Modeling and control of congestion phenomena

Levering, N.A.C.

Publication date
2024

Citation for published version (APA):
Chapter 3

Optimal departure-time advice

3.1 Introduction

There are various settings in which travelers may wish to arrive at a location before a certain time. Examples include the airport, in case they have booked a flight that departs at a known time, and the location of a scheduled job interview. Notably, these travelers will generally want to get an indication of the time it takes to reach the destination beforehand, so as to know at what time to depart. Moreover, in determining their departure time, they will typically be risk-averse, and will not simply account for the expected time it takes to reach the destination, as the inherent randomness of travel times could still lead to a high probability of late arrival at the destination. At the same time, these travelers will also be reluctant to depart overly early, as this would leave them with less time available for other activities. Thus, there is a need for algorithms that generate the optimal departure time: the latest departure time such that a certain user-specified on-time arrival probability can be guaranteed.

In order to determine the on-time arrival probability, it is crucial to have access to the travel-time distribution. This distribution depends on the state of the network, covering both recurrent patterns and non-recurrent events. As the state of the network is constantly evolving, the travel-time distribution will also change over time. In an optimal departure-time advice, one could, e.g., try to exploit that currently present incidents further away from the origin are more likely to be cleared once the corresponding road part is reached by the traveler.

Whereas recurrent patterns are predictable from historic data, the time and location of future non-recurrent events are intrinsically uncertain. As the state of the network (locations of currently present incidents, their elapsed durations, etc.) can be observed, it can be used to identify the current travel-time distribution and hence the on-time arrival probability when one would leave now. However, only knowing the current travel-time distribution does not suffice, as one needs to be able to evaluate the on-time arrival probability for any future departure time as well, which requires access to the travel-time distribution when leaving at any given future time point. Importantly, as the network state evolves over time, these future travel-time distributions depend on the current state of the network.

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.

Relevant 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 [72], 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 [118], 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 [167], whereas Fan et al. [66] and Christman and Cassamano [39] outline a dynamic routing procedure to maximize the on-time arrival probability.

The \( \alpha \)-shortest path criterion succeeds in favoring routes with a low mean travel time, while also penalizing travel-time uncertainty. Using such an objective function, one can strike a proper balance between minimizing the travel time and controlling its randomness [209]. 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. As the \( \alpha \)-shortest path criterion uses the full travel-time distribution (rather than just the travel-time expectation and variance), the uncertainty of a route is captured in more detail than in a moment-based approach (such as the one used in [191]).

An important observation is that the literature on optimal reliable routing, including the above-mentioned works, typically assumes instant departure of the traveler. However, early arrival evidently incurs an (opportunity) cost, so that immediate departure is generally not optimal [195]. 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 Chen et al. [34], 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 [223], assuming log-normally distributed travel times.

As mentioned, travel times cannot be assumed constant; various random effects play a role, as extensively discussed in Mahmassani et al. [199]. To deal with the recurrent patterns, one could work with the framework of [34] and [223], in which the travel-time distribution’s parameters are assumed constant over predefined time intervals. Clearly, non-recurrent effects should be dealt with differently, requiring estimates of incident durations and inter-incident times, as well as knowledge of the current state of the road network.

Various methods have been proposed to incorporate the current state of the network into the travel-time estimation problem. Traffic-theory based methods, such as the one proposed in Celikoglu [30], use recent velocity measurements to infer the current state of the network. These models predict future traffic flow using flow conservation equations and traffic dynamics, but the likelihood of future non-recurrent disruptions is not taken into account. More recently, artificial neural networks have been employed for traffic-flow
The spatio-temporal graph model in Guo et al. [93] is a data-based method that also involves a recent time-series of the traffic flow. However, as the estimation procedure only yields a point estimate of the traffic flow (rather than a distributional estimate), this approach cannot be used in our optimal departure-time problem.

Contributions

In contrast to the referenced works on optimal departure-time advice, this chapter presents a framework that simultaneously addresses the three requirements (i), (ii), and (iii) identified above. Specifically, we determine the optimal departure time in a setting that covers both recurrent and non-recurrent effects, and in which the velocities the vehicle can drive at depend on the current state of the network. The departure time depending on the selected value of the on-time arrival probability, the risk averseness is naturally taken into account.

In our setup, the network state dynamics determine the per-link travel-time distribution, from which its per-path counterpart can be computed. This per-path travel-time distribution enables the evaluation of the on-time arrival probability for any future departure, so that we can determine (for any desired on-time arrival probability) the optimal departure time. By optimizing over paths between an origin and destination, this procedure can be further enhanced to yield the optimal departure time in case only an OD-pair is specified. In numerical experiments, we show that the optimal departure time is greatly affected by both the state of the road network and the time of request. We also examine the efficiency and scalability of our procedure, demonstrating that our procedure can successfully be employed in a real-world road network.

In our approach we chose to describe the velocity dynamics by the Markovian velocity model (MVM) that was introduced in the previous chapter. Where Chen et al. [34] and Yang and Gao [223] 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 [223], the MVM is highly flexible, and covers a broad range of incident duration and inter-incident time distributions. Another advantage of this approach is that the road network by construction satisfies the desirable FIFO property, entailing that we do not have to rely on a stochastic-FIFO assumption as in [34].

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 problem, in which the traveler receives departure time updates while still at the origin. Through a selection of numerical experiments, 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.

Organization

This chapter is organized as follows. Section 3.2 formally describes the optimal departure-time objective. In Section 3.3, we demonstrate a procedure for the computation of travel-
time distributions. Section 3.4 outlines computationally efficient algorithms that use these distributions to output the optimal departure time for a given path or OD-pair, target arrival time, and on-time arrival probability. Numerical examples, to exemplify a selection of properties of the optimal departure time and to study the efficiency of the procedures, are presented in Section 3.5. Finally, Section 3.6 contains conclusive remarks. Various details of our procedures are discussed in Appendices 3.A–3.D.

3.2 Preliminaries

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 recurrent and non-recurrent events that affect the driveable vehicle speeds. The impact of these events on the driveable speeds is modeled through a Markovian background process. At the time the traveler requests advice (which we identify as $t = 0$ throughout this chapter), the state of the background process is known. If, given knowledge of the background process at $t = 0$, one would have access to the travel-time distribution for any departure time (after $t = 0$, that is), one could find the 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 3.2.1 we shortly recap the Markovian velocity model (MVM) of Chapter 2, employed to model the effect of daily patterns and stochastic disruptions on the vehicle speeds. With the notation introduced there, Section 3.2.2 formally introduces the problem of determining the optimal departure time.

3.2.1 Travel-time dynamics: Markovian velocity model

Let $G = (N, A)$ be the (directed) graph which represents the considered road network, with $n \equiv |A|$. From now on, if there is a direct road between the junctions labeled by $k$ and $\ell$, we write $(k, \ell)$ for the corresponding arc (i.e., different from the notation $k\ell$ as used in Chapter 2). Moreover, we alternatively write $A = \{a_1, \ldots, a_n\}$ for the set of arcs in $G$, with $a_i \equiv (k, \ell)$ for some $k, \ell \in N$, and $d_{a_i}$ the length of arc $a_i \in A$. The MVM uses a Markovian background process $B(t)$ to record the evolution of recurrent and non-recurrent events in the considered road network, thereby providing a direct relationship between traffic events and travel times.

In the MVM, $B(t)$ has the form $(Y(t), X_{a_1}(t), \ldots, X_{a_n}(t))$. Here, $X_{a_i}(t)$ is allowed to be any finite-state continuous-time Markov process describing the evolution of incidents on link $a_i$. The ‘common’ Markov process $Y(t)$ captures the recurrent patterns or scheduled events affecting the arc speeds; it may, e.g., contain states that describe the remaining time until the next rush hour. Conditional on the state of the common process $Y(t)$, the processes $X_{a_i}(t)$ and $X_{a_j}(t)$ evolve independently (where $i \neq j$). The process $B(t)$ is therefore described by the transition matrix $Q$ that results from the transition matrix $Q_Y$ of the common process $Y(t)$ in combination with the transition matrices $Q_{Y,a_i}$ of the processes $(Y(t), X_{a_i}(t))$. 
The state space of $B(t)$ is denoted by $I$. To refer to (elements of) subspaces of $I$, we introduce the following notation. For a state $s \in I$ and a subset $C$ of processes in the background process, we let $s(C)$ denote the entries of the state $s$ encoded by the processes in $C$, and $I(C)$ denote the corresponding state space. For example, if $C = \{Y(t), X_{a_2}(t)\}$ and $s = (4, 3, 6, 1, \ldots, 1)$, this means $Y(t) = 4$ and $X_{a_2}(t) = 6$, such that $s(C) = (4, 6)$. For ease of notation, with, e.g., $C$ as above, we may write $s(Y, X_{a_2})$ instead of $s(C)$ or $s(\{Y(t), X_{a_2}(t)\})$ (i.e., omitting the set and process notation).

