Bachelorthesis

Einfluss und Handhabung probabilistischer Fahrzeiten in dynamischen Dial-a-Ride-Modellierungen - eine Simulationsstudie

Lukas Kröger

Würzburg, 15. Juli 2024

Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Betreuerin: Prof. Dr. Marie Schmidt

Abstract

The study is founded on a hard-constrained event-based model for solving the dynamic Dial-a-Ride-Problem (dDARP), relying on precalculated arc travel times. The aim was to assess the tolerability of the model to different levels of travel time variability using a simulated approach. All original constraints except for immediately violated ones originating from new, unforeseeable delays were maintained as long as satisfiable, before time constraints were gradually softened as long as necessary - with differentiation on constraint type level.
First, benchmarks using a lognormal distribution to model travel time showed impractically high CPU times. On the contrary, a follow-up benchmark using the gamma distribution with a CV of 0.5 performed surprisingly well, which suggests that outliers in travel time are most problematic. Simultaneously, the overall results for successful runs showed that neither increased travel time variability itself nor the subsequent softening of MILP constraints led to significant changes in routing decisions nor standard DARP metrics.This could indicate that the suggested softening approach is still too undifferentiated and decreases the motivation to reroute, reschedule or deny requests in order to mitigate the time constraint violations.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Einleitung

Der praktische Hintergrund dieser Thesis liegt in der Routenplanung von Personenbeförderungsdiensten, die durch die Bündelung von Passagieranfragen mit individuellen, aber ähnlichen Routen in Sammelfahrten effizient gestaltet werden soll. Im deutschsprachigen Raum wird dieses Geschäftsmodell meist als *Ridepooling* bezeichnet. Im Gegensatz zu den bereits in vielen Städten etablierten *Anrufsammeltaxis* gibt es hier keinen vorgegebenen Linienplan. Die Route ergibt sich stattdessen aus der möglichst effizienten sequentiellen Bedienung von Passagieranfragen unter gleichzeitiger Berücksichtigung nutzerdefinierter Zeitbeschränkungen.[1]
Ein in Deutschland namhafter Anbieter von Ridepooling-Diensten ist das Volkswagen-Tochterunternehmen MOIA[2], ein Beispiel aus der Region Mainfranken ist der Anbieter CallHeinz[3]. Die Zielgruppe solcher Dienste bilden derzeit vor allem Menschen mit eingeschränkter Mobilität, in erster Linie also Senioren oder Menschen mit Behinderung, deren Anforderungen der breite öffentliche Nahverkehr häufig nicht gerecht wird.
Das zugrundeliegende mathematische Optimierungsproblem wird als *Dial-a-Ride-Problem (DARP)* bezeichnet. Das Dial-a-Ride-Problem ist eine Ausprägung eines Vehicle-Routing-Problems (VRP), welches wiederum eine Erweiterung des in der Informatik und Graphentheorie äußerst populären und NP-schweren Travelling-Salesman-Problems (TSP) darstellt. Demzufolge ist auch die exakte Lösung des Dial-a-Ride-Problems NP-schwer. Es existieren allerdings heuristische Ansätze zur effizienteren, aber suboptimalen Lösung des DARP (siehe Literaturübersicht).
Weiterhin wird hier die dynamische Variante des DARP behandelt (dDARP), bei der neue

[1] Einzelne Bestandteile der Einleitung stammen aus einer im Vorfeld von mir eingereichten Seminararbeit

[2] https://www.moia.io

[3] https://www.callheinz.de/

Anfragen im Unterschied zur statischen Variante auch während des operativen Einsatzes, also während der Fahrt, bearbeitet und integriert werden können.
Maßgeblich fußt diese Thesis auf einer Publikation von Gaul et al. [7] aus dem Jahre 2021, die einen ereignisbasierten Ansatz zur exakten, aber heuristisch unterstützten Lösung des dDARP verwendet. Zusätzlich soll nun ein möglichst realistischer *unsicherer* Kontext einbezogen werden: stochastische Schwankungen innerhalb der Fahrzeiten. Ungeachtet dessen soll aus Laufzeitgründen keine klassische stochastische Optimierung verwendet werden, also keine Probabilistik in die Modellierung selbst integriert werden.

1.2 Literaturübersicht

Das Dial-a-Ride-Problem wurde erstmals von Psaraftis et al. 1980 formuliert [16], der damals bereits zwischen statischer und dynamischer Variante unterschied[4]. Mittlerweile gibt es zahlreiche weitere Varianten und Untervarianten [5][12], angefangen von Fällen mit einzelnen und mehreren Fahrzeugen, über die linienbasierte Form LiDARP [23], die semiflexible Fahrzeiten auf einer vordefinierten Linie zulässt, Varianten mit mehreren Fahrzeugdepots, bis hin zur Erwägung von Passagierumstiegen zwischen einzelnen Fahrzeugen [11].
Unabhängig davon existieren deterministische Formen, bei denen alle Informationen mit Ausnahme der gegebenenfalls dynamisch eintreffenden Anfragen von Beginn an festgelegt sind, neben stochastischen Formen, bei denen unterschiedliche Faktoren unsicher sein können, beispielsweise eine probabilistische Nachfrage, Kantenkosten, Fahrzeiten oder zur Verfügung stehende Betriebsmittel (Fahrzeuge, Personal) [24].
Schließlich lässt sich noch nach Lösungsmethodiken unterscheiden. Ansätze aus der Frühzeit des DARPs setzten fast ausschließlich und aus Gründen der Rechenleistung auf heuristische Verfahren wie lokale Suche, Tabu-Suche oder *Simulated Annealing*. Gerade bei anspruchsvollen Umgebungen mit großer Nachfrage und Entscheidungsspielraum sind heuristische Ansätze häufig auch heute noch zwingend erforderlich [12][14].

1.2.1 Dynamisch-stochastisches Dial-a-Ride-Problem

Die dynamische Variante des Dial-a-Ride-Problems fand bis heute eine geringere Betrachtung als die statische Form, zur stochastischen Abwandlung sDARP gibt es eine abermals niedrigere Literaturdichte und schließlich nur einige wenige Publikationen, die explizit stochastische Fahrzeiten untersuchen. Die bekannteste Publikation dazu von Schilde et al. [19] verwendet darüber hinaus einen äußerst spezialisierten Ansatz – es werden hier

[4] Zur Ermittlung thematisch relevanter Publikationen wurde neben klassischen Suchmaschinen auch das ChatGPT-Plugin *Consensus* verwendet.

auf Basis historischer Daten Unfallwahrscheinlichkeiten einer Route einbezogen. Einzig Chassaing et al. [2] kommen im Jahre 2016 unserem Anwendungsfall sehr nahe, setzen aber, wie bei stochastischer Optimierung üblich, von Beginn an auf Zufallsvariablen zur Modellierung der Kantenfahrzeiten.
Der Auswertung von Literatur zur reinen Modellierung von unsicheren Kantenfahrzeiten widmet sich Kapitel 4.

1.2.2 Stochastische Varianten anderer Vehicle-Routing-Probleme

In einer umfassenden Übersichtsarbeit [13] von Oyola et al. 2015 wurden stochastische Varianten unterschiedlicher Vehicle-Routing-Probleme ausgewertet – mit einem Hauptaugenmerk auf den betrachteten stochastischen Aspekt und dessen Modellierung. Diese ergab, dass Fahrzeiten zumeist als gleich- oder kontinuierlich verteilte Zufallsvariablen behandelt wurden – am häufigsten kamen die (Log-)Normal- und die Gammaverteilung zum Einsatz.

Kapitel 2

Problemstellung

2.1 Vorstellung der Grundidee

Das von Gaul et al. [7] vorgeschlagene Modell beruht auf einem gerichteten und gewichteten Graphen $G = (V, E)$ zur Repräsentation von Ereignissen und deren Abfolgen, mithin also der möglichen Routen. Wir werden die (MILP-)Modellierung für dieses Kapitel größtenteils übernehmen, für die spätere Simulation sind allerdings kleinere gekennzeichnete Anpassungen notwendig.

Die Struktur des Graphen baut primär auf der möglichen Reihenfolge zur Bedienung von Passagieranfragen und weniger auf der geographischen Lage der Start- und Zielorte der Passagiere auf. Knoten repräsentieren dabei Ereignisse - den Einlass oder Auslass von Passagieren -, Kanten auf der anderen Seite eine mögliche Abfolge zweier solcher Ereignisse.

Zeitliche Differenzierung: Bei der dynamischen Variante des DARP spielt der zeitliche Bezug der Objekte eine zentrale Rolle. Wir verwenden dazu Zeitpunkte τ aus einer Zeitpunktliste $\mathcal{T}$. $G(\tau) = (V(\tau), E(\tau))$ verweist dann auf den Graphen zum Zeitpunkt τ.

In der zugrundeliegenden Publikation werden alle Zeitpunkte τ_i Anfragen zugeordnet. τ_3 bezeichnet also den Zeitpunkt, an dem die Anfrage des Passagiers 1 *eingetroffen* ist. Wir werden die Zeitgebung von den Anfragen abkoppeln und allgemein von Elementen τ_j einer zeitlich aufsteigend geordneten Zeitpunktliste $\mathcal{T}$ sprechen. $\mathcal{T}$ kann grundsätzlich alle für die Simulation interessanten Zeitpunkte enthalten. Einschränkungen sind (theoretisch) durch Rechenzeit eines Modellierungsschritts bei sehr großen Instanzen (mehr als 100 Anfragen) und gleichzeitig sehr geringem Zeitabstand zwischen den einzelnen Einträgen der Liste (unter einer Sekunde) gegeben.

2.2 Passagiere und Anfragen

Anfragen und die dazugehörigen Passagiere werden gleichermaßen als $i \in \mathcal{A}$ bezeichnet, was dem Umstand geschuldet ist, dass eine Anfrage auch mit mehreren Sitzplätzen verbunden sein kann. Um sprachlich verständlich und dennoch methodisch korrekt zu bleiben, legen wir fest, dass mit Passagier i stets die gesamte dazugehörige Anfrage gemeint ist, auch wenn es sich dabei um mehrere Personen handelt. Zum Beispiel bedeutet dies, dass der Einstieg des Passagiers 3 in ein Fahrzeug gleichbedeutend mit dem Einstieg aller zur Anfrage 3 zugehörigen Personen ist. Übereinstimmend damit soll es auch nicht möglich sein, nur einzelne Sitzplätze einer Anfrage zuzusagen (wenn weitere benötigt würden) oder diese in verschiedene Fahrzeuge zu platzieren.

2.2.1 DARP-Knoten $\nu \in N$

An dieser Stelle sei nun auch das Konzept des *DARP-Knotens* ν und der inkludierenden Menge N erwähnt, welche nicht zu verwechseln mit den später zu definierenden *Ereignisknoten* sind. Auch sind sie kein Bestandteil des Ereignisgraphen $G(V, E)$. Ein DARP-Knoten repräsentiert einen Einstiegsort oder einen Ausstiegsort eines Passagiers. Für einen sinnvollen Ablauf des Ridepoolingbetriebs verfügt jeder Passagier i über genau einen Einstiegs-DARP-Knoten ν_i^+ und genau einen Ausstiegs-DARP-Knoten ν_i^-.
Die beiden DARP-Knoten dienen auch als Struktur zur Repräsentation der Anfragedaten einer einzelnen Anfrage i:

- ein Zeitfenster $[e_\nu, l_\nu]$
- die Länge der Servicezeit s_ν, die für den Zustieg und Ausstieg benötigt wird
- eine maximal akzeptierte Reisedauer/Reisezeit von L_ν.
- eine Anzahl der benötigten Sitzplätze q_ν

In der Instanzbeschreibung in 3.1.2 wird näher erläutert, welche Vorgaben zu den Parameterwerten des DARP-Knotens bestehen.

2.2.2 Anfrageuntermengen

Nicht nur zu Simulationszwecken, auch zu Modellzwecken ist die Unterscheidung von Passagieren/Anfragen nach deren Zustand bedeutsam. Der Lebenszyklus einer Anfrage wird parallel in Anhang 1 anschaulich gemacht. Zu beachten ist außerdem die bereits im vorigen Abschnitt erwähnte zeitliche Differenzierung der Anfrageteilmengen. Lediglich bei der Obermenge $\mathcal{A}$, die alle Anfragen (einer Instanz) umfasst, ist keine zeitliche Referenzierung notwendig, da diese während des gesamten Simulationsablaufs unverändert bleibt.

Wichtig für uns sind alle folgenden Mengen:

- $\mathcal{A}$ - *alle* in der Instanz (vgl. 3.1.2) enthaltenen Anfragen
- $\mathcal{A}(\tau_j)$ - der Modellierungseinheit zum Zeitpunkt τ_j vorliegende und *akzeptierte*, mithin bei Entscheidungen zu berücksichtigende Anfragen
- $\mathcal{N}(\tau_j)$ - neue Anfragen zum Zeitpunkt τ_i (*new*): Anfragen, die zwischen τ_j und τ_{j-1} eingegangen sind
- $\mathcal{S}(\tau)$ - zum Zeitpunkt τ angenommene, noch nicht im Fahrzeug befindliche Passagiere (*seeker*)
- $\mathcal{S}^{fix}(\tau)$ - zum Zeitpunkt τ angenommene Passagiere, deren Einstiegshaltestelle aktuell angesteuert wird
- $\mathcal{R}(\tau)$ - zum Zeitpunkt τ abgelehnte Anfragen (*rejected*)
- $\mathcal{P}(\tau)$ - zum Zeitpunkt τ bereits in einem Fahrzeug befindliche oder gerade einsteigende Passagiere (*picked-up*)
- $\mathcal{P}^{fix}(\tau)$ - zum Zeitpunkt τ bereits in einem Fahrzeug befindliche Passagiere, deren Ausstiegshaltestelle gerade angesteuert wird
- $\mathcal{D}(\tau)$ - zum Zeitpunkt τ bereits abgesetzte oder aussteigende Passagiere (*dropped-off*)
- Teilmengen der eingeplanten, abgelehnten, aufgenommenen und abgesetzten Passagiere $\mathcal{S}^{neu}(\tau_j)$, $\mathcal{R}^{neu}(\tau_j)$, $\mathcal{P}^{neu}(\tau_j)$ und $\mathcal{D}^{neu}(\tau_j)$, die seit der letzten Lösung des MILP in den dazugehörigen Obermengen hinzugekommen sind. Da Änderungen in diesen Mengen umgehend einen neuen Modellierungsschritt anstoßen, betrifft dies ebenso alle Anfragen zwischen τ_j und τ_{j-1}

Die Mengen $\mathcal{S}^{fix}(\tau)$ und $\mathcal{P}^{fix}(\tau)$ wurden zusätzlich zur Modellierung von Gaul et al. [7] ergänzt und sind für die spätere Simulation insofern entscheidend, als dass Fahrzeuge, die sich gerade auf der Fahrt zwischen zwei Haltestellen/DARP-Knoten befinden, diesen Kurs in keinem Fall ändern werden. Dadurch sind der Modellierung wesentliche Einschränkungen gesetzt, was in Folge auch die Komplexität reduziert.

2.3 Ereignisgraph $G(V, A)$

Der Ereignisgraph repräsentiert den praktikablen Teil des Lösungsraums, besteht also hauptsächlich aus hypothetischen Knoten und Kanten, die kein Bestandteil der tatsächlich umgesetzten Route sein werden.

2.3.1 Ereignisknoten $v \in V(G)$

Definition 1 *Knoten $v \in V$: Die Knoten $v \in G(V)$ nennen wir Ereignisknoten. Sie werden als Q-Tupel dargestellt, welche die Belegungen der Q Sitzplätze eines Fahrzeugs ungeachtet der Reihenfolge zum Ereigniszeitpunkt mit Passagieren v_j kodieren:*

$$v_{i^{+|-}} \sim (v_1 := i^{+|-}, v_2, ..., v_Q) \text{ mit } a, v_j \in \mathcal{A} \cup \{0\} \setminus \{i\}, v_j \leq v_{j+1} \forall j \in \{2, 3, ..., Q\}$$

Das erste Element indiziert die Art des Ereignisses (Zustieg oder Ausstieg) und ist stets ein an der betreffenden Haltestelle zusteigender (bei v^+-Knoten) oder aussteigender Passagier (bei v^--Knoten). Ein Sonderfall sind die Ereignisse 0^+ und 0^-, die den Dienstbeginn bzw. den Dienstschluss am Depotknoten 0 repräsentieren. Die übrigen Sitzplätze sind aufsteigend sortiert, für freie Sitzplätze wird das Tupel mit 0-Einträgen aufgefüllt.
Zu beachten ist, dass es im Unterschied zu den DARP-Knoten für jeden Passagier beliebig viele Ereignisknoten geben kann, begrenzt lediglich durch die Gesamtanzahl der Anfragen und die Realisierbarkeit hinsichtlich Zeitbeschränkungen (siehe Definition 4).

Definition 2 *Wir nennen das erste Element v_1 eines Ereignisknotens dessen signifikantes Element.*

Beispiele für Ereignisknoten (bei $Q = 3$): $(1^+, 2, 0)$ für den Zustieg von Passagier 1, während Passagier 2 bereits im Fahrzeug sitzt; $(2^-, 1, 0)$ für den Ausstieg von Passagier 2, während Passagier 1 noch im Fahrzeug sitzt (im Anschluss an das vorige Ereignis denkbar). Der Depotknoten kann hier auch als $(0^+, 0, 0)$ oder $(0^-, 0, 0)$ benannt werden.
Verknüpft ist jeder Ereignisknoten zusätzlich mit dem Ort des Ereignisses, also dem Start- oder Zielort des signifikanten Passagiers. Isoliert werden wir diese Orte im Folgenden meist als Haltestellen bezeichnen.

Beobachtung 1 *Jeder Ereignisknoten ist über sein signifikantes Element eindeutig mit einem DARP-Knoten ν verknüpft.*

Zusammenfassend lassen sich Ereignisknoten als erst zu errechnende Anzeige der möglichen Sitzplatzbelegungen an einer Haltestelle verstehen, während DARP-Knoten eher pragmatisch orientiert sind und direkt aus den Eingabedaten, der Testinstanz folgen.

2.3.2 Kanten $a \in A(G)$

Definition 3 *Kanten $a \in A(G)$: Eine Kante $a = (v, w)$ repräsentiert die Abfolge zweier Ereignisknoten $v, w \in V(G)$, die von einem Fahrzeug direkt hintereinander passiert werden. Die Kantengewichte c_a sind durch die Entfernung zwischen v und w festgelegt.*

Die *Fahr*zeit t_a einer Kante $a \in A(G)$ folgt direkt aus den Kantengewichten und einer vorab festzulegenden Durchschnittsgeschwindigkeit des Fahrzeugs. Für den weiteren Verlauf ist die Unterscheidung dieser fahrzeugzentrierten Zeit von der passagierzentrierten *Reisedauer/Reisezeit* (Zeitspanne, die ein Passagier im Fahrzeug verbringt) äußerst wichtig.

2.3.3 Mengen innerhalb des Graphen $G(\tau)$

Die vielfältigen Möglichkeiten zur Integration von Passagieranfragen in die Route eines Fahrzeugs machen mehrere Teilmengen der Ereignisknotenmenge sowie der Kantenmengen des Ereignisgraphen G notwendig:

- Teilmengen der Knotenmenge $V(\tau)$ des Graphen $G(\tau)$ (Depot, Einstiegs- und Ausstiegsknoten): $V_{i^+}(\tau)$ (Zustiegsereignisse), $V_{i-}(\tau)$ (Ausstiegsereignisse) für $i \in \mathcal{A}$
- Fixierte Kanten A^{fix}: Hier sind alle Kanten mit eingeschlossen, die bereits befahren wurden oder werden
- Realisierte Kanten A^{real}: Hier sind alle Kanten mit eingeschlossen, die befahren wurden und deren Zielknoten bereits erreicht ist
- Realisierte Knoten V^{real}: Hier sind neben dem Depotknoten V_0^+ wiederum alle Knoten eingeschlossen, die Endknoten einer realisierten Kante sind (durch Induktionsprinzip damit auch alle Startknoten dieser Kanten)

2.3.3.1 Ereignisknotenmengen

Eine präzise Definition ist vor allem für die wichtigsten Teilmengen der Knotenmenge V nötig. Vorab gilt es allerdings, zwei Hilfsfunktionen einzuführen.

Definition 4 *Wir nennen $f^1_{a,b}$ und $f^2_{a,b} \in \{0, 1\}$ mit $a, b \in \mathcal{A}$, den Realisierbarkeitsindikator der Ereignisabfolgen $b^+ \to a^+ \to b^- \to a^-$, bzw. $b^+ \to a^+ \to a^- \to b^-$, mit dem hochgestellten Plus als Einstieg und dem hochgestellten Minus als Ausstieg des zugehörigen Passagiers.*

Realisierbar bedeutet (informell):

- kein Hin- und Herspringen von Passagieren zwischen Fahrzeugen
- Einhaltung der Zeitfenster der Anfragen a und b möglich unter Berücksichtigung der bei dieser Abfolge kumulierten Fahrzeiten

Diese Hilfsfunktionen mögen penibel und unscheinbar wirken, ermöglichen jedoch ein effektives Ausschlussverfahren, das die Zahl generierter Ereignisknoten und Knoten in G deutlich reduziert.

Hierauf aufbauend wird die Menge der möglichen Zustiegsereignisse V_{i+} definiert. Diese fasst alle *möglichen* Ereignisse zusammen, die den Zustieg von Passagier i enthalten - mit variabler Besetzung des Fahrzeugs:

$$V_{i+}(\tau) = \left\{ (v_1, v_2, ..., v_Q) \middle| \begin{matrix} v_1 = i+, v_i \in \mathcal{A}(\tau) \cup \{0\} \setminus \{i\}, \\ f^1_{i,v_j} + f^2_{i,v_j} \geq 1 \, \forall j \in \{2, ..., Q\} \end{matrix} \right\}$$

Die Definition von V_{i-}, der Menge möglicher Ausstiegsereignisse, unterscheidet sich nur durch das erste Element und die Parameter der Funktion f, die zweite Mengenbedingung entspricht der von V_{i+}.

$$V_{i-}(\tau) = \left\{ (v_1, v_2, ..., v_Q) \middle| \begin{matrix} v_1 = i-, v_i \in \mathcal{A}(\tau) \cup \{0\} \setminus \{i\}, \\ f^1_{v_j,i} + f^2_{i,v_j} \geq 1 \, \forall j \in \{2, ..., Q\}, \end{matrix} \right\}$$

2.4 MILP-Modellierung

Wie von Gaul et al. [7] angemerkt, kann das Dial-a-Ride-Problem als erweitertes oder verallgemeinertes Flussproblem verstanden werden, nämlich als Berechnung einer kostenminimalen Zirkulation im Ereignisgraphen G, die allerdings zahlreichen, größtenteils zeitlichen Einschränkungen gerecht werden muss. Wir werden es nun als gemischt-ganzzahliges-lineares Programm (*MILP*) formulieren. Die Darstellung folgt ebenfalls Gaul et al. [7] bis auf eine leicht abgewandelte Variablenbenennung.

