OPTIMAL DEPARTURE TIME ADVICE IN ROAD NETWORKS
WITH STOCHASTIC DISRUPTIONS

RENS KAMPHUIS, NIKKI LEVERING AND MICHEL MANDJES

Abstract

Due to recurrent (e.g. daily or weekly) patterns and non-recurrent disruptions (e.g. caused by incidents),
travel times in road networks are time-dependent and inherently random. This is challenging for travelers
planning a future trip, aiming to ensure on-time arrival at the destination, while also trying to limit the
total travel-time budget spent. The focus of this paper lies on determining their optimal departure time:
the latest time of departure for which a chosen on-time arrival probability can be guaranteed. To model
the uncertainties in the network, a Markovian background process is used, tracking events affecting the
driveable vehicle speeds on the links, thus enabling us to incorporate both recurrent and non-recurrent
effects. It allows the evaluation of the travel-time distribution, given the state of this process at departure,
on each single link. Then, a computationally efficient algorithm is devised that uses these individual link
travel-time distributions to obtain the optimal departure time for a given path or origin-destination pair.
Since the conditions in the road network, and thus the state of the background process, may change
between the time of request and the advised time of departure, we consider an online version of this
procedure as well, in which the traveler receives departure time updates while still at the origin. Finally,
numerical experiments are conducted to exemplify a selection of properties of the optimal departure time
and, moreover, quantify the performance of the presented algorithms in an existing road network — the
Dutch highway network.

KEYWORDS. Latest departure time ◦ Reliable path ◦ Stochastic velocities ◦ Markovian background
process.

AFFILIATIONS. RK, NL and MM are with the Korteweg-de Vries Institute for Mathematics, University
of Amsterdam, Amsterdam. MM is also with EURANDOM, Eindhoven, and with Amsterdam Business
School, University of Amsterdam, Amsterdam. This research project is partly funded by the NWO
Gravitation project NETWORKS, grant number 024.002.003. Date: September 1, 2022.
1. Introduction

Motivation

Travel times in road networks are inherently random. This could lead to late arrival at the destination, which, depending on the importance of the trip, may be highly undesirable. At the same time, travelers will generally also be reluctant to depart far too early, as this would leave them with less time available for other activities. Thus, there is a clear need for algorithms that generate the latest departure time such that a certain user-specified on-time arrival probability can be guaranteed.

The randomness of travel times is due to both recurrent patterns (e.g., peak hours) and non-recurrent events (e.g., traffic accidents, vehicle failures, unfavorable weather conditions). Whereas recurrent patterns are predictable from historic data, the time and location of future non-recurrent events are intrinsically uncertain. It is noted, though, that the current state of the road network is known; think of the locations of the incidents that are currently present, as well as their elapsed durations. As currently present incidents further away from the origin are more likely to be cleared once the corresponding road segment is reached by the traveler, this information can be exploited when determining the optimal departure time.

When devising a procedure to generate an optimal departure time for a given path or origin-destination (OD) pair, various considerations play a role. Ideally, one would like to determine a departure time that (i) takes the risk averseness of the traveler into account, (ii) incorporates the randomness of travel times, due to both recurrent and non-recurrent effects, and (iii) exploits knowledge of the locations of the currently present non-recurrent events. To be able to discuss our contributions, we proceed by an account of the existing literature (where we do not attempt to provide an exhaustive overview, but primarily focus on contributions relevant in the context of our work).

Literature

While there are different notions of optimality, optimal reliable paths can roughly be classified into two categories. Either a path is regarded as optimal if it has the highest on-time arrival probability, as in the seminal work by Frank (1969), or, a path is optimal if it has the lowest expected travel time while also guaranteeing a certain reliability threshold. An example of the latter is the $\alpha$-shortest path as introduced in Ji (2005), which is the path with the lowest expected travel time such that the probability of arriving on time is at least $\alpha$. A study to
identifying the a-priori $\alpha$-shortest path can be found in Nie and Wu (2009), whereas Fan et al. (2005) and Christman and Cassamano (2013) outline a dynamic routing procedure to maximize the on-time arrival probability.

In Loui (1983) it is recognized that drivers may have different preferences, and that the optimal route is the route that optimizes the personal utility function of the individual driver. Often, these utility functions take into account both the expected travel time and the reliability of the route; see e.g. Sen et al. (2001). The typical objective is to identify the route that minimizes the sum of the expected travel time and a term that quantifies the travel-time variance, with weights that reflect the driver’s risk averseness. Such a utility function is intended to strike a proper balance between minimizing the travel time and controlling its randomness (Tilahun and Levinson 2010). The previously mentioned $\alpha$-shortest path criterion succeeds in favoring routes with a low mean travel time, while also penalizing travel-time uncertainty. As this criterion uses the full travel-time distribution (rather than just the travel time variance), the uncertainty of a route can be captured in more detail than in the approach of Sen et al. (2001). Moreover, the degree of the uncertainty penalization can be adjusted by the parameter $\alpha$, which is chosen to reflect the specific driver’s risk averseness.

An important observation is that the literature on optimal reliable routing, including the above-mentioned works, typically assumes instant departure of the traveler. However, besides late arrival, also early arrival evidently incurs an (opportunity) cost, so that immediate departure is generally not optimal (Small 1982). Whereas maximizing the on-time arrival probability would require instant departure, the $\alpha$-shortest path criterion can still be used in case of non-instant departure. The idea is then to identify the latest departure time such that an on-time arrival probability of at least $\alpha$ can be guaranteed. This approach is followed in e.g. Chen et al. (2014), where besides an optimal route, also the optimal departure time is determined under a stochastic first-in-first-out assumption; see also Yang and Gao (2020), assuming log-normally distributed travel times.

As mentioned, travel times cannot be assumed constant; sources of these fluctuations are discussed in Mahmassani et al. (2014). To deal with the recurrent patterns, one could work with the framework of Chen et al. (2014) and Yang and Gao (2020), where time is divided into intervals and the parameters of the travel-time distributions have interval-specific values. These parameters have been inferred from historic data, and hence are considered known. However, travel times are also subject to non-recurrent effects, such as congestion due to incidents. These should clearly be taken into account, as the occurrence or clearance of incidents greatly affects
the travel-time distribution. Obviously, one cannot use techniques designed to deal with recurrent patterns, but one would like to exploit knowledge of the statistics of incident durations and inter-incident times, as well as knowledge of the current state of the road network (in particular the locations of incidents that are currently present).

**Contributions**

It is the main aim of this paper to contribute to the literature on determining optimal departure times, with a focus on including the effect of non-recurrent congestion, aiming at meeting the three requirements (i), (ii), and (iii) identified above. A more detailed account of the contributions of our work is as follows:

- To model the travel-time dynamics such that it can cover both recurrent and non-recurrent effects, we use the Markovian Velocity Model (MVM) as introduced in Levering et al. (2022b). The ‘velocity oriented’ and data-driven MVM, building upon the stochastic disruption models in Psaraftis and Tsitsiklis (1993), Kharoufeh and Gautam (2004), Kim et al. (2005a,b), Thomas and White III (2007), Güner et al. (2012), and Sever et al. (2013), uses a Markovian background process that tracks the events that affect the driveable vehicle speeds on road segments. A particularly attractive aspect of the MVM is that knowledge of the current state of the road network (e.g. locations of incidents) can be naturally incorporated.

- The velocities in the road network are fully determined by the current state of the Markovian background process. The dynamics of the background process determine the travel-time distribution for each link, which can then be used to obtain the travel-time distribution for any route in the network. Having access to the distribution, the probability of on-time arrival for any departure time can be computed. This allows us to set up a procedure that identifies, for any desired on-time arrival probability, the optimal departure time. We do this both for a specified path and for a specified origin-destination pair (i.e., also providing the path that yields the optimal departure time for this OD pair). The risk averseness is naturally taken into account, as the departure time depends on the selected value of the on-time arrival probability.

- Where Chen et al. (2014) and Yang and Gao (2020) exclusively rely on historic estimates to evaluate the travel-time distribution, our procedure also includes the impact of the current state of the road network. Rather than choosing a specific travel-time distribution, such as the log-normal one featuring in Yang and Gao (2020), the MVM is
highly flexible, as extensively argued in Levering et al. (2022a), in particular in terms of modeling the distribution of the incident durations and the inter-incident times. Another advantage of our approach is that the road network by construction satisfies the desirable FIFO property, and we do not have to rely on a stochastic-FIFO assumption as in Chen et al. (2014).

- We recognize that the conditions in the road network, and thus the state of the Markovian background process, may change between the time of request and the advised time of departure. Therefore, we also consider an online version of the optimal departure time problem, in which the traveler receives departure time updates while still at the origin.

- Lastly, we provide a selection of numerical experiments in which we implement our procedure and study the properties of the optimal departure time. We show that the optimal departure time is greatly affected by both the state of road network and the time of request. Moreover, we quantify the (potentially substantial) gains in travel time budget that can be realized by utilizing the online version of the optimal departure time problem. We also examine the efficiency of our procedure, and demonstrate that our procedure can successfully be employed in a real-world road network.

This paper is organized as follows. Section 2 describes our model and objective. Then, in Section 3 a procedure is presented for obtaining the travel-time distribution of individual links, after which computationally efficient algorithms are outlined that output the optimal departure time for a given path or OD pair, target arrival time and on-time probability. Numerical experiments are performed in Section 4 that aim to exemplify a selection of properties of the optimal departure time and studies the efficiency of our procedure. Finally, Section 5 gives some conclusive remarks. Various extensions to our procedure are discussed in the appendix.

2. Model and problem description

We consider a single vehicle that plans an upcoming trip between an OD-pair in a road network, wishing to arrive at the destination before a given time. Travel times in the network are subject to stochastic disruptions, i.e., events that affect driveable vehicle speeds. These disruptions (in the sequel often referred to as incidents) are modeled via a Markovian background process. At the time the traveller requests advice (with which we identify \( t = 0 \) throughout this paper), the state of the background process is known. If one would have access to the travel-time distribution for any departure time (after \( t = 0 \), that is), given knowledge of the background
process at $t = 0$, one could find the \textit{optimal departure time}, i.e., the latest departure time such that the probability of arriving on-time is at least, say, $\eta$. A subtlety is that it should be incorporated that the state of the background process could change between the request time and the departure time.

In Section 2.1 we describe the Markovian Velocity Model (MVM) employed to model the effect of disruptions on the vehicle speeds. With the notation introduced there, Section 2.2 formally introduces the problem of determining the optimal departure time.

2.1. Travel Time Dynamics: Markovian Velocity Model.

Let $G = (N, A)$ be a graph representation of the considered road network, of which the set of nodes $N$ represents the intersections in the road network and the set of directed arcs $A$ represents the roads connecting these intersections. Thus, we have that $(k, \ell) \in A$ if and only if there is a directed road segment from the intersection represented by node $k$ to the intersection represented by node $\ell$. Throughout, we let $n := |A|$ and write $A = \{a_1, \ldots, a_n\}$ for the set of arcs in $G$, with $a_i := (k_i, \ell_i)$ for some $k_i, \ell_i \in N$.

In reality, when traversing a link $a \in A$, the driveable vehicle speed is not always constant, as the occurrence of traffic events potentially affects the velocities. Importantly, the most prominent source of speed variability is formed by random traffic incidents, such as accidents, on-road obstacles, and vehicle break-downs. The MVM, originally introduced in Levering et al. (2022a, b), uses a Markovian background process $B(t)$ to record the evolution of these traffic incidents in the road network. By doing so, the model provides a direct relationship between traffic events and travel times, creating a transparent modelling framework that can be made operational with relatively low effort (as demonstrated in Levering et al. (2022a, b)).