Now, recall that in the MVM, dependence between the velocities on the arcs is allowed. First, it is recognized that there exists global dependence, induced by daily traffic patterns, which are modeled through $Y(t)$. Second, as traffic jams propagate through space and time, velocities on the different arcs are locally dependent. Specifically, as defined above Assumption 2.3.1, denote with $A_{a_i}$ the set of arcs whose congestion level affects the velocity on $a_i \in A$. Then, introducing the notation

$$C_{a_i} \equiv \{Y(t)\} \cup \{X_a(t) \mid a \in A_{a_i}\},$$

dependencies between the velocities on the arcs are realized by setting the speeds at which vehicles are moving on arc $a_i \in A$ equal to $v_{a_i}(s)$ if $B(t) = s$, where, for two states $s, \tilde{s} \in I$, $v_{a_i}(s) = v_{a_i}(\tilde{s})$ if $s(C_{a_i}) = \tilde{s}(C_{a_i})$.

### 3.2.2 Objective: departure time advice

In this subsection we outline the problem of determining the optimal departure time. In doing so, we distinguish between 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 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 for which a certain on-time arrival probability can be guaranteed. If the requested arrival time is given by $t = M > 0$, i.e., $M$ time units after 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

$$t^* \equiv \sup \{t \geq 0 : P(t + T_t \leq M \mid B(0)) \geq \eta\}. \quad (3.1)$$

In (3.1), the random variable $T_t$ represents the travel time of a vehicle departing at time $t$. Importantly, $T_t$ is affected by the current background state $B(0)$, and the distribution of $T_t$ will change once future information about the state of the network becomes available (i.e., through $B(u)$, when $u$ is approaching $t$). In this offline setting, however, we assume that only the current state of the network, $B(0)$, is known. Therefore, the on-time arrival probability at the 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 (3.1). In this case, we put $t^* \equiv -\infty$, and it depends on the preferences of the driver to either depart immediately or to not depart at all.
We now pay closer attention to the conditional probability in (3.1). First, recognize that the travel-time distribution depends on the departure time \( t > 0 \) 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 transition matrix \( Q \) and the known current state \( B(0) = s \). Indeed, using general results for continuous-time Markov chains [109, 169], it follows that this distribution is given by

\[
 p_s^t \equiv (\mathbb{P}(B(t) = s' | B(0) = s))_{s' \in \mathcal{I}} = p_0^s e^{Qt},
\]

with \( p_0^s \), by definition, a row vector of dimension \( |\mathcal{I}| \) with a 1 at the entry that corresponds to the state \( s \in \mathcal{I} \) and zeroes elsewhere. Thus, with \( T[s'] \) denoting the travel time corresponding to departing when the background process is in the state \( s' \), conditioning on the background state at time \( t \) yields

\[
 \mathbb{P}(t + T_t \leq M | B(0) = s) = p_s^t (\mathbb{P}(T[s'] \leq M - t))_{s' \in \mathcal{I}}.
\]

Hence, if we are able to determine the distribution of the travel time \( T[s'] \) for each state \( s' \in \mathcal{I} \), then this would allow us to compute the conditional probability in (3.3) for each \( t \), which in turn would facilitate determining the maximizer \( t^* \) in (3.1). The evaluation of the travel-time distribution is outlined in detail in Section 3.3.

**Online Setting** — In the online setting, the objective is to produce an optimal departure time \( t^* \) that uses that, while the traveler waits for their departure, new information on the state of the background process becomes available. More specifically, for \( u \leq t^* \), the distribution of \( B(t^*) \) given \( B(u) = s \) becomes

\[
 (\mathbb{P}(B(t^*) = s' | B(u) = s))_{s' \in \mathcal{I}} = (\mathbb{P}(B(t^* - u) = s' | B(0) = s))_{s' \in \mathcal{I}} = p_0^s e^{Q(t^* - u)}.
\]

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 progresses, the traveler can request a new departure time that is given by

\[
 t_u^* \equiv \sup \{ t \geq u : \mathbb{P}(t + T_t \leq M | B(u)) \geq \eta \}.
\]

Just as in the offline setting, \( T_t \) should be interpreted as the travel time when departing at time \( t \), but now with the current time being the request time \( u \). Therefore, at time \( u, t_u^* \) is the latest departure time such that the on-time arrival probability is at least \( \eta \). Note that 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 the 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 the time of request will approach the on-time arrival probability at the 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.

### 3.3 Travel-time distribution

Consider a vehicle that departs at \( t = 0 \) to traverse a given path in the network \( G = (N, A) \). In the MVM, 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 3.3.1 below), we use a discretization procedure to obtain an accurate approximation of the travel-time distribution. Specifically, we work with a (sufficiently fine) discretization of the moments at which there is a potential transition in the driveable speed (corresponding to a transition of the background process).

**Remark 3.3.1.** 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 (see Theorem 2.3.5 in Chapter 2), 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., the methods of Abate and Whitt [1] and den Iseger [52], or saddlepoint approximation [27]) is not successful. This is due to the fact that the per-link travel-time distribution is neither discrete nor continuous. To see this, consider link $a$ which takes state $B(0) = s$ upon departure. Now, there is a positive probability that the state of $B(t)$ 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, the per-link travel time has a continuous density on the remainder of the domain.

An easy fix for the challenges discussed in Remark 3.3.1 would be to assume that the driveable vehicle speed on a link is fixed upon entering a link, in that it 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 inspires to work out 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-path travel-time distribution, thereby first demonstrating the method for a simple model setting (Section 3.3.1), after which we point out how to extend this methodology for a general MVM (Section 3.3.2).

### 3.3.1 Demonstration

To optimize our exposition, we first discuss a specific basic MVM, which we refer to as the *compact* MVM. In his model, for $a_i \in A$, the Markov process $X_{a_i}(t)$ is simply such that $X_{a_i}(t) = 1$ if there is no incident on arc $a_i$ at time $t$, and 2 otherwise. There is no impact from recurrent or predicted events, such that $Y(t)$ may be omitted. Moreover, it is assumed that the congestion level at arc $a_i$ only impacts the speed on link $a_i$ itself. That is, the vehicle speed at time $t$ equals $v_1 \equiv v_{a_i}(1)$ if $X_{a_i}(t) = 1$ and $v_2 \equiv v_{a_i}(2)$ otherwise. Consequently, for $i = 1, \ldots, n$, the process $X_{a_i}(t)$ is completely described by its initial state $X_{a_i}(0)$ and its transition rate matrix

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

(3.6)
with $\alpha_i, \beta_i \in \mathbb{R}_{>0}$.

To determine the travel-time distributions in the compact MVM, as a first step, we focus on the travel-time distribution for the traversal of a single link $a \in A$. Note that, since we are solely given the state of $X_a(t)$ at $t = 0$, only the driveable speed upon departure is known. 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 $X'_a(t)$ (with $t = 0, \delta, 2\delta, \ldots$) evolve according to the diagram of Figure 3.1b. Thus, if the process $X'_a(t)$ 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 $p \equiv e^{-\alpha \delta}$, i.e., the probability that the continuous-time Markov chain $X_a(t)$ 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 $X_a(t)$ 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 $X_a(t)$, 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 Appendix 3.A. In a similar fashion, if $X'_a(t) = 2$, the process stays in state 2 with probability $q \equiv e^{-\beta \delta}$, and moves to state 1 with probability $1-q$. Note that we have described the dynamics of $X'_a(t)$, 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.

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 link $a$, recall that $d_a$ is the total distance of the link. Therefore, if at $t = m\delta$ a tuple $(s_0, d_0, p_0)$ with traveled distance $d_0 < d_a$ generates a
3.3. Travel-time distribution

Figure 3.2: Tree-structure corresponding to Example 3.3.2.