2.4.1 Zusätzliche Kantenvariablen und -funktionen

Als binäre Entscheidungsvariable soll x_a dienen, welche genau dann den Wert 1 annimmt, wenn die Kante $a \in A(G)$ von einem Fahrzeug befahren werden soll:

$$x_a = \begin{cases} 1 & a \text{ soll verwendet werden} \\ 0 & a \text{ soll nicht verwendet werden} \end{cases}$$

Die Variable T_v bezeichnet hier den Beginn der Dienstleistung am Knoten v (in anderen Publikationen meist als B_v bezeichnet). Diese sind für die Modellierung entscheidend, da der Dienstbeginn mit den Zeitfenstern des zu- ($v \in V_{i+}$) oder aussteigenden ($v \in V_{i-}$) Passagiers $i \in \mathcal{A}$ an diesem Knoten vereinbar sein muss. Für die spätere Simulation wird allerdings auch der An- und Abfahrtszeitpunkt an diesem Knoten eine Rolle spielen. Erster, $T_v^{arrival}$, steht in keinem fixen Abstand zum Dienstbeginn, für die Abfahrt rechnen wir allerdings stets mit $T_v + s_i$, wobei s_i die Servicezeit für den signifikanten Passagier des Knotens v ist.
$\delta^{in}(v)$ und $\delta^{out}(v)$ zählen die ein- bzw. ausgehenden Kanten des Knotens v auf. Die erzielte Lösung der Modellierung setzt sich also aus x_a und T_v zusammen.

2.4.2 Entscheidungsvariable zur Annahme von Passagieren

Eine für das dynamische Dial-a-Ride-Problem nicht essenzielle Erweiterung ist die Möglichkeit zur Ablehnung von Passagieren, wenn diese nicht ausreichend effizient in die Route eines Fahrzeugs integriert werden können. Gerade für eine probabilistische Simulation ist es sinnvoll, über derartige Flexibilität zu verfügen. Wir definieren dazu p_i als Entscheidungsvariable, die die Annahme eines Passagiers $i \in N(\tau)$ angibt:

$$p_i = \begin{cases} 1 & \text{Passagier } i \text{ wird angenommen} \\ 0 & \text{Passagier } i \text{ wird abgelehnt} \end{cases}$$

Zu beachten ist, dass sich diese Entscheidungsvariable ausdrücklich auf Anfragen bezieht, die zum Zeitpunkt der Lösung des MILPs τ neu sind. Bei der folgenden Iteration des gemischt-ganzzahligen-linearen Programms, also im Allgemeinen für alle $i \in N(\tau)$, werden diese festgeschrieben, indem zusätzliche Nebenbedingungen eingeführt werden, die den erstmalig berechneten Wert für p_i erzwingen. Zusammengefasst bedeutet dies, dass bereits akzeptierte Anfragen nicht mehr abgelehnt werden können und nicht akzeptierte Anfragen nicht mehr nachträglich angenommen werden können - eine derartige Unberechenbarkeit wäre Nutzern nicht zumutbar.
Die Ablehnung von Anfragen muss im Übrigen sorgfältig mit den dadurch eingesparten Kosten abgewogen werden. Grundsätzlich sollte ein Dienst anstreben, so viele Anfragen wie möglich zu bedienen und nur ausgewählte Anfragen abzulehnen. Diese Problemstellung wird im folgenden Abschnitt zur Zielfunktion aufgegriffen.

2.4.3 Zielfunktion

Die einfachste Zielfunktion zum DARP strebt lediglich die Minimierung der Gesamtfahrtkosten (und -fahrzeit) an:

$$\min f_c = \min \sum_{a \in A(\tau)} c_e x_a \tag{2.1}$$

Gemäß unserer Problemdefinition soll auch die Ablehnung von Anfragen möglich sein. Wie bereits angedeutet ist es gleichzeitig erforderlich, dass die Zahl der abgelehnten Anfragen minimiert und in der Zielfunktion berücksichtigt wird. Dazu sei

$$f_n(p) = n - \sum_{i \in \mathcal{A}} p_i \tag{2.2}$$

Darüber hinaus soll die Zielfunktion auch die Reisezeiten individueller Passagiere eindämmen. Beim DARP hat die Minimierung der Gesamtwegstrecke der Fahrzeuge (im Gegensatz zu anderen *Vehicle Routing*-Problemen) nicht die immer höchste Priorität. Passagieren soll nicht zugemutet werden, zu lange im Fahrzeug „mitgeschleppt“ zu werden. Es

ließe sich zwar argumentieren, dass die Einhaltung der Zeitfenster und Maximalreisezeit des Passagiers zur Lösung ausreichend wäre. Auf der anderen Seite soll aber tendenziell vermieden werden, für eine verhältnismäßig kurze Distanz verhältnismäßig lange im Fahrzeug sitzen zu müssen – Passagiere könnten auch dies als weniger befriedigend wahrnehmen, selbst unter Einhaltung der Maximalreisezeit.
Daher soll die Differenz d_a der von der Lösung berechneten Reisezeit des Passagiers von der direkten Fahrzeit zwischen dessen Start- und Zielort ebenfalls minimiert werden.

$$f_r(d) = \sum_{i \in \mathcal{A}} d_i \tag{2.3}$$

Diese drei Gesichtspunkte müssen nun mit noch abzustimmenden Gewichten ω_c, ω_a und ω_d (normiert nach der Gewichtung 1 für die Funktion f_c) in der Gesamtzielfunktion miteinander verknüpft werden:

$$\min f_{cnr} = \min \left(\omega_c \sum_{a \in A} c_e x_a + \omega_a (n - \sum_{i \in \mathcal{A}} p_i) + \omega_d \sum_{i \in \mathcal{A}} d_i \right) \tag{2.4}$$

Für die Einführung von elastischen Nebenbedingungen in Kapitel 4 wird dieser Term noch einmal im Vordergrund stehen und deutlich erweitert, da die Aufweichung von scharfen Nebenbedingungen die Minimierung der Verletzungen erforderlich macht.

2.4.4 M-Konstanten

Die Linearisierung mithilfe der *Groß-M-Methode* innerhalb des folgenden Programms macht weiterhin die exakte Definition der M-Konstanten erforderlich.
Notwendig sind diese zum einen für Nebenbedingung 6, die den „Knoten-Fahrzeit-Abstand“ sicherstellt. Der Mindestabstand für aufeinanderfolgende Knoten $(u, v) \in A(G)$ ist definiert durch die Servicezeit am Startknoten s_u und die Fahrzeit $t_{(u,v)}$ - allerdings nur dann, wenn diese Kante auch befahren werden soll. Um die daher naheliegende Multiplikation des Terms mit der Entscheidungsvariable $x_{(u,v)}$ zu vermeiden, verwenden wir folgende Konstante:

$$M_{(u,v)}(\tau_j) \geq \begin{cases} \tau_j - e_{v(1)} + s_{u(1)} + t_{(u,v)} & \text{falls } x_u^{real} \\ l_{u(1)} - e_{v(1)} + s_{u(1)} + t_{(u,v)} & \text{andernfalls} \end{cases} \tag{2.5}$$

Kanten (v, w), deren Startknoten v bereits realisiert ist, werden separat behandelt, primär um den Wert der M-Konstante möglichst gering zu halten. Die Realisierung wird über die Indikatorvariable x_u^{real} für Knoten $u \in V(G)$ angezeigt, welche im Rahmen der Simulationsbeschreibung in Kapitel 3 näher erläutert wird. Bei möglichen Verspätungen am Knoten v ist dies auch zwingend notwendig, um die ausreichende Größte der M-Konstante sicherzustellen.

Dabei entspricht $u(1)$ und $v(1)$ jeweils dem signifikanten Passagier der Knoten. Multipliziert mit der „umgekehrt“ belegten Entscheidungsvariable ergibt sich ein Term, der bei einer nicht zu verwendenden Kante stets größer ist als die Differenz von T_v und dem Mindestabstand, womit diese Nebenbedingung nicht mehr ausschlaggebend zur Belegung von T_v ist.
Zum anderen ist eine analoge Konstante $M_i \geq l_{i^-} - e_{i^+} - L_i - s_{i^+}$ zur Einhaltung der maximalen Reisezeit 7 einzelner Passagiere i notwendig. Hier ist allerdings eine Deaktivierung der Nebenbedingung für *alle* Knotenpaare v, w erforderlich, bei denen nicht gleichzeitig v dem Einstiegs- *und* w dem Ausstiegsknoten entspricht.

2.5 Ereignisgesteuertes MILP(τ_j)

Zielfunktion:

$$\min f_{cnr} = \min \left(\sum_{a \in A} c_a x_a + \omega_p (n - \sum_{i \in \mathcal{A}} p_i) + \omega_d \sum_{i \in \mathcal{A}} d_i \right) \tag{2.6}$$

Nebenbedingungen:

1. Einhaltung der Passagierzeitfenster

$$e_{i^+} \leq T_v \leq l_{i^+} \quad \forall i \in \mathcal{A}, v \in V_{i^+}(\tau_j) \tag{2.7}$$

$$e_{i^-} \leq T_v \leq l_{i^-} \quad \forall i \in \mathcal{A}, v \in V_{i^-}(\tau_j) \tag{2.8}$$

2. Bedienung aller akzeptierten Passagieranfragen

$$\sum_{\substack{e \in \delta^{in}(v) \\ v \in V_{i^+}(\tau_j)}} x_a = p_i \quad \forall i \in \mathcal{A} \tag{2.9}$$

3. Flusserhaltung - Knoten müssen genau so häufig verlassen wie angefahren werden; zusammen mit den übrigen Nebenbedingungen ist es für alle Nicht-Depotknoten entweder einmal oder gar nicht möglich

$$\sum_{a \in \delta^{in}(v)} x_a = \sum_{a \in \delta^{out}(v)} x_a \quad \forall v \in V(\tau_j) \tag{2.10}$$

4. Begrenzung K der gleichzeitig einsetzbaren Fahrzeuge:

$$\sum_{e \in \delta^{in}(0)} x_a \leq K \tag{2.11}$$

   Hier genügt es zu garantieren, dass der Depotknoten 0 von höchstens K Fahrzeugen verlassen werden kann.

5. Ankunftszeit am Nachfolgeknoten v des Depots muss später liegen als Dienstbeginn T_0 bzw. als der umgehenden Startzeitpunkt weiterer Fahrzeuge zuzüglich Fahrzeit vom Depot:
$$T_v \geq \max(T_v, \tau_j) + t_{(0,v)} x_{(0,v)} \quad \forall (0,v) \in A(\tau_j) \tag{2.12}$$

6. Ankunft am Knoten w muss später erfolgen als die Ankunft am Vorgängerknoten v, addiert mit dortiger Servicezeit s_v sowie Fahrzeit $t_{(v,w)}$ zum Knoten w

   (a) Nicht-lineare Formulierung:
   $$T_v \geq (\max(T_u, \tau_j) + s_{v(1)} + t_{(u,v)}) x_{(u,v)} \quad \forall (u,v) \in A(\tau_j), v \neq 0 \tag{2.13}$$

   (b) Linearisierte Form unter Verwendung der *Groß-M-Methode*:
   $$T_w \geq \max(T_v, \tau_j) + s_{v(1)} + t_{(v,w)} - M_{(v,w)}(\tau_j) \times (1 - x_{(v,w)}) \quad \forall (v,w) \in A(\tau), v \neq 0 \tag{2.14}$$

7. Einhaltung der maximalen Fahrzeit L_i für Passagier i: Hier wird sichergestellt, dass die Zeitdifferenz nur innerhalb dem Paar aus Ein- und Ausstiegsknoten zu Anfrage i berücksichtigt wird, das tatsächlich in der Lösung enthalten ist.

   (a) Nichtlineare Formulierung:
   $$(T_w - T_v - s_i) \sum_{e \in \delta^{in}(v)} x_a \sum_{e \in \delta^{in}(w)} x_a \leq L_i \quad \forall i \in A(\tau_j), v \in V_{i^+}(\tau_j), w \in V_{i^-}(\tau_j) \tag{2.15}$$

   (b) Linearisierte Formulierung:
   $$T_w - T_v - s_{i^+} \leq L_i + M_i(1 - \sum_{e \in \delta^{in}(v)} x_a + 1 - \sum_{e \in \delta^{in}(w)} x_a) \quad \forall i \in A(\tau_j), v \in V_{i^+}(\tau_j), w \in V_{i^-}(\tau_j) \tag{2.16}$$

8. Bereits akzeptierte Anfragen müssen weiterhin akzeptiert bleiben:
$$p_i = 1 \quad \forall i \in S(\tau_j) \tag{2.17}$$

9. Sicherstellung zulässiger Werte für x_a und T_v:
$$x_a \in \{0,1\} \quad \forall a \in A(\tau_j), \qquad T_v \geq 0 \quad \forall v \in V(\tau_j) \tag{2.18}$$

10. Kommunizierte Abholzeit: Das Einstiegsereignis muss, sofern es realisiert werden soll, maximal Γ später erfolgen als kommuniziert (im Unterschied zu [7] von Zeitfensterbedingung separiert)
$$T_v \leq (\gamma_i^+ + \Gamma) \left(1 - \sum_{a \in \delta^{in}(v, \tau_j)} x_a\right) \forall i \in S(\tau_j) \tag{2.19}$$

Zu beachten ist, dass die Kapazitätsbeschränkung der einzelnen Fahrzeuge und die zeitliche Sicherstellung des Einstiegs vor dem Ausstieg eines Passagiers nicht als Nebenbedingung formuliert sind, da sie in der Definition der Knotenmengen von Gaul et al. vorausgesetzt werden.

2.6 Aktionsplan

Für die Simulation ergänzt wurde der *Aktionsplan* Π , welcher Ergebnis eines jeden erfolgreichen Modellierungslaufs (welcher in Kapitel 2 vollständig beschrieben wird) ist. Ein Aktionsplan ist abhängig von der Lösungszeit des MILP τ als Tupel mehrerer Mengen definiert:

$$\Pi(\tau) \sim \begin{pmatrix} \{x_a(\tau)|\forall a \in A\}, \{T_v(\tau)|\forall v \in V\}, \{p_i(\tau)|\forall i \in \mathcal{A}\} \\ \{d_i(\tau)|\forall i \in \mathcal{A}\}, \left\{\left(\gamma_i^+, \gamma_i^-\right)|\forall i \in \mathcal{S}\right\}, \{C_k(\tau)|\forall k \in K\} \end{pmatrix}$$

Dabei enthält die erste Menge für jede Kante $a \in A(G)$ die Entscheidungsvariable x_a, die genau dann den Wert 1 annimmt, wenn e verwendet werden soll. Die zweite Menge enthält die geplanten Ereigniszeitpunkte T_v für alle Knoten $v \in V(G)$, die dritte und vierte Menge jeweils p_i, bzw. d_i für alle bekannten Anfragen $i \in \mathcal{A}$, wobei p_i genau dann den Wert 1 annimmt, wenn die Anfrage i von der Lösung als akzeptabel integrierbar bewertet wurde und d_i für jeden Passagier i die überschüssige Fahrzeit bei Umsetzung der Lösung darstellt. Des Weiteren erhalten wir die kommunizierte Abholzeit γ_i^+ und die kommunizierte Absetzzeit γ_i^- für jeden Passagier i. Diese wäre nicht zwingend der Simulation zu übergeben, da sie keine Auswirkung auf deren Ablauf hat, müsste aber im realen Einsatz selbstverständlich Teil des Aktionsplans sein.
Schließlich erhalten wir noch die Routen C_k aller k Fahrzeuge, wobei noch nicht benötigte Fahrzeuge als Route die leere Menge erhalten. Diese Routen sind stets Kreise mit Start und Ende am Depot. Auch dieser Bestandteil wäre für die Simulation entbehrlich, ermöglicht aber eine unkomplizierte Visualisierung der Fahrzeugrouten in der Simulation. Zudem werden alle Fahrzeugrouten im Rahmen der Lösungsverarbeitung der Modellierungseinheit (Abschnitt 3.2.2) ermittelt, liegen also grundsätzlich vor.

Kapitel 3

Simulation

Der veröffentlichte C++-Code[1] deckt alle Schritte vom Auslesen der Eingabedateien bis zum Konstruieren der gemischt-ganzzahlig-linearen Programme und aller notwendigen Interaktionen mit *ILOG-CPLEX* ab.[2]

3.1 Aufbau

Die Simulation besteht aus zahlreichen Klassen, Strukturen und Modulen. Während die Beziehungen zwischen den Bestandteilen im Anhang .2 visualisiert sind, werden im Folgenden die Funktionen und Zusammensetzung der einzelnen Bestandteile detailliert beschrieben. Sowohl in den UML-Diagrammen als auch an den Kapitelüberschriften befindet sich eine Kennzeichnung, ob der jeweilige Bestandteil im Vergleich zur von Gaul et al. veröffentlichten Implementierung [3]

- neu hinzugefügt
- wesentlich strukturell geändert
- teilweise modifiziert
- übernommen

wurde. Ignoriert werden bei dieser Kategorisierung geringfügige konzeptuelle oder strukturelle Änderungen wie Umbenennungen sowie geänderte Datentypen von Attributen und Variablen oder routinemäßig angewandte *Methodenextraktionen* (Auslagerung von sich

[1] https://gitlab2.informatik.uni-wuerzburg.de/s484411/dyndarp-simulation-with-probabilistic-travel-times

[2] Zu einem Teil der Entwicklungszeit (abhängig von der gerade verwendeten Entwicklungsumgebung) war das Code-Completion-Plugin *Github Copilot* aktiv, das auch auf ein Sprachmodell zurückgreift

[3] https://git.uni-wuppertal.de/dgaul/rolling-horizon-algorithm-for-dynamic-darp

wiederholenden Codesegmenten in separate Funktionen und Methoden). Sollte es nennenswerte Änderungen gegeben haben (teilweise modifiziert), erfolgt in jedem Fall eine Aufzählung der ausschlaggebenden Änderungen.
In der Erläuterung vollständig außen vor bleiben Klassen und Konstrukte für Konsolenrendering und -ausgabe.

3.1.1 Simulation (neu)

DARPSimulation (neu) In der Aufrufhierarchie an oberster Stelle steht die *DARPSimulation*-Klasse (die *main*-Funktion in einer separaten Datei ist hier außen vor gelassen). Das bedeutet, dass diese Klasse alle anderen Klassen, insbesondere also auch das (MILP-)Modell initialisiert und verwaltet. Sie verfügt über eine zentrale Schleife (Algorithmus 7) zur Abarbeitung aller Anfragen und Ereignisse. Speziell zu Simulationszwecken wurden noch drei andere Konstrukte geschaffen, die Klasse *RequestManager*, die alle Anfragen und deren Status verwaltet, die Struktur *ActionPlan*, die dem Aktionsplan Π aus 2.6 entspricht und daher nicht näher beschrieben wird, sowie die Klasse *TimingUnit*, die für die zeitliche Ansetzung und Realisierung von Kanten zuständig ist.

RequestManager (neu) Der *RequestManager* verfügt über sämtliche Anfragemengen aus Abschnitt 2.2.2 und führt bei Bedarf Verschiebungen von Anfragen innerhalb dieser Mengen durch. Insbesondere sind das die Einordnung angenommener und abgelehnter Passagiere in die Mengen $\mathcal{S}$ und $\mathcal{R}$, sowie aufgenommener und abgesetzter Passagiere in die Mengen $\mathcal{P}$ und $\mathcal{D}$.
Erste Aufgabe des *RequestManager* ist das Einfügen der Anfragen aus der *DARPInstance* in eine Prioritätswarteschlange *request_queue*, die nach dem Zeitpunkt des Eintreffens der Anfragen priorisiert wird. Da keine der verwendeten Instanzdateien Eintrittszeitpunkte spezifiziert, legen wir den Eintrittszeitpunkt für alle Anfragen $i \in \mathcal{A}$ stets auf den Zeitpunkt $e_{i^+} - 30$ Minuten, also 30 Minuten vor den Beginn des entsprechenden Einstiegszeitfensters fest.
Bei der Abfrage anstehender Anfragen $i \in \mathcal{A} \setminus \mathcal{A}(\tau)$, also dem Einfügen in die Menge $\mathcal{N}$, verwendet er zudem standardmäßig die Strategie, Anfragen dort in Bündeln von festlegbarer Größe ρ zu hinterlegen, um mehrere Anfragen gleichzeitig in das MILP-Modell integrieren zu können und dadurch effizientere Routen zu ermöglichen. Der Strategie liegt folgende Vorgehensweise zugrunde:

1. Neu eingetroffene, also zuletzt aus *request_queue* extrahierte Anfragen werden zunächst in eine temporäre Datenstruktur $\mathcal{N}^{next}$ eingefügt.

2. In $\mathcal{N}$ werden Anfragen erst eingefügt, wenn $\mathcal{N}^{next}$ mindestens ρ Elemente hat oder bei mindestens einer Anfrage $i \in \mathcal{N}^{next}$ die Mitte des Einstiegszeitfensters über-

schritten wurde, also die aktuelle Zeit $\tau \geq \frac{e_{i+}+l_{i+}}{2}$. Letzteres gilt der Vermeidung von Ablehnungen von Anfragen aufgrund zu später Bearbeitung.

TimingUnit (neu) Der „Zeitgeber“ der Simulation dient mehreren Zwecken. Zum einen aktualisiert und verarbeitet er die zu befahrenden und später die zu realisierenden Kanten (Algorithme 4, 5 und 6), zum anderen ist er auch für das Inkrementieren der Zeitpunkte τ_i zuständig, legt also die Abstände innerhalb der Zeitpunktliste $\mathcal{T}$ fest.

3.1.2 Instanzmodule und Modellspeicher (strukturell geändert)

Eine DARP-Instanz repräsentiert eine denkbare Umgebung eines Ridepoolingbetriebs. Eine Umgebung ist maßgeblich durch die örtlichen Gegebenheiten (Haltestellen und Distanzen zwischen ihnen), die Anzahl und Sitzplätze der zur Verfügung stehenden Fahrzeuge und allen voran durch die Anfragen und deren Anfragenparameter (siehe 2.2.1) gekennzeichnet.
Anzumerken ist, dass die Instanz als Hilfskonstrukt zur Ausführung einer Simulation zu verstehen ist und im realen Einsatz zumindest in dieser Form nicht vorhanden wäre. In diesem Fall sollte der *RequestManager* zunächst sämtliche Nutzerangaben entgegennehmen, aber alle für die Optimierung entscheidenden Daten dem *DARPModel* übergeben, das diese anschließend in einer dedizierten Datenstruktur zwischenspeichert und zudem über Ressourcen zur Abfrage von Reisezeiten zwischen Haltestellen verfügen sollte.
In unserem Fall wird eine zentrale Instanz aus zwei strukturierten Textdateien, der Dienst-/Anfragedatei und der Haltestellendatei gebildet.