For $a_i \in A$, define $\{X_{a_i}(t), t \geq 0\}$ as a two-state continuous Markov process with

$$X_{a_i}(t) = \begin{cases} 1 & \text{if there is no incident on arc } a_i \text{ at time } t \\ 2 & \text{otherwise.} \end{cases} \quad (1)$$

Hence, $X_{a_i}$ provides information about a possible incident on the arc $a_i$ in $G$. We let the velocity of a vehicle traversing $a_i$ depend on the occurrence of incidents in the following way: the vehicle speed at time $t$ equals $v_{a_i}(s_i)$ if $X_{a_i}(t)$ is in state $s_i \in \{1, 2\}$. We assume that for $i$ unequal to $j$ the processes $X_{a_i}(t)$ and $X_{a_j}(t)$ evolve independently. Consequently, for $i = 1, \ldots, n$, the
process $X_{a_i}(t)$ is completely described by its transition rate matrix

$$\begin{align*}
Q_{a_i} &= \begin{bmatrix}
-\alpha_i & \alpha_i \\
\beta_i & -\beta_i
\end{bmatrix},
\end{align*}$$

with $\alpha_i, \beta_i \in \mathbb{R}_{>0}$, and initial state $X_{a_i}(0)$. Thus, observe that $X_{a_i}(t)$ is a process that cyclically switches between an incident and incident-free state, whose durations have exponential distributions with mean $1/\alpha_i$ and $1/\beta_i$ respectively. In conclusion,

$$B(t) := (X_{a_1}(t), \ldots, X_{a_n}(t)),$$

is a Markovian background process with state space $S = \{1, 2\}^n$, that tracks the occurrence of incidents in the road network. The MV M then describes the effect of this background process, characterized by its initial $2^n$-dimensional state $B(0)$ and $(2^n \times 2^n)$ transition rate matrix denoted $Q$, on the arc speeds.

**Remark 1.** It is noted that the MV M as presented in this subsection does not cover the full potential of the MV M as demonstrated in [Levering et al. (2022b)]. However, as the main focus of this paper regards the deduction of the optimal departure time in a network with stochastic velocities, and not the model of these stochastic velocities itself, we use, for expositional reasons, a compact version of the MV M. Specifically, in this paper we only allow the speed on arc $a_i$ to depend on the state of $X_{a_i}(t)$, whereas the full MV M allows the speed to depend on the complete state of $B(t)$. Moreover, we only consider the processes $X_{a_i}(t)$ in the two-state form \([1]\), whereas the full MV M allows any continuous-type Markov process. Thus, by working with phase-type distributions, which are capable of approximating any distribution (on the positive numbers) arbitrary closely, also non-exponential incident durations and inter-incident times could be incorporated. Lastly, whereas we only consider the impact of incidents, the process $B(t)$ can easily be extended with a Markov process $Y(t)$ that captures the recurrent traffic patterns. Importantly, the presented results and algorithms can be extended to include these three generalizations; for the latter two generalizations, this is described in more detail in Appendix A.

The MV M, particularly in its full version (Remark 1), is extremely flexible, and therefore well capable of describing real-world travel-time distributions. Indeed, in [Levering et al. (2022a)] it is shown how a database of incident registrations and loop detector data can be used to operationalize the MV M. Specifically, it is shown how to model the randomness of incidents lengths and inter-incident durations, and set the corresponding driveable speed levels. This
allows the fitting of the mvm in e.g. the Dutch road network, in which a high density of loop detectors and lists of occurred incidents are available.

2.2. Objective: Departure Time Advice.

In this subsection we outline the problem of determining the optimal departure time. In doing so, we distinguish between two settings, called the offline and the online setting. In the offline setting, the optimal departure time is only determined once, based on the information available at the request time, after which the traveler will indeed leave at this time instance. In the online setting, however, the departure time can be updated while the traveler is waiting, as changing conditions may result in a new optimum.

**Offline Setting** — We will first formally define the optimal departure time. Suppose a traveler requests a route at the current time, i.e., \( t = 0 \), and is interested in the latest time of departure such that a certain arrival on-time probability can be guaranteed. If the requested arrival time is given by \( t = M > 0 \), i.e. \( M \) time units from the time of request, and if the desired on-time arrival probability is at least \( \eta \in (0, 1) \), the optimal departure time of a traveler is defined as

\[
t^* := \sup \{ t \geq 0 : P(Y_t \leq M \mid B(0)) \geq \eta \}.
\]  
(3)

In (3), the random variable \( Y_t \) represents the arrival time when departing at time \( t \), and can be written as \( Y_t = t + T_t \), with \( T_t \) the travel time when departing at time \( t \). Importantly, the random variables \( Y_t \) and \( T_t \) are affected by the current background state \( B(0) \). It is noted that the distributions of \( Y_t \) and \( T_t \) will change once future information about the state of the network becomes available (i.e., \( B(s) \), when \( s \) is approaching \( t \)). In this offline setting, however, we assume that only the current state of the network \( B(0) \) is known. Note that the on-time arrival probability at time of departure will generally differ from \( \eta \). Of course, it may happen that there exists no \( t \geq 0 \) that satisfies the condition in (3). In this case, we put \( t^* := -\infty \), and it depends on the preferences of the driver to either depart immediately or to not depart at all.

We will now pay closer attention to the conditional probability in (3). First, recognize that the travel-time distribution depends on the departure time \( t \geq 0 \) only through the state \( B(t) \), which is unknown at time 0. However, it is possible to determine the distribution of \( B(t) \) by using the known current state \( B(0) = s \) and transition matrix \( Q \). Using general results for continuous-time Markov chains [Norris 1997, Ibe 2013], it follows that this distribution is given by

\[
p^s_t := (P(B(t) = s' \mid B(0) = s))_{s' \in S} = p^s_0 e^{Qt},
\]  
(4)
with \( \mathbf{p}_s^t \), by definition, an row vector of dimension \(|S| = 2^n\), with a 1 at the entry that corresponds to the state \( s \in S \) and a 0 at every other entry. Thus, with \( T[s'] \) denoting the travel time corresponding to departing when the background process is in the state \( s' \), we can write
\[
\mathbb{P}(Y_t \leq M \mid B(0) = s) = \sum_{s' \in S} \mathbb{P}(B(t) = s' \mid B(0) = s) \mathbb{P}(T[s'] \leq M - t) = \mathbf{p}_s^t \left( \mathbb{P}(T[s'] \leq M - t) \right)_{s' \in S}.
\] (5)

Hence, if we are able to determine the distribution of the travel time \( T[s] \) for each state \( s \in S \), this would allow us to compute the conditional probability in (5) for each \( t \), which in turn would facilitate determining the maximizer \( t^* \) in (3). The evaluation of the travel-time distribution is outlined in detail in Section 3.

**Online Setting** — In the offline setting, the objective is to produce an optimal departure time \( t^* \) that depends solely on the current state of the background process \( B(0) \). However, when a traveler waits for departure, new information on the state of the background process becomes available. This in turn alters the distribution \( \mathbf{p}_s^t \) in (4). More specifically, for \( u \leq t^* \), the distribution of \( B(t^*) \) given \( B(u) = s \) is
\[
\left( \mathbb{P}(B(t^*) = s' \mid B(u) = s) \right)_{s' \in S} = \left( \mathbb{P}(B(t^* - u) = s' \mid B(0) = s) \right)_{s' \in S} = \mathbf{p}_{t^*-u}^s e^{Q(t^*-u)}.
\] (6)

Since the distribution of \( B(t^*) \) changes as time \( u \) progresses from 0 to \( t^* \), the on-time arrival probability of the driver changes as well. Ideally, as a driver is waiting to depart, their optimal departure time is updated such that it incorporates the latest state of the network. This way, as time passes, the traveler can request a new departure time that is given by
\[
t^*_u := \sup \{ t \geq u : \mathbb{P}(Y_t \leq M \mid B(u) \geq \eta) \}.
\] (7)

Just as in the offline setting, \( Y_t \) should be interpreted as the arrival time, with the current time now being the request time \( u \). Therefore, \( t^*_u \) is the latest departure time such that at time \( u \) the on-time arrival probability is at least \( \eta \), and this online departure time coincides with the offline departure time if \( u = 0 \) (conditional on \( B(u) \) applying at time 0).

Again, just like in the offline setting, the on-time arrival probability at time of departure may differ from \( \eta \). However, the request time will approach the departure time if a driver keeps updating their optimal departure time. Therefore, the on-time arrival probability at time of request will approach the on-time arrival probability at time of departure. Hence, in the online
setting, it is in fact possible to find the latest departure time such that, on departure, a certain on-time probability can be satisfied.

3. Deriving the Optimal Departure Time

Recall that we consider the setting in which a vehicle wants to know its optimal departure time for the traversal of an OD-pair in a network $G = (N, A)$, in which vehicle speeds are described by the Markovian Velocity Model that was presented in Section 2.1. In Section 3.1 we point out how the travel-time distribution can be numerically evaluated by applying a discretization procedure, while in Section 3.2 we discuss how the corresponding granularity can be determined. Then, Section 3.3 outlines how to compute the optimal departure time for a given path, or, if only the destination of the trip is known, how to obtain both the optimal departure time and corresponding path to travel. We consider the case in which the traveler only requests a departure time once, i.e., the offline setting, as well as how to extend this procedure to the case in which the traveler is allowed to receive departure-time advice updates, i.e., the online setting. Note that, even though this section only treats the compact version of the MVM, the optimal departure time procedures extend to the case the Markov process $B(t)$ describes more detailed congestion phenomena, of which two examples are provided in Appendix A.

3.1. Travel-Time Distribution.

Consider a vehicle that departs at $t = 0$ to traverse a given path in the network $G = (N, A)$. The travel-time distribution of this vehicle is completely determined by the velocity dynamics, described through the dynamics of the Markovian background process $B(t)$ and its initial state $B(0)$. As both exact methods and Laplace inversion fail (see Remark 2 below), we use a discretization procedure to obtain an accurate approximation to the travel-time distribution. Specifically, we discretize the moments at which there is a potential transition in the driveable speed (which corresponds to a transition of the background process).

Remark 2. While traveling on a given link, the background process, and thus the driveable vehicle speed, can in principle have infinitely many transitions. Therefore, a closed form distribution function of the per-link travel time is unknown. Since the Laplace-Stieltjes transform is known (Levering et al., 2022b), a natural procedure for obtaining the per-link travel-time distribution would be to rely on the numerical inversion of this transform. Unfortunately, application of common inversion methods (e.g., Abate and Whitt (1992), den Iseger (2009), or the saddlepoint approximation (Butler 2007)), is not applicable. This is due to the fact that
the per-link travel-time distribution is neither discrete nor continuous. To see this, we consider link $a$ with length denoted $d_a$, which takes state $X_a(0) = s$ upon departure. Now, there is a positive probability that the state of $X_a(t)$ does not change while traveling link $a$, and thus, that the link travel time equals $d_a/v_a(s)$. However, as is easily seen, on the remainder of the domain the per-link travel time has a continuous density.

