D$ and $\sigma_D^2$) of this departure time distribution are given by the following equations:

$$\mu_D = \frac{1}{2}(o - \mu_A)\left(\text{erf}\left(\frac{o - \mu_A}{\sqrt{2}\sigma_A}\right) - 1\right) + \frac{\sigma_A e^{-\frac{(o-\mu_A)^2}{2\sigma_A^2}}}{\sqrt{2\pi}} + o + s, \tag{6}$$

$$\begin{aligned} \sigma_D^2 = \frac{1}{4} * \Bigg( 4 * \Bigg( & \frac{\sigma_A(\mu_A - o)e^{-\frac{(o-\mu_A)^2}{2\sigma^2}}}{\sqrt{2\pi}} - \frac{\sigma_A^2}{2} \Bigg) \text{erf}\left(\frac{o - \mu_A}{\sqrt{2}\sigma_A}\right) \\ & -(o - \mu_A)^2 \text{erf}\left(\frac{o - \mu_A}{\sqrt{2}\sigma_A}\right)^2 + \mu_A^2 + o^2 - \frac{2\sigma_A^2 e^{-\frac{(o-\mu_A)^2}{\sigma_A^2}}}{\pi} - 2\mu_A o + 2\sigma_A^2 \Bigg). \end{aligned} \tag{7}$$

Details on the derivations of these equations can be found in Appendix B.

As an example, we define an arrival time distribution with a mean of 9 and a variance of 1, the opening time is equal to 10 ($o$) and the service time ($s$) is set equal to 1. This example is visualised in Figure 2. On the left panel we can see the arrival time distribution in black and the opening time indicated by a black vertical line. The part of the distribution that is located before the opening time (84%) is shaded in orange. On the right panel the real (piecewise) departure time distribution is drawn in black. Since we have a 84% chance to arrive early, the real departure time consists of a mass point of 84% at time 11 (the sum of the opening time and the service time). The other part of the departure time distribution is formed by shifting 1 time unit to the right the part of the arrival time distribution that is located after the opening time.

During the execution of the metaheuristic, this piecewise departure time distribution is approximated as a normal distribution with a mean and variance given by equation (6) and (7) respectively. The approximation for this example is drawn in dashed orange.

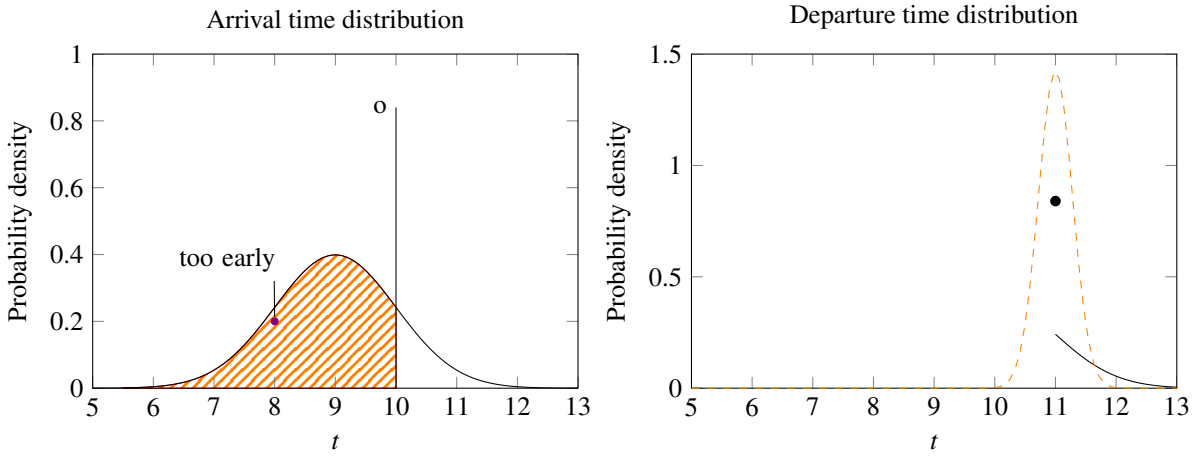
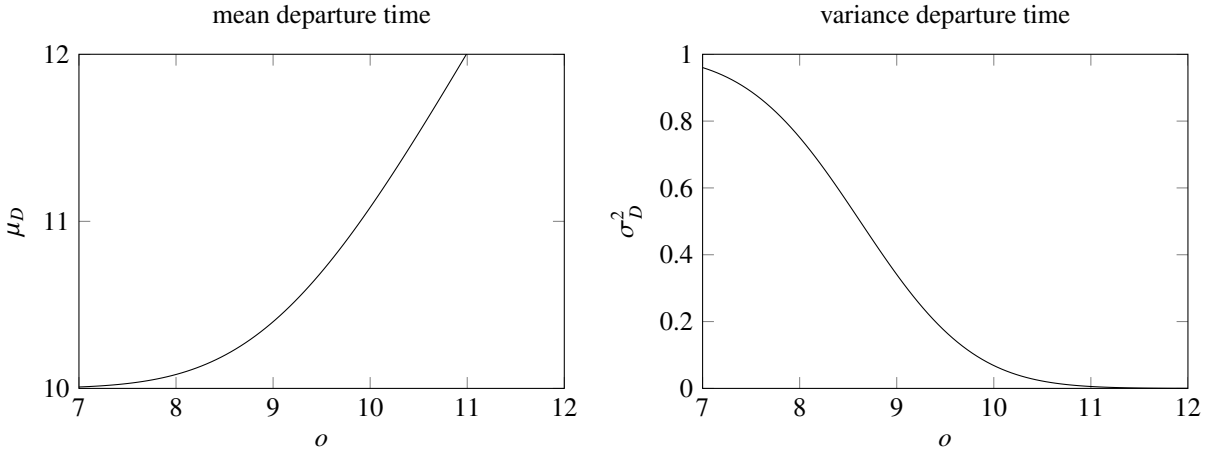



Figure 2: Arrival and departure time distribution in the case of an early arrival with $\mu_A = 9, \sigma_A^2 = 1, s = 1$ and $o = 10$ .

As can be seen in the expression of both the mean and the variance of the departure time distribution, the opening time has an effect on the departure time distribution. An example of this effect for an arrival time distribution with a mean equal to 9 and a variance and service time equal to 1 can be seen in Figure 3. Note that in this figure only the opening time is variable while the mean and the variance of the arrival time distribution are kept equal to 9 and 1 respectively. This means for example that an opening time greater than 9 corresponds to a scenario with a high probability of arriving early. In Figure 3a there can be seen that for a high probability of arriving early (right side of Figure 3a) the mean departure time is equal to the opening time plus the service time. However, for low probability of arriving early (left side of Figure 3a) the mean is equal to the mean arrival time (9) plus the service time (1) of the vertex. In between, a convex function limits the growth of the departure time when the opening time increases. On Figure 3b there can be seen that the variance decreases in a non-linear way when the probability of arriving early increases up to the fact that the departure time becomes deterministic. There can be concluded that arriving early at vertices decreases the variability in a TD-OPSWTW solution due to the waiting before the time window opens.




(a) (b)

Figure 3: Effect of the opening time on $\mu_D$ and $\sigma_D^2$ with $\mu_A = 9, \sigma_A^2 = 1$ and $s$ equal to 1.

*4.1.3. Arriving late at a vertex*

A vertex cannot be served for the part of the arrival time distribution that is larger than the closing time ($c$), so no service time should be added to this part of the arrival time distribution. Recall that the time windows are defined for the start of the service and not for the completion of the service.

When the arrival time distribution is defined as $A \sim \mathcal{N}(\mu_A, \sigma_A^2)$, the random variable $D$ represents the departure time and equals the following piecewise function:

$$D = \begin{cases} A + s & \text{if } A \leq c \\ A & \text{if } A > c. \end{cases}$$

Note that the transformation from $A$ to $D$ is not one to one. In particular, there is a zone of overlap where both sub functions of the piecewise function can contribute to one value of $D$.

The probability density function of $D$ is given by:

$$\phi_D(t) = \begin{cases} \frac{e^{-\frac{(t-\mu_A)^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A} & c + s \leq t \\ \frac{e^{-\frac{(\mu_A+s-t)^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A} & c \geq t \\ \frac{e^{-\frac{(\mu_A+s-t)^2}{2\sigma_A^2}} + e^{-\frac{(t-\mu_A)^2}{2\sigma_A^2}}}{\sqrt{2\pi}\sigma_A} & c < t < c + s. \end{cases} \tag{8}$$

The mean and variance ($\mu_D$ and $\sigma_D^2$) of the resulting departure time distribution are given by the following equations:

$$\mu_D = \frac{1}{2}\left(s\,\mathrm{erf}\left(\frac{c-\mu_A}{\sqrt{2}\sigma_A}\right) + 2\mu_A + s\right), \tag{9}$$

$$\sigma_D^2 = -\frac{1}{4}s^2\mathrm{erf}\left(\frac{c-\mu_A}{\sqrt{2}\sigma_A}\right)^2 - \sqrt{\frac{2}{\pi}}\sigma_A s e^{-\frac{(c-\mu_A)^2}{2\sigma_A^2}} + \frac{s^2}{4} + \sigma_A^2. \tag{10}$$

Details on the derivations of these equations can be found in Appendix C.

As an example, we define the arrival time distribution $A \sim \mathcal{N}(\mu_A, \sigma_A^2)$ with a mean of 16 and a variance of 1, the closing time ($c$) is equal to 17 and the service time ($s$) is set equal to 1. This example is visualised in Figure 4. On the left panel we can see the arrival time distribution in black and the closing time indicated by a black vertical line. The part of the distribution that is located after the closing time is shaded in orange. On the right panel the real (piecewise) departure time distribution is drawn in black and the proposed approximation in orange. Note that the elevated part of the original departure time distribution between 17 and 18 originates from both arrival times that were either on-time or too late. For example an arrival time equal to 16.5 is transformed to a departure time equal to 17.5 but an arrival time equal to 17.5 leads also to the same departure time of 17.5.

During the execution of the metaheuristic, this piecewise departure time distribution is approximated as a normal distribution with a mean and variance given by equations (9) and (10) respectively. The approximation for this example is drawn in dashed orange.