Dienst-/Anfragedatei Bei den verwendeten Testinstanzen muss zwischen zwei Typen unterschieden werden: Zum einen liegt eine nativ dynamische Instanz[4] vor, die aus realen, anonymisierten Anfragedaten eines Wuppertaler Ridepooling-Dienstes[5] stammt, die aus den komprimierten Anfragen mehrerer Tage im Zeitraum Januar und Februar 2021 besteht. Der zweite Typ ist ursprünglich für das statische DARP entworfen worden und besitzt daher leicht abgewandelte Parameter. Hiervon sind zahlreiche Eingabedateien[6] vorhanden, welche ursprünglich von Cordeau et al. im Jahre 2006 [3] entworfen wurden, aber von Gaul et. al [8] verwendet wurden, um einen Performancevergleich mit anderen, exakten (statischen) DARP-Modellierungen zu ermöglichen.
Es ist im Übrigen möglich, diese statischen Instanzen als dynamisch zu behandeln, indem zum Beispiel der Eintreffzeitpunkt einer Anfrage $i \in \mathcal{A}$ basierend auf anderen Parameter

[4] https://git.uni-wuppertal.de/dgaul/rolling-horizon-algorithm-for-dynamic-darp/-/tree/main/data/WSW

[5] https://www.holmich-app.de/

[6] https://git.uni-wuppertal.de/dgaul/event-based-milp-for-darp/-/tree/main/data/a_b_first_line_modified

gesetzt wird, bevorzugt abhängig vom Beginn den Einstiegszeitfensters e_{i^+} der Anfrage i. Zu Beginn gibt jede Dienst-/Anfragedatei req_file die wichtigsten Instanzparameter vor:

- die Anzahl der DARP-Knoten, mithin die zweifache Anzahl der Anfragen
- die zur Verfügung stehenden Fahrzeuge (inklusive Sitzplatzanzahl)
- die Länge des Dienstbetriebs in Minuten
- (nur bei nativ statischen Instanzen) die generelle gültige maximale Reisezeit aller Passagiere

Im Anschluss folgt die Aufzählung aller DARP-Knoten. Für jeden DARP-Knoten ν gibt es dabei folgende Angaben:

1. DARP-Knotennummer, wobei ν_i^+ jeweils mit i nummeriert wird, und ν_i^- mit $n+i$
2. Servicezeit s_i des DARP-Knotens (wir gehen hier davon aus, dass diese für Ein- und Ausstieg übereinstimmt)
3. benötigte Sitzplätze q_i
4. Anfang und Ende des DARPKnoten-Zeitfensters $[e_\nu, l_\nu]$ - im Falle eines Zustiegsknotens entspricht dies $[e_{i^+}, l_{i^+}]$, im Falle eines Ausstiegsknotens $[e_{i^-}, l_{i^-}]$
5. (nur bei nativ dynamischen Instanzen) maximale Reisezeit L_i
6. (nur bei nativ statischen Instanzen) maximale Reisezeit x- und $y-$ Koordinaten des DARP-Knotens

Einstiege und Ausstiege einer Anfrage sind folglich separat spezifiziert, auch wenn es dabei redundante Angaben gibt. Es muss offensichtlich für die Anzahl der Sitzplätze gelten, dass diese sich für Einstieg und Ausstieg nicht unterscheiden dürfen.

Kantenkosten und -fahrzeiten Bei dynamischen Instanzen stellt die Datei $cost\_file$ unterdessen die zwischen den DARP-Knoten entstehenden Kantenkosten zur Verfügung, was folgender Menge entspricht: $\{c_{(\nu_i,\nu_j)} | \forall(\nu_i, \nu_j) \in N \times N\}$. Gleichzeitig sind damit auch die direkt proportionalen Kantenfahrzeiten t_e festgelegt. Bei statischen Instanzen werden die Kantenkosten und die proportionalen Fahrzeiten durch die euklidische Distanz berechnet.

3.1.2.1 Modellspeicher - *DARPState* und *DARPNodeState*

Zur sauberen Trennung zwischen den zuvor beschriebenen gegebenen Instanzdaten und dynamisch berechneten Daten verfügt das DARP-Modell über separate Datenstrukturen zur Speicherung von berechneten Daten mit Bezug zur Instanz (*DARPState*) und zu den DARP-Knoten (*DARPNodeState*). Zu den berechneten Daten gehören zum Beispiel Ankunftszeiten und Auslastungen von Fahrzeugen an DARP-Knoten, durchschnittliche Warte- und Fahrzeiten der Passagiere oder die aktuellen Routen aller eingesetzten Fahrzeuge, durch die mithilfe einer zweifach verketteten Liste iteriert werden kann.
Ungeachtet dessen greift auch das DARP-Modell weiterhin auf die DARP-Instanz- und die DARP-Knoten-Klasse zurück – immer wenn Anfrageparameter wie Zeitfenster benötigt werden. Unter dem Gesichtspunkt der Softwarearchitektur und der Praktikabilität wäre eine noch striktere Datenkapselung zwar denkbar, im praktischen Einsatz wäre eine derartige Instanz allerdings wie angemerkt grundsätzlich nicht vorhanden.

3.1.2.2 Vergleich mit Originalimplementierung

Die entsprechende ursprüngliche Klasse wurde nicht als Instanz bezeichnet, sondern stattdessen als konkrete Ausprägung eines Dial-a-Ride-Optimierungsproblems betrachtet, genannt *DARP*. Auch dynamisch berechnete Daten wurden hier und in der Klasse *DARPNode* abgelegt. Alle für diese Simulation relevanten Daten dieser Klasse aus der Klasse *DARP* wurden entweder in die Klasse *DARPInstance* (statisch) oder in die Klasse *DARPState* (dynamisch) übertragen. Analog wurde bei der Klasse *DARPNode*, die bereits existierte, verfahren.

3.1.3 Modell

DARPModel (neu, teilw. übernommene Funktionen) Die Klasse *DARPModel* ist als vorgelagerte Schicht wiederum für die Verwaltung aller MILP-Modellierungsbestandteile zuständig. Sie stellt eine Schnittstelle zur initialen (Algorithmus 3) und regelmäßigen (8) Lösung für die Simulation zur Verfügung. Dabei greift sie sowohl auf die Klassen für die Grapherstellung (*DARPGraph*), der Vorverarbeitung des aktuellen Lösungsraums (*DARPPreprocessor*) als auch das den Lösungsschritt abschließenden Bau des MILP zurück.

HashMaps (übernommen) Weiterhin verwaltet *DARPModel* Hashtabellen innerhalb der Struktur *HashMaps*, die hauptsächlich zur Referenzierung von Objekten gegenüber der CPLEX-C++-Schnittstelle notwendig sind. Hintergrund ist, dass alle Knoten und Kanten einem Index zugeordnet werden müssen, mithilfe dessen sie den Datenfeldern der Neben-

bedingungen (*IloRangeArrays*[7]) und Variablen (*IloNumVarArray* [8]) des MILP zugeordnet werden können. Diese Indizierung muss streng aufsteigend sein, da die Vergrößerung der Datenfelder ausschließlich explizit erfolgt. Realisiert wird dies mithilfe einer Hashfunktion, die den Knoten und Kanten des Graphen natürliche Zahlen zuweist.
Darüber hinaus erhalten auch die Anfragen eine separate Hashtabelle. Dies macht vor allem mit Blick auf den realen Einsatz Sinn, bei dem Anfragen selbst als komplexere Objekte modelliert werden könnten und sollten.

DARPPreprocessor (übernommen) Diese Klasse erfüllt einen für die Laufzeit ausschlaggebenden Zweck, indem sie vor der Erstellung des Graphen, wann immer dies bereits zu ermitteln ist, Passagierbedienungsreihenfolgen ausschließt, die mit deren Zeitfenstern nicht vereinbar werden.

Übernommene Suchheuristik des $DARPPreprocessors$ Die von Gaul et al. [7] verwendete Heuristik greift unter anderem auch auf die sogenannte 8-Schritt Auswertungsmethode zurück. Diese von Cordeau et al. [4] im Jahre 2003 für die statische Variante des DARP vorgestellte Methode zählt zu den Tabu-Metaheuristiken und ist gleichzeitig das am häufigsten adaptierte Verfahren zur effizienten Berechnung suboptimaler Lösungen für das allgemeine DARP. Die Laufzeit der ursprünglich vorgestellten Variante liegt innerhalb $\mathcal{O}(n^2)$, mittlerweile existieren allerdings Abwandlungen mit quasilinearer Zeitkomplexität ($\mathcal{O}(nlogQ)$, Sohrabi et al. [20]).

DARPGraph (übernommen) Diese Klasse erstellt über eine umfangreiche Routine alle Kombinationen aus Passagieren in Fahrzeugen, also alle Ereignisknoten v der Knotenmenge $V(G)$ des Graphen G unter Berücksichtigung der Ergebnisse der Vorverarbeitung des *DARPPreprocessor*. Alle zu diesen infrage kommenden Kanten $(u, v) \in E(G)$ werden im Anschluss daran erstellt.

3.1.3.1 Vergleich mit Originalimplementierung

Die Klasse *DARPModel* selbst wurde als neue oberste Modellierungsklasse vollständig neu geschaffen, lediglich Funktionen zur Verwaltung der Hashtabellen wurden übernommen. Die restlichen beiden Klassen *DARPPreprocessor* und *DARPGraph* decken sich hingegen mit der Originalimplementierung.

[7] https://www.ibm.com/docs/en/icos/20.1.0?topic=classes-ilorangearray
[8] https://www.ibm.com/docs/en/icos/20.1.0?topic=classes-ilonumvararray

3.1.4 MILP (mit neuem Aufbau übernommen)

Die Struktur *MILP* stellt die Implementierung des in Abschnitt 2.5 vorgestellten gemischtganzzahligen-linearen Programms dar. Diese enthält sämtliche Methoden, zur Initialisierung, Konstruktion und Aktualisierung des MILPs, besteht allerdings aus einigen Unterstrukturen. Die wichtigsten davon sind:

- **MILPConstraints:** Speicherort aller CPLEX-Nebenbedingungsobjekte
- **MILPObjective**: Speicherort aller CPLEX-Zielfunktionsobjekte
- **MILPSolution:** Speicherort aller CPLEX-Lösungsobjekte
- **MILPVariables:** Speicherort aller CPLEX-Variablen

Mit Kapitel 4 werden diese noch um weitere Bestandteile zur Verwaltung von *elastischen Nebenbedingungen* ergänzt.

3.1.4.1 Vergleich mit Originalimplementierung

Konzeptuell entspricht das MILP dem von Gaul et al konzipierten MILP. Lediglich Methoden zur Festschreibung und Realisierung von Kanten, die bei dieser Simulation notwendig sind, fehlten bisher. Die Struktur wurde allerdings modular neu aufgebaut.

3.2 Chronologie einer Simulation

Wir werden nun schrittweise den kompletten Ablauf einer Simulation erläutern. Zu beachten ist, dass der dargestellte Pseudocode selbstverständlich eine Vereinfachung der C++-Implementierung darstellt. Im Gegensatz zu den UML-Diagrammen sollen C++-spezifische Implementierungsdetails genauso wie laufzeitorientierte Umformungen wie die Verwendung von Referenzen oder Zeigern anstelle von Funktionsrückgaben im Pseudocode nicht sichtbar sein. Einige der einzeiligen Anweisungen sind zudem umfangreicher und wurden in separate Funktionen ausgelagert. Parallel dazu wird in 7 das Zwischenspiel zwischen Simulation und *DARPModel* visualisiert.

3.2.1 Initialisierung

3.2.1.1 Simulationsbeginn

Der Initialisierungsblock des ersten Simulationsdurchlaufes Algorithmus 1 stellt die Simulationsumgebung her. Der Hauptzweck ist das Erzeugen einer Instanz aus den beiden Eingabedateien $request\_file$ und $cost\_file$.

Algorithm 1 Erster Simulationsschritt

INPUT

request_file: Eingabedatei mit allen Anfragen; enthält Zeitfenster und Servicezeit
cost_file: Eingabedatei mit allen Kantengewichten
$\mathcal{T}$: Zeitlich geordnete Liste an zu betrachtenden Zeitpunkten

INIT

1: $\tau_j := \tau_0$
2: $instance \leftarrow$ parse_darp_instance($request\_file$, $cost\_file$)
3: $model \leftarrow new$ DARPModel($instance$)
4: $requestManager \leftarrow new$ RequestManager($instance$)
5: $timingUnit \leftarrow new$ TimingUnit($instance$)

MAIN

1: $\mathcal{N}(\tau_0) = requestManager$.retrieve_new_requests()
2: $\Pi(\tau_1) \leftarrow model$.solve_initial($\mathcal{N}(\tau_0)$)
3: $(\mathcal{S}(\tau_1), \mathcal{P}(\tau_1))$ = determine_accepted_requests($\Pi(\tau_1)$)
4: $T_0^{real} \leftarrow \tau_1$
5: update_appoint_queue(τ_1, AP)
6: $RQ(\tau_1) \leftarrow new$ PriorityQueue of values $(u, v) \in E$ ascendingly ordered by keys T_u

Zusätzlich müssen Modellierung (Klasse *DARPModel*), Anfrageverwaltung (Klasse *RequestManager*) und Zeitsetzung (Klasse *TimingUnit*) instanziiert werden. Letztere greift zur Initiierung auf die Instanz zurück und versteht sich fortan aber als eigenständiger Verwalter der Anfragen.

Die Abfrage der zum Zeitpunkt τ_0 bereits eingegangenen Anfragen $\mathcal{N}(\tau_0)$ aus der Anfragewarteschlange markiert gleichzeitig auch den Beginn der Simulation. Mit diesen Anfragen wird ein initialer Optimierungslauf (siehe Abschnitt 3.2.2) gestartet, welcher (wenn erfolgreich) einen Aktionsplan generiert.
Gemäß der Belegung von p_i wird anschließend jede Anfrage $i \in \mathcal{N}(\tau_0)$ entweder in die Menge $\mathcal{S}$ (akzeptiert) oder die Menge $\mathcal{R}$ (abgelehnt) verschoben. Weiterhin muss dem Depotknoten bereits hier ein realisierter Ereigniszeitpunkt zugewiesen werden, um eine zeitliche Realisierung der Depotnachfolgerknoten (siehe Abschnitt 3.2.3) zu ermöglichen.
Eine Prioritätswarteschlange AQ dient als temporäre Datenstruktur zur Abfrage der als Nächstes anstehenden Kanten - Kanten, die demnächst von einem Fahrzeug begonnen werden. Diese wird in Zeile 3 von einer separaten Methode ‘update_appoint_queue‘ (erklärt in Abschnitt 6) befüllt.
Eine weitere Prioritätswarteschlange RQ dient als permanente Datenstruktur von begonnenen Kanten, die primäre Funktion ist allerdings die Abfrage der demnächst anstehenden

Ereignisse, welche den Zielknoten der inkludierten Kanten entsprechen. Die Sortierung beruht hier daher auf dem Zeitpunkt des Zielereignisknotens T_v^{real}. Diese Prioritätswarteschlange wird erst im späteren Simulationsablauf durch die *TimingUnit* befüllt, sobald die ersten Fahrzeuge gestartet sind.

3.2.2 Initiale MILP-Modellierung

Der Konstruktoraufruf des *DARPPreprocessor* in Zeile 2 veranlasst die Initialisierung der f-Funktion (siehe 2.3.3.1). Der *DARPPreprocessor* wird in Zeile 3 erstmals (siehe Abschnitt 3.1.3) mögliche Ein- und Aussteigereihenfolgen hinsichtlich Realisierbarkeit überprüfen. Darauf aufbauend wird ein erster Ereignisgraph $G(\tau_0)$ konstruiert. Ziel dieses Ablaufs ist es, keine Knoten und Kanten zu integrieren, die aufgrund von nicht einhaltbaren Zeitfenstern bereits als nicht realisierbar bekannt sind.

Algorithm 2 solve_initial($\mathcal{N}$): **returns** Π

1: *state* ← *new* DARPState($\mathcal{N}(\tau_0)$)
2: *solver* ← *new* DARPPreprocessor(*instance*,*state*)
3: (f^1, f^2) ← *solver*.check_paths()
4: $G(\tau_0)$ ← build_initial_graph(*instance*,*state*, f^1, f^2)
5: MILPMaps *maps* ← create_maps()
6:
7: MILP $milp(\tau_0)$ ← *new* MILP(*instance*, *state*, G, *maps*)
8: $milp(\tau_0)$.init_MILPVariables()
9: $milp(\tau_0)$.init_MILPSolution()
10: $milp(\tau_0)$.init_MILPObjectives()
//Startzeit τ_0 legt an dieser Stelle frühestmögliche Abfahrt vom Depot fest
11: $milp(\tau_0)$.init_MILPConstraints(τ_0)
12:
13: **if** *cplex*.solve($milp(\tau_0)$.model) **then**
14: $milp(\tau_0)$.process_solution()
15: $\Pi(\tau_0)$ ← generate_action_plan(*solution*) *//Durch Ablehnung verwaiste Kanten und Knoten entfernen*
16: erase_denied($\mathcal{R}(\tau_0)$)
17: **return** $\Pi(\tau_0)$
18: **end if**
19: (Ausnahmebehandlung für nicht lösbares MILP)

Ein ausnahmsweise hervorgehobenes Implementierungsdetail ist die Erstellung mehrerer Hashtabellen für alle Graphbestandteile in Zeile 5, deren Funktionsweise in 3.1.3 erläutert wurde.
Das in Zeile 7 initialisierte *MILP* benötigt alle in den vorigen Schritten erzeugten Objekte als Referenz, die unabhängig von der zeitlichen Referenz τ_0 bestehen bleiben. Nicht dargestellt sind mögliche Parameter zur Lösung des MILPs. Die Initialisierung der Variablen, Lösungs-, Zielfunktions- und Nebenbedingungsobjekte bezieht sich (ausnahmsweise) auf die Arbeitsweise von *ILOG-CPLEX Concert* und wird hier nicht näher erläutert. Das in Kapitel 2 formulierte gemischt-ganzzahlige-lineare Programm soll hier als Erläuterung dieses Aspekts genügen. Nichtsdestotrotz sind diese nicht unerheblichen Objekte Teil des UML-Diagramms(.2).
Das den Abschluss bildende erase_denied() in Zeile 16 wird im Zusammenhang mit weiteren Bereinigungsoperationen in 3.2.4.2 behandelt.

Algorithm 3 process_solution($\mathcal{N}$): returns Π

1: $X \leftarrow cplex$.getValue($x_a \forall a \in A(\tau)$)
2: $T \leftarrow cplex$.getValue($T_v \forall v \in V(\tau)$)
3: $P \leftarrow cplex$.getValue($p_i \forall i \in \mathcal{N}(\tau)$)
4: $D \leftarrow cplex$.getValue($d_i \forall i \in \mathcal{A}i : p_i = 1$)
5: traverse_routes()
6: **for** $i \in \mathcal{S}$ **do**
7: $\quad \gamma_i^+ \leftarrow T_v$ **where** $v \in V_{i^+} : \sum_{e \in \delta^{in}(v)} x_a = 1$
8: $\quad \gamma_i^- \leftarrow T_v$ **where** $v \in V_{i^-} : \sum_{e \in \delta^{in}(v)} x_a = 1$
$\quad$ *//Nach MILP-Nebenbedingung 2 trifft beides genau auf einen Knoten v zu*
9: **end for**

Umgang mit der Lösung Beim Verarbeiten der Lösung mittels process_solution() für alle vorher definierten Lösungsobjekte gibt es hingegen noch einige erwähnenswerte Besonderheiten. Das Lösungsobjekt

$$MILPSolution \sim (X, T, P, D) \tag{3.1}$$

entspricht im Wesentlichen dem späteren Aktionsplan – mit den Vektoren X für die Entscheidungsvariablen, T für die Eintrittszeitpunkte, P für die akzeptierten Anfragen und D für die überschüssige Reisezeit. Die Trennung zwischen Simulation und Modellierung macht es erforderlich, – insbesondere mit Blick auf eine in der Praxis parallele Ausführung der beiden Komponenten – zwischen den beiden Strukturen zu unterscheiden. Anders als beim Aktionsplan wird hier daher auch auf Vektoren zur Repräsentation der Lösungs-

bestandteile zurückgegriffen und auf eine direkte zeitliche Referenzierung verzichtet, die direkt aus dem τ-Parameter des MILP-Objekts folgt.

Weiterhin gilt es, aus den generierten Knoten Routen der Fahrzeuge zu generieren, um anschließend eine Visualisierung durch die Simulation zu ermöglichen. Die Generierung von Routen folgt einem vergleichsweise umfangreichen Algorithmus, der hier aber nicht näher erläutert werden soll. Das Resultat dieses Algorithmus' sind alle Kreise innerhalb der Menge der durch die Entscheidungsvariable gesetzten Kanten.

Der nennenswerteste Aspekt ist schließlich die einmalige Festlegung der kommunizierten Abholzeit Γ_i für alle akzeptierten Passagiere $i \in \mathcal{N}(\tau)$, für die also p_i gesetzt ist und die anschließend in die Menge $\mathcal{S}$ zu übertragen sind. Diese wird demnach unmittelbar aus der ersten Lösung des ersten MILPs generiert, das diesen Passagier berücksichtigt hat.

3.2.3 Modellierung der Kantenrealisierung

Den Kern der Simulation bildet die *TimingUnit* mit den beiden Algorithmen 4 und 5: die Ansetzung und Realisierung von Kanten, also dem Beginn und Abschluss einer Fahrt eines Fahrzeugs zwischen zwei aufeinanderfolgenden Ereignissen. .
Diese Prozedur könnte grundsätzlich vielfältig umgesetzt werden. Im realen Einsatz würde diese womöglich über getrennte *Client*-Geräte innerhalb der Fahrzeuge mit den realen Ankunftszeitpunkten an den zugehörigen Haltestellen befüllt werden, sobald diese erreicht werden.

In einer auf höchstmögliche Realitätstreue ausgelegten Modellierung würde die tatsächliche Fahrzeit $T^{real}_{(u,v)}$ einer Kante (u, v) erst mit dem Erreichen des Zielknotens v feststehen. Das würde allerdings voraussetzen, dass in möglichst geringen regelmäßigen Abständen τ_Δ mithilfe der Wahrscheinlichkeitsdichte der verwendeten Wahrscheinlichkeitsverteilung für alle anstehenden Ereignisse jeweils eine Zufallsvariable $X_v(\tau, \tau_\Delta)$ generiert wird, die die Realisierung einer (gerade befahrenen) Kante (u, v) innerhalb des vergangenen Zeitabschnitts τ_Δ angibt:

$$X_v(\tau, \tau_\Delta) = \begin{cases} 1 & \text{Ereignisknoten } v \text{ im Zeitraum } [\tau - \tau_\Delta, \tau] \text{ erreicht} \\ 0 & \text{andernfalls} \end{cases} \tag{3.2}$$

Der im Folgenden vorgestellte Ansatz ist hinsichtlich seiner Berechnungsabfolge nicht vollständig realitätsgetreu, liefert erwartungsgemäß aber gleiche Ergebnisse bei deutlich einfacherer und performanterer Implementierung.