An easy fix for the challenges discussed in Remark 2 would be to assume that the driveable vehicle speed on a link is fixed upon entering a link, in that is completely determined by the background state upon entering (i.e., not affected by any transitions of the background process while traversing the link). Doing so, the link travel time would reduce to a discrete distribution with travel-time probabilities that can easily be computed. While this procedure clearly gives a decent approximation for short links, it could perform poorly for long links. This inspired us to the following idea: instead of assuming a fixed driveable vehicle speed for an entire link, we only assume fixed velocities for a certain (short) time interval. We now provide a detailed description of the procedure to obtain an approximation for the per-link travel-time distribution (which is also summarized in Algorithm 1 below).

As a first step, we focus on the travel-time distribution for the traversal of a single link $a \in A$. On this link, the driveable speed level is either $v_1 := v_a(1)$ or $v_2 := v_a(2)$ for any $t \geq 0$ (see Section 2.1). Since we are solely given the state of $X_a(t)$ at $t = 0$, only the driveable speed upon departure is known. Now, to be able to compute the travel-time distribution, we discretize the moments at which the background process, and consequently, the speed level, can change. Concretely, given a (typically small) $\delta \in \mathbb{R}_{>0}$, we define the Markov chain $X'_a(t)$ as a discrete-time version of $X_a(t)$, at times $t = 0, \delta, 2\delta, \ldots$. That is, we set $X'_a(0) = X_a(0)$, and let the discrete-time Markov chain $X'_a(t)$ (with $t = 0, \delta, 2\delta, \ldots$) evolve according to the diagram of Figure 1b. Thus, if the process $X'_a(\cdot)$ is in state 1 at some time $m\delta$ (with $m \in \mathbb{N}_0$), it is still there at $(m + 1)\delta$ with probability $p := e^{-\alpha \delta}$, i.e., the probability that the continuous-time Markov chain $X_a(\cdot)$ does not jump to state 2 in a time interval of length $\delta$. Alternatively, the process jumps to state 2 with probability $1 - p$, i.e., the probability that $X_a(\cdot)$ jumps in a time interval of length $\delta$. Note that, even though this latter probability incorporates the event of multiple transitions of the continuous-time process $X_a(\cdot)$, our procedure is justified by the fact that, if $\delta$ is chosen sufficiently small, the probability of more than one such transition within a time interval of length $\delta$ is $o(\delta)$ and therefore negligible; further details regarding the choice of $\delta$ are discussed in Section 3.2. In a similar fashion, if $X'_a(\cdot) = 2$, the process stays in state 2 with
probability $q := e^{-\beta \delta}$, and moves to state 1 with probability $1 - q$. Now that we have described the dynamics of $X'_a(\cdot)$, the corresponding velocities can be defined: if, for $m \in \mathbb{N}_0$, $X'_a(m\delta) = s$, the speed level during the interval $[m\delta, (m+1)\delta)$ equals $v_a(s)$. Hence, during such an interval of length $\delta$, the speed level is constant.

1. Continuous.

\[
\begin{array}{c}
1 \\
\alpha \\
2 \\
\beta
\end{array}
\]

(a) Continuous.

\[
\begin{array}{c}
1 \\
1 - e^{-\alpha \delta} \\
2 \\
e^{-\beta \delta}
\end{array}
\]

(b) Discrete.

Figure 1. Transition rate diagrams for the continuous process $X_a(t)$ and its discretized version $X'_a(t)$.

Note that with the described velocity dynamics, we are able to iteratively compute the travel time distribution on link $a$. That is, consider the case that upon departure, link $a$ is free of incidents, i.e., $X'_a(0) = 1$ (a similar procedure can be followed for the alternative situation). Then, with a constant speed $v_1$ during $[0, \delta)$, the traveled distance at time $\delta$ equals $v_1\delta$ with probability 1. Now, the traveled distance at time $2\delta$ equals $(v_1 + v_2)\delta$ with probability $1 - p$ (the probability that $X'_a(\delta) = 2$), and $2v_1\delta$ with probability $p$ (the probability that $X'_a(\delta) = 1$). These two cases each lead to two potential travel times at time $3\delta$: in case $X'_a(\delta) = 1$, the traveled distance equals $3v_1\delta$ with probability $p^2$, and equals $(2v_1 + v_2)\delta$ with probability $p(1 - p)$, and in case $X'_a(\delta) = 2$, the traveled distance equals $(2v_1 + v_2)\delta$ with probability $(1 - p)(1 - q)$, and $(v_1 + 2v_2)\delta$ with probability $(1 - p)q$. We can iteratively continue these computations, in which every (state, distance, probability)-tuple at $t = m\delta$ generates two tuples for $t = (m+1)\delta$. Thus, any tuple in $t = (m+1)\delta$ has a so-called ancestor in $t = m\delta$.

To obtain the travel-time distribution of this link, recall that $d_a$ is the distance of the link, and thus the total distance to travel. Therefore, if at $t = m\delta$ a tuple $(s_0, d_0, p_0)$ with traveled distance $d_0 < d_a$ generates a tuple $(s_1, d_1, p_1)$ for which $d_1 \geq d_a$, then there is a probability $p_1$ that the travel time of the link equals $m\delta + (d_a - d_0)/v_a(s_1)$. Namely, in this scenario, after $m\delta$ time the vehicle still needs to traverse a distance $d_a - d_0$, which it travels with the constant speed level belonging to state $s_1$. The collection of travel-time values and corresponding probabilities that are iteratively found in this fashion, form the travel-time distribution. We denote this collection for link $a_i$ given $X_{a_i}(0) = j$ as $L^J_{d_a}$. Observe that the tuples $(s, d, p)$ for which $d \geq d_a$ do not serve as an ancestor in a new iteration, such that the new iteration only continues with tuples for which the traveled distance has not yet exceeded the length of the link. Moreover,
since the traveled distance grows with $v_1 \delta$ or $v_2 \delta$ every step, there are only finitely many steps before $d_a$ is exceeded, and consequently, the iterative procedure terminates in a finite number of steps.

![Algorithm 1](image)

**Algorithm 1:** Travel-time distribution of a single link $a$

**Result:** list $L$ of (travel time, probability) values.

Notation: for transparency, omitted subscripts for $v_a(s)$, $X_a(s)$ and $d_a$;

Given: $\delta \in \mathbb{R}_{>0}$, $X(0) = s$, $p = e^{-\alpha \delta}$, $q = e^{-\beta \delta}$;

Initialization: $L = \emptyset$, $S = \{(s, \delta v(s), 1)\}$, $S' = \emptyset$, $m = 0$;

while $S$ nonempty do

1. foreach $(s', d', p')$ in $S$ do
   a. Extract $(s', d', p')$ from $S$;
   b. Compute $p_1 = p'(1\{s'=1\}p + 1\{s'=2\}(1-q))$ and $p_2 = p'(1\{s'=1\}(1-p) + 1\{s'=2\}q)$;
   c. If $d' + v(1)\delta < d$ then append $(1, d' + v(1)\delta, p_1)$ to $S'$. Else append $(m\delta + (d-d')/v(1), p_1)$ to $L$;
   d. If $d' + v(2)\delta < d$ then append $(2, d' + v(2)\delta, p_2)$ to $S'$. Else append $(m\delta + (d-d')/v(2), p_2)$ to $L$;
2. $m += 1$;
3. Set $S = S'$, $S' = \emptyset$;

**Example 1.** Figure 2 displays the iterative steps of the procedure via a tree structure, for obtaining the travel-time distribution on a link $a \in A$ with $d_a = 4$ km, $v_a(1) = 100$ km/h,
and $X_{a}(0) = 1$. In the discretization, we only allow speed transitions at full minutes, i.e., we let $\delta = 1/60$ h. Iteration at a branch of the tree stops in case the traveled distance exceeds 4 km, and results in a travel-time value. For example, the upper branch yields a travel time of $2/60 + (4-10/3)/100 = 1/25$ h, and the branch directly below a travel time of $2/60 + (4-10/3)/60 = 2/45$ h. ♦

Remark 3. We observe that, since we are only working with two constant speed values per time step, the iterative procedure can, in this special case, also be represented by a binomial tree. To be able to generate such a tree and to compute the resulting travel-time distribution, we look, contrary to Algorithm 1, at tuples of the form $(d,p_1,p_2)$, with $d$ is the traveled distance, and $p_1$ ($p_2$, respectively) the probability of the scenario in which the vehicle had speed $v_1$ ($v_2$, respectively) in the last time step. Note that we need to separate these two probabilities, as the corresponding two scenarios affect the probabilities of the next time step differently. Figure 3 shows the binomial tree corresponding to Figure 2.

Importantly, a binomial tree grows by maximally one item per time step, making this procedure particularly efficient. Indeed, after $m$ time steps, the number of ancestors in a binomial tree is only of the order $m$, whereas the number of ancestors in Algorithm 1 would be of order $2^m$. ♦

![Figure 3. Binomial tree representation corresponding to Example 1.](image)

Now, knowing how to compute the travel-time distribution for the traversal of an individual link, we may redirect our focus to the travel-time distribution of a full path $P$. Without loss of generality, we let $P = \{a_1, \ldots, a_m\}$. Writing $s_i$ for the initial state $X_{a_i}(0)$, the travel-time distribution for $a_1$ is known and given by the list $L_{a_1}^s$, consisting of pairs $(t,p)$, with $t$ a travel-time value and $p$ the corresponding probability. Now, let us have a list $L$ of $(t,p)$ pairs that form the travel-time distribution on the subpath $\{a_1, \ldots, a_i\}$. Then, we observe that, if the travel time for $\{a_1, \ldots, a_i\}$ equals $t_1$, the probability that $X_{a_{i+1}} = j$ upon entering $a_{i+1}$ is the
(s_{i+1}, j)-th index of $e^{t_{i}Q_{a_{i+1}}}$, with $Q_{a_{i+1}}$ the transition rate matrix of $a_{i+1}$ as defined in \cite{2}.

Therefore, the travel-time distribution $T[s]$ for traversing $\{a_{1}, \ldots, a_{i+1}\}$ with initial state $s$ is given by the list

$$\{(t_{1} + t_{2}, p_{1} \cdot p_{2} \cdot [e^{t_{1}Q_{a_{i+1}}}]_{(s_{i+1}, j)}) \mid (t_{1}, p_{1}) \in L, (t_{2}, p_{2}) \in L_{a_{i+1}, j}^{j}, j = 1, 2\}. \quad (8)$$

Notably, by iteratively setting $i = 1, \ldots, m - 1$, we obtain the travel-time distribution of the path $P$. To prevent that the number of elements in the list $L$ grows exponentially with the number of links in the path, we aggregate the travel-time values into equally sized bins after every iteration.

### 3.2. Granularity.

Now that we have outlined the discretization procedure that allows us to approximate the per-path travel-time distribution, it remains to choose an appropriate value for the length of the time intervals $\delta$. The choice of $\delta$ affects the number of steps in the iterative procedure, which we call the granularity. It is clear that choosing a relatively large value of $\delta$ renders the algorithm fast. However, for two reasons the value of $\delta$ may not be chosen too large, as we point out now.