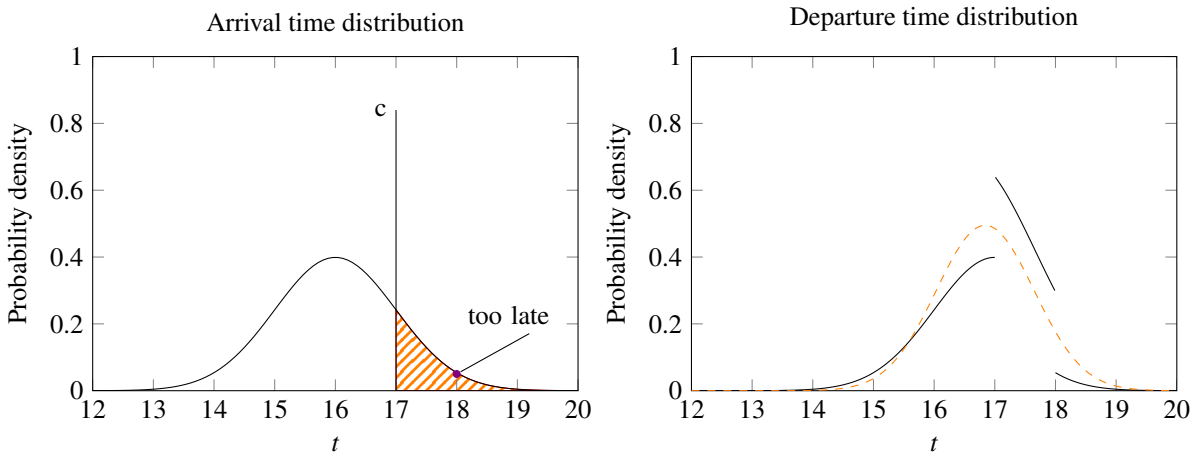



Figure 4: Arrival and departure time distribution in the case of a late arrival with $\mu_A = 16, \sigma_A^2 = 1, s = 1$ and $c = 17$.

Similarly to the effect of the opening time, an example of the effect of the closing time on the mean and variance of the departure time distribution for two different values of the service time is visualised in Figure 5. For the case where $s = 1$, the mean departure time increases and becomes closer to the default value of 17 as the probability of arriving late (higher values of $c$) increases (Figure 5a). The effect of the closing time on the variance resembles in this case a parabolic function which opens downwards, indicating that the variance first decreases for high probabilities of arriving late, then hits a minimum for the 50% probability of arriving late and then increases again when the probability of arriving late decreases (Figure 5b). For the case where the service time equals 4, the curve for the mean departure time follows a similar pattern as discussed above (Figure 5c). However, the curve of the variance shown in Figure 5d deviates strongly from Figure 5b. The variance increases for low probabilities of arriving late and achieves a maximum for the 50% probability whereafter the variance drops for high probabilities of arriving late. Finally, there can be concluded that late arrivals severely complicate the calculation of the departure time distribution.

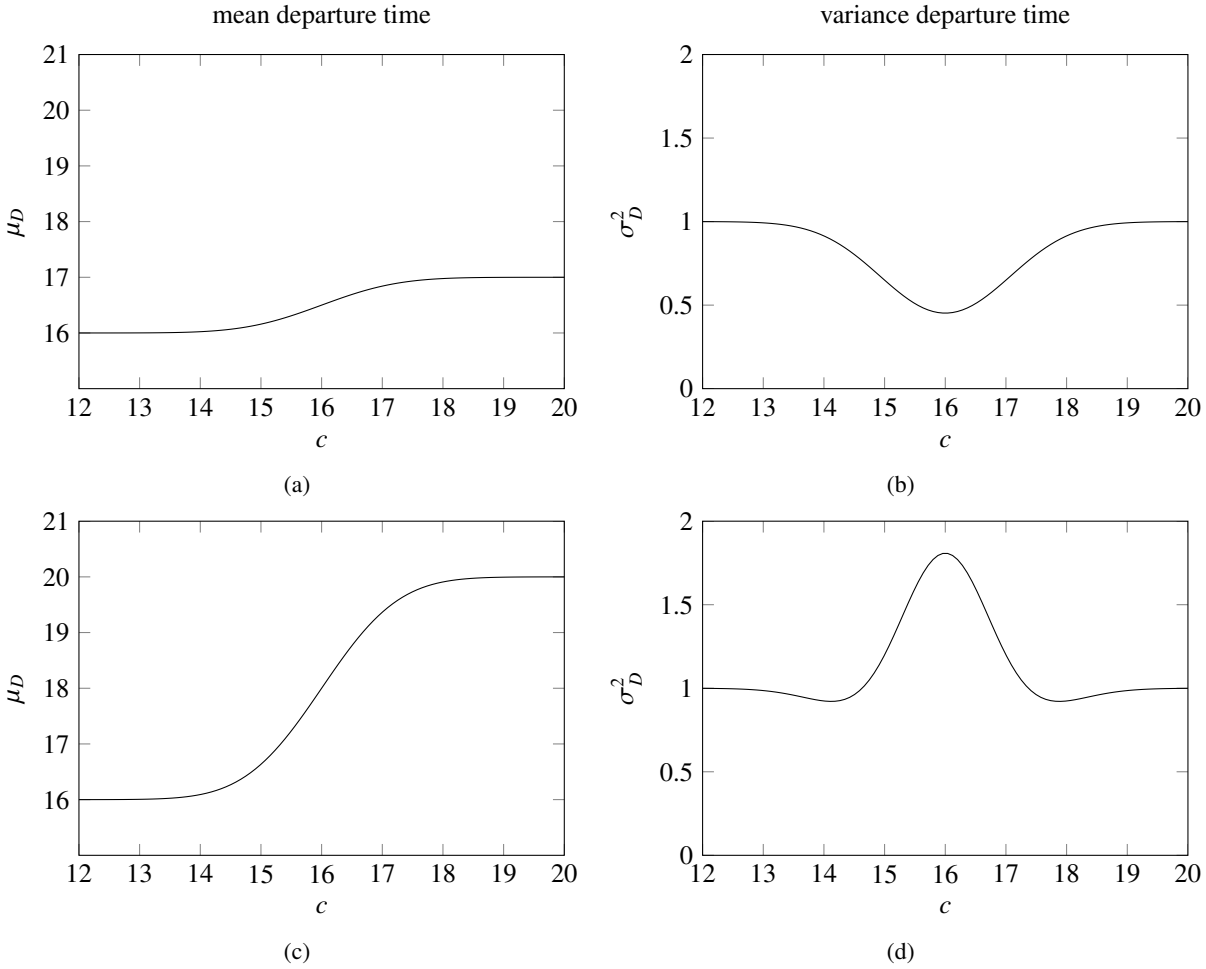



Figure 5: Effect of the closing time on $\mu_D$ and $\sigma_D^2$ with $\mu_A = 16, \sigma_A^2 = 1$ and $s$ respectively equal to 1 and 4.

#### *4.1.4. Monte Carlo simulation*

The pseudo code of the Monte Carlo simulation is provided in Algorithm 1. Since in every iteration the departure time is deterministic, it is possible to find exactly one corresponding travel time distribution with a deterministic departure time. Subsequently, a random percentile of this travel time distribution is chosen as travel time value. The deterministic arrival time therefore equals the sum of the deterministic departure time and the randomly chosen percentile of the travel time distribution. Afterwards, the three cases mentioned above are checked for this arrival time value and a corresponding reward or penalty is obtained. This process is repeated for every vertex in the solution. This algorithm returns the expected reward of the input solutions sequence by dividing the sum of all rewards belonging to the 10,000 iterations by the number of iterations. This maximum number of iterations is set equal to 10,000 since preliminary experiments showed that the objective function no longer shows significant deviations. Although not shown in Algorithm 1, this Monte Carlo simulation is also extended to capture the arrival time distribution at the end vertex of the solution. This allows to measure the difference in mean and variance between the arrival time distribution, originating from the Monte Carlo simulation, and the one produced by the estimation algorithm. This comparison can be found in Section 6.1.

#### *4.1.5. Estimation algorithm*

This algorithm provides an estimate of the departure time distribution at $j$ when travelling from vertex $i$ to vertex $j$ with a departure time $D_i \sim \mathcal{N}(\mu_{D_i}, \sigma_{D_i}^2)$ at $i$. The pseudo code is displayed in Algorithm 2. During the execution,

**Algorithm 1** Monte Carlo simulation- input: solution $\omega$, *maxiterations*

```
sumreward = 0
for iteration = 1 to maxiterations do
    reward = 0
    departuretime = t_0
    for i = 2 to |ω| do
        a = ω[i − 1]
        b = ω[i]
        k = find time slot(departuretime)
        rnd = random number in the open interval (0, 1)
        arrivaltime = departuretime + Φ^{-1}_{T_{a,b,k}}(rnd)
        if arrivaltime < o_b then
            departuretime = o_b + s_b
            reward = reward + r_b
        else if arrivaltime ≤ c_b then
            departuretime = departuretime + s_b
            reward = reward + r_b
        else
            reward = reward − e_b
            departuretime = arrivaltime
        end if
    end for
    sumreward = sumreward + reward
end for
expreward = sumreward/maxiterations
RETURN expreward
```

every distribution is assumed to be normal which helps speeding up the calculation of the convolution product of two distributions.

In general, to construct the arrival time distribution, the departure time distribution is convoluted with the matching normal travel time distribution (lines 10-12). The time slot $k$, corresponding to the mean of the departure time distribution, is selected to find the corresponding travel time distribution (line 1). However, if the proportion of the departure time distribution that is located in another time slot than the time slot that corresponds with the mean of the distribution, is bigger than 1%, the arrival time distribution will be approximated. If a part (larger than 1%) of the departure time distribution is situated in a lower time slot, then $g$ is set equal to $k$. The arrival time distribution is approximated using a normal distribution with a mean equal to equation (3) and a variance equal to equation (4) (lines 2-5). In the other case that a part (larger than 1%) of the departure time distribution is located in a subsequent time slot, then $g$ is set equal to $k+1$. The arrival time distribution is approximated using a normal distribution with a mean equal to equation (3) and a variance equal to equation (4) (lines 6-9).

After the convolution of the departure time and the travel time into the arrival time distribution, three scenarios can occur on arrival at vertex $j$: an early arrival, a late arrival or an on-time arrival. For an early arrival the departure time is approximated using a normal distribution with a mean equal to equation (6) and a variance equal to equation (7) (lines 14-16). In the case of a late arrival the departure time is approximated using a normal distribution with a mean equal to equation (9) and a variance equal to equation (10) (lines 17-19). An on-time arrival requires the addition of a service time to the mean of the arrival time distribution in order to obtain the departure time distribution. The variance remains unchanged (lines 20-22). Finally, this algorithm outputs the expected reward of vertex $j$.