3.2.3.1 Kantenfestsetzung

Im vorgelagerten Schritt muss auf einen Mechanismus zur Auswahl der zunächst „anfallenden“ Kanten zurückgegriffen werden, die vorher definierte Prioritätswarteschlange AQ. Das über 'retrieveMin()' abgefragte Element wird stets das mit dem frühesten noch nicht realisierten Zielereignis v sein. Das Anfangsereignis u eines Elements in dieser Warteschlange muss im Allgemeinen bereits in einer der vorigen Iterationen über T_u^{real} zeitlich festgelegt worden sein, liegt aber womöglich noch in der Zukunft. Der Ereigniszeitpunkt des Depotknotens wurde bereits in der Initialisierung festgelegt und liegt abhängig vom Fortschritt der Hauptschleife bereits in der Vergangenheit. Zu beachten ist, dass x_u^{real} hier gesetzt wird und nicht bei der algorithmischen Realisierung der Vorgängerkante $(w,u) \in A(G) mit x_{(w,u)}^{real} = 1$ innerhalb Algorithmus 5. Dies liegt daran, dass das Fahrzeug bei u möglicherweise warten musste, um einen Passagier aufnehmen zu können. In diesem Fall darf dem *DARPModel* die Realisierung von u erst mitgeteilt werden, wenn dieser Fall eingetreten ist, andernfalls könnte es zu Problem mit der Fahrzeitnebenbedingung 6 in Bezug auf die verwendete M-Konstante kommen.

Algorithm 4 appoint_started_edges(τ_j, AP)

1: $(u,v) \leftarrow AQ(\tau_{j-1})$.retrieveMin()
2: bool $changes \leftarrow$ FALSE
3: **while** !$AQ(\tau_{j-1})$.empty() **and** $T_u^{real} \leq \tau_j$ **do**
4: AQ.pop((u,v))
5: $x_u^{real} = 1$
6: $T_{(u,v)}^{real} \sim$ generate_travel_time(u,v)
7: $i^{+|-} := v_1$
8: $T_{(v)}^{arrival} \leftarrow T_u^{real} + s_i + T_{(u,v)}^{real}$
9: RQ.insert($key = T_v^{fix}, value = (u,v)$)
10: $(u,v) \leftarrow AQ(\tau_{i-1})$.retrieveMin()
11: **end while**

Generierung von Fahrzeiten In Bezug auf die Einleitung des Abschnitts 3.2.3 muss darauf hingewiesen werden, dass die Fahrzeit $T_{(u,v)}^{real}$, also die tatsächlich realisierte Fahrzeit vom Knoten u zum Knoten v, generiert wird, sobald der Simulation bekannt ist, dass die Kante (u,v) begonnen wurde.
Daher wird genau dann, wenn das Anfangsereignis u der Kante bereits in der Vergangenheit liegt, T_u^{real} also kleiner als der aktuell betrachtete Zeitpunkt τ_j ist, die Kante über die Pop-Operation aus der Warteschlange AQ entfernt. Die Fahrzeit wird von einer durch eine Funktion generate_travel_time berechnet. Diese Funktion könnte im Allgemeinen beliebig

implementiert werden, von der bloßen Verwendung der vorberechneten Fahrzeiten t_a für $a \in A(G)$, dazu proportionale Fahrzeiten bishin zur Verwendung von Wahrscheinlichkeitsverteilungen. Unter der Annahme, dass die Gammaverteilung Fahrzeitabweichungen realistisch widerspiegelt (siehe Kapitel 4.2.4.1), werden wir die Kantenfahrzeit später als lognormalverteilte Zufallsvariable modellieren.

Festschreibung des Zielereigniszeitpunkts T_v^{real} Grundsätzlich besteht gemäß Formulierung des MILP (Nebenbedingung 6) zwischen den Ankunftszeiten T_u und T_v zweier aufeinanderfolgenden Knoten $u, v \in V(G)$ ein „Mindestabstand“ von der Servicezeit s_u am Knoten u und der vorberechneten Fahrzeit $t_{(u,v)}$ zwischen den beiden Ereignisorten. Daraus folgt allerdings nicht, dass die vom Solver generierten Werte T_u und T_v tatsächlich genau diese Differenz aufweisen. In vielen Fällen existiert zusätzlich ein Puffer $\Delta_{(u,v)} > 0$, sodass:

$$T_v = T_u + s_u + t_{(u,v)} + \Delta_{(u,v)} \tag{3.3}$$

Für die Integration probabilistischer Fahrzeiten ist dies insofern relevant, als dass Verspätungen zur vollständigen Konfliktvermeidung hinsichtlich der Lösung des MILP nur dann eine Auswirkung auf den realisierten Ereigniszeitpunkt T_v^{real} haben sollten, wenn $T_{(u,v)}^{real} > t_{(u,v)} + \Delta_{(u,v)}$ oder es sich bei v um einen Ausstiegsknoten handelt. Andernfalls könnte das Einstiegszeitfenster der v zusteigenden Passagiere verletzt werden oder aber die maximale Reisezeit dieser zu stark ansteigen, was wiederum zu einer Verletzung der Maximalreisezeit dieser Passagiere führen würde. Die Problematik der zeitlichen Abstimmung zwischen Realität beziehungsweise Simulation in unserem Fall und Modell wird in Abschnitt 3.2.3.3 behandelt.
Für die deterministischen Tests im Folgekapitel werden wir aus diesem Grund für Einstiegsknoten als Zielereigniszeitpunkt (Dienstbeginn am Zielknoten) den Mindestwert T_v vorgeben.

3.2.3.2 Kantenrealisierung

Algorithm 5 realize_appointed_edges(τ_j, AP, T^{fix}): **returns** bool

1: $(u, v) \leftarrow RQ(\tau_{i-1})$.retrieveMin()
2: bool *changes* $\leftarrow$ FALSE
3: **while** !$RQ(\tau_{j-1})$.empty() **and** $T_u^{fix} \leq \tau_j$ **do**
4: RQ.pop((u, v))
5: **if** $v = v_0$ **then**
6: **continue**
7: **end if**
8: *changes* $\leftarrow$ TRUE
9: $x_{(u,v)}^{real} \leftarrow 1$
10: $x_u^{real} \leftarrow 1$
11: **if** $v \in V_i^+$ **then**
12: $T_{(v)}^{real} \leftarrow \max(T_v, T_{fix})$
13: **else**
14: $T_{(v)}^{real} \leftarrow T_{(v)}^{fix}$
15: **end if**//*Zu Ereignisknoten gehörigen DARP-Knoten ermitteln (gemäß Beob. 1)*
16: $\nu \leftarrow$ find($i^{+|-} := v_1$)
17: **if** $v \in V_{i^+}$ **then**
18: $\mathcal{S}(\tau_i)$.remove(i)
19: $\mathcal{P}(\tau_i)$.insert(i)
20: **else if** $v \in V_{i^-}$ **then**
21: $\mathcal{P}(\tau_i)$.remove(i)
22: $\mathcal{D}(\tau_i)$.insert(i)
23: **end if**
24: $a^{active}(\nu) \leftarrow (u, v)$
25: $v^{active}(\nu) \leftarrow v$
26: $(u, v) \leftarrow RQ(\tau_{i-1})$.retrieveMin()
27: **end while**

Die Iteration durch die Prioritätswarteschlange RQ in Algorithmus 5 ist fast identisch mit der Iteration durch AQ in Algorithmus 4. Entscheidender Unterschied ist aber, dass die Realisierung der Kante erst mit dem Erreichen des Endknotens v erfolgt. Wir werden also nur alle Kanten aus RQ entfernen, deren Zielknoten zum Zeitpunkt τ_j schon erreicht ist. Sofern mindestens eine Kante entnommen wurde, wird durch das Setzen des *changes*-Flag signalisiert, dass ein neues Ereignis eingetreten ist, was automatisch eine neue Modellierung des MILPs im Anschluss an den Aufruf dieser Methode zur Folge haben

wird.

Aktualisierung der Passagiermengen und aktiven Knoten und Kanten Zeile 10 bis 21 behandeln die zwingenden Konsequenzen der Realisierung der Kante (u, v). Nach der Abfrage des signifikanten Passagiers des Zielknotens in Zeile 10 muss geprüft werden, ob es sich um einen Abholknoten $\in V_{i^+}$ (Zeile 11) oder einen Absetzknoten $\in V_{i^-}$ (Zeile 6) handelt. Abhängig davon muss er entweder von der Menge der aktuell nur eingeplanten Passagiere $S(\tau_j)$ in die Menge der aktuell aufgenommenen Passagiere $P(\tau_j)$ transferiert werden oder von der Menge der aktuell aufgenommenen Passagiere in die Menge der bereits abgesetzten Passagiere $D(\tau_j)$.
Um die Bereinigung 3.2.4.2 des Modells zu ermöglichen, muss nun außerdem die Realisierung genau dieser Knoten und Kanten des Graphen in Verbindung mit den beiden DARP-Knoten ν_i^+ und ν_i^- des Passagiers gebracht werden. Spätestens hier steht fest, dass die Kante (u, v) und genau dieser Zielknoten v vom Fahrzeug gewählt wurden, um einen Passagier i am Zielknoten aufzunehmen oder herauszulassen. Analog dazu, dass jeder Passagier über genau einen Abhol- und Absetz-DARP-Knoten verfügt, kann nun ein Abholzielknoten als $v^{active}(\nu_i^+)$ festgelegt und ein Absetzzielknoten als $v^{active}(\nu_i^-)$ gesetzt werden. Allerdings ist es auch notwendig, die zum Abholen oder Absetzen führende Kante eindeutig mit den beiden Ereignissen zu verknüpfen, indem diese nun als $a^{active}(\nu_i^+)$ bzw. $a^{active}(\nu_i^-)$ gesetzt wird.
Das Ende des Schleifendurchlaufs markiert die Abfrage des nächst-priorisierten Elements, das im Bedarfsfall im folgenden Durchlauf verarbeitet wird.

3.2.3.3 Vereinbarung von Kantenfestschreibung, -realisierung und Anfrageverarbeitung

Realisierung der Zielereignisse und Festschreibung von Kanten zu trennen, machen eine sorgfältige Abstimmung der Simulationsereignisse und der parallelen Verarbeitung neuer Anfragen durch das Modell notwendig.

Festschreibung von Kanten Es ist einerseits notwendig, dem DARP-Modell zu übermitteln, dass begonnene Kanten in jedem Fall abgeschlossen werden, die Entscheidungsvariable x_a also für diese Kanten $a \in A$ fixiert ist.
Zu diesem Zweck wurden auch die Mengen $\mathcal{S}^{fix}$ und $\mathcal{P}^{fix}$ definiert. Mit diesen lässt sich ausschließen, dass keine Knoten und Kanten erstellt werden, die nicht mehr infrage kommen. Dies geschieht vor dem Hintergrund, dass der *DARPGraph* mögliche $[1..Q]$-Kombinationen aller neuen oder noch nicht aufgenommenen Passagiere $i \in \mathcal{S}$ sowie aller bereits in einem Fahrzeug befindlichen Passagiere $i \in \mathcal{P}$ generiert. Daraus resultieren

wiederum Kanten in einer höheren Größenordnung und für beides jeweils mehrere Nebenbedingungen.
Bei Passagieren $i \in \mathcal{S}^{fix}$, deren Aufnahmeknoten bereits angesteuert wird, oder Passagieren $i \in \mathcal{P}^{fix}$, deren Absetzknoten bereits angesteuert wird, soll dies nun verhindert werden. Dies ist zwar nicht zwingend erforderlich, reduziert die Modellgröße und Laufzeit allerdings - insbesondere bei längeren Kantenfahrzeiten.
Nicht zu vergessen sind die analog dazu bereits existierenden Knoten und Kanten des Graphen, die nicht mehr benötigt werden (Abschnitt 3.2.4.2).

Ermittlung der Abfahrtszeit eines Fahrzeugs an einem Knoten $v \in V$ Hier sind mögliche Ungereimtheiten zwischen Simulation und Modellierung zu beachten.
Denkbar sind beispielsweise folgende Situationen:

1. Ein Fahrzeug ist gerade (zum Zeitpunkt τ) am Einstiegsknoten $v^{active}(i^+) \in V_{i^+}$ des Passagiers $i \in \mathcal{S}^{fix}$ eingetroffen, allerdings weit vor dem Beginn e_{i^+} dessen Zeitfensters. Gemäß der Methode 'realize_appointed_edges' wurde der realisierte Dienstbeginn T_v^{real} daher auf den Wert e_{i^+} gesetzt - der allerdings noch weit in der Zukunft liegt. Wir müssen davon ausgehen, dass das Fahrzeug nach der Aufnahme des Passagiers und der Servicezeit s_i direkt zum Folgeknoten fährt.

2. Ein Fahrzeug ist gerade (zum Zeitpunkt τ) am Ausstiegsknoten $v^{active}(j^-) \in V_{j^-}$ des Passagiers $j \in \mathcal{S}^{fix}$ eingetroffen. Gemäß der Methode 'realize_appointed_edges' wurde der realisierte Dienstbeginn daher auf den Wert τ gesetzt, der möglicherweise auch vor dem Beginn e_{j^+} des Einstiegszeitfensters liegt. Es liegen allerdings aktuell keine weiteren Anfragen vor, für die dieses Fahrzeug eingeplant wurde. Daher wartet dieses Fahrzeug zunächst am Knoten. Zahlreiche Simulationsschritte später wird das Fahrzeug schließlich mit einer weiteren Anfrage $j \in \mathcal{S}^{fix}$) betreut, zu deren Abholpunkt es alsbald aufbricht.

In beiden Fällen ist es notwendig, dem *DARPModel* über das Setzen x^{real} für den entsprechenden Ereignisknoten zu übermitteln, dass dieser erreicht wurde - der Dienstbeginn möglicherweise aber noch in der Zukunft liegt. Für das Ansetzen von Kanten über die Prioritätswarteschlange AQ ist aber lediglich die Abfahrzeit von den Knoten $v^{active}(j^-)$ und $v^{active}(i^+)$ entscheidend, welche hier unterschiedlich zu behandeln ist.

Aktualisierung der Warteschlange AQ Die Prioritätswarteschlange AQ wird zu Beginn der Methode stets neu initialisiert, da gegebenenfalls Kanten aus vorigen Iterationen nicht mehr zu verwenden sind, sofern diese noch nicht begonnen wurde (wird von Modellierung sichergestellt).

Algorithm 6 update_appoint_queue(τ, AP)

1: $AQ(\tau) \leftarrow new$ PriorityQueue of **values** $(u, v) \in E$ ascendingly ordered by **keys** $T_u \in \mathbb{Q}$
2: **for all** $(u, v) \in A(G)(\tau)$ **do**
3: **if** $\Pi(\tau).x_{(u,v)} = 1$ **and not** $x_{(u,v)}^{fixed}$ **and** $v \neq v_0$ **then**
4: **if** $u = v_0$ **then**
5: $T_0^k.add(\tau)$
6: $AQ(\tau)$.insert$(key = \tau, value = (u, v))$
7: **else if** x_u^{real} **then**
8: $AQ(\tau)$.insert$(key = T_u^{real}), value = (u, v))$
9: **else**
10: $AQ(\tau)$.insert$(key = AP.T_u), value = (u, v))$
11: **end if**
12: **end if**
13: **end for**

Eine umschließende Bedingung in Zeile 3 stellt sicher, dass nur Kanten $a \in A(G)$ berücksichtigt werden, die einerseits im Aktionsplan Π als zu verwenden indiziert sind (zugehörige Entscheidungsvariable x_a gesetzt), andererseits allerdings noch nicht festgeschrieben und von einem Fahrzeug begonnen wurden. Letzteres erfasst die Binärvariable $x_{(u,v)}^{fixed}$.
Anschließend muss der *Ansetzzeitpunkt* der Kante $(u, v) \in A$ ermittelt werden. Dieser erfasst den Zeitpunkt, an dem ein Fahrzeug von Knoten u in Richtung v aufbrechen soll.

1. **Es handelt sich um eine am Depotknoten beginnende Kante:** Da die Kante nach der Eingangsbedingung noch nicht festgeschrieben ist, kann es sich dabei nicht um das erste startende Fahrzeug handeln, welches zum Zeitpunkt $T_0 := \tau_1$ gemäß 1 gestartet ist. Daher wird über das Einfügen in die Menge T_0^k der Startzeitpunkt eines weiteren Fahrzeugs $k \in K$ festgelegt, der anschließend auch als Ansetzzeitpunkt der Kante (u, v) in die Warteschlange AQ aufgenommen wird. Wir gehen hier davon aus, dass das bereitstehende Fahrzeug sofort, also zum Zeitpunkt τ des Ausführens der Methode, startet.

2. **Der Startknoten u ist bereits realisiert** (Umsetzung erläutert in Abschnitt 5): In diesem Fall wird die Kante auf das Maximum des dortigen Dienstbeginns T_u^{real} und des aktuellen Zeitpunkts τ gesetzt.

3. **Der Startknoten u wurde noch nicht erreicht:** Die Kante wird mit dem vom Modell berechneten Zeitpunkt T_u des Dienstbeginns am Knoten u als Schlüssel eingefügt. Bei einer zuverlässigen, sehr eng getakteten Abstimmung zwischen Simulation und Modellierung und Rechenzeit für das MILP, die stets unter der Servicezeit aller Knoten verbleiben, wäre die Betrachtung dieses Falls nicht zwingend erforderlich.

3.2.4 Regelmäßige Simulations- und Modellierungsschritte

3.2.4.1 Hauptschleife

Algorithm 7 Hauptprozedur der Simulation

1: **for** $\tau_j \in \mathcal{T}$ **do**
2: $changes \leftarrow FALSE$
3: $A^{new\_fixed}(\tau)$.insert($timingUnit$.appoint_started_edges(τ_j)
4: $changes \leftarrow timingUnit$.realize_appointed_edges(τ_j)
5: $\mathcal{N}(\tau_j) = requestManager$.retrieve_new_requests()
6: **if not** $\mathcal{N}(\tau_j)$.empty() **then**
7: $changes \leftarrow$ TRUE
8: **end if**
9: **if** $changes$ **then**
10: $\Pi(\tau_j) \leftarrow model$.solve($A^{new\_fixed}(\tau), x^{real}, T^{real}, \mathcal{N}(\tau_j), \mathcal{P}(\tau_j), \mathcal{D}(\tau_j)$)
11: ($\mathcal{S}(\tau_i)$, $\mathcal{P}(\tau_i)$) = determine_accepted_requests($\Pi(\tau_i)$)
12: update_appoint_queue(τ_i, AP)
13: $A^{new\_fixed}(\tau)$.clear()
14: **end if**
15: **end for**

Wir iterieren in der Hauptprozedur der Simulation - Algorithmus 7 - nun durch die Liste aller vorher definierten Zeitpunkte. Die Zeitpunktliste ist bewusst flexibel gehalten, um auch einen Vergleich mit der ursprünglichen Lösung von Gaul et al. zu ermöglichen. Mit Zeile 3 werden gegebenenfalls begonnene Kanten angesetzt, ein Vorgang, der zusammen mit der Realisierung von Kanten (Zeile 4) in Abschnitt 3.2.3 erläutert wird.
Sofern es weder neu realisierte Kanten noch neue Anfragen (Zeile 5) gibt, wäre ein erneutes Lösen eine unnötige Verzögerung des Ablaufs. Daher wird nur im Falle eines neuen Ereignisses (hier nicht als Ereignis*knoten* zu verstehen, sondern als bisher unbekannte Information) eine erneute Modellierung und Lösung vorgenommen.
Da im Gegensatz zur initialen Lösung nun nicht nur neue Anfragen, sondern gegebenenfalls auch Knotenereigniszeitpunkte und damit gleichzeitig Änderungen der Passagiermengen aufgetreten sind, unterscheidet sich der Aufruf der Modellierungeinheit von dem innerhalb 1. Die Methode 'DARPModell::solve()' verfügt daher nicht nur über andere Parameter, sondern erfüllt auch einen anderen Zweck (vgl. 3.2.4.2) als die Methode 'DARPModell::solve_initial()'.
Unverändert bleibt hingegen das Befüllen der Prioritätswarteschlange AQ über die Methode 'update_appoint_queue' auf Basis der Lösung des MILPs, das sich über den letzten Abschnitt der einzelnen Schleifendurchläufe erstreckt.

3.2.4.2 Regelmäßiger Modellierungsschritt (Algorithmus 8)

Algorithm 8

solve($\tau, A^{new\_fixed}(\tau), A^{new\_real}(\tau), T^{real}, V^{active}, A^{active}, \mathcal{N}(\tau), \mathcal{P}(\tau), \mathcal{D}(\tau)$) : **returns** Π

1: *state*.insert_new_requests($\mathcal{N}(\tau)$)
2: *milp*(τ).process_fixed_edges($A^{new\_fixed}(\tau)$)
3: *milp*(τ).process_realized_edges($A^{new\_real}(\tau), T^{real}$) *//Durch festgelegte Abhol- und Absetzereignisse verwaiste Knoten, Kanten und MILP-Elemente entfernen bzw. festsetzen*
4: clean_model($A^{new\_fixed}, V^{active}, A^{active}$)
5:
6: $(f^1, f^2) \leftarrow$ *solver*.check_new_paths($\mathcal{N}(\tau), \mathcal{S}(\tau), \omega_a, \omega_d$)
7: $G(\tau)$.update_graph($f^{1|2}, \mathcal{N}(\tau), \mathcal{S}(\tau)$)
8: *maps*.update($\mathcal{N}(\tau)$)
9:
10: *milp*(τ).update_MILPVariables()
11: *milp*(τ).update_MILPObjectives()
12: *milp*(τ).update_MILPConstraints(τ)
13:
14: **if** *milp*(τ).solve() **then**
15: $\Pi(\tau) \leftarrow$ *milp*(τ).process_solution(τ_j)
16: erase_denied($\mathcal{R}(\tau)$)
17: **return** $\Pi(\tau)$
18: **end if**
19: (Ausnahmebehandlung für nicht lösbares MILP)

Nach dem Einfügen neuer Anfragen in Zeile 1, muss der Ereignisgraph $G(V(\tau), E(\tau))$ bereinigt werden. Andernfalls würden bei der Aktualisierung des Graphen zahlreiche nicht mehr realisierbare Knoten und Kanten entstehen, die sich auf verwaiste Knoten beziehen. Dieser umfangreiche Prozess wird in Abschnitt 3.2.4.2 behandelt.