First, for a given link $a \in A$, we used in Section \ref{3.1} the approximation

$$\mathbb{P}(X_{a}(\delta) = 1 \mid X_{a}(0) = 1) \approx p := e^{-\alpha \delta}.$$ 

Note that $e^{-\alpha \delta}$ is the probability that $X_{a}(0)$ has no transitions in the time interval $\delta$. However, the probability $\mathbb{P}(X_{a}(\delta) = 1 \mid X_{a}(0) = 1)$ also contains the events in which $X_{a}(\cdot)$ jumps multiple times, and returns to its original state to remain in that state until the end of the time interval $\delta$. Therefore, in order to ensure that the approximations we used in Section \ref{3.1} are justifiable, we require the probability of two or more transitions conditional on $X_{a}(0) = 1$ to be negligible. With $E(\gamma)$ an exponentially distributed random variable with parameter $\gamma$ and $f_{\gamma}(\cdot)$ its density, the probability of two or more transitions conditional on $X_{a}(0) = 1$ is given by

$$\int_{0}^{\delta} f_{\alpha}(t) \int_{0}^{t-\delta} f_{\beta}(s) \, ds \, dt = \int_{0}^{\delta} \alpha e^{-\alpha t} (1 - e^{-\beta(\delta - t)}) \, dt$$

$$= 1 - e^{-\alpha \delta} - \frac{\alpha}{\alpha - \beta} (e^{-\beta \delta} - e^{-\alpha \delta}) = 1 - \frac{\alpha e^{-\beta \delta} - \beta e^{-\alpha \delta}}{\alpha - \beta}.$$

Due to the fact that this expression is symmetric in $\alpha$ and $\beta$, it also equals the probability of two or more transitions conditional on $X_{a}(0) = 2$. This concretely means that, for a given small value of $\varepsilon_{1} > 0$ (for instance 0.01), the interval length $\delta$ should be chosen sufficiently small so
that
\[
1 - \frac{\alpha e^{-\beta \delta} - \beta e^{-\alpha \delta}}{\alpha - \beta} < \varepsilon_1. \tag{9}
\]
As \( \delta \) should be chosen small, we can simply evaluate (9) working with its second-order Taylor approximations in \( \delta \) at 0. Indeed, as \( \delta \downarrow 0 \),
\[
1 - \frac{\alpha e^{-\beta \delta} - \beta e^{-\alpha \delta}}{\alpha - \beta} = \frac{\frac{1}{2} \alpha^2 \beta \delta^2 - \frac{1}{2} \alpha \beta^2 \delta^2}{\alpha - \beta} + o(\delta^2) = \frac{1}{2} \alpha \beta \delta^2 + o(\delta^2).
\]
We thus find that (9) reduces to
\[
\delta < \left( \frac{2 \varepsilon_1}{\alpha \beta} \right)^{1/2}. \tag{10}
\]
From (10), it is clear that \( \delta \) should be chosen smaller for higher transition rates \( \alpha, \beta \). As higher transition rates result in a higher probability of the occurrence of two or more transitions within a fixed time interval, it is also intuitively clear that \( \delta \) should be decreasing in \( \alpha, \beta \) in order to constrain this approximation error.

Second, we used that the driveable vehicle speed on a link is fixed for the time period \( \delta \). If, however, a transition from, say, state 1 to state 2 occurs at time \( t < \delta \), the actual traversed distance in the time interval \( \delta \) would be \( v_1 t + v_2 (\delta - t) \) as opposed to our approximated \( v_1 \delta \). This approximation error is more substantial if the difference between the velocities \( v_1 \) and \( v_2 \) increases. In case \( X_a(0) = 1 \), we thus want to choose \( \delta \) sufficiently small such that for a chosen small \( \varepsilon_2 > 0 \), the expected error is bounded:
\[
(1 - e^{-\alpha \delta}) \left| \int_0^\delta \frac{\alpha e^{-at}}{1 - e^{-\alpha \delta}} (v_1 t + v_2 (\delta - t)) \, dt - v_1 \delta \right| < d \cdot \varepsilon_2.
\]
This can be rewritten to
\[
|v_1 - v_2| \left( \frac{1}{\alpha} (1 - e^{-\alpha \delta}) - \delta \right) < d \cdot \varepsilon_2. \tag{11}
\]
A similar condition can be obtained for the case \( X_a(0) = 2 \), but with the \( \alpha \) replaced by \( \beta \) in (11). Note that the error \( \varepsilon_2 \) is multiplied by the length \( d \), since we are interested in the error relative to the length of a link. Just as we did for the first approximation error, we can simplify (11) further by considering its Taylor approximations in \( \delta \) at 0. Doing this for both \( X_a(0) = 1 \) and \( X_a(0) = 2 \), we find
\[
\delta < \left( \frac{2 \varepsilon_2}{|v_1 - v_2|} \right)^{1/2} \quad \text{and} \quad \delta < \left( \frac{2 \varepsilon_2}{d \cdot |v_1 - v_2|} \right)^{1/2}. \tag{12}
\]
Combining conditions \ref{eq:10} and \ref{eq:12}, we conclude that both the probability of two or more transitions and the expected error are sufficiently small, whenever we pick

$$\delta < \min \left\{ \left( \frac{2\varepsilon_1}{\alpha \beta} \right)^{1/2}, \left( \frac{2\varepsilon_2}{|v_1 - v_2| \max\{\alpha, \beta\}} \right)^{1/2} \right\},$$

\hfill (13)

### 3.3. Optimal Departure Time.

So far, we have described how to compute the travel-time distribution $T[s]$ for a vehicle traversing a path $P$ and departing when $B(t)$ is in state $s$, for any $s \in S$. Using knowledge of the dynamics of the background process $B(t)$, this allows us to compute the on-time arrival probability for a path $P$ and any departure time $t \geq 0$. Observing that the on-time arrival probability is monotonically decreasing in the departure time, by performing an elementary bisection we can determine the optimal (the latest, that is) time to depart on path $P$ for a given on-time arrival probability. This monotonicity can moreover be used to extend the results to the case in which the user only specifies the origin and destination, instead of the specific path to travel. The procedure then compares the departure times for different paths, selects the latest, and outputs both $t^*$ and the corresponding path to travel.

#### 3.3.1. Optimal departure time for a path — offline setting.

One natural way to obtain the on-time arrival probability on a path $P$ for a departure time $t \geq 0$ was presented in Section 2.2, namely, computing the product in \ref{eq:5}. To this end, we need to evaluate $P(T[s'] \leq M-t)$ for all $s' \in S$. Note that the distribution of $T[s']$ can be derived by the presented discretization procedure, which outputs a list of (travel time, probability)-pairs. By summing all probabilities for which the corresponding travel-time value does not exceed $M-t$, we obtain a value for $P(T[s'] \leq M-t)$.

However, computing the arrival probability via \ref{eq:5} is typically time consuming, as it requires us to compute the distribution of $T[s]$ for all $s \in S$. Fortunately, there is an alternative procedure for which only one travel-time distribution needs to be derived. That is, realize that departing at time $t \geq 0$ to traverse path $P$ can be viewed as departing at time 0 and, before entering $P$, first a fictional link has to be traversed for which the travel time equals $t$ with probability 1. Therefore, we directly obtain the distribution of the arrival time $Y_t$ by computing the travel time distribution of this partly fictional path (with departure at time 0). Then, the on-time arrival probability can be found by summing all probabilities for which the corresponding travel time does not exceed $M$. 
Now, since we are able to compute the on-time arrival probability on the path \( P \) for a given departure time \( t \) and, moreover, this probability is monotonically decreasing in the departure time, we can use bisection to find the optimal departure time for the given on-time arrival probability \( \eta \). The monotonicity of the on-time probability follows directly from Proposition 1 of Levering et al. (2022b), which gives that for a path consisting of a single link, \( t' \leq t \) implies \( t' + T_{t'} \leq t + T_t \),

and therefore, \( Y_{t'} \leq Y_t \). Since the minimum and maximum velocity on all links of the path are known, the minimum travel time \( t_{\text{min}} \) and the maximum travel time \( t_{\text{max}} \) are known as well, so that

\[
I_0 := \left[ \max\{0, M - t_{\text{max}}\}, \max\{0, M - t_{\text{min}}\} \right]
\]

serves as natural starting interval for the bisection method. First, we check if \( I_0 \) equals \([0,0]\), since in that case, the minimum travel time is at least \( M \), and \( t^* := -\infty \). Second, we check the on-time arrival probability at the left boundary. If this probability is below \( \eta \), \( t^* := -\infty \) as well. In case neither is true, we apply the bisection algorithm until we obtain the latest departure time for which the on-time probability is at least \( \eta \) (which is guaranteed to exist by the first two checks).

Algorithm 2 now summarizes the complete procedure for obtaining the optimal departure time \( t^* \) to traverse a path \( P \) for a given on-time arrival probability \( \eta \) in the offline setting, in which a traveler requests the value of \( t^* \) once. It is important to note that we can precompute the travel-time distribution for every link \( a_i \) in the network, and for every initial state \( s \) of this link. We can then store these distributions as the lists \( L^s_{a_i} \). Then, upon an optimal departure time request of a vehicle, we can directly use these distributions, and do not need to compute them on-the-spot. Thus, the computational costs are only determined by Part II of the algorithm.

3.3.2. Optimal departure time for an OD-pair — offline setting. We can now extend the results to the natural setting in which a user does not specify the complete path to travel, but only the destination point \( k^* \). Then, the traveler does not just request the optimal departure time to reach this endpoint with on-time arrival probability \( \eta \), but also requests the specific path that guarantees this probability \( \eta \). Note that, with \( \mathcal{P} \) the set of all paths to the endpoint, we can simply compute the optimal departure time for all \( P \in \mathcal{P} \), and output the path with the latest departure time. However, since the size \(|\mathcal{P}|\) of such paths is typically huge, the above procedure is not applicable in practical settings. Therefore, we present two alternative methods: an exact
Algorithm 2: Optimal departure time offline setting

Result: optimal departure time for traversing path \((a_1, \ldots, a_m)\) within time \(M\) with probability \(\eta\), given \(B(0) = s\).

Part I (precomputations):

\[
\text{for } i = 1, \ldots, m \text{ do} \\
1. \text{for } a_i, \text{ compute } \delta \text{ via (13)}; \\
2. \text{Compute } L_{a_i}^1 \text{ and } L_{a_i}^2 \text{ with Algorithm 1 and } \delta \in \mathbb{R} \text{ from step 1};
\]

Part II (on-the-spot computations):

1. Define the function:

\[
\text{OnTimeProbability} \ (t, s, M) \\
\text{Set } L = \{(t, 1)\}; \\
\text{for } i = 1, \ldots, m \text{ do} \\
\quad \text{Set } L = \{(t_1 + t_2, p_1 \cdot p_2 \cdot [e^{Q_{a_i} t_1}]) | (t_1, p_1) \in L, (t_2, p_2) \in L_{a_i}^j, j = 1, 2\}; \\
\text{Return } \sum_{(t_1, p_1) \in L : t_1 \leq M} p_1;
\]

2. Compute \(t_{\text{min}} = \sum_{i=1}^{m} d_{a_i} / v_{a_i}(2)\) and \(t_{\text{max}} = \sum_{i=1}^{m} d_{a_i} / v_{a_i}(1)\);

3. Set \(I_0 = [\max\{0, M - t_{\text{max}}\}, \max\{0, M - t_{\text{min}}\}]\);

4. if \(I_0 = [0, 0]\) or \(\text{OnTimeProbability}(\max\{0, M - t_{\text{max}}\}, s, M) \leq \eta\) then

\[
\text{Return } t^* = -\infty;
\]

else

\[
\text{Use the bisection method with initial interval } I_0 \text{ on the function} \\
\text{OnTimeProbability}(t, s, M), \text{ until obtain latest departure time for which output} \\
is \text{at least } \eta. \text{ Return this departure time } t^*.
\]

procedure that is still somewhat computationally demanding, and a very efficient, near-optimal method:

- Bisection method: similar to Algorithm 2, the first procedure uses a bisection algorithm to obtain the optimal departure time and corresponding path. That is, for a given departure time \(t\), it uses a label-correcting algorithm to output the path with maximum on-time arrival probability. This algorithm (Algorithm 3) will be described in more detail below. Now, as the maximum of monotonically decreasing functions, the on-time arrival probability outputted by the label-correcting algorithm is again a monotonically decreasing function of the departure time \(t\). Consequently, bisection can indeed be
employed to find the optimal departure time and corresponding path. Note that even though the bisection method is guaranteed to find the optimal path and departure time, the computational complexity of the label-correcting algorithm may prohibit practical application.