<table>
<thead>
<tr>
<th>Algorithm 3.1: Travel-time distribution of a single link $a$ (compact MVM).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Result:</strong> list $L$ of (travel time, probability) values.</td>
</tr>
<tr>
<td><strong>Notation:</strong> for transparency, omitted subscripts for $v_a(s)$, $X_a(s)$ and $d_a$;</td>
</tr>
<tr>
<td><strong>Input:</strong> $\delta \in \mathbb{R}_{&gt;0}$, $X(0)=s$, $p = e^{-\alpha \delta}$, $q = e^{-\beta \delta}$;</td>
</tr>
<tr>
<td><strong>Initialization:</strong> $L = \emptyset$, $S = {(s, \delta v(s), 1)}$, $S' = \emptyset$, $m = 0$;</td>
</tr>
<tr>
<td><strong>while</strong> $S$ non-empty <strong>do</strong></td>
</tr>
<tr>
<td>1. <strong>foreach</strong> $(s', d', p')$ in $S$ <strong>do</strong></td>
</tr>
<tr>
<td>a. Compute $p_1 = p'(1{s'=1} p + 1{s'=2}(1-q))$ and</td>
</tr>
<tr>
<td>$p_2 = p'(1{s'=1}(1-p) + 1{s'=2} q)$;</td>
</tr>
<tr>
<td>b. <strong>for</strong> $i = 1, 2$: if $d' + v(i) \delta &lt; d$ <strong>then</strong> append $(i, d' + v(i) \delta, p_i)$ to $S'$. <strong>else</strong></td>
</tr>
<tr>
<td>append $(m \delta + (d-d')/v(i), p_i)$ to $L$;</td>
</tr>
<tr>
<td>2. Set $S = S'$, $S' = \emptyset$ and $m = m + 1$;</td>
</tr>
</tbody>
</table>

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 $m \delta + (d_a - d_0)/v_a(s_1)$. 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 $a$ given $X_a(0) = s$ as $L_a[s]$. Observe that the tuples $(s, d, p)$ for which $d \geq d_a$ do not serve as an ancestor in a new iteration, implying that the iterative procedure, which is summarized in Algorithm 3.1, terminates in a finite number of steps.

**Example 3.3.2.** Figure 3.2 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, $v_a(2) = 60$ 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 3.3.3.** We observe that, since we are only working with two speed values per time step, the iterative procedure can, in the compact MVM, also be represented by a binomial tree. To be able to generate such a tree and compute the resulting travel-time
distribution, we look, contrary to Algorithm 3.1, at tuples of the form \((d, p_1, p_2)\), with \(d\) the traveled distance, and \(p_1\) (resp. \(p_2\)) the probability of the scenario in which the vehicle had speed \(v_1\) (resp. \(v_2\)) 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 3.3 shows the binomial tree corresponding to Figure 3.2.

Importantly, a binomial tree grows maximally one branch per time step, making this procedure particularly efficient. Indeed, after \(m\) time steps, the number of ancestors in a binomial tree is only of order \(m\), whereas the number of ancestors in Algorithm 3.1 would be of order \(2^m\).

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 \(P = \{a_1, \ldots, a_m\}\). 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 \(L_{a_1}[s_1]\), 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 \(e^{t_1 Q_{a_{i+1}}}\), with \(Q_{a_{i+1}}\) the transition rate matrix of \(a_{i+1}\) as defined in (3.6). Therefore, the travel-time distribution \(T[s]\) for traversing \(\{a_1, \ldots, a_{i+1}\}\) with initial state \(s\) is given by the list

\[
\{(t_1 + t_2, p_1 \cdot p_2 \cdot [e^{t_1 Q_{a_{i+1}}}]_{(s_{i+1}, j)}) \mid (t_1, p_1) \in L, (t_2, p_2) \in L_{a_{i+1}}[j], \quad j = 1, 2\}.
\]

(3.7)

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. Specifically, with a minimum (resp. maximum) travel-time value of \(\tau_{\text{min}}\) (resp. \(\tau_{\text{max}}\)), \(n_b\) the number of bins, and \(\hat{\tau} \equiv (\tau_{\text{max}} - \tau_{\text{min}})/n_b\), we map, for \(m = 0, 1, \ldots, n_b - 1\), all \((t, p)\) pairs with \(t - \tau_{\text{min}} \in [m\hat{\tau}, (m+1)\hat{\tau})\) to \((\tau_{\text{min}} + m\hat{\tau}, p)\), and then join all pairs with the same first element to a single tuple by summing their corresponding probabilities.
3.3.2 General methodology

Algorithm 3.1 is designed under the compact mvm setting. In case the MVM is more comprehensive, it is evident that we can use a similar discretization technique. This claim is corroborated by Examples 3.3.4 and 3.3.5 below, that both describe a generic example structure for the background process of the full MVM.

Example 3.3.4. In the compact mvm, the impact of daily traffic patterns is ignored. However, in road networks, the incident duration, inter-incident time, and vehicle speeds may be dependent on the time of day and the day of the week. One way to tackle such time-dependencies is by dividing the days of the week in periods $\Theta_1, \ldots, \Theta_\ell$ over which these effects are essentially constant.

Representing recurrent events, the boundaries of the periods $\Theta_1, \ldots, \Theta_\ell$ are relatively deterministic. This means that the time between boundaries of subsequent periods can be modeled with Erlang distributions. Indeed, for given $k \in \mathbb{N}$ and $Z_i \sim \text{Exponential}(k/\tau)$, $Z_1, \ldots, Z_k$ independent, we have that $\sum_{i=1}^{k} Z_i$ is Erlang$(k, k/\tau)$ distributed, and

$$\mathbb{E}\left[\sum_{i=1}^{k} Z_i\right] = \tau, \quad \text{Var}\left[\sum_{i=1}^{k} Z_i\right] = \tau^2/k.$$  

Thus, if the time between the boundaries of $\Theta_j$ is $\tau_j$, we can model it by an Erlang$(k_j, k_j/\tau_j)$ distribution; it has mean $\tau_j$ as desired, and we can achieve a low variance by choosing $k_j$ large enough. Thus, to model $\Theta_1, \ldots, \Theta_\ell$, the background process of the compact MVM is extended with the common process $Y(t)$ whose state space consists of the $k_1 + \cdots + k_\ell$ Erlang phases; $k_1, \ldots, k_\ell$ are chosen large enough to make the times spent in the intervals $\Theta_1, \ldots, \Theta_\ell$ sufficiently deterministic. The process $Y(t)$ visits these states cyclically, modeling the daily and weekly traffic patterns, with $Y(t) = y$ encoding being in $\Theta_j$ at time $t$ if $y$ belongs to one of the $k_j$ Erlang phases corresponding to $\Theta_j$. Then, the vehicle speed on $a_i \in A$ at time $t$ equals $v_{a_i}(s)$ if $B(t) = s$, with $v_{a_i}(s) = v_{a_i}(\tilde{s})$ for $s, \tilde{s} \in \mathcal{I}$ if $s(Y, X_{a_i}) = \tilde{s}(Y, X_{a_i})$. Thus, the speed on a link depends both on the time of day (via $Y(t)$) and the presence of an incident on the link itself (via $X_{a_i}(t)$).

As the velocity dynamics on link $a_i$ are fully described by the Markov process $(Y(t), X_{a_i}(t))$, the corresponding travel-time distribution can be found by applying a discretization procedure similar to the one described in the previous subsection. We do, however, need to take into account that $Y(t)$ affects the velocities on all arcs. Therefore, to be able to compute travel-time distributions on paths, we store the (travel time, probability)-pairs in lists $L_{a_i}[(y, s_i), (\tilde{y})]$. Here, $(y, s_i)$ denotes the state of $(Y(t), X_{a_i}(t))$ at $t = 0$ (i.e., the time the link is entered), and $\tilde{y}$ denotes the state of $Y(t)$ in the final iteration step.

Example 3.3.5. In the compact model, the duration of an incident follows an exponential distribution. In this example, we let $B(t) = (X_{a_1}(t), \ldots, X_{a_n}(t))$ (i.e., we only deal with non-recurrent events), and we consider the extension of the compact MVM for which the incident distribution on each link is a mixture and/or convolution of exponential distributions (i.e., a phase-type distribution, as already referred to in Remark 2.3.3). Moreover, we allow local dependence between the velocities on the arcs, such that, for each $a_i \in A$, the set $C_{a_i}$ may consist of the background processes of neighboring links.