Fixierung und Festschreibung von Kanten Eine besondere Bedeutung für die Simulation kommen den Aufrufen von process_fixed_edges(A^{new_fixed}) und process_realized_edges(A^{new_real}) in Zeile 2 und 3 zu. Die erste Methode fixiert die Entscheidungsvariable x_a für alle im Vorfeld dieses Modellierungsschritts (und nach dem vorigen Schritt) über Algorithmus 4 neu festgeschriebenen Kanten $a \in A^{new\_fixed}$ über das Einfügen einer neuen Nebenbedingung:

$$x_a = 1 \forall a \in A^{new\_fixed} \Leftrightarrow (x_a \leq 1 \wedge x_a \geq 1) \forall a \in A^{new\_fixed}$$

Zur Vollständigkeit ist hier auch die dem Solver zu übergebende explizite, aber äquivalente Nebenbedingung aufgeführt. process_realized_edges(A^{new_real}) schreibt hingegen den Ereigniszeitpunkt für alle neu realisierten Kanten fest:

$$T_v \leq T_v^{real} + \epsilon \quad \forall (u,v) \in A^{new\_real}$$
$$T_v \geq T_v^{real} - \epsilon \quad \forall (u,v) \in A^{new\_real}$$

Die Konstante ϵ kann Ungenauigkeiten bei der Lösung durch den Solver ausgleichen, die insbesondere bei der Verwendung von Gleitkommazahlen auftreten können. Je nach gewünschter zeitlicher Präzision und verwendetem Solver sowie der Rechnerarchitektur kann deren Wert unterschiedlich zu setzen sein; wir verwenden den Wert 10^{-7}.
Zusätzlich werden wir innerhalb dieser Methode grundsätzlich sicherstellen, dass keine Konflikte durch die vorgestellten Simulationsarbeitsweise (im speziellen Algorithmus 5) sowie Fahrzeiten, die kürzer als erwartet ausfielen, auftreten. Dies spielt insbesondere in Bezug auf die Untersuchungen im Folgekapitel eine Rolle. Wir werden nun für Kante $(u,v) \in A^{new\_real}$ jeweils ein oder zwei Nebenbedingungen entfernen, die unter den soeben genannten Bedingungen jederzeit verletzt werden könnten:

- die Fahrzeitnebenbedingungen (Nebenbed. 6)
- (falls es sich um einen Ausstiegsknoten handelt) der Beginn des Ausstiegszeitfensters (Nebenbed. 1) mit $i^- := v(1)$

Erstere würden mit der vorgestellten Modellierungsweise keine Fahrzeiten T_e^{real} akzeptieren, die kürzer als die erwartete Fahrzeiten t_e ist. Zweitere könnte grundsätzlich immer dann verletzt werden, wenn beim errechneten Zweitpunkt des Dienstbeginns T_v am Absetzknoten $v = v^{active}(\nu)$ eines Passagiers $i \in \mathcal{A}$ ein Puffer vorliegt (siehe 3.2.3.1) - insbesondere also dann, wenn die tatsächliche Fahrzeit T_e^{real} bei der hinführenden Kante $e = e^{active}(i)$ kleiner als T_v ausfällt.
Im Rahmen des Folgekapitels (für beliebige Fahrzeitabweichungen) werden die Vorkehrungen innerhalb dieser Funktion um einige weitere Aspekte erweitert.

Bereinigung des Modells Für die Modellierung nicht zwingend notwendig, für den Speicherbedarf und die Rechenzeit aber äußerst wirkungsvoll ist die Bereinigung des Ereignisgraphen $G(V,A)$ und des gemischt-ganzzahlig-linearen Programms. Auf Basis der aktiven Knoten und Kanten, die anschließend jeweils eindeutig mit einem Passagier und DARPKnoten verknüpft ν_i sind, können *verwaiste* Knoten $v \in V^{alt}$ und Kanten $a \in A^{alt}$ des Graphen entfernt werden. *Verwaist* sind also alle

- Knoten, die definitiv nicht mehr von einem Fahrzeug verwendet werden, von denen also bereits feststeht, dass der zugehörige DARP-Knoten (ein- oder aussteigender

Passagier) mit einem anderen Graphknoten realisiert wurde oder zumindest festgeschrieben wurde

- ein- und ausgehende Kanten der im vorigen Punkt beschriebenen Knoten
- alle eingehenden Kanten des neuen aktiven Knotens, mit Ausnahme der gerade befahrenen Kante
- weitere mit den betroffenen DARP-Knoten verknüpfte Knoten und Kanten, die nicht verwendet wurden – zur vollständigen Auslese ist eine relativ umfangreiche Suche durch die Knotenmengen des Graphen notwendig.

Zusätzlich müssen die MILP-Variablen, die sich auf diese Knoten und Kanten beziehen, gelöscht oder fixiert werden. Insbesondere die Entscheidungsvariablen x_a aller verwaisten Kanten $a \in A^{alt}$ sind hiervon betroffen. Diese muss zwingend (über das Einfügen einer neuen Nebenbedingung) auf den Wert 0 fixiert werden, was analog zur Festschreibung von realisierten Kanten (siehe vorigen Abschnitt 3.2.4.2) umsetzbar ist.

Weitere Schritte Wie bereits im initialen Modellierungslauf geht der Graphgenerierung bzw. Aktualisierung zudem die Erkundung realisierbarer DARP-Knoten-Abfolgen durch den *DARPPreprocessor* voraus.
Auch die weiteren Schritte stellen – zumindest der hier gezeigten Ebene des *DARPModel* – im Wesentlichen eine Fortsetzung des initialen Durchlaufs dar. Dazu gehören die Aktualisierung der Hashtabellen, die Aufrufe zur Aktualisierung der Variablen, Nebenbedingungen und der Zielfunktion, der Aufruf der Solvers sowie die Abfrage der Lösung.

Kapitel 4

Unsicherheiten beim dynamischen Dial-a-Ride-Problem

4.1 Relevante *unsichere* Faktoren beim DARP

Der Begriff *Unsicherheit* wird in unterschiedlichen Kontexten verwendet und ist daher nicht einheitlich definiert. Unsichere Faktoren sind für uns im Folgenden Umstände, deren tatsächliche Ausprägung (welche wir im vorigen Kapitel auch als *Realisierung* bezeichnet haben) vor dem Optimierungsprozess nicht feststeht, deren Verhalten wir allerdings mit Wahrscheinlichkeitsverteilungen annähernd simulieren können.
Zur Einordnung dieser Arbeit werden wir kurz auf Unsicherheiten eingehen, denen Ridepooling-Dienste für gewöhnlich ausgesetzt sind:

- unsicheres Anfragevolumen
- unsichere räumliche Lage benötigter Haltestellen
- Ausfälle von Fahrzeugen oder Personal
- unsichere Fahrzeiten zwischen zwei Haltestellen
- technische Probleme mit der Buchungsplattform

Für alle genannten Aspekte könnte bei Bedarf eine möglichst effiziente Strategie zur Vorbereitung darauf oder zum Umgang damit erarbeitet werden. Wir werden nun die stets vorhandene unsichere Fahrzeit zwischen Haltestellen untersuchen.

4.2 Fahrzeitunsicherheiten

4.2.1 Charakterisierung von Fahrzeitunsicherheiten

Wir werden im Folgenden Umstände, die sich auf die Fahrzeit auswirken, als störende und/oder beeinflussende Faktoren bezeichnen. Wir interessieren uns nicht nur für den Einfluss dieser Faktoren auf die Fahrzeiten selbst, sondern auch auf deren Streuung. Als *störend* bezeichnen wir einen Faktor dann, wenn dieser die Fahrzeit erwartungsgemäß erhöht oder eine erhöhte Streuung bewirkt. Umstände, die die Fahrzeit und deren Streuung sowohl verringern als auch erhöhen können, je nach Ausprägung, zählen zu den restlichen beeinflussenden Faktoren.
Die Zahl denkbarer Einflüsse ist unerschöpflich, und diese lassen sich zeitgleich vielfältig kategorisieren. Wir wollen uns daher auf eine Auswahl der wichtigsten Einflüsse konzentrieren. Auch sind nicht alle Einflüsse eindeutig zu kategorisieren, wir verwenden dann die überwiegend zutreffende Kategorie.

4.2.2 Allgemeine Einflüsse

4.2.2.1 Raumbezogene Einflüsse

Zunächst betrachten wir raumbezogene Einflüsse. Diese Faktoren haben zumeist örtlich begrenzte Auswirkungen auf Fahrzeiten und deren Schwankungen. Sie schließen die Struktur des Straßennetzes, die Verteilung von Gewerbe- und Wohngebäuden und die Kapazitäten einzelner Straßen mit ein. Da diese Arbeit sich nicht auf eine bestimmte Stadt oder Region beziehen soll, können wir die genannten Faktoren im weiteren Verlauf nicht berücksichtigen.
Es gibt allerdings einen wichtigen räumlichen Einfluss auf Fahrzeitschwankungen, der weitgehend ortsunabhängig ist, nämlich die Größe der Distanz selbst. Untermauert von zahlreichen Studien [18][10][6] lässt sich dies recht einfach erklären: Kürzere Distanzen sind im Verhältnis wesentlich stärker von den im folgenden Abschnitt behandelten Störfaktoren betroffen und weisen daher eine grundsätzlich stärkere Streuung auf.
Im Falle von Ridepoolingdiensten kommen noch kleinere zeitliche Ungereimtheiten beim Dienstablauf hinzu. Grundsätzlich zu erwartende Differenzen zwischen gebuchter und realisierter Servicezeit sind hier ebenso wie ungenaue Übermittlungen von Echtzeitdaten zwischen Fahrzeugen und Schnittstelle des Ridepoolingdienstes zu nennen. Schließlich könnten Fahrer dazu neigen, auf längeren Strecken drohende oder vorliegende Verspätungen durch ihren Fahrstil auszugleichen.
Wir können diesen Einfluss demzufolge als annähernd omnipräsent betrachten und werden ihn daher bei der Fahrzeitgenerierung berücksichtigen.

4.2.2.2 Szenarien

Eine ausgeprägte Rolle spielen Szenarien im Straßenverkehr. Die Differenzen zwischen Fahrzeiten im Berufsverkehr, im normalen Werktagsverkehr, an Feiertagen oder nachts sind ausgesprochen groß. Auch besondere Veranstaltungen können Fahrzeiten in bestimmten Bezirken deutlich erhöhen.
Da die bevorzugte Umgangsweise hiermit die (Neu-)Berechnung von Fahrzeiten für diese Szenarien darstellen sollte und szenarienabhängigen Schwankungen durch ihre starke Ausprägung im Allgemeinen weniger effektiv durch Anpassungen der DARP-Modellierung begegnet werden kann, werden Szenarien im weiteren Verlauf nicht berücksichtigt.
Nichtsdestotrotz sei darauf hingewiesen, dass Szenarien sich über die erwartete Fahrzeit hinaus auch auf die Schwankung dieser auswirken, welche wir daher modellieren und variieren wollen. Der Einfluss solcher Szenarien auf Fahrzeitschwankungen wurde neben empirischen Studien[15] auch mithilfe von Simulationen gezeigt [9].

4.2.3 Störende Einflüsse

4.2.3.1 Kategorisierung nach Wirkungsspektrum

Wie bei den allgemeinen Einflüssen lassen sich hier lokale und globale Faktoren ausmachen. Straßensperrungen aufgrund von Baustellen oder temporäre Absenkungen der erlaubten Höchstgeschwindigkeit sind dabei als lokale Störfaktoren einzuordnen.

Der wichtigste globale Faktor ist das Wetter. Wir gehen für diese Einordnung zum einen davon aus, dass ungünstige Wetterbedingungen im gesamten Einzugsgebiet des Ridepoolingdienstes ähnliche Auswirkungen haben. Zum anderen nehmen wir vereinfacht an, dass städteübergreifend, aber auf eine Klimazone begrenzt, ähnliche Wetterbedingungen im Mittel ähnliche Auswirkungen haben.

Nach Tsapakis et al. [22] wurden die Wetterbedingungen am Beispiel der Stadt London nach ihrer Ausprägung auf die Fahrzeit folgendermaßen nach Einflussstärke aufsteigend eingeordnet: leichter, mittlerer, starker Regen, gefolgt von leichtem und schließlich starkem Schnee. Nicht berücksichtigt wurde hierbei Glatteis. Zahlreiche Erhebungen belegen zudem einhellig den Einfluss von ungünstigen Wetterbedingungen auf die Unfallwahrscheinlichkeit und deren mittleren Schweregrad. Wir können davon ausgehen, dass daraus nicht nur eine Erhöhung der erwarteten Fahrzeit, sondern auch eine signifikant verstärke Streuung der Fahrzeiten resultiert.

Wetterbedingungen lassen sich wie Stoßzeiten auch effektiv als Szenarien modellieren. Im Rahmen unserer uninformierten Fahrzeitgenerierung werden wir daher auch das Wetter

nicht explizit berücksichtigen.
Über Fahrzeiten hinaus sei an dieser Stelle angemerkt, dass Wetterbedingungen möglicherweise auch einen signifikanten Einfluss auf das aktuelle Anfragevolumen des Ridepoolingdiensts haben. Dabei könnte es sowohl zu einer erhöhten Nachfrage (z.B. aufgrund notwendiger Fahrten bei Regen) als auch zu einer Anfragereduktion (bei schweren Unwettern oder für Fahrten zu Freizeitaktivitäten) kommen.

4.2.3.2 Kategorisierung nach Interdependenz

Einige Störfaktoren wirken auf die Fahrzeit in erster Linie *additiv*, erhöhen die Fahrzeit also um annähernd konstante Werte. Dazu zählen Verpassen der Grünphase einer einzelnen Ampel und (in bedingtem Ausmaß) auch lokale Störfaktoren wie Baustellen.
Zusätzlich liegen im Verkehrsgeschehen allerdings auch sogenannte *multiplikative* Faktoren vor. Solche Einflüsse erhöhen die Fahrzeit (annähernd) multiplikativ, also um einen bestimmten Faktor. Insbesondere gilt dies für die zuvor erwähnten ungünstigen Wetterbedingungen.
Staus können Fahrzeiten sowohl additiv (wenn begrenzt auf einzelne Verkehrsabschnitte), als auch multiplikativ (sich weiter erstreckende Staus oder Folgestaus) erhöhen. Selbiges gilt für Ampeln, die gehäuft auf einer Route liegen.
Die Existenz von multiplikativen Faktoren zählt im Gegensatz zu den meisten bisher genannten Einflüssen zu den für uns zu berücksichtigenden Aspekten, da unterschiedliche Modellierungsansätze unterschiedlich gut dazu geeignet sind, solche zu modellieren.

4.2.4 Simulation von Fahrzeitunsicherheiten

4.2.4.1 Auswahl der Wahrscheinlichkeitsverteilung

Für die Modellierung realer Fahrzeiten existieren vielfältige Ansätze. Anhand realer Daten über maschinelles Lernen können geeignete Wahrscheinlichkeitsverteilungen ausgewählt [17] werden. Die meisten Publikationen befassen sich allerdings mit der Erarbeitung möglichst genauer Modelle für die Vorhersage von Reisezeiten anhand von Prädiktoren. Dies geschieht meist über die Verwendung geeigneter Wahrscheinlichkeitsverteilungen oder Methoden des maschinellen Lernens [1].
Da das Ziel dieser Arbeit keineswegs eine möglichst genaue Fahrtzeitvorhersage ist und auch keine umfangreichen realen Daten zur Verfügung stehen, wollen wir uns in diesem Abschnitt primär der Suche nach einer optimalen Wahrscheinlichkeitsverteilung für Fahrzeiten innerhalb größerer Städte widmen.
Geeignete Wahrscheinlichkeitsverteilungen lassen sich bereits anhand ihrer Charakteristiken eingrenzen. Offensichtlich ausgeschlossen sein müssen Fahrzeiten, die physikalische Begrenzungen überschreiten, d.h. ausgesprochen niedrige Fahrzeiten, die nur aus einer

unzulässigen oder nicht erreichbaren Durchschnittsgeschwindigkeit hervorgehen und insbesondere jegliche negative Fahrzeit. In Übereinstimmung hiermit weisen Fahrzeiterhebungen in der Regel eine rechtsschiefe Häufigkeitsverteilung auf [21].
Weiterhin wäre es für die Wahrscheinlichkeitsverteilung wünschenswert, in der Lage zu sein, die im vorigen Abschnitt genannten *multiplikativen* Prozesse zu modellieren.
Schließlich wäre es von Vorteil, wenn die Standardabweichung der zugehörigen Zufallsvariablen abhängig vom erwarteten Mittel dieser ist. Andernfalls müsste die Parametrisierung vom Betrag der absoluten Fahrzeit abhängig gemacht werden. Grundsätzlich gilt dies für die Gammaverteilung.
Häufig wird aber die logarithmische Normalverteilung, auch Log-Normalverteilung genannt, vorgezogen[21]. Im Gegensatz zur Gammaverteilung berücksichtigt diese auch multiplikative Faktoren und spiegelt zudem Ausreißer bei Fahrzeit realistischer wider. Sie repräsentiert die Wahrscheinlichkeitsverteilung eines Produkts von Normal(μ, σ^2)-verteilten Zufallsvariablen X. Wenn wir unsere Zufallsvariable $T^{real}_{(u,v)}$ als logarithmisch normalverteilt betrachten, wäre der Logarithmus dieser Zufallsvariable wiederum normalverteilt.

4.2.4.2 Implementierung

Um die stärkere Streuung bezüglich kürzerer Strecken zu modellieren, soll sich die Standardabweichung σ dynamisch am absoluten Wert der Fahrzeit $t_{(u,v)}$ einer Kante $(u, v) \in A(G)$ orientieren. Wir möchten diese dennoch in einem bestimmten „Rahmen“ halten. Dazu legen wir eine obere Schranke σ_{max} und eine untere Schranke σ_{min} fest, zwischen denen sich die dynamisch gewählte Standardabweichung σ bewegen soll.
Da nicht-stetige Methoden, dies zu modellieren, zum Beispiel regelbasierte Abstufungen oder ein linearer Verlauf von σ innerhalb eines Intervalls $[t_{min}, t_{max}]$ mit dem Wertebereich $[\sigma_{min},\sigma_{mx}]$ wenig flexibel sind, bedienen wir uns eines Tricks: Wir verwenden einen Term, dessen Wert in Abhängigkeit der Fahrzeit $t_{(u,v)}$ und einer Verfallsrate λ exponentiell verfällt und später zur Minimalabweichung σ_{min} addiert wird:

$$(\sigma_{max} - \sigma_{min}) \times e^{-\lambda \times t_{(u,v)}} \tag{4.1}$$

Wir nehmen hier ohne Beschränkung der Allgemeinheit an, dass die unter Schranke σ_{min} stets einem Drittel des Wertes der oberen Schranke σ_{max} entspricht und verwenden den Wert $\lambda = 1$. Der Wert von σ_{max} soll für die späteren Tests variiert werden. Zur Veranschaulichung ist in Grafik 4.1 der Verlauf der resultierenden Standardabweichung mit $\sigma_{max} = 1,5$ dargestellt. Beispielhaft sind in Grafik 4.2 die Wahrscheinlichkeitsdichtefunktionen der resultierenden Log-Normalverteilungen für verschiedene, gängige Werte von $\tau_{(u,v)}$ aufgeführt.[1]

[1]Die Python-Skripte zur Generierung aller Graphen zu den Wahrscheinlichkeitsverteilungen wurden von ChatGPT aufgesetzt.

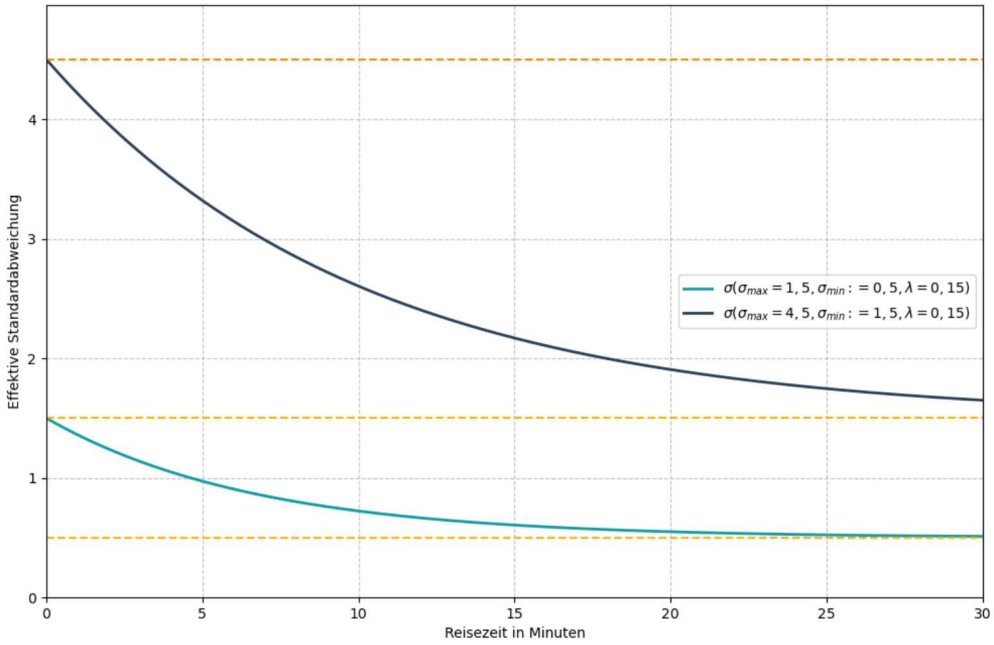


Abbildung 4.1: σ in Abhängigkeit von $t_{(u,v)}$ mit $\sigma_{max} := 1,5$ und $\sigma_{max} := 4,5$

Die vollständige Berechnung der Standardabweichung ist schließlich in Algorithmus 9 formalisiert.