- **k-shortest path method**: we simply compute the optimal departure time for a small subset of \( \mathcal{P} \). Concretely, we compute the optimal departure time for the subset of \( k \) shortest paths (e.g. in distance) to the destination. Thus, with \( \mathcal{P}' \) the set of \( k \) shortest path as found via Yen’s algorithm (Yen, 1970, 1971), we use Algorithm 2 to compute the optimal departure time for all \( p \in \mathcal{P}' \), and output the path with latest departure time. Note that, for \( k \) large enough, \( \mathcal{P}' \) will typically contain the path which yields the latest departure time. Importantly, as the optimal departure times for the different paths can be computed in parallel, this method can indeed be employed in a highly efficient way.

We are left with describing the label-correcting algorithm, used within the bisection method, to output the path with maximum on-time arrival probability for a given departure time \( t \).

This algorithm, outlined in Algorithm 3, is an A∗-algorithm in the same spirit as the algorithm presented in Chen et al. (2014), assigning a label set to every node in the graph and updating these sets iteratively. The label set of a node is used to store travel-time distributions of paths from the origin to that node, when departing from the origin at time \( t \). Initially, the label set of the origin consists of a single element, namely the distribution \( L_0 = \{(t, 1)\} \), whereas the other label sets start empty.

The iteration uses a queue \( q \), whose elements are tuples of length four. Every tuple consists of a node in the graph \( k \), a path from the origin to this node \( (P) \), the list of (travel time, probability)-pairs forming the travel-time distribution of this path \( (L) \), and an upper bound of the maximum on-time arrival probability for any path from origin to destination that has \( P \) as subpath \( (\alpha) \). This upper bound \( \alpha \) is computed as \( \mathbb{P}(T_L + t_{\min}(k, k^*) \leq M) \), in which \( t_{\min}(k, k^*) \) is the minimum travel time from \( k \) to the destination \( k^* \) and \( T_L \) a random variable with distribution \( L \). At the start, \( q = \{\{(\alpha, L_0, k_0, \text{path: } \{k_0\}\})\} \), with \( \alpha = 0 \) if \( t + t_{\min}(k_0, k^*) > M \) and \( \alpha = 1 \) otherwise. In case \( \alpha = 0 \), all paths from \( k_0 \) to \( k^* \) will have a zero probability of on-time arrival, thus the algorithm stops and outputs probability 0. Otherwise, we continue. Now, every iteration step, the element with minimum \( \alpha \)-value is extracted from the queue. Then, for every neighbor \( k' \) of the node \( k \) that is not in \( P \), the travel-time distribution \( L' \) for traversing subsequently \( P \) and the link \( (k, k') \) can be computed via (8). Clearly, neighbors \( k' \in P \) are omitted as extending \( P \) with \( (k, k') \) would yield a suboptimal path containing a loop. After
computing $\alpha'$, the upper bound to the maximum on-time arrival probability for the travel-time distribution $L'$, we insert $L'$ into the label set of $k'$ and the tuple $\{\alpha', L', k', P+k'\}$ into $q$. This ends the current iteration step, and a new minimum element is extracted from $q$. The algorithm is terminated if the third element from the extracted tuple from $q$ equals the destination $k^\star$.

Now, to improve the speed of the procedure, we perform one extra step before inserting a new tuple in the label set of $k'$ and the queue $q$. That is, we check if the distribution $L'$ dominates one of the distributions already stored in the label set of $k'$, or vice versa. In this setting, dominance refers to (first-order) stochastic dominance: for two paths $p_1, p_2$ from $k_0$ to $k'$ with respectively travel-time distributions $L_1$ and $L_2$, we say that $L_1$ dominates $L_2$ if $\mathbb{P}(T_{L_2} \leq t) \leq \mathbb{P}(T_{L_1} \leq t)$ for all $t > 0$ and $\mathbb{P}(T_{L_2} \leq t) < \mathbb{P}(T_{L_1} \leq t)$ for at least one $t > 0$. Now, if $L_1$ would indeed dominate a distribution $L_2$ in the label set of $k'$, then, for every path $p'$ from $k'$ to the destination $k^\star$, the on-time arrival probability of the path consisting of $p_1$ and $p'$, is at least as high as the on-time arrival probability of the path consisting of $p_2$ and $p'$. Thus, a path with a travel-time distribution that is dominated by the travel-time distribution of at least one other path to the same node, is never part of the optimal path from the origin $k_0$ to the destination $k^\star$. Therefore, this subpath can be disregarded in subsequent iterations. Thus, in the algorithm, there is an extra check to see if $L'$ is dominated by a distribution in the label set of $k'$, or vice versa. Dominated distributions are removed from the label set, and the corresponding tuples are removed from the queue.

3.3.3. Optimal departure time for a path — online setting. We have already determined the optimal departure time $t^\star$ for a given path in the offline setting in Section 3.3.1. Note that this departure time depends solely on the current state of the background process $B(0)$. Therefore, if case $t^\star > 0$, the traveler should wait $t^\star$ time units until departure. However, during the time the traveler is waiting for departure, new information on the state of the background process becomes available. In the online setting, this new information is used to update the optimal departure time. Specifically, we will consider the case where the traveler requests a new optimal departure time every $\Delta > 0$ time units. We note that the value of $\Delta$ should be chosen thoughtfully: while a higher frequency of updating the optimal departure time (i.e., a smaller value for $\Delta$) allows for more up-to-date departure time advice, the computational burden also increases.

Determining the optimal departure time using the information available at time $i\Delta$ for $i \in \mathbb{N}_0$ can then be done in a very similar way as in Algorithm 2 with some minor adjustments. Recall
Algorithm 3: Optimal path for a given departure time

Result: path from \( k_0 \) to \( k^* \) with highest on-time arrival probability, given departure time \( t, B(t) = s \).

1. With \( L_0 = \{ (t, 1) \} \), set \( D_{k_0} = \{ L_0 \} \) and \( D_{k_i} = \emptyset \) for all other \( i \);
2. If \( t + t_{\text{min}}(k_0, k^*) > M \) quit and return 0. Else continue;
3. Set the queue \( q = \{ \{ 1, L_0, k_0, \text{path: } \{ k_0 \} \} \} \);
4. If \( k = k^* \) quit and return \( (\alpha, P) \). Else continue;
5. \textbf{foreach} neighbor \( k' \) of \( k \) not in \( P \) do
   \hspace{1em} a. Compute new travel-time distribution \( L' \) via (8) with \( a'_{i+1} = (k, k') \);
   \hspace{1em} b. Compute \( \alpha' = \mathbb{P}(T_{L'} + t_{\text{min}}(k', k^*) \leq M) \);
   \hspace{1em} c. \textbf{if} \( L' \) \textbf{is not dominated by an element from} \( D_{k'} \) \textbf{then}
      \hspace{2em} Remove all paths dominated by \( L' \) in \( D_{k'} \) and insert \( L' \) into \( D_{k'} \). Add \( \{ \alpha', L', k', P + k' \} \) to \( q \);
6. Return to step 3.

that the optimal departure time determined at time \( i\Delta \) is written as \( t^*_{i\Delta} \). Of course, a traveler cannot indefinitely keep updating their departure time; eventually, the driver will need to depart. Whenever we find an \( i \) such that
\[
t^*_{i\Delta} - i\Delta < \Delta,
\]
it means that the optimal departure time is less than \( \Delta \) time units removed from the request time. If this is the case, the driver should not wait another \( \Delta \) time units to request a new departure time, since its optimal departure time is earlier than the next request time. Therefore, for such an \( i \), we set \( t^*_{i\Delta} \) as the optimal departure time.

Note that it could happen that \( t^*_{i\Delta} < i\Delta \). That is, at request time \( i\Delta \), the traveler finds that they should have already departed in order to have an on-time arrival probability of at least \( \eta \). In this case, the departure advice is to leave immediately at the request time \( t = i\Delta \). Note that this scenario can only happen if at time \( (i-1)\Delta \) it was the case that \( t^*_{(i-1)\Delta} - (i-1)\Delta \geq \Delta \) as otherwise the driver should have already departed before time \( i\Delta \). Therefore, the departure advice of time \( i\Delta \) is not going to deviate more than \( \Delta \) from the actual optimal departure time as defined in (7).

This procedure is now summarized in Algorithm 4.
Algorithm 4: Optimal departure time online setting

Result: optimal departure time for traversing path \((a_1, \ldots, a_m)\) within time \(M\) with probability \(\eta\), given \(B(0) = s\).

1. Carry out Algorithm 2;
2. Set \(t_0\) as the output of Algorithm 2;
3. if \(t_0 = 0\) then
   | Return \(t^* = 0\);
else
   | Set \(j = 0\);
   | while \(t_{j\Delta} - j\Delta \geq \Delta\) do
     | 1. Set \(I_j = \{\max\{0, M - j\Delta - t_{\text{max}}\}, \max\{0, M - j\Delta - t_{\text{min}}\}\}\);
     | 2. Use the bisection method with initial interval \(I_j\) on the function \(\text{OnTimeProbability}(t', s, M - j\Delta)\), until obtain latest departure time for which output is at least \(\eta\). Set this departure time to \(t_{j\Delta}\);
3. if \(\text{OnTimeProbability}(\max\{j\Delta, M - t_{\text{max}}\}, s) \leq \eta\) then
   | Return \(t^* = j\Delta\);
else
   | 1. Set \(j = j + 1\);
   | 2. Obtain \(B(j\Delta)\) and set \(s = B(j\Delta)\);
Return \(t^* = t_{j\Delta}\)

4. Numerical Experiments

Now that we have derived the optimal departure time, we will perform a set of numerical experiments in order to discuss a selection of properties of the optimal departure time, and to demonstrate the efficiency our procedure. For these experiments, we consider a road network inspired by the highways of Amsterdam, see Figure 4. We examine a driver that is currently located at the red-colored vertex 7 and wishes to travel to the green-colored vertex 2 by either traversing the red or blue route. We start by considering a baseline setting, which is selected for illustration purposes only and does not necessarily reflect the true parameters of this network (where we remark that experiments on a larger network with realistic parameters are discussed in Section 4.5). It is given by the following parameters:

- there are currently no incidents, which means that \(B(0) = \{1\}^{12}\);
4.5
4.7
11.8
8.
4.
4.5
10.7
3.6
6.
5.2
9.3
3.8
1
2
3
4
5
6
7
8
9

Figure 4. The graph used in the numerical experiments. The edge values denote the length of the edge in kilometers. We examine a driver that wants to travel from vertex 7 to vertex 2.

- on each link, the velocity in case no incident occurred on that link is 100 km/h and is 40 km/h otherwise, or, \( v_{a_i}(1) = 100 \text{ km/h} \) and \( v_{a_i}(2) = 40 \text{ km/h} \) for each \( i = 1, \ldots, 12 \).