In case the incident distribution on link $a_i \in A$ is a mixture of an Erlang-2 (with probability $p$) and Erlang-1 (with probability $1-p$) distribution, as depicted in Figure 3.4a,
we can use the discrete-time Markov chain in Figure 3.4b to approximate the travel-time distribution on link \( a_i \) with a similar procedure as Algorithm 3.1 (again, details regarding the choice of \( \delta \) are discussed in Appendix 3.A). In an analogous fashion, we can determine approximate link travel-time distributions for any mixture and/or convolution of exponential distributions. However, to account for the between-arc dependencies when translating the per-link distributions into per-path distributions, we store the (travel time, probability)-pairs in lists \( L_{a_i}[s(C_{a_i}), \tilde{s}(C_{a_i})] \), where \( s(C_{a_i}) \) and \( \tilde{s}(C_{a_i}) \) denote the state of \( \{X_a\}_{X_a \in C_{a_i}} \) at the start of the first and at the end of the final iteration step, respectively.

In general, for the full MVM, the travel-time distribution on a link \( a_i \in A \) is stored as lists of (travel time, probability)-pairs \( L_{a_i}[s(C_{a_i}), \tilde{s}(C_{a_i})] \), where \( s(C_{a_i}) \) and \( \tilde{s}(C_{a_i}) \) denote the restricted state of \( B(t) \) at the start of the first and the end of the final iteration step, respectively. Now, for the computation of the travel-time distribution when departing in state \( B(0) \) on a path \( P = \{a_1, \ldots, a_m\} \), denote

\[
\hat{C}_{a_i} \equiv \left( \bigcup_{1 \leq j \leq i} C_{a_j} \right) \cap \left( \bigcup_{i \leq j \leq m} C_{a_j} \right).
\]

If \( \hat{C}_{a_i} \neq C_{a_i} \), there is a process \( X_{a_i}(t) \) that affects the speeds on an arc traveled before and an arc traveled after arc \( a_i \), that does not affect the speeds on arc \( a_i \) itself. Thus, to be able to use an iterative methodology as presented for the compact MVM, the travel-time distribution on \( a_i \) should track the state of \( X_{a_i}(t) \) as well. Therefore, as a first step, for each \( a_i \in P \) with \( \hat{C}_{a_i} \neq C_{a_i} \), we create lists \( L_{a_i}[s(C_{a_i}), \tilde{s}(C_{a_i})] \) that yield the travel-time distribution on \( a_i \) whenever the restricted states of \( B(\cdot) \) are \( s(C_{a_i}) \) and \( \tilde{s}(C_{a_i}) \) at the first and last iteration step, respectively. Given that only the set \( \hat{C}_{a_i} \subseteq C_{a_i} \) impacts the velocities on \( a_i \), the travel-time values in \( L_{a_i}[s(C_{a_i}), \tilde{s}(C_{a_i})] \) are similar to those in \( L_{a_i}[s(C_{a_i}), \tilde{s}(C_{a_i})] \).

However, the transition probabilities differ, as we should also account for the dynamics of \( X_{a_j} \in \hat{C}_{a_i} \setminus C_{a_i} \). Since these are, conditional on the common process \( Y(\cdot) \), independent of
the processes in $C_{a_i}$, $L_{a_i}[s(\hat{C}_{a_i}), \tilde{s}(\hat{C}_{a_i})]$ equals
\[
\left\{ \left( t, p \cdot \prod_{a_j: X_{a_j} \in \hat{C}_{a_i} \setminus C_{a_i}} \tilde{p}_{a_j}(s, \tilde{s}, t) \right) \left| (t, p) \in L_{a_i}[s(C_{a_i}), \tilde{s}(C_{a_i})] \right. \right\}, \tag{3.8}
\]
with
\[
\tilde{p}_{a_j}(s, \tilde{s}, t) \equiv \mathbb{P} \left( Y(t), X_{a_j}(t) = \tilde{s}(Y, X_{a_j}) \right| (Y(0), X_{a_j}(0)) = s(Y, X_{a_j}), Y(t) = \tilde{s}(Y) \right) = \frac{[e^{tQ_{Y,a_j}}](s(Y, X_{a_j}), \tilde{s}(Y, X_{a_j}))}{[e^{tQ_{Y}}](s(Y), \tilde{s}(Y))},
\]
where the numerator is the $(i_1, i_2)$-th element of the matrix exponential $\exp\{tQ_{Y,a_j}\}$, $i_1$ (resp. $i_2$) being the index of $s(Y, X_{a_j})$ (resp. $\tilde{s}(Y, X_{a_j})$) in $I(Y, X_{a_j})$. The denominator is defined in an analogous way.

Now, for $i = 2, \ldots, m$, let $L_{i-1}[s(\hat{C}_{a_{i-1}})]$ be a list of (travel time, probability)-pairs on \{a_1, \ldots, a_{i-1}\} for which the restricted state $s(\hat{C}_{a_{i-1}})$ is attained upon arrival of the vehicle at the endpoint of $a_{i-1}$. Then, $L_i[s(\hat{C}_{a_i})]$, the travel-time distribution on the subpath \{a_1, \ldots, a_i\} for which the restricted state $s(\hat{C}_{a_i})$ is attained upon finishing the traversal of this subpath, is simply given by the (unordered) list
\[
\left\{ (t_1 + t_2, p_1 \cdot p_2 \cdot \prod_{a_j: X_{a_j} \in \hat{C}_{a_i} \setminus C_{a_{i-1}}} \tilde{p}_{a_j}(B(0), s, t_1)) \left| (t_1, p_1) \in L_{i-1}[s(\hat{C}_{a_{i-1}})] \wedge (t_2, p_2) \in L_{a_i}[s(\hat{C}_{a_i}), \tilde{s}(\hat{C}_{a_i})] \right. \right\}, \tag{3.9}
\]
That is, each travel time is the sum of travel times on \{a_1, \ldots, a_{i-1}\} and \{a_i\} from lists in which the final restricted state of \{a_1, \ldots, a_{i-1}\} is coherent with the initial restricted state of \{a_i\}. The corresponding probability is found by multiplying the respective probabilities of attaining these two travel times. If $\hat{C}_{a_i}$ contains processes that are not part of $\hat{C}_{a_{i-1}}$, this probability should again be multiplied by the probability of finding these processes in the restricted state $s(\hat{C}_{a_i} \setminus \hat{C}_{a_{i-1}})$ upon entering $a_i$.

We may iteratively compute the travel-time distributions on the subpaths, and retrieve the travel-time distribution $T[B(0)]$ on \{a_1, \ldots, a_m\} as the union of $L_m[s(\hat{C}_{a_m})]$ over the potential states $s(\hat{C}_{a_m})$. Again, to prevent that the number of elements in the travel-time distribution list grows exponentially with the number of links in the path, we aggregate the travel-time values into equally sized bins after every iteration.

### 3.4 Computing the optimal departure time

Recall that we consider the setting in which a traveler wants to know their optimal departure time for the traversal of an OD-pair in a network $G = (N, A)$, in which vehicle speeds are described by the mvm. So far, we have outlined 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 $s \in \mathcal{I}$. 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, the use of an elementary bisection algorithm enables us to 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 procedure to the case in which the user only specifies the OD-pair, rather than the specific path to travel, by comparing the departure times of different candidate paths and outputting the latest departure time (and corresponding path). It is noted that, while we have consistently used the MVM to describe the underlying velocity dynamics in this chapter, this is by no means crucial, in that in principle any alternative travel-time model whose link travel times satisfy the FIFO property could be relied upon.

In this section, we show how to compute the optimal departure time for a given path (Section 3.4.1), or, if only the destination of the trip is known, how to obtain both the optimal departure time and the corresponding path to travel (Section 3.4.2). Whereas we initially consider the case in which the traveler only requests a departure time once, i.e., the offline setting, we extend this procedure to the case in which the traveler receives departure-time advice updates, i.e., the online setting (Section 3.4.3).

Note that in the computations it is assumed that the state \( B(0) \) is known. In the context of the full MVM there is the complication that, in case of, e.g., an incident, one knows the elapsed incident duration, but not the state \( B(0) \) that the underlying phase-type distribution is in. In Appendix 3.B, we discuss how knowledge of the elapsed incident duration allows the computation of the distribution vector of \( B(0) \), which will then replace \( p_0^* \) in (3.2).