**Algorithm 2** Estimation algorithm -input: vertex $i$, vertex $j$, departure time $D_i \sim \mathcal{N}(\mu_{D_i}, \sigma^2_{D_i})$

1: $k = find\,timeslot(\mu_{D_i})$
2: **if** $\Phi_{D_i}(tp_k) > 0.01$ **then**
3: $\quad g = tp_k$
4: $\quad \mu_{A_j}$ = eq. 3 $(g, D_i, T_{i,j,k-1}, T_{i,j,k})$
5: $\quad \sigma^2_{A_j}$ = eq. 4 $(g, D_i, T_{i,j,k-1}, T_{i,j,k})$
6: **else if** $\Phi_{D_i}(tp_{k+1}) - \Phi_{D_i}(tp_k) < 0.99$ **then**
7: $\quad g = tp_{k+1}$
8: $\quad \mu_{A_j}$ = eq. 3 $(g, D_i, T_{i,j,k}, T_{i,j,k+1})$
9: $\quad \sigma^2_{A_j}$ = eq. 4 $(g, D_i, T_{i,j,k}, T_{i,j,k+1})$
10: **else**
11: $\quad \mu_{A_j} = \mu_{D_i} + \mu_{T_{i,j,k}}$
12: $\quad \sigma^2_{A_j} = \sigma^2_{D_i} + \sigma^2_{T_{i,j,k}}$
13: **end if**
14: **if** $\Phi_{A_j}(o_j) > 0.01$ **then**
15: $\quad \mu_{D_j}$ = eq. 6 $(o_j, A_j)$
16: $\quad \sigma^2_{D_j}$ = eq. 7 $(o_j, A_j)$
17: **else if** $\Phi_{A_j}(c_j) < 0.99$ **then**
18: $\quad \mu_{D_j}$ = eq. 9 $(c_j, A_j)$
19: $\quad \sigma^2_{D_j}$ = eq. 10 $(c_j, A_j)$
20: **else**
21: $\quad \mu_{D_j} = \mu_{A_j} + s_j$
22: $\quad \sigma^2_{D_j} = \sigma^2_{A_j}$
23: **end if**
24: RETURN $D_j$

### 4.2. Pre-processing

A set of neighbours is defined for each vertex $i \in V_c$. This set, $NB_i$, is computed before the start of the solution procedure. A vertex $j$ is considered a neighbour of vertex $i$ if the following equation holds:

$$o_i + s_i + ff_{i,j} \leq c_j \tag{11}$$

In this equation, $ff_{i,j}$ represents the time-independent travel time. Subsequently, these neighbours are first stored in a list, sorted according to the following equation:

$$\frac{r_j}{s_j + ff_{i,j}} \tag{12}$$

In this way, neighbours with a higher reward and a smaller service and time-independent travel time are ranked first. If the number of neighbours in this list is greater than $NB_{max}$, only the first $NB_{max}$ neighbours are stored as the set of neighbours for the vertex under consideration. These neighbour sets are used in the construction method and the local search moves. In the remainder of this text, a vertex $j$ is called a neighbouring vertex of vertex $i$ if $j$ is included in the neighbour set of $i$. Various values where tested for $NB_{max}$ and a value equal to 45 seemed a good trade-off between a scalable computational time and solution quality.

### 4.3. Stochastic ant colony system (SACS)

The structure of the SACS is similar to the ant colony system proposed for the TD-OPTW as described by Verbeeck et al. [90] and consists of 3 phases which are executed on a set of solutions during a predetermined amount of iterations $N_c$.

Firstly, ants, which represent solutions, are created in a construction phase. The second phase consists of an improvement phase where the ants are improved using three local search moves. Thirdly, in the global pheromone update phase the pheromone values belonging to the arcs of the best ant (of the iteration) are increased. An overview of the algorithm is provided in Algorithm 3 and the different components are described below. The best solution found in an iteration is defined as $\omega_{ib}$, $\omega_{gb}$ represents the best solution found during the entire optimization procedure and $F(\omega_x)$ refers to the objective function (expected reward) of solution $\omega_x$.

The ACS is executed until 10,000 trial solutions have been created. The exact value of $N_c$ is therefore adjusted to the value of `max_ants`. For example when 20 ants are used, 500 iterations were allowed in order to generate 10,000 trial solutions.

**Algorithm 3** Ant colony system for TD-OPSWTW - input parameters: $\alpha, \beta, \rho, \texttt{max\_ants}, \tau_{init}, N_{ni}^{max}$

$\omega_{ib} = 0$, $\omega_{gb} = 0$, $N_{ni} = 0$, $\texttt{iteration} = 0$
**while** $\texttt{iteration} < N_c$ **do**
  Initialize parameters: $\tau$, $\eta$ ($\tau_{init}$)
  **for** $i = 1$ to $\texttt{max\_ants}$ **do**
    Construct solution & local pheromone update ($\tau$, $\eta$, $\alpha$, $\beta$, $\rho$)
    Swap
    Calculate $\texttt{max\_start}$
    Insert ($\texttt{max\_start}$)
    Replace ($\texttt{max\_start}$)
  **end for**
  $\omega_{ib} = \arg\max(F(\omega_1), F(\omega_2), ..., F(\omega_{\texttt{max\_ants}}))$
  **if** $F(\omega_{ib}) > F(\omega_{gb})$ **then**
    $\omega_{gb} = \omega_{ib}$
    $N_{ni} = 0$
  **else**
    $N_{ni} = N_{ni} + 1$
  **end if**
  Global pheromone update ($\tau$, $\omega_{ib}$, $N_{ni}$, $N_{ni}^{max}$)
  $\texttt{iteration} = \texttt{iteration} + 1$
**end while**
Monte Carlo simulation ($\omega_{gb}$)
RETURN $\omega_{gb}$

#### *4.3.1. Construction phase*

The construction phase, displayed in Algorithm 4, starts with an empty solution. During each iteration, a random vertex from the candidate list is selected using a roulette wheel selection method. During the execution of this method, the probability of selection is based on a memory structure ($\tau$) and a heuristic ratio ($\eta$). The candidate list consists of all unvisited vertices that can be inserted into the solution and increase the expected reward of the solution when added. This increase in expected reward of the solution consists of the increase in expected reward, due to adding the unvisited vertex to the solution, minus the change in expected reward when travelling from this newly added vertex to the end vertex. In short, this means that late arrivals are allowed during the construction phase as long as it is beneficial. Finally, this phase stops when the candidate list is empty except for the end vertex. The memory structure follows the same logic as described in [90]. The pheromone values of arcs used during the construction phase are decreased while the values of the arcs belonging to the iteration best solution are increased.

#### *4.3.2. Using local search moves in a stochastic context*

Three local search moves are used in the ACS solution method: swap, insert and replace. They are visually represented in Figure 6. Firstly, the swap local search move tries to exchange two solution member vertices in order to save travel time. We define $y$ and $z$ as the vertices that will be exchanged, $k$ and $l$ as the predecessor and successor of $y$ and $m$ and $n$ as the predecessor and successor of $z$. After verification that $k$ and $l$ are neighbours of $z$ and that $m$ and $n$ are neighbours of $y$, the vertices $y$ and $z$ are temporarily exchanged. Otherwise, this swap pair $(y, z)$ is discarded before further evaluation. Afterwards, the new arrival time for the vertices between $k$ and $n$ is calculated while accounting for time window restrictions. Next, the decrease in travel time (calculated from $k$ to $n$) of the possible swap is calculated using the estimation algorithm. Only if this decrease in mean and variance is positive, the swap is executed. However, it is unsure that this exact decrease in travel time can be gained also for the whole solution due to waiting times and the time-dependent stochastic nature of succeeding travel times after vertex $n$. Nevertheless, since the mean and variance of the departure time distribution at $n$ are reduced, we are sure that departing from vertex $n$ guarantees that our new arrival time at the end depot will be equal to or smaller than the previous one. Therefore, this approach is chosen to avoid a complete evaluation of the solution. The swap local search move is run in a first-improving manner and stopped when no more feasible improvements can be found.

**Algorithm 4** SACS construction phase- input parameter: $\alpha, \beta$

$i = 0$
$\omega[i] = 1$
$\mu_{D[i]} = 0$
$\sigma_{D[i]} = 0$
**while** $\omega[i] \neq v$ **do**
  $CL = \emptyset$
  $a = \omega[i]$
  **for** all available neighbours $b$ of $a$ except $v$ **do**
    $X$=estimation algorithm$(a, b, D[i])$
    $expreward = \Phi_X(c_b) * r_b - (1 - \Phi_X(c_b)) * e_b$
    $V$=estimation algorithm$(b, v, X)$
    $endexpreward = expreward - (1 - \Phi_V(c_v)) * e_v$
    **if** $endexpreward > 0$ **then**
      add vertex $b$ to $CL$
      store $X$ and $expreward$
    **end if**
  **end for**
  **if** CL $\neq \emptyset$ **then**
    **for** all vertices $r \in CL$ **do**
      calculate $l_r$ $(\alpha, \beta)$
    **end for**
    roulette wheel selection method
    add selected vertex to $\omega$ and update the solution
  **else**
    $\omega[i + 1] = v$
  **end if**
  $i = i + 1$
**end while**
local pheromone update

Secondly, the insert and replace local search moves make use of a local evaluation metric called *max_start*. This metric is created by calculating for every vertex in the current solution, the latest possible start time of the service at the vertex, before the objective function of the solution decreases, using the 99th percentile of the travel time distribution. Other percentiles (50th, 77.1th, 90th) were tested but the results did not deviate much. This can be explained by the relatively high penalty for arriving late at the end vertex. This process is shown in Algorithm 5. Note that the latest possible start time is searched given the current total length of the route. This means that if the solution already contains a late arrival at the end depot, this increase in travel time compared to $t^{max}$ is taken into account as feasible time during the calculation of *max_start* (lines 7-9).

**Algorithm 5** Calculation of *max_start* - input solution $\omega$

1: $sz = |\omega|$
2: latestarrival=$\Phi^{-1}_{D[sz]}(0.99)$
3: **if** latestarrival $\leq t^{max}$ **then**
4: $\quad max\_start[sz] = t^{max}$
5: $\quad t_a = t_0 + t^{max}$
6: **else**
7: $\quad max\_start[sz]$ =latestarrival
8: $\quad t_a$ =latestarrival
9: **end if**
10: $i = sz - 1$
11: **while** $i > 0$ **do**
12: $\quad z = \omega[i+1]$
13: $\quad y = \omega[i]$
14: $\quad k$ = find time slot $(t_a)$
15: $\quad t_d = t_a - \Phi^{-1}_{T_{y,z,k}}(0.99)$
16: $\quad$ **if** $t_d > c_y + s_y$ **then**
17: $\quad\quad t_d = c_y + s_y$
18: $\quad$ **end if**
19: $\quad max\_start[i] = t_d$
20: $\quad t_a = t_d - s_y$
21: $\quad i = i - 1$
22: **end while**

Concerning the insert procedure, a vertex $y$ can only be inserted between $x$ and $z$ into the solution when the 99th percentile of the new arrival time distribution at $z$ (calculated using the estimation algorithm) is smaller than the *max_start* of vertex $z$. When vertex $y$ is actually included in the solution, the algorithm updates the arrival times from $x$ to $y$, as well as the arrival times at the vertices succeeding vertex $y$. This is necessary, because the insertion of vertex y has most likely caused a change in waiting and/or travel time for the arcs of the solution succeeding vertex $y$. Secondly, for the vertices preceding vertex $z$, a recalculation of the *max_start* local evaluation metric is required. The insert local search move is executed in a best-improving manner and stopped when no more feasible improvements can be found.

A similar logic is applied for the replace move which tries to replace an included vertex by a non-included vertex with a higher expected reward. The replace local search move is run in a first-improving manner and stopped when no more feasible improvements can be found.