In the following experiments we may deviate from the baseline setting in order to magnify certain properties of the optimal departure time. In what follows, we note that whenever we refer to a travel-time distribution, we actually intend to refer to the approximated travel-time distribution as described in Section 3. For these approximations, travel-time values are aggregated into 100 bins after every iteration (i.e., after every step of (8)). Figure 5 illustrates that, for both routes in Figure 4, the approximated travel-time distribution, obtained by the method of discretization as described in Section 3, closely resembles the actual travel-time distribution, obtained with 100,000 simulation runs. Here, on each link, both the rate of incidents and the rate of clearance are one per hour, or, \( \alpha_i = \beta_i = 1 \text{ h}^{-1} \) for each \( i \). Similar performance results were obtained for other networks and paths under various parameter settings. For the experiments we implemented the networks and algorithms in Wolfram Mathematica 12.0 on an Intel® Core™ i7-8665U 1.90GHz computer.

4.1. Effect of considering the on-time arrival probability.

This experiment aims to show that, by having access to the travel-time distribution, a more suitable choice can be made in the route selection problem. Concretely, in contrast to only considering expected travel times, the risk-averseness of an individual driver can be incorporated, resulting in different departure times and routes. We consider the baseline setting outlined above, again with \( \alpha_i = \beta_i = 1 \text{ h}^{-1} \), but with the incident rate on each link of the red route
increased to two per hour. Since the red route is shorter compared to the blue route, while also being more prone to incidents, it is not immediately clear which route is optimal for the driver.

Suppose that a driver wants to arrive at vertex 7 before time $t = 0.5$. We first consider the problem in which the driver wants to arrive at $t = 0.5$ in expectation. We find that the departure times for both routes in this case are very comparable, namely $t = 0.229$ for the red route and $t = 0.230$ for the blue route; see the dashed lines in Figure 6. In other words, if a driver were concerned with the expected arrival time, they are likely to be indifferent between the two routes.

Instead of the requirement of arriving on-time in expectation, we will now consider the problem in which a driver wants to arrive on-time at vertex 7 with a certain probability. We can do this by utilizing the arrival time distribution for both routes. From Figure 6 we learn that the blue route allows for a later departure time, and thus should be preferred in case the desired on-time arrival probability $\eta$ is either between 0.4 and 0.6, or greater than 0.75. In particular, for e.g. $\eta = 0.9$, the departure time of the blue route is about 0.033 h, or about 2 min, later. For routes with an approximately equal expected travel time of only 16 min, the difference between their corresponding departure times is remarkable (i.e., it corresponds to as much as 10–15% of the travel time). Without having access to the travel-time distribution, which is provided by our modeling framework, this difference would have gone unnoticed.

4.2. The role of the background state upon departure.

We consider a driver who plans to traverse the red route in Figure 4, with the intend to depart as late as possible while guaranteeing an on-time arrival probability of $\eta = 0.9$ for arrival time.
The departure time as a function of the on-time arrival probability corresponding to the red and blue route in Figure 4, where the arrival time is chosen as \( t = 0.5 \). The dashed lines indicate the departure time for both routes in case of a desired arrival time at \( t = 0.5 \) in expectation.

At time 0, the moment the vehicle requests its optimal departure time, the background process is in a known initial state \( B(0) \). However, for any departure time \( t > 0 \), the background state \( B(t) \) is random, and may very well be different from \( B(0) \) due to the occurrence or clearance of incidents in the network during the time interval \([0, t] \). Therefore, when determining the optimal departure time \( t^* \), the possibility of the background state having changed at time \( t \) should be taken into account, which is done by using the distribution of \( B(t) \) conditional on the initial state \( B(0) \). The importance of doing so is illustrated in Figure 7.

(a) \( B(0) \) is a state without incidents.

(b) \( B(0) \) is a state with solely incidents.

Figure 7. Advised departure times for a vehicle that wants to travel the red route in Figure 4 with 90% on-time arrival probability. Departure times are obtained by assuming that at departure time \( t \) the background process is in state \( B(0) \) (blue), or by using the distribution of \( B(t) \) conditional on \( B(0) \) (yellow).
Figures 7a and 7b show the advised departure time of our procedure for different request times, and compare the results with a simplified procedure, in which it is assumed that upon departure at time $t > 0$ the background process is still in state $B(0)$. In the simplified procedure, the travel-time distribution is the same for every departure time $t$. Consequently, the optimal departure time under this procedure, which we denote by $\tilde{t}$, is independent of the request time. Observe that when the difference between the request time and $\tilde{t}$ grows, so does the difference between the outputs of the two procedures. Specifically, in both plots, this difference even exceeds four minutes, which is substantial considering that the travel time is in the interval of $[12.06, 30.15]$ minutes.

In Figure 7a, we see that whenever the request time is before 13:44 hrs, $\tilde{t} > t^\star$. Therefore, the on-time arrival probability for the simplified procedure will be below the desired 90%. Here, since $B(0)$ is a state without incidents, the simplified procedure assumes that the network is incident-free upon departure, whereas our procedure takes the possibility of changes in the background process into account. This explains why $\tilde{t} > t^\star$. The opposite phenomenon can be seen in Figure 7b. We thus see that the optimal departure time is greatly affected by the state of the background process.

4.3. Effect of request moment & initial state.

Besides showing the importance of incorporating the dynamics of $B(t)$, Figures 7a and 7b also display that $t^\star$ depends on the initial state upon the time of request. This effect is even more clear from Figure 8 which shows the optimal departure time $t^\star$ for traversing the red path (path 1) of Figure 4 under three different initial states. Note that, similar to the preceding experiment, the driver wishes to arrive before 14:00 hrs with 90% certainty. Moreover, we again let $\alpha_i = 0.5 \text{ h}^{-1}$ and $\beta_i = 2 \text{ h}^{-1}$.

First, we note that, being in steady state, a request time before 12:45 hrs yields similar departure times for the three initial states. This is no longer the case when the request moment is close to the arrival time. As expected, the latest departure time is obtained for the incident-free background state. Importantly, in case there is an incident at request time, the location of this incident has a significant impact on the departure time $t^\star$. That is, with the location of the incident at the end of the path (state 2), there is a high probability of clearance before arrival at the incident link, such that the corresponding departure time is still relatively close to the departure time of the incident-free state. However, in case the location of the incident is at the start of the path (state 3), there is only a small probability that this incident is cleared.
upon arrival at the incident-link. Consequently, the vehicle must depart considerably earlier in state 3 than in state 2.

**Figure 8.** Departure times for vehicle that wants to travel the red path (path 1) or the blue path (path 2) in Figure 4 and wants to arrive before 14:00h with 90% certainty. We consider three initial states: incident-free (state 1), only an incident at link (1, 2) (state 2) and only an incident at link (7, 9) (state 3).

In case the driver would only give the OD-pair, there are other paths from node 7 to node 2 that potentially outperform the red path, of which the the blue path of Figure 4 (path 2) is an example. Note that as the incidents in state 2 and 3 are not located on path 2, all three initial states considered in Figure 8 will result in the same departure time for traversal via path 2. As is shown in Figure 8, the vehicle would prefer path 2 over path 1 in steady state. However, this dominance no longer applies when the request time is close to the arrival time, as, in that case, path 2 is only preferred over path 1 if there is an incident located at the start of path 1.

From this experiment we learn that our modeling procedure effectively exploits knowledge of the locations of the currently present non-recurrent events in the network. It not only incorporates the presence of incidents on the routes, but also distinguishes between the locations of these incidents. For example, incidents on a link at the end of a path affect the departure time to a lesser extent, as our framework incorporates the high probability of incident clearance before arrival at this link. This is of course not the case for incidents closer to the origin.

4.4. **Online vs Offline Departure Time.**

As our modeling framework allows for the real-time implementation of the changing conditions in the road network, we now implement the online version of the optimal departure time problem
outlined in Algorithm 4 in which the traveler receives departure time updates while still at the origin. We again consider the baseline setting defined in the beginning of Section 4, with the rate of incidents \( \alpha_i = 0.1 \text{ h}^{-1} \) and the rate of clearances \( \beta_i = 2 \text{ h}^{-1} \). We will study a driver that wants to travel from vertex 7 to vertex 2 using the red route in Figure 4.

We compute the optimal departure of this driver using both the online and the offline setting, utilizing Algorithm 4 and Algorithm 2 respectively. We do this for a selection of requested arrival times: we let the length of the interval between request and desired arrival time be \( M = 10 \text{ h}, M = 1 \text{ h}, \) and \( M = 30 \text{ min} \). In addition, we consider a range of on-time arrival probabilities \( \eta \). As the online optimal departure time is random, we approximate its expectation by performing 10,000 repetitions and computing its mean. Lastly, we subtract the offline departure time from the approximated expected online departure time. The findings are shown in Figure 9.

![Figure 9. The difference between the expected online departure time, approximated using 10,000 repetitions, and the offline departure time. This is done for a selection of arrival times \( M \) and background states \( B(0) \).](image)

In the left graph of Figure 9 (that corresponds to \( M = 10 \text{ h} \)), we see that the online setting gives a later departure time compared to the offline setting in case the driver demands a high on-time arrival probability (\( \eta > 0.85 \)). This difference in departure time of 3 minutes is quite substantial, as the expected travel time of this route is roughly 15 minutes (i.e., 20% of the duration of the trip). This can be explained as follows. Since the requested arrival time is 10 hours from now, and since only the current state of the network is known in the offline setting, the actual state of the network on departure is highly uncertain. Therefore, if a driver demands a high on-time arrival probability in the offline setting, a very conservative belief about the state of the network on time of departure must be used. Either this belief is not far off, and the online and offline departure time will not much differ, or this belief was indeed too
pessimistic and the network is actually in a more favorable state upon departure such that the online setting will give a later optimal departure time. On average, we see that the departure time corresponding to the online setting will be later than that of the offline setting. Note that the converse is also true: for lower values of the on-time arrival probability, the online setting will find an earlier departure time compared to the offline setting.

With this in mind, it is not surprising that the difference between the online and offline optimal departure time is smaller when $M = 1$ h, and is even smaller in the $M = 30$ min case. As the time of request moves closer to the departure time, the offline setting is able to determine the optimal departure time based on a more recent state of the network. This way, the state of the network upon departure is less uncertain and the offline method will more closely resemble the online method.

Lastly, in Figure 9, we also consider the effect of the initial background state $B(0)$ on the difference between the optimal departure times for the online and offline setting. For $M = 10$ h, the initial background state plays no role, as the distribution of the background state upon departure, given any initial state, is effectively in the steady state. This is not the case for $M = 1$ h and $M = 30$ min, and we see that the difference between the online and offline procedure is greatest for the case in which there is an incident on the nearby link (7, 9) (the dotted line), and smallest in case there are no incidents (the dashed line).

As expected, more is to be gained from the online procedure when there is more uncertainty about the state of the network upon departure, which is in particular the case when the initial state contains incidents. Especially when these incidents are located near the origin, such as link (7, 9), the online method is able to more accurately incorporate the state of nearby links as they will be seen by the traveler upon departure, as opposed to links that are further away from the origin, such as link (1, 2).

4.5. Efficiency.

To empirically study the computational complexity of our procedures, we consider the complete Dutch highway network as depicted in Figure 10. This network consists of 659 nodes (i.e., highway ramps) and 1378 links (i.e., roads connecting ramps). The parameters are chosen to reflect the real-life setting and estimated via the fitting procedures presented in Levering et al. (2022a).