Algorithm 9 calculate_std_dev($t_{(u,v)}$) **returns** double

1: $\sigma_{min} \leftarrow \frac{\sigma_{max}}{3}$
2: $\lambda \leftarrow 0.1$
3: $\sigma \leftarrow \sigma_{min} + (\sigma_{max} - \sigma_{min}) \times e^{-\lambda \times t_{(u,v)}}$
4: **return** σ

Gemäß der Definition der Log-Normalverteilung werden in Algorithmus 10 anschließend die passenden Parameter berechnet, die in einem logarithmischen Verhältnis zu der für uns anschaulicheren Standardabweichung σ der zugrundeliegenden Normalverteilung stehen. Der Variationskoeffizient, definiert als $VarK = \frac{\sqrt{Var[X]}}{E[X]}$ mit der $\mathcal{N}(\mu, \sigma^2)$-verteilten Zufallsvariable X dient dabei zur Berechnung des erst zu ermittelnden Formparameters $\sigma_{\mathcal{LN}}$ für die Log-Normalverteilung. Auch der Parameter $\mu_{\mathcal{LN}}$ muss dahingehend gewählt werden, dass anschließend $E[X] = t_{(u,v)} = e^{\mu_{\mathcal{LN}} + \frac{\sigma^2}{2}}$ gilt, die Verteilung also erwartet im Mittel den Wert $t_{(u,v)}$ generiert.
Bei der Rückgabe wird schließlich, auch wenn es angesichts der verwendeten Verteilung unwahrscheinlich ist, sichergestellt, dass keine nicht praktikable Fahrzeit $T^{real}_{(u,v)}$ verwendet

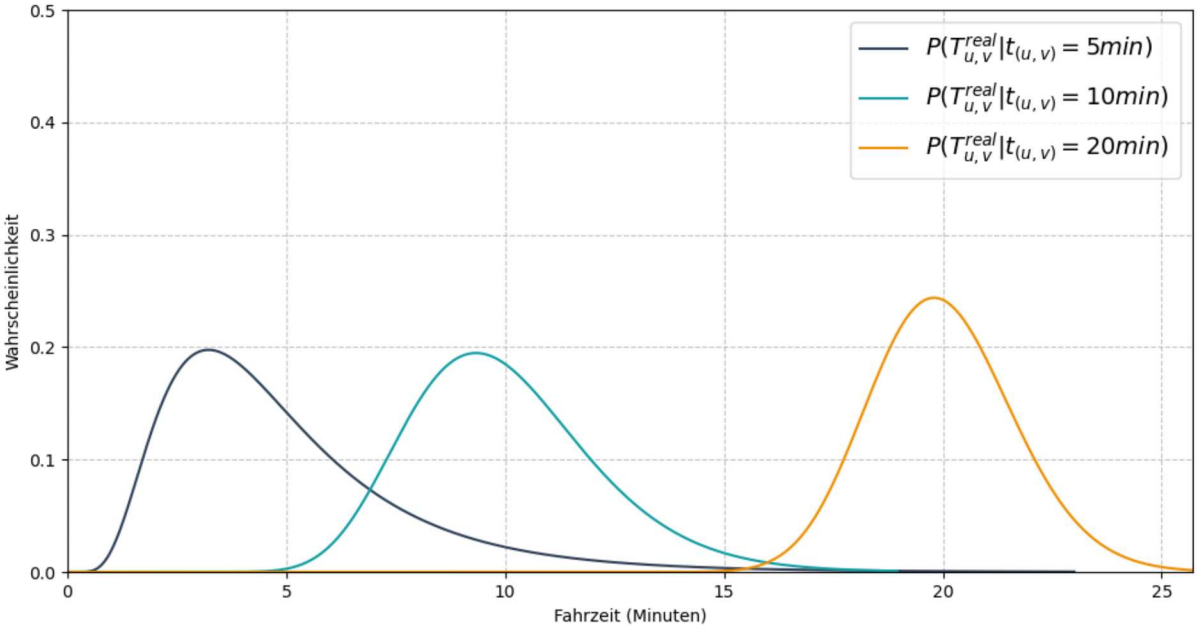


Abbildung 4.2: Resultierende Wahrscheinlichkeitsdichte bei $\sigma_{max} := 4,5$ für die realisierte Fahrzeit jeweils in Abhängigkeit einer vorberechneten Kantenfahrzeit $t_{(u,v)}$

Algorithm 10 generate_travel_time($t_{(u,v)}$): **returns** double

1: $\sigma \leftarrow$calculate_std_dev($t_{(u,v)}$)
2: $VarK \leftarrow \frac{\sigma}{t_{(u,v)}}$
3: $\sigma_{\mathcal{LN}} \leftarrow \sqrt{\log(1 + VarK^2)}$
4: $\mu_{\mathcal{LN}} \leftarrow \log(t_{(u,v)}) - \frac{\sigma^2_{\mathcal{LN}}}{2}$
5: $T_{(u,v)} \sim$Lognormal$(\mu_{\mathcal{LN}}, \sigma^2_{\mathcal{LN}})$
6: **return** $\max(0.75 \times t_{(u,v)}, T_{(u,v)})$

wird, die kleiner als $\frac{3}{4} \times t_{(u,v)}$ ist.

4.2.5 Praktische Auswirkungen von Fahrzeitschwankungen auf die vorgestellte Simulation

4.2.5.1 Versuchsaufbau

Im Folgenden beziehen wir uns mit dem Begriff *Instanz* auf die verwendeten Eingabedateien, nicht etwa auf die Klasse *DARPInstance* aus Kapitel 3. Wir werden die dynamische Instanz *no-011-6*, die über 85 Anfragen und 7 Fahrzeuge verfügt, die auf realen Anfragen basiert, mit einer kleinen, generierten Testinstanz *a2-16* vergleichen, die lediglich 16 Anfragen über 2 Fahrzeuge bedienen soll - mit dem Hauptaugenmerk darauf, bei wie vielen Durchläufen das generierte MILP von ILOG CPLEX nicht mehr zu lösen ist.

Handhabung der Nebenbedingungen Allerdings soll zwischen zwei Ausgangslagen unterschieden werden: In der ersten Umgebung soll sich die Methode fix_realized_edges() in Algorithmus 8 wie vorgestellt verhalten.
In einem zweiten Anlauf sollen nach jeder Knotenrealisierung (die mit der Realisierung der hinführenden Kante zusammenfällt), also für alle v mit $(u, v) \in A^{new\_real}$ zusätzliche Modellbereinigungen durchgeführt werden: Grundsätzlich sollen alle sich ausschließlich auf diesen *konkreten* Knoten und den signifikanten Passagier dieses Knotens beziehenden Nebenbedingungen gelöscht werden:

- sämtliche Zeitfenster des Knotens v, welche vom signifikanten Passagier gemäß Nebenbed. 1 vererbt werden
- (sollte es sich bei v um einen Zustiegsknoten handeln) die Abholverzögerungsgrenze für v, welche vom signifikanten Passagier gemäß Nebenbed. 1 vererbt werden
- (sollte es sich bei v um einen Ausstiegsknoten handeln) die maximale Reisezeit für alle Knotenpaare (u, v) mit $u \in V_{i^+}(\tau)$, welche vom signifikanten Passagier gemäß Nebenbed. 7 vererbt wurde.

Letzter Schritt könnte reduziert werden, im Zuge der Graphbereinigung sollte zum Zeitpunkt τ allerdings nur noch der zugehörige realisierte Einstiegsknoten innerhalb V_{i^+} enthalten sein.
Die aufgeführten Nebenbedingungen teilen ebenso die Eigenschaft, „der Vergangenheit anzugehören“ - insofern, als dass es für das *DARPModel* weder weiteren Optimierungsspielraum noch eine Möglichkeit gäbe, Verletzungen dieser Nebenbedingungen ungeschehen zu machen.
Des Weiteren gibt es für die erweiterte Modellbereiningung eine Anpassung an 4. $T^{real}_{(v)}$:=max(T_v, T_{fix} in Zeile 12 können wir hier durch $T^{real}_{(v)}$:=max($e_{v_{i^+}}, T_{fix}$) ersetzen, da wir nicht mehr auf verletzte Maximalreisezeitbeschränkungen Rücksicht nehmen müssen, die aufgrund verfrühter Einstiege auftreten.

Fahrzeitvariabilität Unabhängig davon wollen wir die Simulation zwei unterschiedlich großen Fahrzeitstreuungsfaktoren aussetzen, vergleichbar mit jeweils einem Szenario, in dem es hauptsächlich geringfügige Abweichungen von den vorhergesagten Fahrzeiten gibt (wie zum Beispiel bei Nachtverkehr), und einem solchen, in dem mit größeren Abweichungen von vorhergesagten Fahrzeiten zu rechnen ist (wie zum Beispiel bei starkem Regen oder Berufsverkehr). Als Vergleichswert dienen deterministische Durchläufe unter Verwendung der vorberechneten Kantenfahrzeit unter Beibehaltung der unmittelbaren Nebenbedingungen.
Wir werden folglich jede Instanz insgesamt in fünf verschiedenen Umgebungen testen, bedingt durch zwei unabhängige Variablen:

- Variabilität - Deterministisch, Klein: $\sigma_{max} = 1.5$, Groß: $\sigma_{max} = 4.5$
- Modellbereinigung (nicht für deterministische Durchläufe) - Standard, Erweitert

4.2.5.2 Messmethodik

Pro Umgebung und Instanz sollen wiederum 100 Durchläufe ausgeführt werden. [2] Wie bereits angedeutet, liegt der Fokus auf den jeweils erfolgreichen und fehlgeschlagenen Durchläufen. Als fehlgeschlagen zählen Durchläufe genau dann, wenn währenddessen zu einem beliebigen Zeitpunkt τ das zugehörige MILP(τ) nicht mehr lösbar ist. Falls es zu einer Lösung gekommen ist, werden allerdings noch einige Kennzahlen ausgelesen und verbucht. Die Kennzahlen sind zum Teil aus der Lösung von Gaul et al. [7] übernommen und werden jeweils hinsichtlich ihrer Aussagekraft grob eingeordnet, welche im Allgemeinen mit steigender Instanzabhängigkeit abnimmt.
Personenkilometer sollen die insgesamt zurückgelegten Strecken in Beziehung zur Belegung der Fahrzeuge setzen. Sie werden aufsummiert über die Produkte aus der Länge jeder zurückgelegten Kante (Strecke in Kilometern, welche aus den Kantengewichten folgt) mit der Anzahl der während der Kantenpassierung im Fahrzeug befindlichen Passagiere.
Gebuchte Personenkilometer sind eng verwandt, werden aber anders ermittelt. Es werden hier für alle Anfragen die Produkte aus der *direkten* Strecke vom Start- zum Zielort und der Anzahl an Personen/Sitzplätzen dieser Anfrage aufsummiert.

- **Anzahl akzeptierter Anfragen**;
  bedingte Instanzabhängigkeit (wird im Folgenden vernachlässigt)
- **Gesamtroutenkosten** (Summe aller Gewichte der verwendeten Kanten). Zu beachten ist, dass das Gewicht c_a einer Kante $a \in A$ **nicht** von einer abweichenden Fahrzeit T_a^{real} (im Vergleich zu t_a) beeinflusst wird. Diese Kennzahl ist selbstverständlich ausschließlich im Kontext einer bestimmten Instanz aussagekräftig;
  ausgesprochen starke Instanzabhängigkeit
- **Durchschnittliche Reisezeit der Passagiere**;
  starke Instanzabhängigkeit
- **Durchschnittliche Wartezeit der Passagiere:** Differenz zwischen realisiertem Einstiegszeitpunkt und Beginn des Einstiegszeitfensters von Passagier $i \in \mathcal{A}$);
  moderate Instanzabhängigkeit

[2] Zur Erstellung der zur automatisierten Durchlaufsausführung und -auswertung notwendigen *bash*- bzw. *python*-Skripte wurde der noch relativ junge Chatbot *claude.ai* der Organisation Anthrophic PBC in Anspruch genommen. Dies betrifft alle durchgeführten Tests der Arbeit.

- **Durchschnittsbelegung der Fahrzeuge**, wobei Zeiträume, in denen Fahrzeuge leer sind, hier außen vor gelassen werden;
  weniger starke Instanzabhängigkeit

- **Anteil aller Leerfahrten an Gesamtstrecke:** Leerfahrten sind solche, bei denen sich auf einer Kante, also zwischen zwei Knoten, keine Passagiere im Fahrzeug befinden, was zum Beispiel auf die Fahrt vom Depot zu einem Einstiegsknoten zutrifft;
  weniger starke Instanzabhängigkeit

- **Verhältnis von gebuchten Kilometern zu den Gesamtroutenkosten**;
  geringe Instanzabhängigkeit;

- **Rechenzeit des Durchlaufs inklusive Auswertung**;
  ausgesprochen starke Instanzabhängigkeit

Anhand der Mittelwerte, dem Median, der Minima und Maxima dieser Werte wird deutlich werden, welche Auswirkungen die verschiedenen (Simulations-)Umgebungen auf die hypothetische Wirtschaftlichkeit des Dienstes sowie den Komfort der Passagiere haben.

4.2.5.3 Testumgebung

Grundsätzlich gilt für alle Durchläufe, dass die aktuelle Zeit nach jedem Simulationsschritt um eine Minute erhöht wird (es gilt entsprechend $\tau_j - \tau_{j-1} = 1 \forall j \in \mathbb{N}^+$), bis alle Anfragen bearbeitet wurden und solange noch anzusetzende oder realisierbare Kanten existieren, also zumindest eine der beiden Prioritätswarteschlangen AQ und RQ noch Elemente enthält. Nach Möglichkeit sollen drei neue Anfragen gleichzeitig vom *RequestManager* freigegeben werden ($\rho = 3$, vgl. Arbeitsweise des *RequestManagers* 3.1.1).
Von Gaul et al. [7] übernommen und eingesetzt wurden eine Abholverzögerungsgrenze Γ von 5 Minuten sowie folgende Gewichtungen für die Zielfunktion des MILPs:

- $\omega_c = 1 \frac{1}{km}$,

- $\omega_d = 60 \frac{1}{abgelehne Anfrage}$

- $\omega_d = 0,1 \frac{1}{min}$

Der dabei verwendete Rechner verfügt über einen Core i5 6600K mit 3,4 GHz Taktfrequenz und 8 GB RAM.

4.2.6 Diskussion der Ergebnisse

4.2.6.1 Erfolgsquote

Nachfolgend sind die erfolgreichen und fehlgeschlagenen Durchläufe für beide Instanzen aufgeführt, wobei als Vergleichswert die selbstverständlich durchgehend erfolgreichen deterministischen Durchläufe aufgeführt sind.

Umgebung	Erfolgreich	Fehlgeschlagen	Gesamt
Deterministisch	100	0	100
$\sigma_{max} = 1{,}5$, Entfernung unm. Nebenbed.	100	0	100
$\sigma_{max} = 1{,}5$, Beibehaltung unm. Nebenbed.	84	16	100
$\sigma_{max} = 4{,}5$, Entfernung unm. Nebenbed.	78	22	100
$\sigma_{max} = 4{,}5$, Beibehaltung unm. Nebenbed.	48	52	100

Tabelle 4.1: Anzahl erfolgreicher und fehlgeschlagener Durchläufe für Instanz *a2-16*

Umgebung	Erfolgreich	Fehlgeschlagen	Gesamt
Deterministisch	100	0	100
$\sigma_{max} = 1{,}5$, Entfernung unm. Nebenbed.	23	77	100
$\sigma_{max} = 1{,}5$, Beibehaltung unm. Nebenbed.	5	95	100
$\sigma_{max} = 4{,}5$, Entfernung unm. Nebenbed.	0	100	100
$\sigma_{max} = 4{,}5$, Beibehaltung unm. Nebenbed.	0	100	100

Tabelle 4.2: Anzahl erfolgreicher und fehlgeschlagener Durchläufe für Instanz *no-011-6*

Es sollte bereits deutlich werden, dass die Entfernung der „unmittelbaren Nebenbedingungen“ einen stark positiven Effekt auf die Erfolgsquote hat. Nichtsdestotrotz wird sichtbar, dass die Durchläufe der großen Testinstanz *no-116-6* bei einem σ_{max}-Wert von 4,5 auch mit dieser Strategie durchgängig fehlschlugen.

4.2.6.2 Kennzahlen

Die vollständigen Kennzahlen sind im Anhang hinterlegt. Ausgelassen wurden dabei die kritischen Zeitfensterverletzungen, die es für beide Instanzen in keiner Umgebung gab, die akzeptierten Anfragen (bei beiden Instanzen wurden stets alle Anfragen akzeptiert), sowie für Instanz *a2-16* die Verletzung der maximalen Reisezeitbeschränkungen, die dort in keiner Umgebung auftraten. Ebenso sind für die große Instanz keine Kennzahlen für $\sigma_{max} = 4,5$ gelistet, da es hier wie bereits aufgeführt keine erfolgreichen Durchläufe gab. Die Kennzahlen, die die Qualität der Routen widerspiegeln, also Routenkosten, Durchschnittsbelegung, Anteil Leerfahrten und gebuchte Kilometer/Gesamtroutenkosten, zei-

gen minimale Abweichungen für beide Instanzen. Dies deutet darauf hin, dass es trotz der Fahrzeitschwankungen äußerst selten zu spontanen Umplanungen der Routen kam.
Auch dass der Median der meisten Kennzahlen [3] fast ausschließlich mit dem der deterministischen Durchläufe übereinstimmt, bestätigt, dass die Abweichungen keine nennenswerten Änderungen der Routen bewirkt haben.
Kurz werden wir aber auf die durchschnittliche Warte- und Fahrzeiten der Passagiere eingehen. Die Abweichungen der Durchschnittswerte für die Durchläufe mit Entfernung der unmittelbaren Nebenbedingungen sind höchstwahrscheinlich damit erklärbar, dass Einstiege dort in appoint_started_edges() häufig früher angesetzt wurden.
Schließlich sei allerdings darauf hingewiesen, dass die Überschreitung der Abholverzögerung für die deterministischen Durchläufe, welche korrekt berechnet wurden, tatsächlich in den Durchläufen reproduzierbar waren. Daher scheint diese Nebenbedingung noch nicht vollständig korrekt implementiert zu sein.

4.3 Ansätze zum Umgang mit unsicheren Fahrzeiten

4.3.1 Bedarfsgesteuerte Routenplanung

Auf den ersten Blick liegt es nahe, die aktuell erwartete Fahrzeit dynamisch mithilfe eines Routenplanungsdienstes zu ermitteln. Angesichts der großen Anzahl an möglichen unterschiedlichen Kantengewichten - $O(|N \times N|) = O((2n)^2 = O(n^2)$ (siehe 3.1.2) -, die zudem bei schwankendem Verkehrsaufkommen möglicherweise mehrfach neu abgefragt werden müssen, wird dies hier als wenig praktikabel bewertet.

4.3.2 Berücksichtigung von Szenarien

In jedem Fall ist es sinnvoll, Kantenkosten jeweils für unterschiedliche Szenarien im Voraus zu berechnen. Bei jeder Fahrt könnte dann mithilfe einer einfachen Heuristik ein passendes Szenario gewählt werden, welches anschließend die Kantenfahrzeit festlegt. Diese Heuristik könnte beispielsweise die aktuelle Tageszeit und den Wochentag zur Wahl des Szenarios heranziehen.

4.3.3 Vorgeschlagene Lösung

Im Rahmen dieser Arbeit wäre ein szenarienbasierter Ansatz ausgesprochen schwer zu evaluieren. Die Wahl der Szenarien und zugehörigen Kantenfahrzeiten ließe sich nur mit sehr umfangreichen Verkehrsdaten treffen und auswerten.

[3] siehe komplette statistische Auswertung unter https://gitlab.informatik.uni-wuerzburg.de/s484411/dyndarp-simulation-with-probabilistic-travel-times im Ordner tests/csv$_{r}esults$

Zudem wird es auch bei einer sorgfältigen Differenzierung der Szenarien Abweichungen von den zugehörigen, vorberechneten Kantenfahrzeiten geben, deren Ausprägung natürlich auch vom jeweiligen Szenario abhängt.
Daher soll die Stabilität des vorgstellten dDARP-Modells nun zusätzlich mithilfe von *elastischen* Nebenbedingungen erhöht werden. Der Definition und Umsetzung wird sich das folgende Kapitel ausführlich widmen.

Kapitel 5

Einführung *elastischer* Nebenbedingungen

Der von mir geprägte Begriff „elastische Nebenbedingung“, wird hier als deutsche Übertragung für das Konzept *Soft Constraints* verwendet, welche im Gegensatz zu einer scharfen Nebenbedingung über einen elastischen Puffer verfügt, der die Über- oder Unterschreitung der ursprünglichen scharfen Schranke ermöglicht, diese aber anschließend innerhalb der Zielfunktion sanktioniert. Wir sprechen bei einer Umformung einer solchen, ursprünglich scharfen Nebenbedingung in eine elastische Form, von einer Aufweichung dieser Nebenbedingung.

5.1 Identifikation „aufweichbarer“ Nebenbedingungen

Trotz der großen Zahl an Nebenbedingungen lässt sich eine recht deutliche Hierarchie dieser ausmachen. Das Hauptkriterium für einen Passagier $i \in \mathcal{A}$ ist das Erreichen des Zielorts bis zu einem bestimmten Zeitpunkt, der in diesem Modell dem Ende des Ausstiegszeitfensters l_{i^-} entspricht. In vielen Fällen ist zudem eine Abfahrt vor einem bestimmten Zeitpunkt für den Passagier nicht möglich, hier der Beginn des Einstiegszeitfensters e_{i^+}. Es sollte also deutlich sein, dass die entsprechenden Nebenbedingungen nicht die ersten Kandidaten für elastische Nebenbedingungen sind.

5.1.1 Auszuschließende Nebenbedingungen

Es gibt allerdings auch einige Nebenbedingungen, für die eine Aufweichung ausgeschlossen sein muss. Für die Flusserhaltung (Nebenbedingung 3), die Aufnahme von akzeptierten Passagieren (Nebenbedingung 8) und die Begrenzung der Fahrzeugzahl hätte eine Aufweichung mitunter fatale Folgen und könnte in einen nicht mehr in die Realität übertragbaren Aktionsplan münden.

Beim Zeitabstand zwischen zwei aufeinanderfolgenden Knoten (Nebenbedingung 6) wäre eine Aufweichung zwar denkbar, allerdings unter der Einschränkung, dass im Zuge der Umsetzung eines resultierenden Aktionsplans beliebige andere Zeitbeschränkungen verletzt werden.

5.1.2 Mögliche Kandidaten

Wir wollen selektiver vorgehen und zwischen den unterschiedlichen Zeitbeschränkungen sowie einzelnen Passagieren unterscheiden. Bereits eingeplante oder aufgenommene Passagiere $i \in \mathcal{S}(\tau) \cup \mathcal{P}(\tau)$ müssen, auch wenn das ursprüngliche MILP nicht mehr erfüllbar ist, in jedem Fall transportiert werden. Eine Auswahl der mit diesen Passagieren zusammenhängenden Zeitbeschränkungen muss also zur Aufweichung infrage kommen. Für neue Anfragen allerdings wäre eine Aufweichung vermeidbar und würde zudem die Relevanz der vom Passagier angegebenen Zeitfenster infrage stellen.
Auf Ebene der Zeitfenster sollte zudem sichtbar sein, dass das frühzeitige Absetzen eines Passagiers i (also vor e_{i^-}) keinen Komfortverlust für Passagiere darstellen sollte. Das verspätete Abholen eines Passagiers (also nach l_{i^+}) wirkt problematischer, könnte aber ebenso vertretbar sein, wenn der Passagier trotzdem innerhalb des Absetzzeitfensters erwartungsgemäß am Ziel ankommt und zudem die kommunizierte Abholzeit nicht stärker als zugelassen verfehlt wird.
Beide Zeitfenster machen wir daher als erste Kandidaten zur Aufweichung aus. Da eine verfrühte Aufnahme und ein verspätetes Absetzen wie anfangs erwähnt möglichst vermieden werden sollten, verblieben die maximalen Reisezeiten L_i der Passagiere i (Nebenbedingung 7) und die Abholverzögerung (Nebenbedingung 10) als weitere Ansatzpunkte.
Wir nehmen an, dass ein Passagier ein verzögertes Abholen deutlich wahrscheinlicher erkennt und daher als Störung wahrnimmt als eine Überschreitung seiner Maximalreisezeit, die bei geringen Überschreitungen mutmaßlich unentdeckt bleibt.

5.1.3 Einordnung der aufweichbaren Nebenbedingungen

Als Resultat ergibt sich folgende aufsteigend nach erachteter Relevanz geordnete Platzierung aller aufzuweichenden, exakt spezifizierten Nebenbedingungen:

1. unkritische Zeitfenster:
   (a) betrifft Einstiegsknoten: $\forall i \in \mathcal{S}\, \forall v \in V_{i^+}$ bezüglich l_i^+
   (b) betrifft Ausstiegsknoten: $\forall i \in \mathcal{P}\, \forall v \in V_{i^-}$ bezüglich e_i^-
2. maximale Reisezeit, betrifft Knotenpaare:
   $\forall i \in \mathcal{S} \cup \mathcal{P}\ \forall (u, v), u \in V_{i^-}, v \in V_{i^+}$ bezüglich L_i

3. Abholverzögerung, betrifft Einstiegsknoten:
   $\forall i \in \mathcal{S} \cup \mathcal{P}, v \in V_{i^-}$ bezüglich γ_i^+

4. obere Ausstiegszeitfenster, betrifft Ausstiegsknoten:
   $\forall i \in \mathcal{S} \cup \mathcal{P}\ \forall v \in V_{i^-}$ bezüglich l_i^-

Gerade in Bezug auf den letzten Punkt ist darauf hinzuweisen, dass ein Abholen eines Passagiers i vor dem entsprechenden Zeitfensterbeginn e_i^+ möglicherweise allein deshalb nicht in die Realität zu übertragen ist, weil der Passagier zu diesem Zeitpunkt noch nicht vor Ort ist.