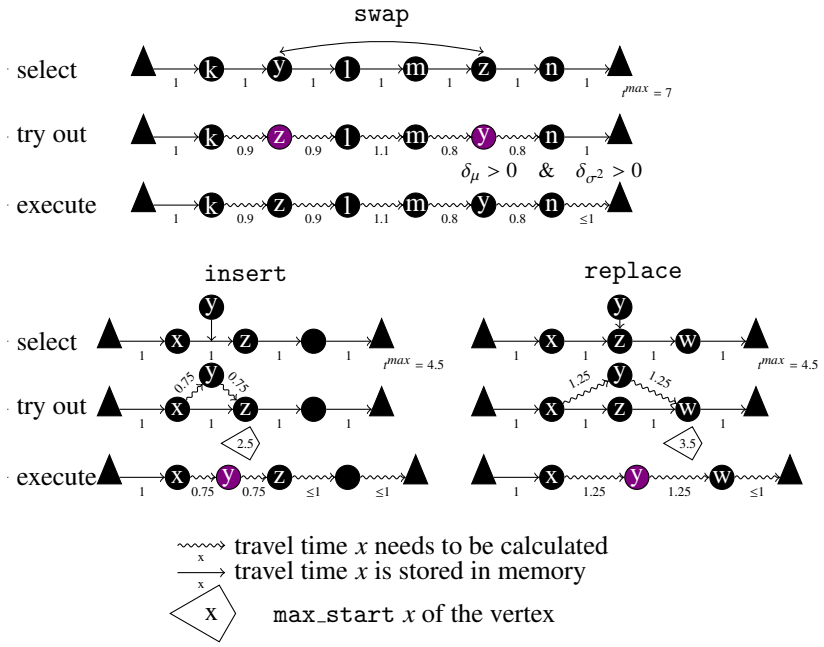



Figure 6: Overview of local search moves

## 5. Instance generation

This section discusses in detail, the generation of realistic test instances for the TD-OPSWTW based on the available data. These test instances are based on the ones proposed for the TD-OPTW in [90]. In turn, these instances were based on the real road network of Belgium, the Netherlands and Luxembourg. From this road network containing 84,720 vertices ($V$) and 116,683 arcs ($A$), 36 time-dependent stochastic test instances are extracted with a number of vertices ranging from 20 to 100. The difference resides in the fact that the travel times are now stochastic and a penalty is defined for every vertex. This penalty, also for the end vertex, is set independent of the magnitude of the lateness. This implies that in theory visiting additional vertices and collecting the reward before going to the end vertex might counter the collected penalty at the end vertex. In order to avoid this behaviour, the closing time of the regular vertices is set smaller than or equal to $t^{max}$. The penalty for regular vertices is equal to 20% of their respective reward. The penalty for arriving late at the end vertex is set equal to the highest reward over all regular vertices of the instance. The instances are also available at http://www.mech.kuleuven.be/en/cib/op/. The accompanied historical travel time data, consisting of travel time observations for every time period of 15 minutes for a representative Tuesday, is used to fit a set of normal travel time distributions for every arc during 96 time slots ($\kappa = 96$). Since for every arc and time slot combination only one travel time estimate is available, the standard deviation for a particular combination of an arc and time slot is set equal to the variability over all time slots of the arc. The average coefficient of variation ($cv = \frac{\sigma}{\mu}$) is therefore equal to 0.5%, which is rather low. Other coefficients of variation and penalty ratios ($pr = \frac{r_i}{e_i}$) will be used when the proposed algorithm is tested (Section 6.2).

The time-independent and time-dependent stochastic travel times between each pair of vertices representing serviceable customers to visit $V_c \in V$, are constructed. The time-independent travel time ($tt_{i,j}$) is calculated using Dijkstra's shortest path algorithm and assuming that arcs can be traversed at their maximum allowed speed. These estimates are used to solve the problem instances as deterministic time-independent orienteering problems as will be described in Section 6.2. Furthermore, these estimates are also used to construct the neighbourhood structure as described in Section 4.

The stochastic time-dependent travel time distributions ($T$) are calculated using an adapted version of Dijkstra's algorithm inspired by the research of Demeyer et al. [17]. The Dijkstra algorithm [18] is a label-setting algorithm with labels representing the lowest cost between the start vertex and the destination vertex. During initialization, the label of the start vertex is set to zero and all other labels are set to infinity. The set $P$ contains all (non-permanent) vertices

whose labels have been updated by the algorithm. In every iteration, the vertex with the lowest label is removed from the set $P$ and made permanent. Then, all labels of the neighbouring vertices are updated. More specifically, if the sum of the label of the investigated (permanent) vertex and of the arc cost is smaller than the previous label of the neighbouring vertex, then the label is changed to this sum. This process is repeated until the destination vertex has been made permanent. The time-dependent stochastic Dijkstra's algorithm requires that labels are no longer single values, but probabilistic distributions of travel times. Two operations need to be defined: comparing labels and updating labels. Deciding which of two labels is the best is no unambiguous process. Usually a fixed percentile such as the 90th percentile is used to compare the labels. In the work of Demeyer et al. [17], 5 equally spread percentiles (10th, 30th, 50th, 70th, 90th) are used during the update of the labels. However, in this research the labels are compared on the 99th percentile since we want to obtain reliable routes. Furthermore, it is assumed that the travel times on the arcs are independent (no correlation between the arcs) and normally distributed $(\mu, \sigma^2)$, which enables us to easily calculate the arrival distribution as the sum of the departure time distribution and the travel time distribution:

$$\mu_{arrival} = \mu_{departure} + \mu_{travel}$$
$$\sigma^2_{arrival} = \sigma^2_{departure} + \sigma^2_{travel}$$

Subsequently, the arrival time distribution becomes the departure time distribution when subsequent arcs are added to a route. So unlike the research of Demeyer et al. [17], no discrete approximation of the distribution is used.

We validated our stochastic implementation using a Monte Carlo simulation on the final path returned by the Dijkstra algorithm. Our Monte Carlo simulation was also used to verify the performance of the stochastic Dijkstra procedure of Demeyer et al. [17]. We could conclude that their procedure significantly underestimates the variance of the travel time on the proposed path. A possible explanation provided by Bickel et al. [8] is that the distribution is approximated using 5 percentiles that do not match the moments of the normal distribution.

The resulting time-dependent travel time distributions for each arc between two vertices are stored. As an indication of the performance of this procedure, for the set of 20, 50 and 100 vertices respectively 2, 6 and 13 minutes of calculation time is required.

## 6. Results

The SACS was programmed in C++ and all tests in this section were performed on a computer with an i5 2.6 GHz processor and 8 GB of memory. For the calculation of the cumulative and inverse normal distribution function, the Boost C++ 1.55.0 library was used.

### *6.1. Validation of the estimation algorithm*

The estimation algorithm is compared with the Monte Carlo procedure, both described in Section 4.1, based on the final solution sequence $(\omega_{gb})$ outputted by the SACS. Recall from Section 4.1 that we approximated the real distribution for the three different cases using a normal distribution with adapted mean and variance. Therefore, the same solution sequence can have a different objective function when evaluated by the estimation algorithm or the Monte Carlo procedure.

The average difference after five runs over all instances concerning the arrival time distribution at the end vertex is smaller than 0.01% for the mean arrival time at the end vertex and 0.67% for the variance. This results in an objective function deviation of only 0.01%. As a comparison, during the initial research phase, the SACS was also executed using an inferior estimation algorithm. During this algorithm, each normal arrival/departure time distribution was discretised using four-point Gaussian quadratures of a normal distribution. The benefits and downsides of this approximation are discussed by Bickel et al. [8]. The corresponding action to each of the three cases discussed in Section 4.1 is then applied separately to the four different percentiles of the arrival/departure distribution. When comparing this method to the results of the Monte Carlo procedure, this way of working leads to an average deviation of 0.04% for the mean arrival time, 7.9% for the variance and 0.15% for the objective function. The simplest estimation algorithm consists of taking the convolution of the departure time distribution and the travel time distribution (which corresponds with the mean of the departure time distribution) while ignoring the three different cases. This leads to an average deviation of 0.02% for the mean, 8.19% for the variance and 0.32% for the objective function.

Table 2: Overview on the performance of SACS and ACS

| | SACS | | | ACS ($\pi$=77.1th) | | |
|---|---|---|---|---|---|---|
| $\|V_c\|$ | % imp | cpu (s) | sd | % imp | cpu (s) | sd |
| 20 | 21.9 | 1.2 | 0.23 | 21.0 | 0.1 | 1.95 |
| 50 | 22.7 | 3.8 | 0.20 | 21.5 | 0.4 | 0.25 |
| 100 | 26.8 | 5.9 | 0.67 | 26.0 | 0.7 | 0.55 |
| Overall | 23.8 | 3.7 | 0.37 | 22.8 | 0.4 | 0.92 |

### 6.2. *Evaluation of the SACS algorithm*

To the best of our knowledge, this is the first time the TD-OPSWTW is proposed. Therefore, the SACS algorithm cannot be compared to other solution methods. However, to evaluate its performance, all TD-OPSWTW instances were also solved as deterministic and time-independent orienteering problems with time windows (OPTW) according to the mixed integer problem formulation presented in [90] using a commercial solver. Afterwards, this optimal solution is evaluated in a stochastic context using the Monte Carlo simulation. Then, a comparison can be made between this stochastically evaluated OPTW solution and a genuine stochastic solution proposed by the SACS. More specifically, the percentage difference between the expected reward of SACS solution method and the expected reward of the evaluated optimal OPTW solution is calculated as follows :

$$\%\, imp = \frac{SACS\ expreward - OPTW\ expreward}{OPTW\ expreward} * 100 \tag{13}$$

Other performance metrics are the standard deviation of the improvement and the required CPU time. These performance metrics of the SACS algorithm are displayed in Table 2.

Secondly, the SACS is compared with the ACS for the TD-OPTW using a predefined travel time percentile on every arc. The resulting deterministic TD-OPTW solutions are subsequently validated using a Monte Carlo simulation. This way of working differs from the two stage approach used in [24, 25] for the fact that the route is never aborted and penalties are imposed for late arrivals. Table 3 provides an overview of the effect of executing the algorithm with different percentiles of the travel time distribution on every arc. Constructing routes with the 1st percentile corresponds to searching for an optimistic solution, while using the 99th percentile corresponds to finding a limited-risk or pessimistic solution. In the last column the average improvement is displayed when the best solution is taken over all solutions corresponding to the 5 different travel time percentiles. Based on this table, using the 77.1th percentile of the travel time distribution results in the best performance for this set of instances. In general, the optimal percentile ($\pi$) is instance specific and depends on the penalty ratio and the coefficient of variation of the network.