The computational performance of Algorithm 2 is given in Table 1. To this end, we consider the blue path in Figure 10 and compute the optimal departure time for subpaths of different lengths
from the origin. It can be observed that the run-time of the algorithm increases super-linearly in the number of links in the path.

For the performance of the two procedures proposed in Section 3.3.2 designed to output the optimal path and departure time in case a traveler gives the desired arrival time for an OD-pair, we consider two OD pairs that differ considerably in length (i.e., minimum number of links a vehicle needs to traverse to get from the origin to the destination). Concretely, OD-pair 1 has a length of 5 (Figure 10 purple path), whereas OD-pair 2 has a length of 32 (Figure 10 red path). For the experiments, we look at a traveler with an on-time arrival probability objective of 0.8, and set $M = 1$ h for OD-pair 1 and and $M = 2$ h for OD-pair 2. Now, as only incidents located in an area around (one of) the shortest paths between an OD-pair will significantly affect the departure time and route advice, we focus on the states $B(0)$ that encode incident instances on one of the three shortest paths (in km) between the origin and destination. With the probability of more than two incident instances being low, we further restrict our focus and only consider the set of states $B(0)$ that reflect one or two incident occurrences on the set of three shortest paths between an origin and destination, and use $B^*$ to denote this set. The setting in which the three shortest paths are completely incident-free is treated separately.

![The Dutch highway system, with blue, purple and red path highlighted.](image)

**Figure 10.** The Dutch highway system, with blue, purple and red path highlighted.

| Number of links | 1   | 5   | 10  | 15  | 20  |
|-----------------|-----|-----|-----|-----|-----|
| Seconds         | $<10^{-4}$ | 0.46 | 1.36 | 4.05 | 6.60 |

**Table 1.** Computational performance optimal departure time path (in sec.).
Recall that, contrary to the $k$-shortest path method, the procedure which employs bisection on the label correcting algorithm from Algorithm 3 is exact, and outputs the optimal departure time and corresponding path. Now, for both OD pairs, whenever $B(0)$ is such that there is no incident on one of the three shortest paths, the $k$-shortest path method already yields the optimal path and departure time for $k = 1$. Note that this setting will be frequently encountered: the stationary probabilities of such initial states are 0.994 and 0.985 for OD pairs 1 and 2, respectively. In case $B(0) \in B^*$, $k = 1$ is typically not sufficient for obtaining the optimal departure time. To assess the performance for different values of $k$, we compute, for every state $B(0) \in B^*$ the optimal departure time. A quantification of the performance is expressed in terms of the mean absolute percentage error (MAPE), defined as

$$\text{MAPE} := \frac{1}{|B^*|} \sum_{s \in B^*} \frac{|t^*_s - t^k_s|}{t^*_s},$$

with $t^*_s$ the optimal departure time and $t^k_s$ the departure time as outputted by the $k$-shortest path procedure, given $B(0) = s$.

The procedure based on Algorithm 3 is exact, and thus has zero MAPE. The procedure based on the $k$-shortest path algorithm is not exact, but, as can be observed from Table 2, already provides near-optimal results for $k = 3$. Moreover, the computational savings when using the $k$-shortest path method with moderate values of $k$ is significant, especially for OD-pair 2. Note that, in these experiments, we have only used one core, whereas the run-time of the $k$ shortest path algorithm can be further reduced when applying parallel computing, as the optimal departure times of the $k$ paths are computed individually. Importantly, in Appendix B, we argue that, in case of the $k$ shortest path method, even more computational efficiency can be realized by allowing additional precomputations.

| OD-pair | Run-time (sec.) | MAPE |
|---------|----------------|------|
| Algorithm 3 | 4.0 | 393.8 | 0% |
| $k$-shortest paths | | |
| $k = 1$ | 0.9 | 3.1 | 4.5% | 2.9% |
| $k = 3$ | 1.9 | 9.6 | 0.0% | 0.7% |
| $k = 5$ | 4.8 | 16.2 | 0.0% | 0.5% |
| $k = 10$ | 16.2 | 34.6 | 0.0% | 0.4% |
| $k = 15$ | 26.0 | 55.7 | 0.0% | 0.0% |

Table 2. Computational performance optimal departure time OD-pair (in sec.).
5. Conclusive remarks

In this paper we have developed a procedure for determining the optimal departure time in road networks with stochastic disruptions. We accomplished this by first developing an iterative procedure by which the travel-time distribution can be numerically evaluated for each departure time. Then, these distributions enabled us to develop efficient algorithms that identify the optimal departure time. We performed a selection of numerical experiments that exemplify various properties of the optimal departure time, and we demonstrated the efficiency of our procedure by applying it to an existing (large) road network. Lastly, we outlined an extension of our framework to traffic networks with more general dynamics, and we provided a speed-up technique for our procedure.

We defined the optimal departure time as the latest time of departure such that a selected on-time arrival probability is still guaranteed. By allowing the departure time to depend on the on-time arrival probability, the risk averseness of drivers can be taken into consideration. Next, in order to account for both recurrent and non-recurrent congestion, we used the Markovian Velocity Model that relies on a background process that tracks the events affecting the velocities in the traffic network. Doing so, our model also successfully exploits the knowledge of the locations of the currently present events in the road network. This allowed us to develop an online version of the optimal departure time problem, in which the traveler (while still at the origin) receives departure time updates, that incorporate the most recent state of the road network.

Our numerical experiments illustrated the extent to which the optimal departure time is affected by the state of the background process and the time of request. We also demonstrated that the route selection process is affected by using the latest departure time as an objective function. Moreover, we were able to quantify the substantial reduction in travel time budget that can be obtained by utilizing the online version of the problem. Lastly, we have demonstrated that our procedure can also be successfully employed in a real-world road network, as the run-time of our procedure, even in large road networks, remains manageable.

Several directions for follow-up research can be thought of. First, one could focus on empirically validating the approach presented in this paper. Secondly, one may develop an interface by which a driver can, explicitly or implicitly, reveal their risk averseness, after which the optimal departure time can be communicated to the driver. Thirdly, our framework could be extended to a setting that also allows for adaptive routing once the driver has already departed. Lastly,
different notions of optimality for the departure time, depending on the travel-time distribution, may be considered. An example could be a variant that, in case of late arrival, also takes into account by how much the desired arrival time has been exceeded.

ACKNOWLEDGEMENTS

The authors would like to thank dr. Marko Boon (TU Eindhoven) for his input on the implementation of the procedures, as well as the helpful feedback on the manuscript.

REFERENCES

Abate, J. and Whitt, W. (1992), ‘The Fourier-series method for inverting transforms of probability distributions’, Queueing systems 10(1), 5–87.

Butler, R. W. (2007), Saddlepoint Approximations with Applications, Cambridge University Press.

Chen, B. Y., Lam, W. H. K., Sumalee, A., Li, Q. and Tam, M. L. (2014), ‘Reliable Shortest Path Problems in Stochastic Time-dependent Networks’, Journal of Intelligent Transportation Systems 18(2), 177–189.

Christman, A. and Cassamano, J. (2013), Maximizing the probability of arriving on time, in ‘International Conference on Analytical and Stochastic Modeling Techniques and Applications’, Springer, pp. 142–157.

den Iseger, P. (2009), Laplace transform inversion on the entire line, Technical report, Available at SSRN 1355451.

Fan, Y., Kalaba, R. E. and Moore, J. E. (2005), ‘Arriving on time’, Journal of Optimization Theory and Applications 127(3), 497–513.

Frank, H. (1969), ‘Shortest paths in probabilistic graphs’, Operations Research 17(4), 583–599.

Güner, A. R., Murat, A. and Chinnam, R. B. (2012), ‘Dynamic routing under recurrent and non-recurrent congestion using real-time information’, Computers and Operations Research 39(2), 358–373.

Ibe, O. (2013), Markov processes for stochastic modeling, second edn, Elsevier.

Ji, X. (2005), ‘Models and algorithm for stochastic shortest path problem’, Applied Mathematics and Computation 170(1), 503–514.

Kharoufeh, J. P. and Gautam, N. (2004), ‘Deriving link travel-time distributions via stochastic speed processes’, Transportation Science 38(1), 97–106.
Kim, S., Lewis, M. E. and White III, C. C. (2005a), ‘Optimal vehicle routing with real-time traffic information’, *IEEE Transactions on Intelligent Transportation Systems* **6**(2), 178–188.

Kim, S., Lewis, M. E. and White III, C. C. (2005b), ‘State space reduction for non-stationary stochastic shortest path problems with real-time traffic information’, *IEEE Transactions on Intelligent Transportation Systems* **6**(3), 273–284.

Levering, N., Boon, M. and Mandjes, M. (2022a), ‘Estimating probability distributions of travel times by fitting a Markovian velocity model’. Pre-print, arXiv:2208.13704 [physics.soc-ph].

Levering, N., Boon, M., Mandjes, M. and Núñez Queija, R. (2022b), ‘A framework for efficient dynamic routing under stochastically varying conditions’, *Transportation Research Part B: Methodological* **160**, 97–124.

Loui, R. P. (1983), ‘Optimal paths in graphs with stochastic or multidimensional weights’, *Communications of the ACM* **26**(9), 670–676.

Mahmassani, H. S., Kim, J., Chen, Y., Stogios, Y., Brijmohan, A. and Vovsha, P. (2014), *Incorporating reliability performance measures into operations and planning modeling tools*, Transportation Research Board.

Nie, Y. M. and Wu, X. (2009), ‘Shortest path problem considering on-time arrival probability’, *Transportation Research Part B: Methodological* **43**(6), 597–613.

Norris, J. (1997), *Markov chains*, Cambridge University Press.

Psaraftis, H. N. and Tsitsiklis, J. N. (1993), ‘Dynamic shortest paths in acyclic networks with Markovian arc costs’, *Operations Research* **41**(1), 91–101.

Sen, S., Pillai, R., Joshi, S. and Rath, A. K. (2001), ‘A mean-variance model for route guidance in advanced traveler information systems’, *Transportation Science* **35**(1), 37–49.

Sever, D., Dellaert, N., van Woensel, T. and de Kok, T. (2013), ‘Dynamic shortest path problems: Hybrid routing policies considering network disruptions’, *Computers & Operations Research* **40**(12), 2852–2863.

Small, K. A. (1982), ‘The scheduling of consumer activities: work trips’, *The American Economic Review* **72**(3), 467–479.

Thomas, B. W. and White III, C. C. (2007), ‘The dynamic shortest path problem with anticipation’, *European Journal of Operations Research* **176**(2), 836–854.

Tilahun, N. Y. and Levinson, D. M. (2010), ‘A moment of time: reliability in route choice using stated preference’, *Journal of Intelligent Transportation Systems* **14**(3), 179–187.

Yang, Z. and Gao, Z. (2020), ‘Finding reliable paths considering the earliest arrival time and the latest departure time with 3-parameter lognormal travel times’, *IEEE Transactions on*
Appendix A. Recurrent congestion and non-exponential incident duration

For expositional reasons we have considered a compact version of the mvm. That is, the algorithms were designed under the setting that the only events affecting arc speeds are incidents and, moreover, that both the duration of these incidents and the time between these incidents are exponentially distributed. In Remark 1, we have claimed that we can handle a more comprehensive version of the mvm as well. To corroborate this claim, we will provide two examples that show how to adapt the algorithms of Section 3 in case a more detailed version of the mvm is used. In the first example, we consider a setting in which the per-link incident duration is not exponentially distributed, whereas in the second example, we discuss including recurrent events into the departure time framework.