3.4.1 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 3.2.2, namely, computing the product in (3.3). To this end, we need to evaluate \( \mathbb{P}(T[s'] \leq M-t) \) for all \( s' \in \mathcal{I} \). 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 \( \mathbb{P}(T[s'] \leq M-t) \).

However, computing the arrival probability via (3.3) is typically time consuming, as it requires us to compute the distribution of \( T[s] \) for all \( s \in \mathcal{I} \). 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 \( P \) is equivalent to departing at time 0 and, before entering \( P \), first traversing a fictional link for which the travel time equals \( t \) with probability 1. Therefore, we directly obtain the arrival-time distribution by computing the travel-time distribution of this partly fictional path (with departure at time 0). Then, the on-time arrival probability follows by summing all probabilities for which the corresponding travel time does not exceed \( M \). Specifically, for a path \( P = \{a_1, \ldots, a_m\} \), executing \( m \) iterations of (3.9) yields, for each \( s(\hat{C}_{a_m}) \in \mathcal{I}(\hat{C}_{a_m}) \), the travel-time distribution estimate \( \mathcal{L}_m[s(\hat{C}_{a_m})] \) for traversing \( P \) and observing state \( s(\hat{C}_{a_m}) \) upon
arrival at the endpoint, such that the corresponding on-time arrival probability equals

\[ \sum_{s(\hat{C}_{am}) \in \mathcal{I}(\hat{C}_{am})} \sum_{(t,p) \in \mathcal{L}_m[s(\hat{C}_{am})]: t \leq M} P. \]

Now, since we are able to compute the on-time arrival probability on the path \( P \) for a given departure time \( t \) and, moreover, since 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 2.3.4, which gives that for a path consisting of a single link, \( t' \leq t \) implies

\[ t' + T_v \leq t + T_t. \]

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

\[ I_0 \equiv [\max\{0, M - t_{\text{max}}\}, \max\{0, M - t_{\text{min}}\}] \]

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 3.2: Optimal departure time offline setting.

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

Initialization: let \( t_{\text{min}} = \sum_{i=1}^{m} d_{ai} \cdot \min_{s \in \mathcal{I}} v_{ai}(s)^{-1}, \)
\( t_{\text{max}} = \sum_{i=1}^{m} d_{ai} \cdot \max_{s \in \mathcal{I}} v_{ai}(s)^{-1}, \) and \( I_0 = [\max\{0, M - t_{\text{max}}\}, \max\{0, M - t_{\text{min}}\}] \);

1. if \( I_0 = [0, 0] \), quit and return \( t^* = -\infty \), else continue;
2. for \( i = 1, \ldots, m \) do
   - if \( \hat{C}_{ai} \neq C_{ai} \) then compute \( L_{ai}[s(\hat{C}_{ai}), s(\hat{C}_{ai})] \) via (3.8);
3. Define the function: \( \text{OnTimeProbability} \ (t, s, M) \)
   a. set \( C_0 = C_0 = \emptyset \);
   b. for each \( s(Y) \in \mathcal{I}(Y) \), set \( \mathcal{L}_0[s(Y)] = \{(t, [e^{tQ_Y}]_{Y(0), s(Y)})\} \);
   c. for \( i = 1, \ldots, m \) do
      - for each \( s(\hat{C}_{ai}) \in \mathcal{I}(\hat{C}_{ai}) \) let \( \mathcal{L}_i[s(\hat{C}_{ai})] \) equal (3.9);
   return \( \sum_{s(\hat{C}_{am}) \in \mathcal{I}(\hat{C}_{am})} \sum_{(t,p) \in \mathcal{L}_m[s(\hat{C}_{am})]: t \leq M} P \);
4. if \( \text{OnTimeProbability}(\max\{0, M - t_{\text{max}}\}, s, M) \leq \eta \) then
   return \( t^* = -\infty \);
else
   - Apply bisection on \( \text{OnTimeProbability}(\cdot, s, M) \) with initial interval \( I_0 \), until the departure time input yields output \( \eta \). return this departure time \( t^* \).
Algorithm 3.2 now summarizes the complete procedure for obtaining the optimal departure time \( t^\star \) 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 \( t^\star \) once. Importantly, we can precompute the travel-time distribution for every link \( a \) in the network, in terms of the lists \( L_a[s(C_a), \hat{s}(C_a)] \). 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.

### 3.4.2 OD-pair — Offline setting

We can now extend the results to the natural setting in which a user (residing at a given origin) does not specify the complete path, but only the destination \( k^\star \). 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 may not be applicable in practical settings. Therefore, we present two alternative methods: an exact procedure that is relatively computationally demanding, and a very efficient, near-optimal method:

- **Bisection method**: similar to Algorithm 3.2, 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 3.4), together with a detailed description, can be found in Appendix 3.C. 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**: simply compute the optimal departure time for the subset of \( k \) shortest paths (e.g., in distance) to the destination. Then, with \( \mathcal{P}' \) the set of \( k \)-shortest paths as found via Yen’s algorithm [224, 225], Algorithm 3.2 can be used to compute the optimal departure times for all \( P \in \mathcal{P}' \), and to output the path with the latest departure time. 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 be employed highly efficiently.

### 3.4.3 Online setting

In the online setting, new information on the state of the network, becoming available during the time the traveler is waiting for departure, is used to update the optimal departure time. In this subsection, we 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...
3.5 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
Figure 3.5: The graph used in the numerical experiments. The edge values denote the lengths of the edges in kilometers. We examine a driver that wants to travel from vertex 7 to vertex 2.

Figure 3.6: The simulated (solid line) and approximated (dashed line) travel-time distribution of the red and blue route in Figure 3.5.

to demonstrate the efficiency our procedure. For these experiments, we consider a road network inspired by the highway network around Amsterdam (Figure 2.9), schematically depicted in Figure 3.5.

We examine a driver currently located at the red-colored vertex 7, wishing 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 3.5.4). We specify this setting as follows:

- velocity dynamics are modeled by the compact MVM (as described in Section 3.3.1), with, on each link, the incident-free velocity 100 km/h and the velocity during incidents 40 km/h, or, $v_{ai}(1) = 100$ km/h and $v_{ai}(2) = 40$ km/h for each $i = 1, \ldots, 12$;
- there are currently no incidents, which means that $B(0) = \{1\}^{12}$.

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 3.4. For these approximations, travel-time values are aggregated into 100 bins after every iteration (i.e., after every step of (3.7)
3.5. Numerical experiments

Figure 3.7: The departure time as a function of the on-time arrival probability corresponding to the red and blue route in Figure 3.5, the target arrival time being 0.5. The dashed lines indicate the departure time for both routes in case of a desired arrival time of 0.5 in expectation.

or (3.9)). Figure 3.6 illustrates that, for both routes in Figure 3.5, the approximated travel-time distribution closely resembles the actual travel-time distribution, obtained with $10^5$ simulation runs. Note that the non-smoothness of the simulated distribution reflects the fact that the true travel-time distribution is non-smooth; cf. Remark 3.3.1. In Figure 3.6, 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 background processes and parameter settings.

3.5.1 Effect of considering reliable paths

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 than 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 $t = 0.5$. If the driver wants to arrive at $t = 0.5$ in expectation, the departure times for both routes nearly coincide (Figure 3.7), such that the driver is indifferent between the two routes. However, the driver will no longer be indifferent in the case they want to arrive on-time with a certain probability. Specifically, Figure 3.7 shows that the blue route allows for a later departure time if the desired on-time arrival probability $\eta$ is either between 0.4 and 0.6, or larger than 0.75. For example, for $\eta = 0.9$ 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).

3.5.2 The role of the request moment & initial state

We consider a driver who plans to traverse the red route in Figure 3.5, with the intent to depart as late as possible while guaranteeing an on-time arrival probability of $\eta = 0.9$ for arrival before 14:00 hrs. We again consider the baseline setting, and set $\alpha_i = 0.5 \text{ h}^{-1}$ and
\( \beta_i = 2 \text{ h}^{-1} \) for all \( i \).

Figures 3.8a and 3.8b show the advised departure time, as produced by our methodology, for different request times, and compare the results with a simplified procedure. This simplified procedure assumes that, upon departure at some time \( t > 0 \), the background process is still in state \( B(0) \); hence, the associated optimal departure, denoted by \( \tilde{t} \), is independent of the request time. Observe that the difference between the outputs of the two procedures may even exceed four minutes, which is substantial considering that the travel time is in the interval of \([12.06, 30.15]\) minutes. We conclude that the simplified procedure leads to unreliable departure advice.