The SACS slightly outperforms the regular ACS on quality (1%) but needs significantly more computation time than the deterministic version of the ACS. However, this computation time is relative as in reality the ACS needs to be executed for every percentile of the travel time distribution in order to determine which percentile returns the best result. The solutions proposed by the SACS are on average 23.8% (0.37 standard deviation) better than executing the optimal solution sequence calculated for a stochastic environment. This is expected as the average coefficient of variation of the instances is rather low (0.5%). It turns out that not taking into account the time-dependent and stochastic nature of the travel time has a significant impact on the quality of the proposed solutions in a realistic problem context. Finally, it can be concluded that accounting for time-dependent and stochastic travel times during the optimization process augments the performance of the proposed solutions but at the same time also requires higher computational efforts.

If we enhance the Monte Carlo simulation by adding a recourse action that prohibits a departure to a vertex which has a 100% probability of arriving late, the SACS was still able to improve the evaluated optimal OPTW solutions by 12.2%. This recourse action is based on the fact that in real-life a driver is not going to depart to a customer if the sum of the departure time and the deterministic time-independent travel time to this customer is larger than the closing time at this customer.

Table 3: The avg % imp for the ACS for different percentiles of the travel time distribution

| | Percentile of travel time distribution ($\pi$) | | | | | |
|---|---|---|---|---|---|---|
| $\|V_c\|$ | 1st | 22.9th | 50th | 77.1th | 99th | best |
| 20 | 7.9 | 13.2 | 19.0 | 21.0 | 21.0 | 22.1 |
| 50 | 16.4 | 18.4 | 21.9 | 21.4 | 21.5 | 22.6 |
| 100 | 16.5 | 21.9 | 26.8 | 26.0 | 25.7 | 27.3 |
| Overall | 13.6 | 17.8 | 22.6 | 22.8 | 22.7 | 24.0 |

### *6.3. Impact of stochastic travel times*

To measure the impact of not taking into account stochastic travel times during the optimisation process, the SACS approach is compared against the solution to the certainty equivalence problem using the ACS. When solving the certainty equivalence problem, the mean of the stochastic travel time is taken as travel time estimate.

The detailed results are displayed in Table 4 and a summary is provided in Table 5. The first four columns of Table 4 list respectively the name, amount of vertices, the maximum allowed travel time and time window restrictiveness for each problem instance. The labels S, M, L display the span (closing time minus opening time) of the time windows (small, medium, large). The succeeding column displays the expected reward (expreward) of the evaluated optimal ACS solution. In the 6th column, the expected reward of the TD-OPSWTW solution created by the SACS is given. Finally, the last two columns display the percentage improvement (imp) of the SACS versus the evaluated TD-OPTW solution and also the diversity (div) between the two solution sequences. The diversity between two solutions A and B is calculated as the sum of the number of vertices in A not present in B and the number of vertices in B not present in A, divided by the total amount of vertices present in A and B (start and end vertex not included).

The SACS on average can improve the expected reward with 1.3% in comparison with the ACS. A time-dependent stochastic orienteering problem has on average a solution that differs 28.3% from the time-independent and deterministic one. Based on Table 5, the SACS is more likely to improve instances with short time windows and a shorter allowed maximum travel time as more late arrivals can be avoided. Instances with more vertices and a higher maximum allowed travel time are more difficult to improve by the SACS and therefore negative improvements are sometimes obtained. As the problem becomes less constrained, the advantage of the deterministic approach is rewarded because of its simplicity.

Table 4: Impact of stochasticity per problem instance

| name | $\|V_c\|$ | $t^{max}$ | TW S,M,L | ACS($\pi$=50th) expreward | SACS expreward | imp % | div % |
|---|---|---|---|---|---|---|---|
| 20.1.1 | 20 | 8 | S | 124.7 | 157.0 | 25.9 | 33.3 |
| 20.1.2 | 20 | 8 | M | 173.0 | 173.0 | 0.0 | 7.7 |
| 20.1.3 | 20 | 8 | L | 173.0 | 181.6 | 4.9 | 28.6 |
| 20.2.1 | 20 | 10 | S | 188.0 | 188.0 | 0.0 | 33.3 |
| 20.2.2 | 20 | 10 | M | 190.7 | 195.0 | 2.3 | 50.0 |
| 20.2.3 | 20 | 10 | L | 194.5 | 195.0 | 0.3 | 14.3 |
| 20.3.1 | 20 | 12 | S | 275.9 | 272.0 | -1.4 | 9.1 |
| 20.3.2 | 20 | 12 | M | 239.7 | 239.6 | 0.0 | 15.8 |
| 20.3.3 | 20 | 12 | L | 259.0 | 259.0 | 0.0 | 23.8 |
| 20.4.1 | 20 | 14 | S | 255.8 | 274.0 | 7.1 | 21.7 |
| 20.4.2 | 20 | 14 | M | 275.0 | 275.0 | 0.0 | 4.4 |
| 20.4.3 | 20 | 14 | L | 268.0 | 268.0 | 0.0 | 21.7 |
| 50.1.1 | 50 | 8 | S | 287.8 | 288.0 | 0.1 | 14.3 |
| 50.1.2 | 50 | 8 | M | 274.0 | 274.0 | 0.0 | 15.8 |
| 50.1.3 | 50 | 8 | L | 280.5 | 289.0 | 3.0 | 20.0 |
| 50.2.1 | 50 | 10 | S | 298.0 | 298.0 | 0.0 | 4.8 |
| 50.2.2 | 50 | 10 | M | 310.0 | 310.0 | 0.0 | 50.0 |
| 50.2.3 | 50 | 10 | L | 326.9 | 327.5 | 0.2 | 12.0 |
| 50.3.1 | 50 | 12 | S | 324.4 | 337.0 | 3.9 | 65.2 |
| 50.3.2 | 50 | 12 | M | 410.3 | 412.1 | 0.4 | 12.0 |
| 50.3.3 | 50 | 12 | L | 366.0 | 366.0 | 0.0 | 46.2 |
| 50.4.1 | 50 | 14 | S | 461.9 | 470.0 | 1.7 | 24.1 |
| 50.4.2 | 50 | 14 | M | 439.0 | 434.9 | -0.9 | 17.2 |
| 50.4.3 | 50 | 14 | L | 446.0 | 450.0 | 0.9 | 24.1 |
| 100.1.1 | 100 | 8 | S | 272.0 | 266.0 | -2.2 | 50.0 |
| 100.1.2 | 100 | 8 | M | 278.0 | 278.0 | 0.0 | 26.3 |
| 100.1.3 | 100 | 8 | L | 343.0 | 343.0 | 0.0 | 11.1 |
| 100.2.1 | 100 | 10 | S | 351.0 | 351.0 | 0.0 | 23.1 |
| 100.2.2 | 100 | 10 | M | 350.3 | 361.0 | 3.0 | 68.0 |
| 100.2.3 | 100 | 10 | L | 370.0 | 370.0 | 0.0 | 38.5 |
| 100.3.1 | 100 | 12 | S | 435.9 | 437.0 | 0.3 | 15.2 |
| 100.3.2 | 100 | 12 | M | 450.0 | 444.0 | -1.3 | 61.3 |
| 100.3.3 | 100 | 12 | L | 470.0 | 466.0 | -0.9 | 14.3 |
| 100.4.1 | 100 | 14 | S | 478.0 | 479.0 | 0.2 | 76.5 |
| 100.4.2 | 100 | 14 | M | 484.7 | 489.0 | 0.9 | 24.3 |
| 100.4.3 | 100 | 14 | L | 538.0 | 525.0 | -2.4 | 42.1 |
| | | | avg | | | 1.3 | 28.3 |

These results are road network and instance specific but nonetheless they reflect that including stochastic information in the model can improve the objective value.

Recall that the coefficient of variation ($cv = \frac{\sigma}{\mu}$) for the original set of instances was rather low (0.5%) and the penalty ratio ($pr = \frac{r}{e}$) was equal to 20% for the regular vertices while the penalty of the end depot was set equal to the highest score of the instance. Therefore, the focus of the algorithm lies in generating routes with a high probability to arrive on time.

In order to test a more diverse set of stochastic situations, the same test setup was performed with higher coefficients of variation and also different penalty ratios for all vertices including the end depot. To achieve this, for every

Table 5: Impact of stochasticity versus problem characteristics

| $\|V_c\|$ | %imp | %div | $t^{max}$ | %imp | %div | TW | %imp | %div |
|---|---|---|---|---|---|---|---|---|
| 20 | 3.3 | 22.0 | 8 | 3.2 | 26.8 | S | 3.0 | 30.9 |
| 50 | 0.8 | 25.5 | 10 | 0.6 | 23.7 | M | 0.4 | 29.4 |
| 100 | -0.2 | 37.5 | 12 | 0.1 | 22.5 | L | 0.5 | 24.7 |
| | | | 14 | 0.8 | 28.5 | | | |

arc and timeslot combination the standard deviation of the travel time ($\sigma$) was set equal to $cv * \mu$. The percentage average improvement of the ACS solution method against the evaluated optimal OPTW solution over all instances is displayed in Table 6. Firstly, there can be seen that the average improvement is higher than in Table 4 for these higher levels of the coefficient of variation and the penalty ratio. The lowest result obtained in Table 6 equals 12.5% while the overall average improvement for the original instances displayed in Table 4 is 1.3%. Therefore, the logical conclusion that can be made is that it is worthwhile to take stochasticity into account when the variability of the travel times is high. Secondly, given a certain coefficient of variation a higher penalty ratio always leads to a bigger increase in overall average improvement. Thirdly, given a certain penalty ratio the overall average improvement decreases when the coefficient of variation increases. This is due to an overall decrease in performance of the proposed solution method and the insert local search move in particular. This is a consequence of the `max_start` local evaluation metric that uses the 99% travel time percentile. Therefore, a vertex can only be inserted when it is almost certain that the insertion will not lead to an additional lateness for subsequent vertices. This leads to very few candidates for feasible insertion when the variation is extremely high. For example for a $cv$ equal to 30%, the results can be improved by using a lower percentile of the travel time (e.g. 77.1%) during the calculation of the `max_start` local evaluation metric.

The average improvement against the evaluated optimal OPTW solution over all instances if we would apply the ACO metaheuristic with two different percentiles is shown in Tables 7 and 8. When comparing the results from these Tables with Table 6, it can be concluded that the ACS clearly outperforms the ACO solution method for instances with a higher penalty ratio and coefficient of variation.