Es ist allerdings darüber hinaus ausgeschlossen, dass die nun einzig verbleibenden, harten passagierbezogenen Beschränkungen e_i^+ zusammen mit den weiterhin ausschlaggebenden Kantenfahrzeitbeschränkungen 6 zur Unerfüllbarkeit des MILPs führen: Das Fahrzeug wartet bei verfrühtem Eintreffen schlicht am gewählten Knoten $v^{active}(i^+)$, bis der Zeitpunkt e_i^+ erreicht ist. Alle daraus zusammen mit den Kantenfahrzeiten resultierenden Verzögerungen bei Folgeereignissen werden durch die aufgeweichten Nebenbedingungen „aufgefangen“.

5.2 Vorgehensweise

5.2.1 Überblick

Der verwendete Ansatz ließe sich als „Aufweichungskaskade“ bezeichnen. Diese findet dann jeweils im Rahmen des Schritts *Lösen* innerhalb eines MILP-Modellierungsschritts in Grafik 7 statt. Es folgt eine strukturierte Beschreibung der angestrebten Vorgehensweise:

1. Die Simulation beginnt wie vormalig beschrieben und verwendet alle scharfen Nebenbedingungen aus 2.5.

2. Nach jeder Kantenrealisierung werden alle sich auf den betroffenen Passagier und ausschließlich auf die Vergangenheit beziehenden Nebenbedingungen gelöscht (vgl. 4.2.5.1).

3. Falls Fahrzeitabweichungen offensichtlich weitreichendere Folgen hatten, also über die Verletzung der Zeitfenster des signifikanten Passagiers bezüglich des erreichten Knotens hinaus, und in Folge keine Lösung ermittelbar ist, wird eine Kaskade an Aufweichungen ausgelöst:

   (a) Löschen/Deaktivierung aller unkritischen Zeitfenster

(b) Wenn weiter erforderlich: Deaktivieren aller Maximalreisezeiten, Ersetzung durch elastische Nebenbedingungen

(c) Wenn weiter erforderlich: Deaktivieren aller Abholverzögerungsnebenbedingungen und Ersetzung durch elastische Nebenbedingungen

(d) Letzter Ausweg, wenn erforderlich: Aufweichung der verbliebenen, kritischen obere Schranke der Passagierzeitfenster bezüglich l_{i^-} für alle $i \in \mathcal{A}(\tau)$

4. Unabhängig vom Ablauf der Kaskade werden im Anschluss alle neu zu erstellenden Nebenbedingungen weiterhin scharf definiert.

In dieser Übersicht noch unerwähnt blieb, dass provisorisch von Beginn an elastische Nebenbedingungen für die maximale Fahrzeit erstellt werden. Kurz gefasst ist dies der im Vergleich zu anderen Nebenbedingungen größeren Anzahl solcher Nebenbedingungen geschuldet, welche gemäß Definition für alle Paare aus Einstiegs- und Ausstiegsknoten eines Passagiers entstehen und in Kombination mit der aufsteigenden Referenzierung der zugehörigen *IloRangeArray*-Objekte (siehe 3.1.3) eine nachträgliche Erstellung erschweren würden.

5.2.2 Formalisierte elastische Nebenbedingungen

Bei der konkreten Formalisierung einer solchen elastischen Nebenbedingung setzen wir auf etablierte Standards. Eine Schlupfvariable λ wird zu einer oberen Schranke addiert oder von einer unteren Schranke abgezogen und ermöglicht somit die Überschreitung beziehungsweise Unterschreitung der ursprünglichen Schranke. Am Beispiel der komplexesten betroffenen Nebenbedingung 7, der maximalen Reisezeit, sähe diese unter Verwendung der Schlupfvariable $\lambda^L_{(w,v)}$ folgendermaßen aus:

$$T_w - T_v - s_{i^+} \leq \lambda^L_{(w,v)} + L_i + M_i(1 - \sum_{e \in \delta^{in}(u)} x_a + 1 - \sum_{e \in \delta^{in}(v)} x_a) \quad \forall i \in A(\tau_j), u \in V_{i^+}(\tau_j), v \in V_{i^-}(\tau_j) \tag{5.1}$$

Für die kommunizierte Abholzeit verwenden wir die Schlupfvariable λ^Γ_v, die anschließend zur oberen Schranke der Nebenbedingung 10 addiert wird:

$$T_v \leq \lambda^\Gamma_v + (\gamma^+_i + \Gamma)\left(1 - \sum_{a \in \delta^{in}(v,\tau_j)} x_a\right) \forall i \in S(\tau_j) \tag{5.2}$$

Im Falle der Zeitfenster greifen wir schließlich auf die Schlupfvariable λ_{v^-} zurück, die zur oberen Schranke von T_v von Nebenbedingung 1, Gleichung 2.8 addiert wird (man beachte, dass die untere Schranke bereits im ersten Schritt der Kaskade entfernt wurde):

$$T_v \leq \lambda_{v^-} + l_{i^-} \quad \forall i \in \mathcal{A}, v \in V_{i^-}(\tau_j) \tag{5.3}$$

5.2.3 Berücksichtigung in der Zielfunktion

Die Überschreitungen der Maximalreisezeiten werden von folgender Unterzielfunktion erfasst:

$$p_L = \sum_{i \in \mathcal{N} \cup \mathcal{S} \cup \mathcal{P}} \left(\sum_{v \in V_{i+}} \left(\sum_{w \in V_{i-}} \lambda^L_{(v,w)} \right) \right) \tag{5.4}$$

Für die Überschreitung der Abholverzögerung verwenden wir:

$$p_\Gamma = \sum_{i \in \mathcal{N} \cup \mathcal{S}} \left(\sum_{v \in V_{i+}} \lambda^\Gamma_v \right) \tag{5.5}$$

Für die Überschreitung eines kritischen Zeitfensters, also eines Absetzens eines Passagiers nach dem Ende dessen Zeitfensters:

$$p_{tw} = \sum_{i \in \mathcal{N} \cup \mathcal{S} \cup \mathcal{P}} \left(\sum_{v \in V_{i+}} \lambda_{v^-} \right) \tag{5.6}$$

Als Resultat erhalten wir nun abhängig von der aktuellen Position der Kaskade unterschiedliche Zielfunktionen. Bei Stufe 3b gilt

$$\min f^{soft}_L = \min \left(f_{cnr} + \omega_L p_L \right) \tag{5.7}$$

Die folgenden Stufen werden bei Bedarf ergänzt, dabei gilt für Stufe 3c:

$$\min f^{soft}_\Gamma = \min \left(f^{soft}_L + \omega_\Gamma p_\Gamma \right) \tag{5.8}$$

und für Stufe 3d schließlich:

$$\min f^{soft}_{tw} = \min \left(f^{soft}_\Gamma + \omega_{tw} p_{tw} \right) \tag{5.9}$$

In Anbetracht des beschränkten Zeitrahmens und der später deutlich werdenden ungleichen verteilten Notwendigkeit, jene Nebenbedingungen aufzuweichen, wurden die Strafkoeffizienten intuitiv gewählt:

- $\omega_L = 5 \frac{1}{min}$
- $\omega_p = 15 \frac{1}{min}$
- $\omega_{tw} = 45 \frac{1}{min}$

5.3 Implementierung und Architekturanpassungen

Die Aufweichkaskade greift, wenn im Modellierungsschritt in 7 nach dem Vorgang „Lösung durch ILOG CPLEX“ keine Lösung ermittelt werden konnte. Zudem wurden für die elastischen Nebenbedingungen separate Unterstrukturen der Klasse *MILP* in Anhang 5 für die elastischen Nebenbedingungen selbst, die Schlupfvariablen, die Unterzielfunktionen und die Strafkoeffizienten hinzugefügt.

5.4 Evaluierung

5.4.1 Methodik

Wie im vorigen Kapitel werden die Instanz der realen Anfragen *no_011_6* mit der Instanz *a2-16* mit jeweils $\sigma_{max} = 1,5$ und $\sigma_{max} = 4,5$ vergleichen. Primär für die große Testinstanz *no_011_6* wurde ein oberes Zeitlimit von 2 Minuten festgelegt, bei dem der Durchlauf abgebrochen und entsprechend eingeordnet wird, zudem werden nur 50 Durchläufe pro Instanz durchgeführt. Aufgrund einer sehr häufigen Überschreitung dieser wurden im Anschluss nochmals jeweils 50 Durchläufe mit einer oberen Grenze von 5 Minuten durchgeführt, die im folgenden Abschnitt gemeinsam ausgewertet werden.

5.4.2 Diskussion der Ergebnisse

Angesichts der sehr knappen Zeit soll hier die Laufzeit im Vordergrund stehen. Wie bereits angemerkt, war für die erste Reihe an Durchläufen ein Zeitlimit von 2 Minuten festgelegt, was für die größere Instanz zu einer unverhältnismäßig starken Anzahl von fehlgeschlagenen Durchläufen führte. Allerdings gab es dort auch trotz der elastischen Nebenbedingungen fehlgeschlagene Durchläufe. Die Aufweichkaskade scheint hier also noch zuverlässig zu verhindern, dass es eine Lösung gibt.

Umgebung	Erfolgreich	Fehlge.	Zeitüberschreitung	Gesamt
Limit von 2 min, $\sigma_{max} = 1,5$	50	0	0	50
Limit von 2 min, $\sigma_{max} = 4,5$	50	0	0	50

Tabelle 5.1: Anzahl erfolgreicher und fehlgeschlagener Durchläufe für Instanz *a2-16*

Umgebung	Erfolgreich	Fehlge.	Zeitüberschreitung	Gesamt
Limit von 2 min, $\sigma_{max} = 1,5$	44	4	2	50
Limit von 2 min, $\sigma_{max} = 4,5$	11	2	37	50

Tabelle 5.2: Anzahl erfolgreicher und fehlgeschlagener Durchläufe für Instanz *no-116-6*

Obere Zeitgrenze von 5 Minuten Wie angekündigt wurde eine zweite Reihe an Durchläufen durchgeführt mit einer zeitlichen Obergrenze von 5 Minuten:
Auch hier gab es für die größere Instanz aber immerhin in der Hälfte aller Fälle eine Zeitüberschreitung - die durchschnittlichen Durchführungszeiten für die erfolgreichen Durchläufe waren dabei 41,14 Sekunden für $\sigma_{max} = 1,5$ und 158,74 Sekunden $\sigma_{max} = 4,5$,

Umgebung	Erfolgreich	Fehlge.	Zeitüberschreitung	Gesamt
Limit von 5 min, $\sigma_{max} = 1,5$	50	0	0	50
Limit von 5 min, $\sigma_{max} = 4,5$	50	0	0	50

Tabelle 5.3: Anzahl erfolgreicher und fehlgeschlagener Durchläufe für Instanz *a2-16*

Umgebung	Erfolgreich	Fehlge.	Zeitüberschreitung	Gesamt
Limit von 5 min, $\sigma_{max} = 1,5$	47	3	0	50
Limit von 5 min, $\sigma_{max} = 4,5$	25	0	25	50

Tabelle 5.4: Anzahl erfolgreicher und fehlgeschlagener Durchläufe für Instanz *no-116-6*

was nochmals den generell starken Einfluss des Parameters auf die Ausführungszeit untermauert. Es handelt sich also nicht um einzelne Ausreißer, die verhältnismäßig lange brauchen.
Die ausführlichen Auswertungen für die erfolgreichen Durchläufe beider Instanzen zeigen wiederum geringe Abweichungen in den Kennzahlen bei präsenten Nebebedingungsverletzungen [1]. Ein denkbarer Rückschluss an dieser Stelle ist also eine möglicherweise zu geringe Flexibilität des *DARPModels*, Verletzungen von Zeitfenstern mit Umplanungen der Routen zu begegnen.

5.4.3 Zweiter Durchlauf mit Gammaverteilung

Bei den Durchläufen für die größere Instanz *no_011_6*, insbesondere mit $\sigma_{max} = 4,5$, wurde deutlich, dass der Anteil an aufgrund überschrittener Zeitbeschränkung abgebrochenen Durchläufen zu hoch war. Da in einer vorigen Implementierungsphase zur Modellierung der Kantenfahrzeiten zunächst die Gammaverteilung gewählt wurde und in manuellen Tests damit keine größeren Laufzeitprobleme aufgetreten sind, wurde diese nochmals als Vergleich eingeführt.

5.4.4 Implementierung

Algorithm 11 generate_travel_time_gamma($t_{(u,v)}$): **returns** double

1: $\alpha \leftarrow \frac{1}{VarK^2}$
2: $\beta \leftarrow t_{(u,v)} \times VarK^2$
3: **return** Gamma(α, β)

[1] https://gitlab.informatik.uni-wuerzburg.de/s484411/dyndarp-simulation-with-probabilistic-travel-times im Ordner tests/csv$_{r}esults$

Um eine knappe Formulierung (Algorithmus 11) zu ermöglichen, werden wir den Variationskoeffizienten $VarK$ als vorab definierbare Steuerung der Variabilität verwenden.
Beispielhaft sind in Abbildung 5.1 und 5.2 analog zum vorigen Kapitel die Wahrscheinlichkeitsdichtefunktionen der neuen realisierten Fahrzeiten $T^{real\,\prime}_{(u,v)}$ visualisiert - mit zwei verschiedenen Variationskoeffizienten. Bei den Abbildungen sollte deutlich werden, dass die Streuungen für hohe Werte von $t_{(u,v)}$ tatsächlich deutlich stärker ausgeprägt sind als bei Abbildung 4.2 - selbst bei $VarK = 0.3$. Für niedrige Werte fallen die Abweichungen hingegen deutlich schwächer aus, zudem ist die Rechtsschiefe weit weniger ausgeprägt.

Kumuliert werden wir es hier, gerade beim Fall $VarK = 0,5$, mit deutlich stärkeren Abweichungen zu tun haben, lediglich extreme Ausreißer bei geringen Fahrzeiten treten weitaus seltener auf.

5.4.5 Ergebnisse

Abschließend ist in Tabelle 5.5 die Erfolgsquote mit den beiden getesteten Umgebungen für die „problematische“ Instanz *no-116-6* dargestellt.

Umgebung	Erfolgreich	Fehlge.	Zeitüberschreitung	Gesamt
Limit von 5 min, $VarK = 0,25$	95	4	1	100
Limit von 5 min, $VarK = 0,5$	90	1	9	100

Tabelle 5.5: Anzahl erfolgreicher und fehlgeschlagener Durchläufe unter Verwendung der Gammaverteilung für die Instanz *no-116-6*

Für die erfolgreichen Durchläufe können wir von einer durchschnittlichen Rechenzeit von 46,47 Sekunden für den Fall $VarK = 0,25$ und 58,23 Sekunden für den Fall $VarK = 0,5$. Es ist äußerst bemerkenswert, dass selbst die in den vorigen Abbildungen deutlich gewordenen starken Streuungen dieser verwendeten Gamma-Parametrisierung nicht annähernd den Einfluss auf die Laufzeit ausüben konnten wie die verwenden Log-Normalverteilungen. Dieser Einfluss der Log-Normalverteilung ist zum aktuellen Zeitpunkt einzig über das häufigere Auftreten von Ausreißern bei geringen Kantenfahrzeiten $t_{(u,v)}$ zu erklären.

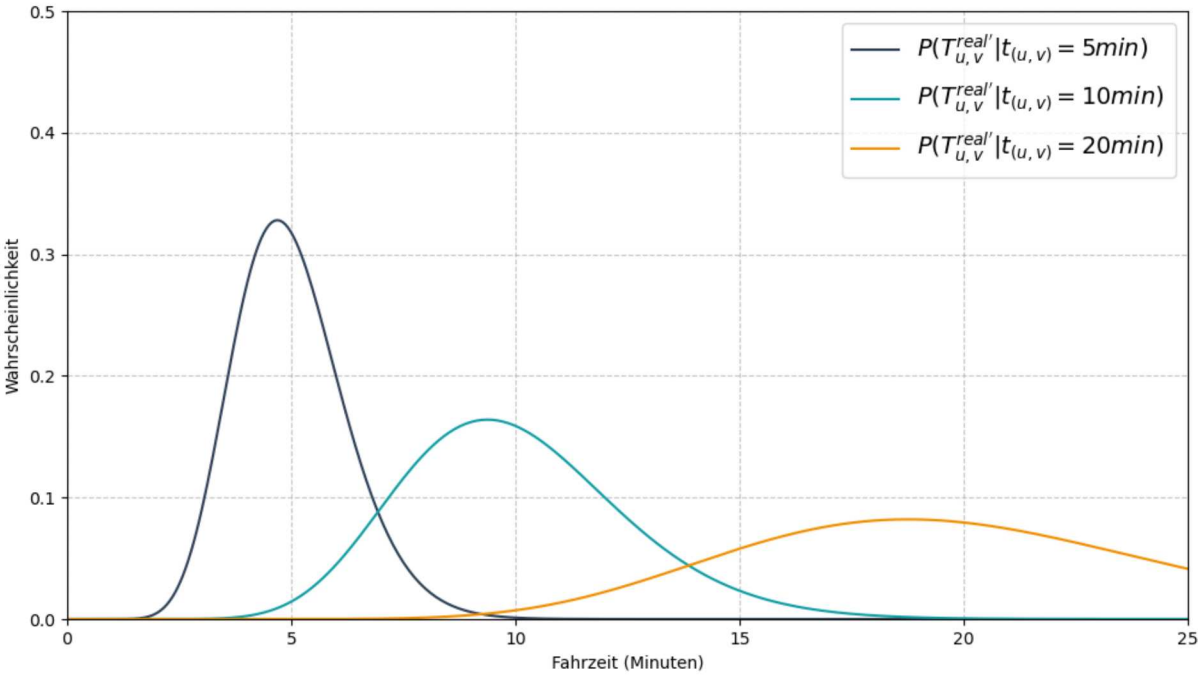


Abbildung 5.1: Wahrscheinlichkeitsdichtefunktion abhängig von vorberechneter Kantenfahrzeit bei $VarK = 0,25$

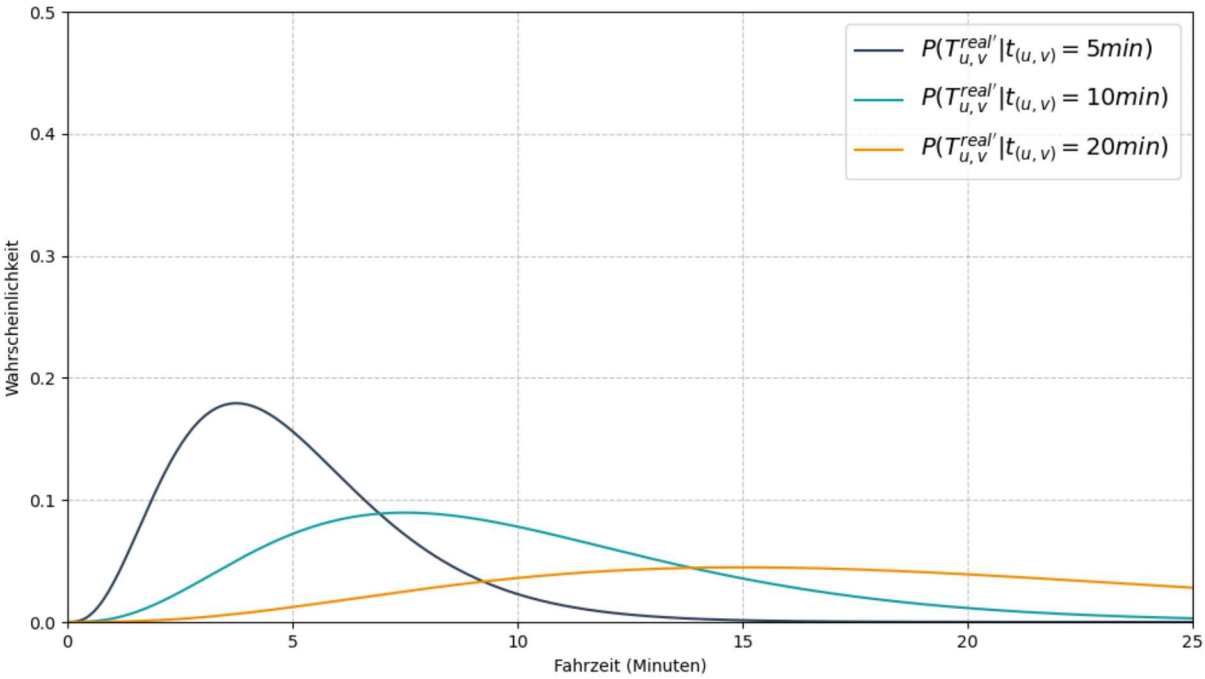


Abbildung 5.2: Wahrscheinlichkeitsdichtefunktion abhängig von vorberechneter Kantenfahrzeit bei $VarK = 0,5$

Kapitel 6

Fazit und Ausblick

Primär hat diese Thesis - insbesondere natürlich die Simulation selbst - gezeigt, dass die vorgeschlagene Vorgehensweise eine ausgesprochen hohe Toleranz gegenüber stärker schwankenden Fahrzeiten aufweist und dabei nur selten zu suboptimalen Routen führt. Bei Ausreißern, mit denen im realen Einsatz immer gerechnet werden muss, gibt es allerdings noch deutliche Laufzeitprobleme.
Andererseits deuten die Ergebnisse aus Kapitel 5 darauf hin, dass „spontane“ Verletzungen von Nebenbedingungen ein größeres Problem als daraus resultierende Verzögerungen darstellen (wobei eine klare Trennung hier schwierig ist), da zum einen Verspätungen ausgeglichen werden können und zum anderen das Modell ausreichend Zeit zur Umplanung besitzt.
Schließlich könnte es sein, dass die vorgeschlagene Aufweichkaskade noch zu undifferenziert vorgeht, indem alle Anfragen gleich behandelt werden und nur nach Nebenbedingungstyp unterschieden wird.

6.1 Mögliche Erweiterungen

6.1.1 Fahrzeiten

Es sind bereits zahlreiche Ansatzpunkte für Erweiterungen zur weiteren Verbesserung der Praxistauglichkeit bekannt, denen allerdings aufgrund des begrenzten Zeitrahmens nicht näher nachgegangen werden konnte. Einschränkende Annahmen, die getroffen wurden, waren die statistische Unabhängigkeit von Kantenfahrzeiten untereinander, was selbstverständlich in der Praxis nicht der Fall ist. Diesem Umstand wird allerdings von den wenigsten Publikationen zu stochastischen Vehicle-Routing-Problemen Rechnung getragen, da eine realistischere Modellierung hier äußerst aufwendig wäre und eine Berücksichtigung des zugrundeliegenden Straßennetzes erfordern würde.
Stattdessen gäbe es möglicherweise ausreichende performante Wege, die tatsächlich auf

einen Routenplanungsdienst mit Live-Verkehrsdaten zurückgreifen. Dieser müsste allerdings speziell auf diesen Zweck ausgelegt sein und in der Lage sein, Unterrouten dynamisch zwischenzuspeichern.