In Figure 3.8a, we see that \( \tilde{t} > t^* \) whenever the request time is before 13:44 hrs. 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 \( \tilde{t} > t^* \). The opposite phenomenon can be seen in Figure 3.8b.

Besides showing the importance of incorporating the dynamics of \( B(t) \), Figure 3.8 also displays that \( t^* \) depends on the initial state upon the time of request. Indeed, our modeling procedure exploits knowledge 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; see Figure 3.9 for the optimal departure time \( t^* \) corresponding to traversing the red path (path 1) of Figure 3.5 under three different initial states.

First note that Figure 3.9 shows 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 closer to the target arrival time. As expected, the latest departure time corresponds to the incident-free background state (state 1). Second, observe that in case there is an incident at the request time, the location of this incident has a significant impact on the departure time \( t^* \). 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
3.5. Numerical experiments

Figure 3.9: Departure times for a driver that wants to travel the red path (path 1) or the blue path (path 2) in Figure 3.5, and wants to arrive before 14:00 hrs 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).

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.

We proceed by comparing multiple paths from node 7 to node 2. Specifically, we verify whether the blue path of Figure 3.5 (path 2) outperforms the red path (path 1). Note that as the incidents in state 2 and 3 are not located on path 2, all three initial states considered in Figure 3.9 will result in the same departure time for traversal via path 2. As is shown in Figure 3.9, 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.

Besides incident locations, our modeling procedure may also exploit knowledge of (i) the time of day and (ii) the characteristics of the present incidents. For example, if we would capture the daily patterns by using the \textit{mvm} as presented in Example 3.3.4, and consider a fixed request time, the length of the interval between the advised optimal departure time and target arrival time will depend on the time of day. This phenomenon is illustrated by Table 3.1, which uses request time 05:45 hrs, and shows that the time that is reserved for driving the red route of Figure 3.5 is higher for arrival times within rush hours. In this experiment, the (inter-)incident rates are chosen to reflect the real-life setting; we will detail how this parameterization is obtained in Chapter 4. The speed on a link is set 20 km/h if there is an incident on that link, and otherwise either 80 km/h (evening rush), 70 km/h (morning rush) or 100 km/h (free-flow speed). If we would, alternatively, use the \textit{mvm} as presented in Example 3.3.5, then different incident distributions (which may encode different incident types) indeed result in different optimal departure times. This behavior is seen in Table 3.2, which considers traversing the red route, with, upon request, an incident located on the last link, for different distributions of the incident length. Again, the other (inter-)incident rates are chosen to reflect the real-life setting (as detailed in Chapter 4), now with the speed on a link 20 km/h if there is an incident on that link, and otherwise 40 km/h (incident on neighboring link), 60 km/h (incident on neighbor of neighboring link), or 100 km/h (free-flow speed). Unsurprisingly, non-severe
Table 3.1: Reserved driving times for the red route of Figure 3.5, with full \( \text{mvm} \) as in Example 3.3.4, an incident-free request time 05:45 hrs, and \( \eta = 0.9 \).

<table>
<thead>
<tr>
<th>Desired arrival time (hrs)</th>
<th>06:15</th>
<th>07:30</th>
<th>17:45</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M - t^* ) (min.)</td>
<td>12.1</td>
<td>17.3</td>
<td>15.1</td>
</tr>
</tbody>
</table>

Table 3.2: Reserved driving times (\( M - t^* \), in min.) for the red route of Figure 3.5 with \( \eta = 0.9 \), the full \( \text{mvm} \) of Example 3.3.5, and different incident distributions for the last link (rate parameters in hours), given that, upon request, there is an incident with an elapsed time of 20 minutes on this link.

<table>
<thead>
<tr>
<th>Distribution incident duration</th>
<th>( M ) (min.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
<td>Mean (min.)</td>
</tr>
<tr>
<td>Exponential(4)</td>
<td>15.0</td>
</tr>
<tr>
<td>Erlang(2,8)</td>
<td>15.0</td>
</tr>
<tr>
<td>Erlang(2,1/2)</td>
<td>240.0</td>
</tr>
</tbody>
</table>

incidents (which correspond to a low mean) only impact the driver if its desired arrival time is close to the request time. Moreover, a higher volatility leads to more unpredictability, and therefore, typically, earlier optimal departure times.

### 3.5.3 Online vs offline departure time

Our modeling framework being capable of incorporating changing conditions in the underlying road network, we have implemented the online version of the optimal departure-time problem outlined in Algorithm 3.3, 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 3.5, with the rate of incidents \( \alpha_i = 0.1 \, \text{h}^{-1} \) and the rate of clearances \( \beta_i = 2 \, \text{h}^{-1} \).

We study a driver that wants to travel from vertex 7 to vertex 2 using the red route in Figure 3.5.

We compute the optimal departure time of this driver using both the online and the offline setting, by employing Algorithm 3.3 and Algorithm 3.2, respectively. We do this for a selection of requested arrival times \( M \) and 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 3.10.

In the left graph of Figure 3.10 (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 (\( \eta > 0.87 \)). 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 upon 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 the time of departure must be used. Either this belief is not far off, and the online and offline departure time will hardly 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. Thus, 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, on average, find an earlier departure time compared to the offline setting.

From the above observation, it is not surprising that the difference between the online and offline optimal departure time is smaller for $M = 30$ min than for $M = 1$ h. 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 3.10, 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 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 largest 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, the potential gain of the online procedure is larger in instances with 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).

### 3.5.4 Efficiency

To empirically study the scalability of our procedures, we consider the complete Dutch highway network as depicted in Figure 3.11. This network consists of 659 nodes (i.e., highway junctions, such as ramps and intersections) and 1378 links (i.e., roads connecting these junctions). We work with the full MVM, whose background process and parameters are chosen to reflect the real-life setting, and are estimated via the fitting procedures that
The computational performance of Algorithm 3.2 is given in Table 3.3. To this end, we consider the blue path in Figure 3.11, 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 3.4.2, 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., the minimum number of links a vehicle needs to traverse to get from the origin to the destination). Specifically, OD-pair 1 has a length of 5 km (Figure 3.11, purple path), whereas OD-pair 2 has a length of 32 km (Figure 3.11, red path). For the experiments, we consider a traveler with an on-time arrival probability objective of 0.8, and set $M = 1$ h for OD-pair 1 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 a destination, and use $B^*$ to denote this set. The setting in which the three shortest paths are completely incident-free is treated separately.

Recall that, contrary to the $k$-shortest path method, the procedure which employs bisection on a label correcting algorithm (Algorithm 3.4) 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
3.6 Concluding remarks

<table>
<thead>
<tr>
<th>OD-pair</th>
<th>Run-time (sec.)</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Bisection method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k = 1$</td>
<td>0.9</td>
<td>3.1</td>
</tr>
<tr>
<td>$k = 3$</td>
<td>1.9</td>
<td>9.6</td>
</tr>
<tr>
<td>$k = 5$</td>
<td>4.8</td>
<td>16.2</td>
</tr>
<tr>
<td>$k = 10$</td>
<td>16.2</td>
<td>34.6</td>
</tr>
<tr>
<td>$k = 15$</td>
<td>26.0</td>
<td>55.7</td>
</tr>
</tbody>
</table>

$k$-shortest path method

Table 3.4: Performance of the procedures from Section 3.4.2 for two OD-pairs.

yields the optimal path and departure time for $k = 1$. Note that this setting will be encountered frequently: 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. A quantification of the performance of the $k$-shortest path method for different $k$ is expressed in terms of the mean absolute percentage error (MAPE), defined as

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

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 bisection method 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 3.4, already provides near-optimal results from $k = 3$ on. Moreover, the computational savings when using the $k$-shortest path method with moderate values of $k$ are 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 independently. Importantly, in Appendix 3.D, we argue that, in case of the $k$-shortest path method, the computational efficiency can be even further improved by using additional precomputations.

3.6 Concluding remarks

The objective of this chapter was to develop a procedure for determining the optimal departure time in road networks with stochastic disruptions and recurrent patterns. To this end, we first developed an iterative procedure to numerically evaluate the travel-time distribution for any departure time. Then, these distributions enabled us to develop efficient algorithms that identify the optimal departure time, i.e., the latest departure time that keeps the on-time arrival probability above a given target value. We performed a selection of numerical experiments that exemplify various properties of the optimal departure time, and we demonstrated the efficiency and scalability of our approach by applying it to an existing (large) road network.