Table 6: Impact on the average improvement of ACS (%) for different coefficients of variation (%) and penalty ratios ( %)

| $pr$ \ $cv$ | 10 | 20 | 30 |
|---|---|---|---|
| 25 | 16.7 | 14.2 | 12.5 |
| 50 | 22.0 | 19.1 | 17.9 |
| 100 | 42.7 | 38.2 | 36.5 |

Table 7: Impact on the average improvement (%) of ACO ($\pi$=77.1th) (%) for different coefficients of variation and penalty ratios ( %)

| $pr$ \ $cv$ | 10 | 20 | 30 |
|---|---|---|---|
| 25 | 6.81 | 1.95 | -1.62 |
| 50 | 15.75 | 10.80 | 6.31 |
| 100 | 41.65 | 34.89 | 29.57 |

Table 8: Impact on the average improvement (%) of ACO ($\pi$=99th) for different coefficients of variation (%) and penalty ratios ( %)

| $pr$ \ $cv$ | 10 | 20 | 30 |
|---|---|---|---|
| 25 | -2.76 | -13.95 | -21.82 |
| 50 | 5.59 | -6.72 | -15.15 |
| 100 | 29.43 | 13.82 | 3.80 |

### 6.4. Practical insights

When it is 100% certain that a vertex is visited on time, its variance of the departure time is equal to the variance of its arrival time. The probability of arriving early at a vertex decreases the variance of the departure time (Section 4.1.2). As mentioned in Section 4.1.3, the probability of a late arrival can lead to an increase or decrease of the variance of the departure time ($\sigma_D^2$) depending on the ratio of the variance of the arrival time ($\sigma_A^2$) and the service time ($s$). When $s$ is much higher than $\sigma_A^2$ (more than twice the size), $\sigma_D^2$ can become higher than $\sigma_A^2$. In reality the service time is also likely to be higher than $\sigma_A^2$ as the service time is usually several minutes while $\sigma_A^2$ is zero at the beginning of the solution sequence and increases towards the end of the solution sequence as variance propagates through the system. Therefore, in some cases, a possible late arrival at the end of the solution sequence decreases the variance of the route.

Contrary to the deterministic version, a swap operator can, apart from reducing the travel time, also increase the expected reward in a stochastic context. For example, if the swap operator is used on a solution with a late arrival at a vertex, the reduction in travel time can lead to the fact that the traveller no longer arrives late at this vertex. This means that the expected reward obtained at this vertex will increase. Therefore, an update of the expected reward starting from vertex $z$ is necessary after a successful swap move has been executed. Two additional (less restrictive) decision criteria have been tried for evaluating possible swap pairs apart from comparing both the difference between the mean and variance values of the departure time at $n$: only comparing the difference between the mean departure times at $n$ and comparing the difference between the 99th percentiles of the departure time at $n$. The problem with these last two criteria is that a reduction in mean departure time can go along with a significant increase in variance of the departure time which might cause a decrease in the objective function. This can happen because the probability of arriving late at subsequent vertices might increase as well. There exist multiple reasons for this phenomenon. Firstly, executing a swap move means that a set of arcs is replaced by another set of arcs in the solution and this new set of arcs might have a higher variance. Secondly, as already mentioned above, late arrivals at the end of the solution decrease the variance of the route. If after the execution of a swap move the late arrivals become on time arrivals, this in turn might lead to an increase in route variability which in most cases means a late arrival at the end vertex. Only a full evaluation of the solution for every swap pair can ensure that this adverse effect is not present. But since a full evaluation is too computationally expensive, we opted to compare the difference of a possible swap move on both the mean and variance.

Finally, comparing the 99th percentile of the updated arrival time distribution against the latest possible start (calculated using the 99th percentile of the travel time) whether to decide if a vertex can be inserted (replaced) means that there is also at least a 1% probability that you arrive late at the inserted (replaced) vertex and/or subsequent vertices. Furthermore, inserting a vertex also leads to an increase in variance that propagates through the system. However, the adverse effects of this assumption are marginal and the expected reward of inserting a vertex largely compensates for the very small expected penalty of a late arrival towards the end of the solution sequence. Nonetheless, the insert and replacement moves might cause small violations to the time budget but this effect is preferred as similar results cannot be achieved using the deterministic ant colony system and its local search moves.

## 7. Conclusion

This research proposes a solution method for the orienteering problem with time windows and time-dependent stochastic travel times (TD-OPSWTW). This specific problem formulation applies to congestion related issues in realistic routing problems that deal with uncertainty. In previous work [90], we discussed and solved the TD-OPTW without considering stochastic travel times. Although we started from the same solution framework (ACS), the solution approach is adapted and improved significantly in order to deal with stochastic travel times. Moreover, a number of original contributions and innovative insights in routing problems with stochastic travel times and time windows are presented in this work. The stochastic ant colony system (SACS) uses a swap local search move and an insert and replace local search move equipped with a local evaluation metric which speeds up the insertion/replacement process. The combination of time-dependency, time windows and stochastic travel time severely complicates the calculation of departure and arrival time distributions. Therefore, three prominent complications were defined in this paper and the way they were handled by the proposed estimation algorithm is explained and might also be useful for other vehicle routing problems with time windows and time-dependent stochastic travel times.

The proposed solution method is evaluated based on solution quality and computational performance on realistic problem instances based on the real road network of Belgium, the Netherlands and Luxembourg. Promising results were obtained in a reasonable amount of computation time. The results of the SACS are compared with optimal solutions for the regular OP with time windows, evaluated in a stochastic environment. The solutions proposed by the SACS are on average 23.8% better than executing the optimal solution sequence of the regular OPTW in a time-dependent stochastic environment. Secondly, the solutions generated by the ACS for TD-OPTW, using different percentiles of the travel time distribution as input, were also evaluated in a stochastic environment and compared to the solutions of the SACS. The SACS performs better than the ACS, independent of the used travel time percentile, as it takes into account the stochastic nature of this problem. However, the SACS also requires a greater computation time. An important practical insight is that a probability of a late arrival at a customer can lead to either an increase or decrease of the variance of the departure time. This effect depends on the ratio of the variance of the arrival time and the length of the service time. In general, at the beginning of the solution sequence, the service time is likely to be higher than the variance of the arrival time as the service time comprises usually several minutes while the variance of the arrival time is equal to zero at that point. However, the variance of the arrival time tends to increase towards the end of the solution sequence. In short, a probability of a late arrival in the beginning of the solution sequence increases the variance of the arrival time whereas a possible late arrival at the end of the solution sequence might decrease the variance of the arrival time. The combination of these results should motivate vehicle trip planners to consider the time-dependent and stochastic nature of the travel times in the future.

Possible extensions to this work should take into account that customers not served today must be served in the near future. Including this consideration could lead to an interesting extension and significantly impacts the set of customers selected on a given day. Determining a threshold value for the service time that influences the oscillation of the variance of the departure time forms a second research opportunity. Adapting the estimation algorithm in order that other types of travel time distributions (such as the gamma distribution) can be taken into account forms a third research opportunity.

## Acknowledgments

This research was funded by the Agency for Innovation by Science and Technology in Flanders (IWT).

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government – department EWI.

## References

[1] Abbaspour, A., & Samadzadegan, F. (2011). Time-dependent personal tour planning and scheduling in metropolises. *Expert Systems with Applications*, *38*, 12439–12452.

[2] Aghezzaf, E.-H., Zhong, Y., Raa, B., & Mateo, M. (2012). Analysis of the single-vehicle cyclic inventory routing problem. *International Journal of Systems Science*, *43*, 2040–2049.

[3] Ahn, B.-H., & Shin, J.-Y. (1991). Vehicle-routeing with time windows and time-varying congestion. *The Journal of the Operational Research Society*, *42*, 393–400.

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

[5] Balseiro, S., Loisea, I., & Ramonet, J. (2011). An ant colony algorithm hybridized with insertion heuristic for the time dependent vehicle routing problem with time windows. *Computers & Operations Research*, *38(6)*, 954–966.

[6] Barros, A., & Evers, L. (2012). The stochastic scouting problem. In *Proceedings of the Congreso Latino-Iberoamericano de Investigacion Operative - Simposio Brasilairo de Pesquisa Operacional CLAIO-SBPO*.

[7] Berry, D. S., & Belmont, D. M. (1951). Distribution of vehicle speeds and travel times. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (pp. 589–602). Berkeley, Calif.: University of California Press.

[8] Bickel, P., Lake, L. W., & Lehman, J. (2011). Discretization, simulation, and swanson's (inaccurate) mean. *SPE Economics & Management*, *3*, 128–140.

[9] Boyce, D. E., Nie, Y., Bar-Gera, H., Liu, Y., & Hu, Y. (2010). *Field Test of a Method for Finding Consistent Route Flows and Multiple-Class Link Flows in Road Traffic Assignments*. Technical Report Transportation Center, McCormick School of Engineering and Applied Science.

[10] Butt, S., & Cavalier, T. (1994). A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, *21*, 101–111.

[11] Campbell, A. M., Gendreau, M., & Thomas, B. W. (2011). The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, *186*, 61–81.