Example 2. In the current model setting, the duration of an incident follows an exponential distribution, although generally, this is not always a realistic assumption. In Levering et al. (2022a), in which the mvm is used to describe travel times in the Dutch highway network, it was e.g. shown that there are indeed links for which the exponential distribution does not represent the incident duration well. It is moreover argued, that, for these links, the incident duration is well described by an Erlang-2, mixture-Erlang or hyperexponential distribution. Below we argue how to obtain the optimal departure time in case the incident duration is indeed modeled by one of these distributions.

In this example, we consider the setting in which the incident distribution on a link \( a \in A \) is a mixture of an Erlang-2 (with probability \( p \)) and Erlang-1 (with probability \( 1 - p \)) distribution, as depicted in Figure 11a. Now, given \( X_a(0) = j \), the link travel-time distribution of \( a \) can be found with a procedure in a similar spirit as Algorithm 1, using the discrete Markov chain in Figure 11b. Let \( L_a^j \) again denote the list of (travel time, probability)-pairs that arises. Note that, as there are now five potential non-symmetric transitions instead of two symmetric ones,
in determining the granularity, $\delta$ should satisfy five conditions of the form in (10), as well as five conditions of the form in (12).

In a similar fashion, we can determine approximate link travel-time distributions for any mixture Erlang, Erlang-2 or hyperexponential incident duration. Then, computation of the path travel time is straightforward, as we can use (8) to compute the travel time on a path, replacing $j = 1, 2$ by an iteration over the number of states in the state space of the link. A similar adaptation allows the use of Algorithm 2 for determining the optimal departure time of a path, and the use of both proposed methods for the optimal departure time of an OD-pair.

The only difficulty one encounters when working with these more general phase-type incident durations is the identification of $B(0)$. The complication is that, if there is an incident upon departure, the current state of the network is unknown as there are multiple states that correspond to the occurrence of an incident. However, knowledge of the elapsed incident duration allows the computation of the distribution vector of $B(0)$, which will then replace $p_s^1$ in (4).

Indeed, writing $Y$ for the duration of the current incident, $t$ for the (known) elapsed duration of the incident, and $F_1$ ($F_2$, respectively) for the event that the incident duration has an Erlang-1 (Erlang-2, respectively) distribution, we have, for $j = 2, 3$:

$$
\mathbb{P}(X_a(0) = j \mid Y \geq t) = \mathbb{P}(X_a(0) = j \mid Y \geq t, F_2) \mathbb{P}(F_2 \mid Y \geq t)
$$

\[
= \frac{p \mathbb{P}(X_a(0) = j \mid Y \geq t, F_2) \mathbb{P}(Y \geq t \mid F_2)}{(1 - p) \mathbb{P}(Y \geq t \mid F_1) + p \mathbb{P}(Y \geq t \mid F_2)}. 
\]
This fraction can be computed by realizing that for $E_1, E_2 \sim \text{Exponential}(\lambda_b)$ and $E_3 \sim \text{Exponential}(\lambda_c)$, we have
\[
\begin{align*}
P(Y \geq t \mid F_1) &= P(E_3 > t) = e^{-\lambda_c t} \\
P(Y \geq t \mid F_2) &= P(E_1 + E_2 > t) = (1 + \lambda_b t) e^{-\lambda_c t}
\end{align*}
\]
and
\[
\begin{align*}
P(X_a(0) = 2 \mid Y \geq t, F_2) &= P(E_1 > t) = e^{-\lambda_b t} \\
P(X_a(0) = 3 \mid Y \geq t, F_2) &= P(E_1 \leq t, E_1 + E_2 > t) = \int_0^t P(E_2 \geq t - s) \lambda_b e^{-\lambda_b s} ds = \lambda_b te^{-\lambda_b t}.
\end{align*}
\]
The probability that $X_a(0) = 4$ can be computed in a similar fashion.

\section*{Example 3.}

In our compact version of the MVM, we solely consider the impact of non-recurrent incidents on the vehicle speeds, and ignore e.g. daily traffic patterns. We are, however, able to capture these recurrent events as well. To this end, we propose a similar strategy as in Levering et al. (2022a). There it is shown that the incident duration, inter-incident time, and vehicle speeds, are dependent on the time-of-day, but that these time-dependencies can be tackled by working with periods of the day $\Theta_1, \ldots, \Theta_\ell$ over which these effects are essentially constant. For every such period, the per-link incident and inter-incident distribution are estimated, as well as the corresponding driveable speed levels. These can then be used when computing (and storing) the per-link travel-time distributions.

Representing recurrent events, the boundaries of the periods $\Theta_1, \ldots, \Theta_\ell$ are quite predictable. Therefore, the time between these boundaries can be modeled by Erlang distributions. Indeed, for given $k \in \mathbb{N}$ and $Z_i \sim \text{Exponential}(k/t)$, $Z_1, \ldots, Z_k$ independent, we have that $\sum_{i=1}^k Z_i$ is Erlang$(k, k/t)$ distributed, and
\[
\begin{align*}
E\left[\sum_{i=1}^k Z_i\right] &= t, \\
\text{Var}\left[\sum_{i=1}^k Z_i\right] &= t^2/k.
\end{align*}
\]
Thus, modeling the time between the boundaries of $\Theta_j$ with mean $t_j$ by an Erlang-$k_j$ distribution with mean $t_j$, we can achieve a low variance by choosing $k_j$ large enough. Then, to include the periods $\Theta_1, \ldots, \Theta_\ell$ in the MVM, we expand the background process $B(t)$ with a Markov process $Y(t)$, whose state space consists of the $k_1 + \cdots + k_\ell$ Erlang phases that model the times between their boundaries. The process $Y(t)$ visits these states cyclically, with $Y(t) = y$ encoding presence in period $\Theta_j$ at time $t$ if $y$ belongs to one of the $k_j$ Erlang phases modeling the boundaries of $\Theta_j$. Now, working with the extended $B(t)$, we set the velocity of a vehicle traversing $a_i \in A$.
in the following way: the vehicle speed at time \( t \) equals \( v_a(s_i, y) \) if \( X_a(t) = s_i \in \{1, 2\} \) and \( Y(t) = y \). This way, the speed on a link depends both on the time-of-day (via \( Y(t) \)) and the presence of an incident (via \( X_a(t) \)).

As the velocity dynamics on \( a_i \) are fully described by the Markov process \( (X_a(t), Y(t)) \), we can apply a discretization procedure in a similar fashion as in Section 3.1 to obtain the link travel-time distribution on \( a_i \). Then, the list \( L_{a_i}^{N} \) represents the travel time distribution on \( a_i \) given that \( (X_a(t), Y(t)) \) is initially in state \((j, y)\). We do, however, need to pay special attention to the computation of the travel-time distribution on paths, as the process \( Y(t) \) affects the velocities on all arcs. Therefore, instead of (travel time, probability)-pairs, we let the lists \( L_{a_i}^{N} \) contain (travel time, probability, state)-tuples, in which the state is the state of all velocities on arcs. Therefore, instead of (travel time, probability)-pairs, we let the lists \( L_{a_i}^{N} \) contain (travel time, probability, state)-tuples, in which the state is the state of all velocities on arcs.

The path-extension procedure in (14) can now be used to compute the optimal departure time with one of the algorithms presented in Section 3.3. Note that, to use these algorithms, we need knowledge on the state \( Y(0) \). Similar to the previous phase-type example, this state can not directly be observed, but a probability distribution over the possible states of \( Y(t) \) can be computed. That is, given the request time, we do know the current period \( \Theta_i \) and the elapsed time \( t > 0 \) between the start of this period and the request time. Denoting with \( y_1, \ldots, y_k \) the subsequent Erlang states that model the duration of \( \Theta_i \), and with \( \lambda \) their transition rate, we have

\[
\mathbb{P}(Y(0) = y_i \mid Y(-t) = y_1, Y(0) \in \{y_1, \ldots, y_k\}) = \frac{\mathbb{P}(Y(0) = y_i \mid Y(-t) = y_1)}{\mathbb{P}(Y(0) \in \{y_1, \ldots, y_k\} \mid Y(-t) = y_1)} = \frac{\mathbb{P}(S_{i-1} < t, S_i \geq t)}{\mathbb{P}(S_k \geq t)},
\]

with \( S_j = \text{Erlang}(j, \lambda) \). Conditioning on the value of \( S_{i-1} \), it is now easily derived that, for \( E_i \sim \text{Exponential}(\lambda) \),

\[
\frac{\mathbb{P}(S_{i-1} < t, S_i \geq t)}{\mathbb{P}(S_k \geq t)} = \frac{\int_0^t \mathbb{P}(E_i > t-s)f_{S_{i-1}}(s)ds}{\mathbb{P}(S_k \geq t)} = \frac{e^{-\lambda t} \int_0^t (\lambda s)^i-2/(i-2)! ds}{\mathbb{P}(S_k \geq t)} = \frac{(\lambda t)^{i-1}}{\lambda(i-1)! \sum_{n=0}^{k-1} \frac{1}{n!}(\lambda t)^n}.
\]

\( \diamond \)
We will discuss a simple but effective way to speed-up the computation of the optimal departure time for certain paths or OD pairs. We have limited the computational costs by storing per-link travel-time distributions. We could, however, expand the work that is carried out in the precomputations and store the travel-time distribution for some sets of subsequent links as well. Concretely, for a given path $P'$, we are able to compute the list $L^j_{P'}$ of (travel time, probability)-pairs that arises for each initial state $j$ the background process of $P'$ can attain. Then, when computing the optimal departure time on a path $P$ that has $P'$ as subpath, we can simply view $P'$ as one link and use the stored travel time distributions for $P'$ in the algorithms. Notably, this will speed up the computation of the optimal departure time in both Algorithm 2 and the $k$-shortest path method, whenever (one of) the path(s) contains subpath(s) for which the travel-time distribution is stored. In contrast, as the A*-algorithm works with individual links rather than with paths, the speed-up technique is not applicable to Algorithm 3.

Remark 4. Availability of the precomputed travel-time distribution of a subpath $P'$ reduces the number of iterations necessary to compute the travel-time distribution on a complete path $P$. Note that this does not directly yield a speed-up, as the number of operations within the iteration adding $P'$ is potentially large. That is, with $P'$ consisting of $k$ individual links, this iterative step works – instead of with two – with $2^k$ (travel time, probability) lists, corresponding to the number of states in the background process of $P'$. However, as the operations concerning different background states can be carried out in parallel, the impact of these additional operations is greatly reduced.

There is an additional advantage when precomputing the travel-time distribution for a set of paths. That is, with only a negligible loss in accuracy, we are able to substantially reduce the size of the state space and, consequently, further speed up the computations. Concretely, if one wants to find the optimal departure time for a path $P$ containing a subpath $P'$ for which the distribution is stored, and if there are states in the state space of $P'$ that have an extremely small probability of occurring, then the distributions that correspond to these states can simply be neglected. This means that, in the computation of the travel-time distribution on the path $P$, we will omit lists $L^j_{P'}$ of (travel time, probability)-pairs on $P'$ for which

$$\mathbb{P}(\exists t \in [0, M] : (X_{a'}(t))_{a' \in P'} = j \mid B(0) = s) < \varepsilon,$$
for some small $\varepsilon > 0$. Examples of procedures to generate upper bounds for such hitting probabilities are provided in Levering et al. (2022b).