The optimal departure time being parameterized by the selected on-time arrival probability, the driver’s risk averseness has been naturally incorporated. 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 also 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 state of the background process and the time of request have impact on the optimal departure time. Moreover, we were able to quantify the (potentially 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 consider a large-scale empirical validation of the approach presented in this chapter. 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 optimal departure time 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.

3.A Granularity

In Section 3.3, we have outlined a discretization procedure that allows us to approximate the per-link travel-time distribution. In this procedure, the choice of \( \delta \) affects the number of steps in the iteration, 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 here.

First, consider the compact MVM. Then, for a given link \( a \in A \) we used the approximation

\[
\mathbb{P}(X_a(\delta) = 1 \mid X_a(0) = 1) \approx p \equiv e^{-\alpha \delta}.
\]

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 \( X_a(t) \) 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 3.3 are justifiable, we require the probability of two or more transitions in the time interval \( \delta \) to be negligible. With \( f_\gamma(\cdot) \) the density of an exponentially distributed random variable with parameter \( \gamma \), the probability of two or more transitions conditional
on $X_a(0) = 1$ is given by
\[
\int_0^\delta f_a(t) \int_0^{t - \delta} f_\beta(s) \, ds \, dt = \int_0^\delta \alpha e^{-\alpha t} (1 - e^{-\beta(\delta - t)}) \, dt = \frac{1 - e^{-\alpha \delta} - \alpha}{\alpha - \beta} \left( e^{-\beta \delta} - e^{-\alpha \delta} \right) = 1 - \frac{\alpha e^{-\beta \delta} - \beta e^{-\alpha \delta}}{\alpha - \beta}.
\]

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
\[
1 - \frac{\alpha e^{-\beta \delta} - \beta e^{-\alpha \delta}}{\alpha - \beta} < \varepsilon_1.
\]

As $\delta$ should be chosen small, we can simply evaluate (3.10) working with its second-order Taylor approximation in $\delta$ at 0. Indeed, as $\delta \downarrow 0$,
\[
1 - \frac{\alpha e^{-\beta \delta} - \beta e^{-\alpha \delta}}{\alpha - \beta} = \frac{1}{2} \alpha^2 \beta^2 \delta^2 - \frac{1}{2} \alpha \beta^2 \delta^2 + o(\delta^2) = \frac{1}{2} \alpha \beta \delta^2 + o(\delta^2).
\]

We thus find that (3.10) reduces to
\[
\delta < \left( \frac{2\varepsilon_1}{\alpha \beta} \right)^{1/2}.
\]

From (3.11), 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 $t \in (0, \delta)$, 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 $\varepsilon_2 > 0$, the expected error is bounded:

\[
(1 - e^{-\alpha \delta}) \left| \int_0^\delta \frac{\alpha e^{-\alpha t}}{1 - e^{-\alpha \delta}} (v_1 t + v_2 (\delta - t)) \, dt - v_1 \delta \right| < d \cdot \varepsilon_2.
\]

This can be rewritten to
\[
|v_1 - v_2| \left( \frac{1}{\alpha} (1 - e^{-\alpha \delta}) - \delta \right) < d \cdot \varepsilon_2.
\]

A similar condition can be obtained for the case $X_a(0) = 2$, but with the $\alpha$ replaced by $\beta$ in (3.12). 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 (3.12) 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
\[
\delta < \left( \frac{2\varepsilon_2}{|v_1 - v_2| \alpha} \right)^{1/2} \quad \text{and} \quad \delta < \left( \frac{2\varepsilon_2}{|v_1 - v_2| \beta} \right)^{1/2}.
\]
Combining conditions (3.11) and (3.13), we conclude that both the probability of two or more transitions and the expected error are sufficiently small, whenever we pick

$$\delta < \min \left\{ \left( \frac{2\varepsilon_1}{\alpha \beta} \right)^{1/2}, \left( \frac{2\varepsilon_2}{|v_1-v_2| \max\{\alpha, \beta\}} \right)^{1/2} \right\}. \quad (3.14)$$

For the full MVM, similar considerations will result in a suitable choice for $\delta$. For example, observe that in Figure 3.4b, there are five non-symmetric transitions instead of two symmetric ones. Therefore, in determining the granularity, $\delta$ should now satisfy five conditions of the form (3.11), as well as five conditions of the form (3.13).

### 3.3B Initial state

Throughout this chapter it is assumed that $B(0)$, the state of the background process upon request, is known. Due to the presence of Intelligent Transportation Systems, among which traffic cameras with incident detection software, we do know which links of a highway network contain an incident. Therefore, when working with the compact MVM, the corresponding background state is clear. However, this may not be the case for the full MVM. If there would, for example, be a detected incident on a link $a$ whose incident duration has an Erlang-2 distribution, then, as there are two states in $X_a(t)$ that encode this incident, it is not clear in which of the two states the process is at the request time, and thus, $B(0)$ is unknown.

Below, we discuss how knowledge of the past (i.e., the time before the request moment) allows the computation of the distribution vector of $B(0)$, which will then replace $p_{s_0}$ in (3.2). Specifically, we will outline how such a vector can be computed for the setting of Examples 3.3.4 and 3.3.5. Similar computations may be executed for other MVM structures.

Note that, in Example 3.3.4, the state of $Y(t)$ cannot 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 $\Theta_l$ 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 $\Theta_l$, and with $\lambda$ their transition rate, we have, with $\Upsilon = \{y_1, \ldots, y_k\}$,

$$p_{y_1, y_i} \equiv \mathbb{P}(Y(0) = y_1 \mid Y(-t) = y_i, Y(0) \in \Upsilon) = \frac{\mathbb{P}(Y(0) = y_i \mid Y(-t) = y_1) \mathbb{P}(Y(0) \in \Upsilon \mid Y(-t) = y_1)}{\mathbb{P}(S_{i-1} < t, S_i \geq t)},$$

with $S_j = \sum_{i=1}^j E_i$ for $E_1, E_2, \ldots$, i.i.d. random variables with $E_1 \sim \text{Exponential}(\lambda)$. Conditioning on the value of $S_{i-1}$, and realizing that $S_j$, for $j \geq 1$, has an Erlang$(j, \lambda)$ distribution, it is now easily derived that,

$$p_{y_1, y_1} = \frac{\mathbb{P}(S_1 \geq t)}{\mathbb{P}(S_k \geq t)} = \frac{1}{\sum_{n=0}^{k-1} \frac{1}{n!}(\lambda t)^n},$$
and, for \( i \geq 2 \),
\[
p_{y_{i-1}, y_i} = \frac{\int_0^{t} \mathbb{P}(E_i > t - s) f_{S_{i-1}}(s) ds}{\mathbb{P}(S_k > t)} \quad = \quad \frac{e^{-\lambda t} \int_0^{t} \frac{\lambda^{i-1}s^{i-2}}{(i-2)!} ds}{\mathbb{P}(S_k > t)} \quad = \quad \frac{(\lambda t)^{i-1}}{(i-1)!} \sum_{n=0}^{k-1} \frac{1}{n!} (\lambda t)^n.
\]

In the setting of Example 3.3.5, 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 of \( B(0) \). If, for example, \( X_a(t) \) would be structured as Figure 3.4a, writing \( V \) for the duration of the current incident, \( t \) for the (known) elapsed duration of the incident, and \( F_1 \) (resp. \( F_2 \)) for the event that the incident duration has an Erlang-1 (resp. Erlang-2) distribution, we have, for \( j = 2, 3 \):
\[
\mathbb{P}(X_a(0) = j \mid V \geq t) = \mathbb{P}(X_a(0) = j \mid V \geq t, F_2) \frac{\mathbb{P}(V \geq t \mid F_2)}{\mathbb{P}(F_2) \mathbb{P}(V \geq t \mid F_2)}
\]
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
\[
\mathbb{P}(V \geq t \mid F_1) = \mathbb{P}(E_3 > t) = e^{-\lambda_c t},
\]
\[
\mathbb{P}(V \geq t \mid F_2) = \mathbb{P}(E_1 + E_2 > t) = (1 + \lambda_b t)e^{-\lambda_b t},
\]
and
\[
\mathbb{P}(X_a(0) = 2 \mid V \geq t, F_2) = \mathbb{P}(E_1 > t) = e^{-\lambda_b t},
\]
\[
\mathbb{P}(X_a(0) = 3 \mid V \geq t, F_2) = \mathbb{P}(E_1 < t, E_1 + E_2 > t)
\]
\[
= \int_0^t \mathbb{P}(E_2 \geq t - s) \lambda_b e^{-\lambda_b s} ds = \lambda_b t e^{-\lambda_b t}.
\]
The probability that \( X_a(0) = 4 \) can be computed in a similar fashion.