[12] Chao, I., Golden, B., & Wasil, E. (1996). Theory and methodology a fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, *88*, 475–489.
[13] Chen, H., Hsueh, C., & Chang, M. (2006). The real-time time-dependent vehicle routing problem. *Transportation Research Part E*, *42*, 383–408.
[14] Clark, S., & Watling, D. (2005). Modelling network travel time reliability under stochastic demand. *Transportation Research Part B: Methodological*, *39*, 119 – 140.
[15] Corthout, R., Himpe, W., Viti, F., Frederix, R., & Tampère, C. M. (2014). Improving the efficiency of repeated dynamic network loading through marginal simulation. *Transportation Research Part C: Emerging Technologies*, *41*, 90 – 109.
[16] Dabia, S., Ropke, S., van Woensel, T., & De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, *47(3)*, 380–396.
[17] Demeyer, S., Audenaert, P., Pickavet, M., & Demeester, P. (2014). Dynamic and stochastic routing for multimodal transportation systems. *IET Intelligent Transport Systems*, *8(2)*, 112–123.
[18] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*, 269–271.
[19] Donati, A., Montemanni, R., Casagrande, N., Rizzoli, A., & Gambardella, L. (2008). Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, *185*, 1174–1191.
[20] Ehmke, J. F., Campbell, A. M., & Urban, T. L. (2015). Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research*, *240*, 539 – 550.
[21] Ehmke, J. F., Steinert, A., & Mattfeld, D. C. (2012). Advanced routing for city logistics service providers based on time-dependent travel times. *Journal of Computational Science*, *3*, 193–205.
[22] Evers, L., Barros, A. I., Monsuur, H., & Wagelmans, A. (2014). Online stochastic UAV mission planning with time windows and time-sensitive targets. *European Journal of Operational Research*, *238*, 348 – 362.
[23] Evers, L., Dollevoet, T., Barros, A., & Monsuur, H. (2014). Robust UAV mission planning. *Annals of Operations Research*, *222*, 293–315.
[24] Evers, L., Glorie, K., van der Ster, S., Barros, A., & Monsuur, H. (2012). *The Orienteering Problem under Uncertainty Stochastic Programming and Robust Optimization compared*. Technical Report Econometric Institute Report EI 2012-21 Erasmus University, Econometric Institute.
[25] Evers, L., Glorie, K., van der Ster, S., Barros, A. I., & Monsuur, H. (2014). A two-stage approach to the orienteering problem with stochastic weights. *Computers & Operations Research*, *43*, 248 – 260.
[26] Fan, Y., Kalaba, R., & Moore, J. (2005). Arriving on time. *Journal of Optimization Theory and Applications*, *127*, 497–513.
[27] Figliozzi, M. A. (2012). The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review*, *48*, 616 – 636.
[28] Fleischmann, B., Gietz, M., & Gnutzmann, S. (2004). Time-varying travel times in vehicle routing. *Transportation Science*, *38*, 160–173.
[29] Fomin, F., & Lingas, A. (2002). Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, *83*, 57–62.
[30] Fu, L. (1999). Improving paratransit scheduling by accounting for dynamic and stochastic variations in travel time. *Transportation Research Record: Journal of the Transportation Research Board*, *1666*, 74–81.
[31] Fu, L. (2002). Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. *Transportation Research Part B: Methodological*, *36*, 485–506.
[32] Fu, L., & Rilett, L. (1998). Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Research Part B: Methodological*, *32*, 499–516.
[33] Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., & Linaza, M. (2013). Integrating Public Transportation in Personalised Electronic Tourist Guides. *Computers & Operations Research*, *40*, 758–774.
[34] Gavalas, D., Konstantopoulos, C., Mastakas, K., & Pantziou, G. (2014). A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, *20*, 291–328.
[35] Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., & Vathis, N. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research*, *62*, 36 – 50.
[36] Gendreau, M., Ghiani, G., & Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers & Operations Research*, *64*, 189 – 197.
[37] Golden, B., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, *34*, 307–318.
[38] Haghani, A., & Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research*, *32*, 2959–2986.
[39] Hashimoto, H., Yagiurab, M., & Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, *5*, 434–456.
[40] He, R. R., Liu, H. X., Kornhauser, A. L., & Ran, B. (2002). *Temporal and Spatial Variability of Travel Time.*. Technical Report Center for Traffic Simulation Studies. UC Irvine: Center for Traffic Simulation Studies.
[41] Ichoua, S., Gendreau, M., & Potvin, J. Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, *144*, 379–396.
[42] Ilhan, T., Iravani, S. M. R., & Daskin, M. S. (2008). The orienteering problem with stochastic profits. *IIE Transactions*, *40*, 406–421.
[43] Kantor, M., & Rosenwein, M. (1992). The orienteering problem with time windows. *Journal of Operational Research Society*, *43 (6)*, 629–635.
[44] Kaparias, I., Bell, M. G. H., & Belzner, H. (2008). A New Measure of Travel Time Reliability for In-Vehicle Navigation Systems. *Journal of Intelligent Transportation Systems*, *12*, 202–211.
[45] Kharoufeh, J. P., & Gautam, N. (2004). Deriving link travel-time distributions via stochastic speed processes. *Transportation Science*, *38*, 97–106.
[46] Kok, A. L., Hans, E. W., & Schutten, J. M. J. (2012). Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & Operations Research*, *39 (5)*, 910–918.
[47] Kok, A. L., Hans, E. W., Schutten, J. M. J., & Zijm, W. H. M. (2010). A dynamic programming heuristic for vehicle routing with time-

dependent travel times and required breaks. *Flexible Service and Manufacturing Journal*, *22*, 83–108.

[48] Kritzinger, S., Tricoire, F., Doerner, K., & Hartl, R. (2011). Variable neighborhood search for the time-dependent vehicle routing problem with soft time windows. In C. Coello (Ed.), *Learning and Intelligent Optimization* (pp. 61–75). Springer Berlin Heidelberg volume 6683 of *Lecture Notes in Computer Science*.

[49] Kuo, Y. (2010). Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering*, *59*, 157 – 165.

[50] Kwon, J., Coifman, B., & Bickel, P. (2000). Day-to-day travel-time trends and travel-time prediction from loop-detector data. *Transportation Research Record: Journal of the Transportation Research Board*, *1717*, 120–129.

[51] Lau, H. C., Yeoh, W., Varakantham, P., Nguyen, D. T., & Chen, H. (2012). Dynamic stochastic orienteering problems for risk-aware applications. *CoRR*, *abs/1210.4874*, 1–11.

[52] Lecluyse, C., Sörensen, K., & Peremans, H. (2013). A network-consistent time-dependent travel time layer for routing optimization problems. *European Journal of Operational Research*, *226*, 395–413.

[53] Lecluyse, C., Van Woensel, T., & Peremans, H. (2009). Vehicle routing with stochastic time-dependent travel times. *4OR: Quarterly journal of Operational Research*, *7*, 363–377.

[54] Li, J., Wu, Q., Li, X., & Zhu, D. (2010). Study on the time-dependent orienteering problem. In *International Conference on E-Product E-Service and E-Entertainment (ICEEE)* (pp. 1 – 4).

[55] Li, X., Tian, P., & Leung, S. (2010). Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *International Journal of Production economics*, *125*, 137–145.

[56] Liu, X., Chien, S., & Kim, K. (2012). Evaluation of floating car technologies for travel time estimation. *Journal of Modern Transportation*, *20*, 49–56.

[57] Maden, W., Eglese, R., & Black, D. (2010). Vehicle routing and scheduling with time-varying data: A case study. *Journal of the Operational Research Society*, *61*, 515–522.

[58] Malandraki, C., & Daskin, M. (1992). Time dependent vehicle routing problems: formulations, properties and heuristic algorithms. *Transportation Science*, *26*, 185–200.

[59] Mufalli, F., Batta, R., & Nagi, R. (2012). Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans. *Computers & Operations Research*, *39*, 2787 – 2799.

[60] Nguyen, P. K., Crainic, T. G., & Toulouse, M. (2013). A tabu search for Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows. *European Journal of Operational Research*, *231*, 43–56.

[61] Noland, R. B., & Polak, J. W. (2002). Travel time variability: A review of theoretical and empirical issues. *Transport Reviews*, *22*, 39–54.

[62] Osvald, A., & Stirn, L. Z. (2008). A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food. *Journal of Food Engineering*, *85*, 285 – 295.

[63] Papapanagiotou, V., Montemanni, R., & Gambardella, L. (2014). Objective function evaluation methods for the orienteering problem with stochastic travel and service times. *Journal of applied Operational research*, *6(1)*, 16–29.

[64] Papapanagiotou, V., Weyland, D., Montemanni, R., & Gambardella, L. (2013). A sampling-based approximation of the objective function of the orienteering problem with stochastic travel and service times. *Lecture Notes in Management Science*, *5*, 143–152.

[65] Potvin, J., Xu, Y., & Benyahia, I. (2006). Vehicle routing and scheduling with dynamic travel times. *Computers & Operations research*, *33*, 1129–1137.

[66] Rahmani, M., Jenelius, E., & Koutsopoulos, H. N. (2015). Non-parametric estimation of route travel time distributions from low-frequency floating car data. *Transportation Research Part C: Emerging Technologies*, *58*, 343–362.

[67] Royset, J. O., & Reber, D. N. (2009). Optimized routing of unmanned aerial systems for the interdiction of improvised explosive devices. *Military Operations Research*, *14*, 5–19.

[68] Russell, R. A., & Urban, T. L. (2008). Vehicle routing with soft time windows and erlang travel times. *The Journal of the Operational Research Society*, *59*, 1220–1228.

[69] Schilde, M., Doerner, K., & Hartl, R. (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, *238*, 18 – 30.

[70] Schilde, M., Doerner, K., Hartl, R., & Kiechle, G. (2009). Metaheuristics for the biobjective orienteering problem. *Swarm Intelligence*, *3*, 179–201.

[71] Setak, M., Habibi, M., Karimi, H., & Abedzadeh, M. (2015). A time-dependent vehicle routing problem in multigraph with fifo property. *Journal of Manufacturing Systems*, *35*, 37 – 45.

[72] Soler, D., Albiach, J., & Martinez, E. (2009). A way to optimally solve a time-dependent vehicle routing problem with time windows. *Operations Research Letters*, *37*, 37–42.

[73] Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., & Van Oudheusden, D. (2011). The planning of Cycle Trips in the Province of East Flanders. *OMEGA: International Journal of Management Science*, *39*, 209–213.

[74] Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., & Van Oudheusden, D. (2008). A personalised tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, *22*, 964–985.

[75] Sun, S., Duan, Z., & Yang, D. (2015). Urban Freight Management with Stochastic Time-Dependent Travel Times and Application to Large-Scale Transportation Networks. *Discrete Dynamics in Nature and Society*, (p. 10).

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

[77] Taş, D., Dellaert, N., van Woensel, T., & de Kok, T. (2014). The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C: Emerging Technologies*, *48*, 66 – 83.

[78] Taş, D., Gendreau, M., Dellaert, N., van Woensel, T., & de Kok, A. G. (2014). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operations Research*, *236*, 789–799.

[79] Tang, H., & Miller-Hooks, E. (2005). Algorithms for a stochastic selective travelling salesperson problem. *The Journal of the Operational Research Society*, *56*, 439–452.

[80] Taniguchi, E., Thompson, R., Yamada, T., & van Duin, J. (2001). *City Logistics: Network Modelling and Intelligent Transport Systems*. Pergamon.
[81] Teng, S. Y., Ong, H. L., & Huang, H. C. (2004). An integer l-shaped algorithm for time-constrained traveling salesman problem with stochastic travel and service times. *Asia-Pacific Journal of Operational Research*, *21*, 241–257.
[82] Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, *35 (9)*, 797–809.
[83] Van Woensel, T., Kerbache, L., Peremans, H., & Vandaele, N. (2008). Vehicle routing with dynamic travel times: A queueing approach. *European Journal of Operational Research*, *186*, 990–1007.
[84] Vansteenwegen, P., Souffriau, W., & Van Oudheusden, D. (2011). The orienteering problem: a survey. *European Journal of Operational Research*, *209*, 1–10.
[85] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2009). Iterated Local Search for the Team Orienteering Problem with Time Windows. *Computers & Operations Research*, *36*, 3281–3290.
[86] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2009). Metaheuristics for tourist trip planning. In K. Sörensen, M. Sevaux, W. Habenicht, & M. J. Geiger (Eds.), *Metaheuristics in the Service Industry* (pp. 15–31). Springer Berlin Heidelberg volume 624 of *Lecture Notes in Economics and Mathematical Systems*.
[87] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2011). The City Trip Planner: A Tourist Expert System. *Expert Systems with Applications*, *38*, 6540–6546.
[88] Vansteenwegen, P., & Van Oudheusden, D. (2007). The mobile tourist guide: An OR opportunity. *OR Insights*, *20 (3)*, 21–27.
[89] Verbeeck, C., Sörensen, K., Aghezzaf, E.-H., & Vansteenwegen, P. (2014). A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, *236 (2)*, 419–432.
[90] Verbeeck, C., Vansteenwegen, P., & Aghezzaf, E.-H. (2015). A fast solution method for the time-dependent orienteering problem with time windows. *Annals of Operations Research: under review*, (pp. 1–20).
[91] Wang, X., Golden, B. L., & Wasil, E. A. (2008). The vehicle routing problem: Latest advances and new challenges. chapter Using a Genetic Algorithm to Solve the Generalized Orienteering Problem. (pp. 263–274). Boston, MA: Springer US.
[92] Wen, L., & Eglese, R. (2015). Minimum cost VRP with time-dependent speed data and congestion charge. *Computers & Operations Research*, *56*, 41 – 50.
[93] Yildirimoglu, M., & Geroliminis, N. (2013). Experienced travel time prediction for congested freeways. *Transportation Research Part B: Methodological*, *53*, 45 – 63.
[94] Zhang, S., Ohlmann, J. W., & Thomas, B. W. (2014). A priori orienteering with time windows and stochastic wait times at customers. *European Journal of Operational Research*, *239*, 70 – 79.
[95] Zhang, T., Chaovalitwongse, W. A., & Zhang, Y. (2014). Integrated Ant Colony and Tabu Search approach for time dependent vehicle routing problems with simultaneous pickup and delivery. *Journal of Combinatorial Optimization*, *28*, 288–309.
[96] Zheng, F., & Van Zuylen, H. (2013). Urban link travel time estimation based on sparse probe vehicle data. *Transportation Research Part C: Emerging Technologies*, *31*, 145 – 157.