### 3.C Label-correcting algorithm

The exact procedure described in Section 3.4.2 relies on a label-correcting algorithm that outputs the path with maximum on-time arrival probability for a given departure time \( t \). This algorithm, outlined in Algorithm 3.4 below, is an \( A^* \) algorithm in the same spirit as the algorithm presented in Chen et al. [34], 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 \( t \).

Updating the label sets includes the extension of travel times on a path \( P \) to a path \( P \cup \{a\} \) for some \( a \in A \). We can do this essentially as in (3.9), except for the fact that \( \hat{C}_a \) is not well-defined in this case, because the arcs that will be traversed after arc \( a \) are unknown. However, we can determine the potential paths to travel beforehand, and define \( \hat{C}_a \) accordingly, in the sense that we let it contain all arcs that the label-correcting
algorithm may use. Specifically, if we define the weight of a link as the maximum travel time on that link, and apply Dijkstra’s algorithm on this network, the travel-time value of the best path will yield an upper bound on the first non-zero point of the travel-time distribution of the path with highest on-time arrival probability. Then, we may alternatively set the weights of the links in the network as the minimum travel times on those links, and compute, for each \( a' \in A \), the minimum travel time from its nodes to both the origin and the destination. This will yield a lower bound for the travel time on a path from the origin to the destination via link \( a' \), and, if this lower bound does not exceed the determined upper bound of the optimal path, there is a possibility this arc will be visited by the label-correcting algorithm. In that case, we let, for each \( a \in A \), \( \hat{C} \alpha \) contain the process \( X_{a'} \).

Initially, the label set of the origin consists of a single element, namely the distribution by the label-correcting algorithm. In that case, we let, for each \( a \in A \),\( \mathcal{L} \),\( \mathcal{L}_0 = \{ \mathcal{L}_0[s(Y)] \} \), with

\[
\mathcal{L}_0[s(Y)] = \{ (t, [e^{t\Omega_Y}]_{Y(0), s(Y)}) \} \tag{3.15}
\]

for each \( s(Y) \in \mathcal{T}(Y) \), whereas the other label sets start empty. The iterative procedure to update the label sets uses a (priority) 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 \( (P) \), a list of \((\text{travel time, probability})\)-pairs forming the travel-time distribution of this path \( (\mathcal{L}) \), 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 distributed according to \( L \). At the start, \( q = \{ \{ \alpha, \mathcal{L}_0, k_0, \text{path: } \{ k_0 \} \} \} \), 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 \( \mathcal{L}' \) for traversing subsequently \( P \) and the link \( (k, k') \) can be computed via (3.9). 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 \( \mathcal{L}' \), we insert \( \mathcal{L}' \) 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 \( k^* \).

Now, to improve the speed of the procedure, we perform one extra step before inserting \( \mathcal{L}' \) in the label set of \( k' \) and a new tuple in the queue \( q \). That is, we check if the distribution \( \mathcal{L}' \) dominates one of the distributions already stored in the label set of \( k' \), or vice versa. In this context, ‘dominance’ refers to (first-order) stochastic dominance: for paths \( P_1 \) and \( P_2 \) from \( k_0 \) to \( k' \), with their respective travel-time distributions \( \mathcal{L}_1 \) and \( \mathcal{L}_2 \), we say that \( \mathcal{L}_1 \) dominates \( \mathcal{L}_2 \) if \( \mathbb{P}(T_{\mathcal{L}_2} \leq t) \leq \mathbb{P}(T_{\mathcal{L}_1} \leq t) \) for all \( t > 0 \) and \( \mathbb{P}(T_{\mathcal{L}_2} \leq t) < \mathbb{P}(T_{\mathcal{L}_1} \leq t) \) for at least one \( t > 0 \). Now, if \( \mathcal{L}_1 \) would indeed dominate a distribution \( \mathcal{L}_2 \) in the label set of \( k' \), then, for every path \( P' \) from \( k' \) to the destination \( k^* \), 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
3.D Speed-up techniques

We will discuss a simple but effective way to speed-up the computation of the optimal departure time for certain paths or OD-pairs. So far, 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. That is, for a given path $P' = \{a_1, \ldots, a_m\}$, we are able to compute the list $L_{\ell_0}^{s(C_{P'})}$ of (travel time, probability)-pairs that arises for each initial state $s(C_{P'})$ the background process of $P'$ can attain, i.e., $C_{P'} = C_{a_1} \cup \cdots \cup C_{a_m}$. 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 3.2 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 3.4.

**Remark 3.D.1.** Availability of the precomputed travel-time distribution of a subpath $P'$ reduces the number of iterations necessary to compute the travel-time distribution on

<table>
<thead>
<tr>
<th>Algorithm 3.4: Optimal path for a given departure time.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Result:</strong> path from $k_0$ to $k^*$ with highest on-time arrival probability, given</td>
</tr>
<tr>
<td>departure time $t$ and initial state $B(0)$.</td>
</tr>
<tr>
<td>1. Determine the set of arcs $A'$ that are potentially visited;</td>
</tr>
<tr>
<td>2. for each $a \in A$, set $\hat{C}<em>a = C_a \cup {X_a}</em>{a \in A'}$, and compute the lists $L_a[s(\hat{C}_a), \hat{s}(\hat{C}_a)]$</td>
</tr>
<tr>
<td>via (3.8);</td>
</tr>
<tr>
<td>3. Set $D_{k_0} = {L_0}$ and $D_{k_i} = \emptyset$ for all other $i$;</td>
</tr>
<tr>
<td>4. if $t + t_{\min}(k_0, k^*) &gt; M$ quit and <strong>return</strong> 0. else continue;</td>
</tr>
<tr>
<td>5. Let $q = {(1, L_0, k_0, \text{path: } {k_0})}$;</td>
</tr>
<tr>
<td>6. Extract $q^* = {\alpha, L, k, P}$ with minimum first entry from $q$;</td>
</tr>
<tr>
<td>7. if $k = k^*$ quit and <strong>return</strong> $(\alpha, P)$. else continue;</td>
</tr>
<tr>
<td>8. for each neighbor $k'$ of $k$ not in $P$ do</td>
</tr>
<tr>
<td>a. Compute distribution $L'$ via (3.9) with $a_i = (k, k')$ and $L_{i-1} = L$;</td>
</tr>
<tr>
<td>b. Compute $\alpha' = P(T_{L'} + t_{\min}(k', k^*) \leq M)$;</td>
</tr>
<tr>
<td>c. if $L'$ is not dominated by an element from $D_{k'}$ then</td>
</tr>
<tr>
<td>Remove all paths dominated by $L'$ in $D_{k'}$ and insert $L'$ into $D_{k'}$. Add</td>
</tr>
<tr>
<td>${(\alpha', L', k', P + k')}$ to $q$;</td>
</tr>
</tbody>
</table>
| 9. **return** to step 3.

never part of the optimal path from the origin $k_0$ to the destination $k^*$. Therefore, this subpath can be disregarded in subsequent iterations. This means that 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.
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. For example, in the compact MVM setting, 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 remains limited.

An additional advantage of precomputing the travel-time distribution for a set of paths, is that, with just a negligible loss in accuracy, we can substantially reduce the size of the state space and, consequently, further speed up the computations. Specifically, if one wants to find the optimal departure time for a path $P$ containing a subpath $P'$ for which the travel-time 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 $L_{P'}^{\tilde{s}(C_{P'})}$ of (travel time, probability)-pairs on $P'$ for which

$$\mathbb{P} \left( \exists t \in [0, M] : (X_{a'}(t))_{X_{a'} \in C_{P'}} = \tilde{s}(C_{P'} \setminus \{Y\}) \mid B(0) = s \right) < \varepsilon,$$

for some small $\varepsilon > 0$. We have seen examples of procedures to generate upper bounds for such hitting probabilities in Appendix 2.C.