## Appendix A. Width of departure time distribution spanning multiple time slots

If $D \sim \mathcal{N}(\mu_0, \sigma_0^2)$ represents the departure time and $g > 0$ the start of a time slot that divides the departure time distribution into two parts. Furthermore, let $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ be the travel time corresponding to the first time slot and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ the travel time corresponding to the second time slot. Then, the random variable $A$ represents the arrival time and is equal to the following piecewise function:

$$A = \begin{cases} D + X_1 & \text{if } D \leq g \\ D + X_2 & \text{if } D > g. \end{cases} \tag{A.1}$$

To find the probability density function of A, the following help functions need to be calculated:

- $h_1(z)$ which is the probability density function of $(D + X_1)|(D \leq g)$
- $h_2(z)$ which is the probability density function of $(D + X_2)|(D > g)$

The probability density function of $(D+X_1)|(D \leq g)$ is the joint product of the individual probability density functions or:

$$\phi_{f_1} = \frac{\phi_D}{\mathcal{P}(D \leq g)} * \phi_{x_1} \tag{A.2}$$

Note that $\mathcal{P}$ stands for the probability. This function is inserted in Mathematica as follows:

```
f1 = (1/(Subscript[\[Sigma],0]*Sqrt[2*\[Pi]])*Exp[-(d - Subscript[\[Mu],0])^2/
(2 *Subscript[\[Sigma],0]^2)])/(1/2 (1 +Erf[(g - Subscript[\[Mu],0])/
(Subscript[\[Sigma],0]*Sqrt[2])]))*1/(Subscript[\[Sigma],1]*Sqrt[2*\[Pi]])*
```

```
Exp[-(Subscript[x,1] - Subscript[\[Mu],1])^2/(2 *Subscript[\[Sigma],1]^2)];
domain[f1] = {{d,-Infinity,Infinity},{Subscript[x,1],-Infinity,Infinity}}
&& {Element[Subscript[\[Mu],0],Reals],Element[Subscript[\[Mu],1],Reals],
Subscript[\[Sigma],0] > 0,Subscript[\[Sigma],1] > 0,g > 0};
```

Afterwards, we apply the following transformation on $f_1$ $(D, X_1) \rightarrow (Z = D + X_1, V = X_1)$. The joint probability density function of $(Z, V)$ is called $g_1(z, v)$. Note that the transformation equation $(Z = D + X_1, V = X_1)$ induces dependency between $Z$ and $V$. In particular, since $Z = V + D$ and $D < g$, it follows that $Z < V + g$. We execute this transformation in Mathematica using the *Transform* function and a *Boole* statement.

```
g1 = Transform[{z == d + Subscript[x,1],v == Subscript[x,1]}, f1]  Boole[z < v + g]
domain[g1] = {{z, -Infinity, Infinity},{v, -Infinity,Infinity}} &&
{Element[Subscript[\[Mu],0],Reals],Element[Subscript[\[Mu],1], Reals],
Subscript[\[Sigma],0] > 0,Subscript[\[Sigma],1] > 0, g > 0};
```

Finally, $h_1$ is the marginal probability density function of $Z = D+X_1$. This last procedure is performed in Mathematica using the *Marginal* function:

```
h1 = Marginal[z, g1]
```

Similarly, $h_2$ is calculated as follows: The probability density function of $(D + X_2)|(D > g)$ is the joint product of the individual probability density function or:

$$\phi_{f2} = \frac{\phi_D}{\mathcal{P}(D > g)} * \phi_{x_2} \tag{A.3}$$

Afterwards, we apply the following transformation $(D, X_2) \rightarrow (Z = D + X_2, V = X_2)$, the joint probability density function of $(Z, V)$ is called $g_2(z, v)$. $h_2$ is the marginal probability density function of $Z = D + X_2$. These procedures are done in Mathematica as follows:

```
f2 = (1/(Subscript[\[Sigma],0]*Sqrt[2*\[Pi]])*Exp[-(d - Subscript[\[Mu],0])^2/
(2 *Subscript[\[Sigma],0]^2)])/(1/2(1 -Erf[(g - Subscript[\[Mu],0])/
(Subscript[\[Sigma],0]*Sqrt[2])]))*1/(Subscript[\[Sigma],2]*Sqrt[2*\[Pi]])
*Exp[-(Subscript[x,2] - Subscript[\[Mu],2])^2/(2*Subscript[\[Sigma],2]^2)];
domain[f2] = {{d, -Infinity, Infinity},{Subscript[x, 2], -Infinity,Infinity}} &&
{Element[Subscript[\[Mu],0],Reals],Element[Subscript[\[Mu],2], Reals],
Subscript[\[Sigma],0] > 0,Subscript[\[Sigma],2] > 0, g > 0};
g2 = Transform[{z == d + Subscript[x,2],v == Subscript[x,2]},f2]  Boole[z > v + g]
domain[g2] = {{z, -Infinity, Infinity},{v, -Infinity,Infinity}}
&&{Element[Subscript[\[Mu],0], Reals],Element[Subscript[\[Mu],2], Reals],
Subscript[\[Sigma],0]>0,Subscript[\[Sigma],2]>0,g > 0};
h2 = Marginal[z, g2]
```

The probability density function of A is now equal to:

$$A = \mathcal{P}(D \leq g) * h_1(z) + \mathcal{P}(D > g) * h_2(z) \tag{A.4}$$

This is calculated using the *Prob* function in Mathematica as follows:

```
Dep = 1/(Subscript[\[Sigma],0]*Sqrt[2*\[Pi]])*Exp[-(d - Subscript[\[Mu],0])^2/
(2*Subscript[\[Sigma],0]^2)];
domain[Dep] = {d, -Infinity,Infinity} && {Element[Subscript[\[Mu],0], Reals],
Subscript[\[Sigma],0] > 0};
prob = Prob[g, Dep]
Arr = h1*prob + (1 - prob)*h2 // Simplify
```

To calculate the mean and the variance let $f_3(d, x_1, x_2)$ denote the joint probability density function of $(D, T_1, T_2)$ which, by independence, is the product of the 3 individual normal probability density functions. Together with the following function $z$, Mathematica can calculate the mean and variance of $A$ using respectively the *Expect* and *Var* function.

```
f3 = (1/(Subscript[\[Sigma],0]*Sqrt[2*\[Pi]])*Exp[-(d - Subscript[\[Mu],0])^2/(
2 *Subscript[\[Sigma],0]^2)])*(1/(Subscript[\[Sigma],1]*Sqrt[2*\[Pi]])*
Exp[-(Subscript[x,1] - Subscript[\[Mu],1])^2/(2 *Subscript[\[Sigma], 1]^2)])*(1/(
Subscript[\[Sigma],2]*Sqrt[2*\[Pi]])*Exp[-(Subscript[x,2] - Subscript[\[Mu],2])^2/(
2 *Subscript[\[Sigma], 2]^2)]);
domain[f3] = {{d,-Infinity,Infinity},{Subscript[x,1],
-Infinity,Infinity},{Subscript[x,2],-Infinity, Infinity}} && {Element[Subscript[\[Mu],0],
Reals], Element[Subscript[\[Mu],1], Reals],Element[Subscript[\[Mu],2], Reals],
Subscript[\[Sigma],0] > 0, Subscript[\[Sigma],1] > 0,Subscript[\[Sigma],2] > 0, g > 0};
z = Piecewise[{{d + Subscript[x,1], d <= g},{d + Subscript[x,2], d > g}}];
meana= Expect[z,f3]
vara = Var[z, f3]
```

## Appendix B. Early arrival

Define normal arrival time function and help function z.

```
A = Exp[-(t - Subscript[\[Mu],a])^2/(2 Subscript[\[Sigma],a]^2)]/(Sqrt[2 Pi]
Subscript[\[Sigma],a]);
domain[A] = {t, -Infinity, Infinity} &&Subscript[\[Mu],a] > 0,
Subscript[\[Sigma],a] > 0, o > 0, s > 0};
z = If[t <= o, o + s, t + s];
```

Calculation of the mean using the *Expect* function

```
meand = Expect[z,A]
```

Calculation of the variance using the *Var* function

```
vard = Var[z,A]
```

Calculation of the continuous part of the probability density function using the *Prob* function

```
cdf = Prob[If[t < o, o + s, t + s] < d, A]
domain[cdf] = {d, -Infinity, Infinity}&& {Subscript[\[Mu],a] > 0,
Subscript[\[Sigma],a] > 0, o > 0, s > 0};
pdf = D[cdf, d]
domain[pdf] = {d, -Infinity, Infinity}&& {Subscript[\[Mu],a] > 0,
Subscript[\[Sigma],a] > 0, o > 0, s > 0};
```

## Appendix C. Late arrival

Define normal arrival time function and help function z.

```
A = Exp[-(t - Subscript[\[Mu],a])^2/(2 Subscript[\[Sigma],a]^2)]/(Sqrt[2 Pi]
Subscript[\[Sigma],a]);
domain[A] = {t, -Infinity, Infinity} &&{Subscript[\[Mu],a] > 0,
Subscript[\[Sigma],a] > 0, c > 0, s > 0};
z = If[t <= c, t + s, t];
```

Calculation of the mean using the *Expect* function

```
meand = Expect[z, A]
```

Calculation of the variance using the *Var* function

```
vard = Var[z, A]
```

Calculation of the probability density function using the *Prob* function

```
cdf = Prob[If[t < c, t + s, t] < d, A]
domain[cdf] = {d, -Infinity, Infinity}&& {Subscript[\[Mu],a] > 0,
Subscript[\[Sigma],d] > 0, c > 0, s > 0};
pdf = D[cdf, d] // Simplify
domain[pdf] = {d, -Infinity, Infinity}&& {Subscript[\[Mu],a] > 0,
Subscript[\[Sigma],a] > 0, c > 0, s > 0};
```