6.1.2 Optimierung der MILP-Modellierung

Hier lassen sich ebenfalls einige Verbesserungsmöglichkeiten ausmachen. Zum einen ist es natürlich ratsam, die Strafkoeffizienten innerhalb der Zielfunktion für verletzte Nebenbedingungen feiner abzustimmen, was allerdings umfangreiche Tests und detaillierte Vorgaben zur Relevanz einzelner Nebenbedingungen notwendig machen würde.
Eine Verbesserung mit vergleichsweise überschaubarem Aufwand sollte die gezieltere Aufweichung von Nebenbedingungen sein. Statt alle Nebenbedingungen eines Typs aufzuweichen, könnte dies auch nur für die miteinander im Konflikt stehenden Nebenbedingungen geschehen.
Eine offensichtliche Einschränkung besteht im Vernachlässigen aller akut betroffenen Nebenbedingungen nach der Realisierung von Knoten/Ereignissen. Um an dieser Stelle „unangenehme Überraschungen“ zu vermeiden, müsste die Modellierung des Optimierungsproblems grundsätzlich in Richtung stochastische Optimierung angepasst werden, was eine Neuformulierung und vermutlich deutliche Performanceeinbußen zur Folge hätte.

6.1.3 Weitere Anregungen

Schließlich gäbe es auch abseits vom Umgang mit probabilistischen Fahrzeiten selbst Potenziale, beispielsweise die Navigation temporär nicht beanspruchter Fahrzeuge. Diese bleiben nach Vorgabe am letzten Absetzknoten, sobald sie mit einer neuen Anfrage betreut werden. Eine Heuristik könnte hier beurteilen, ob dieser Standort strategisch sinnvoll ist, je nach Lage innerhalb des Einzugsgebiets und der Anfragewahrscheinlichkeit im Umkreis dieses Knotens.

τ	(beliebiger) Zeitpunkt
$\tau_i \in \mathcal{T}$	Zeitpunkt i in Zeitpunktliste
$i \in \mathcal{A}$	beliebige Anfrage aus der Anfragegesamtmenge
$k \in K$	Fahrzeug innerhalb der Menge der Fahrzeuge
t_a	(fixe) Fahrzeit der Kante $a \in A(G)$
$\nu_i \in N$	DARP-Knoten des Passagiers $i \in \mathcal{A}$ in DARP-Knotenmenge N ($\|N\| = 2\|A\|$)
	- bildet ein Tupel aus den nachfolgenden Elementen:
$[\nu_i ::] e_\nu$	Beginn des Zeitfensters für Ein- bzw. Ausstieg des Passagiers i,
	der zum DARP-Knoten ν gehört (mithin $\nu = i^+$ oder $\nu = i^-$)
$[\nu_i ::] l_\nu$	Ende des Zeitfensters für Ein- bzw. Ausstieg des Passagiers i,
	der zum DARP-Knoten ν gehört (mithin $\nu = i^+$ oder $\nu = i^-$)
$[\nu_i ::] s_i$	Servicezeit für Einstieg und Ausstieg des Passagiers i
$[\nu_i ::] q_i$	benötigte Sitzplätze des Passagiers i
$[\nu_i ::] L_i$	maximale Reisezeit des Passagiers i

Tabelle 6.1: Aus der Instanz direkt oder indirekt resultierende Objekte

$\mathcal{N}(\tau)$	neu aufgetretene Anfragen zum Zeitpunkt τ
$\mathcal{S}(\tau)$	zum Zeitpunkt τ bereits eingeplante Anfragen
$\mathcal{R}(\tau)$	zum Zeitpunkt τ abgelehnte Anfragen
$\mathcal{P}(\tau)$	zum Zeitpunkt τ im Fahrzeug befindliche Passagiere/Anfragen
$\mathcal{D}(\tau)$	zum Zeitpunkt τ abgesetzte Passagiere/Anfragen
$\mathcal{S}^{fix}(\tau)$	zum Zeitpunkt τ in der Abholung befindliche Anfragen (Fahrzeug auf eingehender Kante des aktiven Abholknotens)
$\mathcal{P}^{fix}(\tau)$	zum Zeitpunkt τ in der Absetzung befindliche Anfragen (Fahrzeug auf eingehender Kante des aktiven Absetzknotens)
$v^{active}(\nu)$	realisierter Knoten für Einstieg/Ausstieg des zu Passagier i gehörenden DARP-Knotens (mithin $\nu = i^+$ oder $\nu = i^-$)
$a^{active}(\nu)$	realisierte Kante resultierend in Einstieg/Ausstieg von Passagier i
$T^{real}_{(u,v)}$	realisierte Fahrzeit der Kante $(u, v) \in A(G)$
$x^{fixed}_{(u,v)}$	Binärvariable zur Indizierung der Festschreibung der Kante (u, v)
$x^{real}_{(u,v)}$	Binärvariable zur Indizierung der Realisierung der Kante $(u, v) \in A(G)$
T^{fix}_v	festgeschriebener Zeitpunkt des Eintreffens bei $v \in V(G)$
T^{real}_v	realisierter Ereigniszeitpunkt/Dienstbeginn bei $v \in V(G)$

Tabelle 6.2: Legende für simulationsspezifische Objekte

$G(\tau) = (V(\tau), A(\tau))$	Ereignisgraph G mit Knoten- und Kantenmenge zum Zeitpunkt τ
V_{i^+}, V_{i^-}	Ein- und Ausstiegsknoten zu Anfrage i
ω_c	Gewichtung/Bestrafung für Kriterium c in der Zielfunktion
$\Pi(\tau)$	Aktionsplan zum Zeitpunkt τ, besteht aus nachfolgenden Elementen
$[\Pi(\tau) ::]T_v$	vom Modell berechneter Dienstbeginn am Knotens v
$[\Pi(\tau) ::]x_a$	Entscheidungsvariable für Kante $a \in A(G)$
$[\Pi(\tau) ::]p_i$	Entscheidungsvariable für Passagier $i \in \mathcal{A}$
$[\Pi(\tau) ::]d_i$	überschüssige Reisezeit für alle akzeptierten Anfragen $i \in \mathcal{A} : p_i = 1$
$[\Pi(\tau) ::]\gamma_i^{+\|-}$	kommunizierte Abhol- und Absetzeit für Passagier $i \in \mathcal{A} : p_i = 1$
Γ	erwünschte obere Grenze der Abholverzögerung
$[\Pi(\tau) ::]C_k$	Route des Fahrzeugs $k \in K$

Tabelle 6.3: Legende für modellierungsspezifische Objekte

.1 Lebenszyklus einer Anfrage

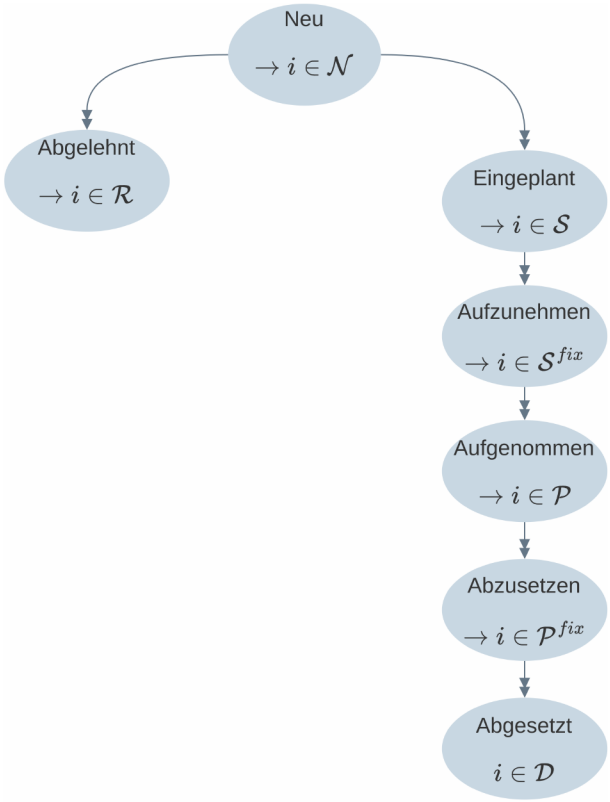


Abbildung 1: Visualisierung der Interaktion zwischen Simulation und Modellierungseinheit

.2 UML-Diagramme

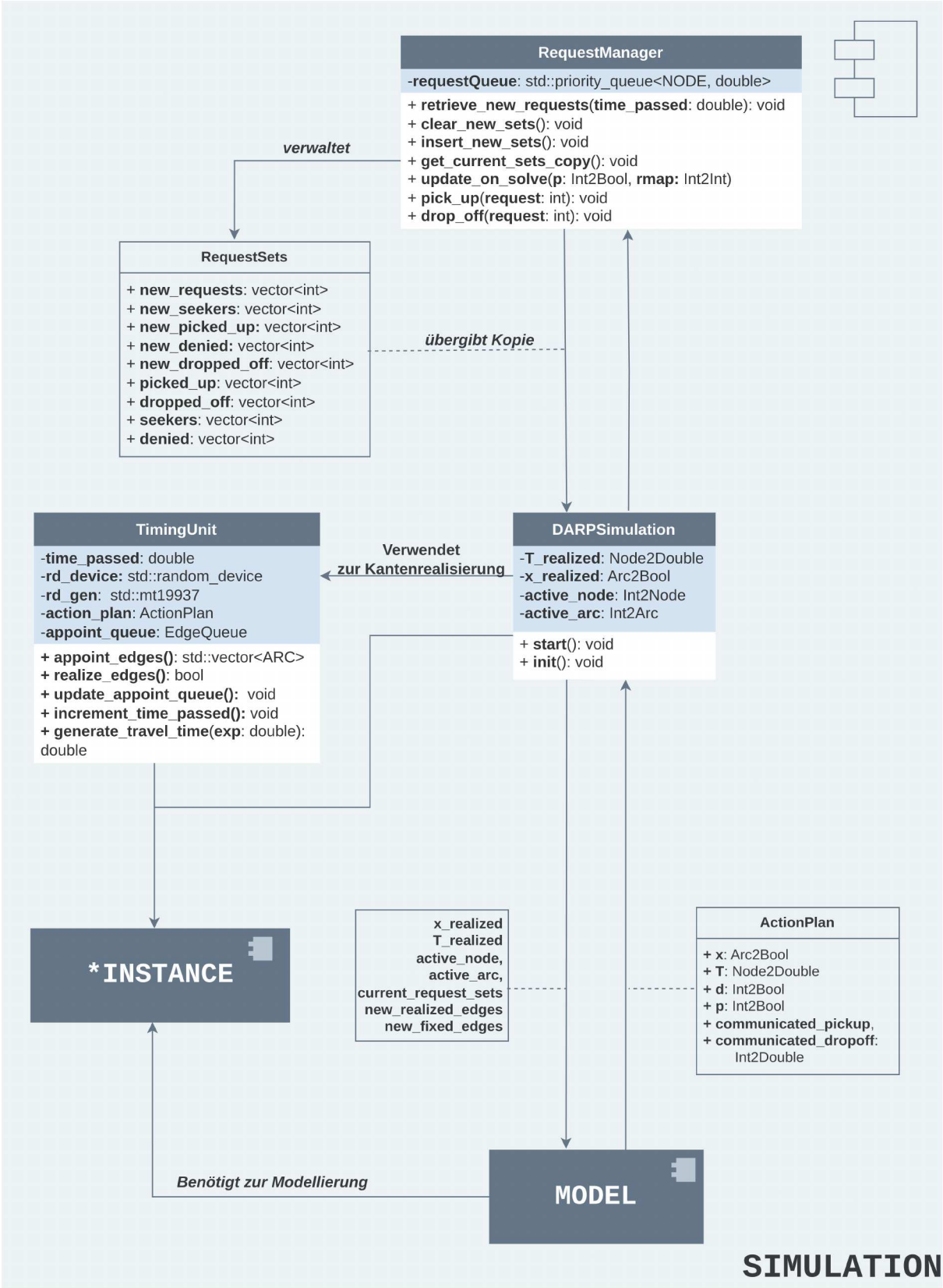
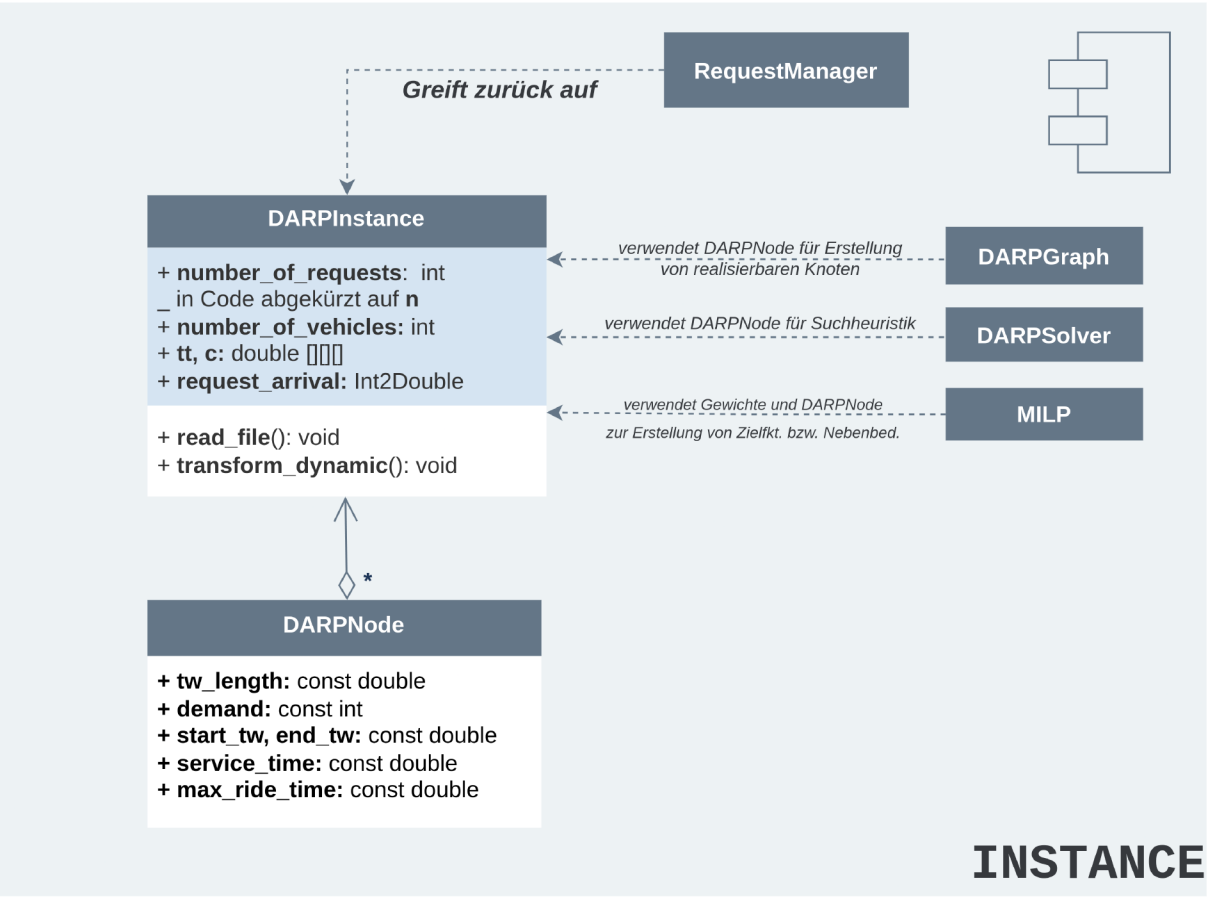


Abbildung 2: UML-Diagramm des Simulationsmoduls



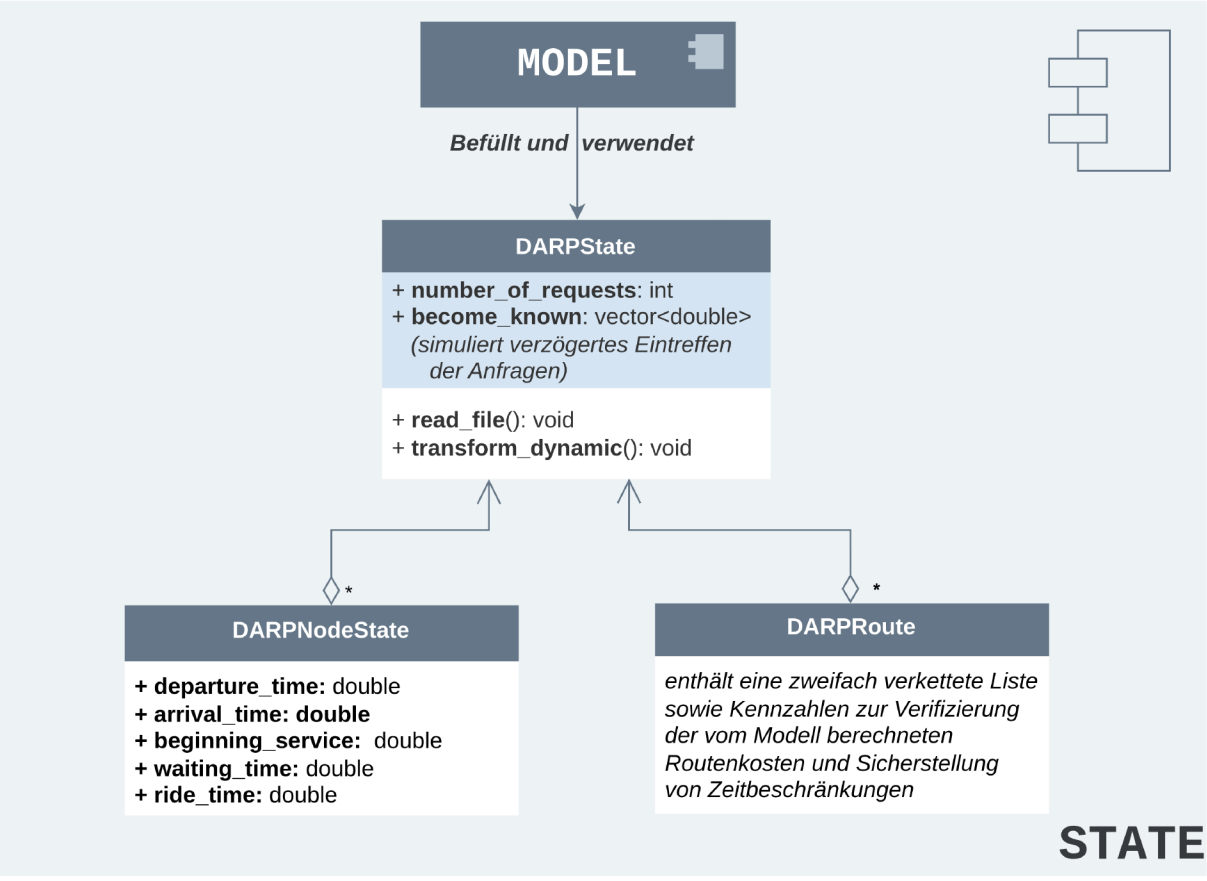


Abbildung 3: UML-Diagramm des Instanz- und Modellierungsinstanzmoduls

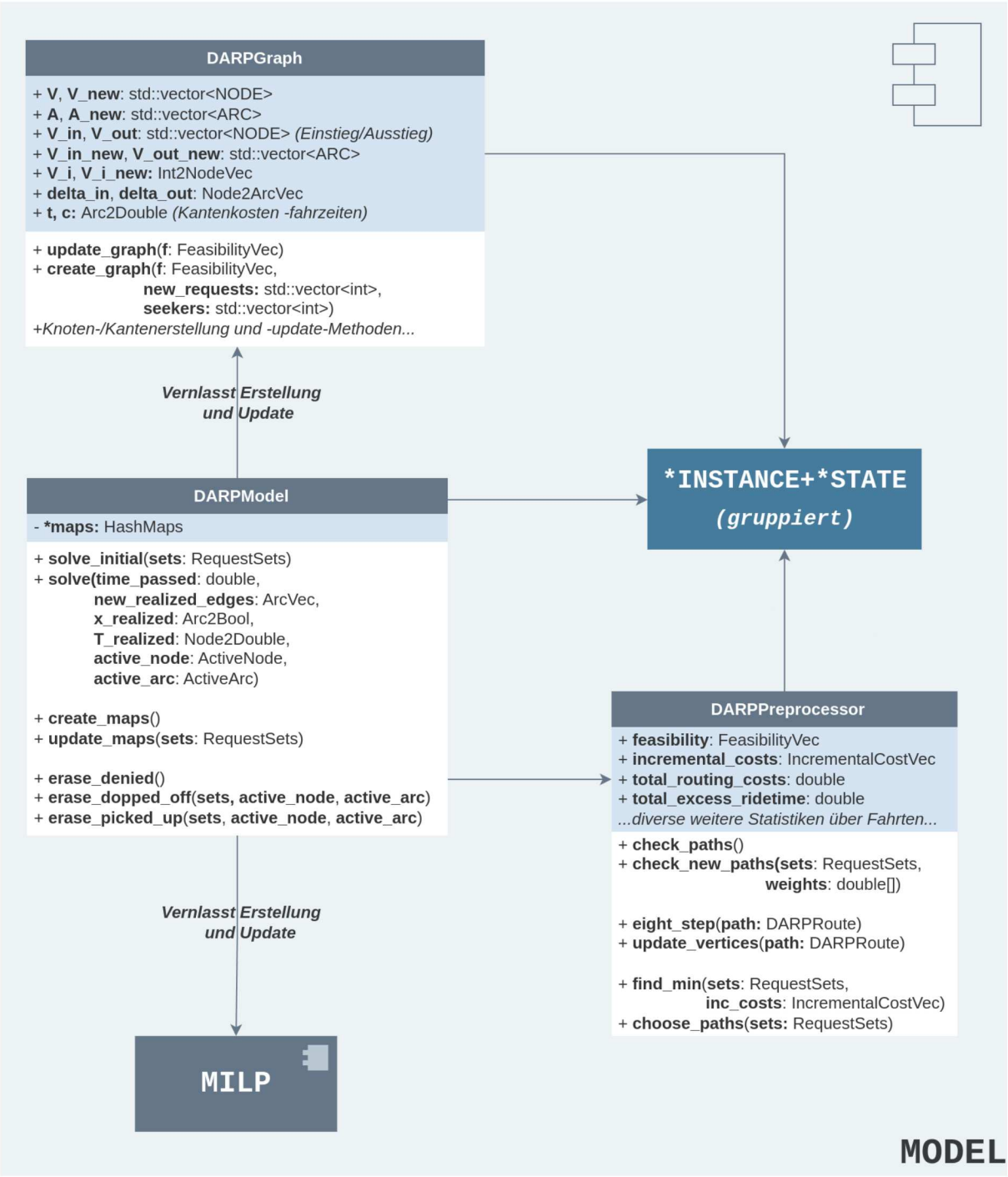


Abbildung 4: UML-Diagramm der Modellierungseinheit

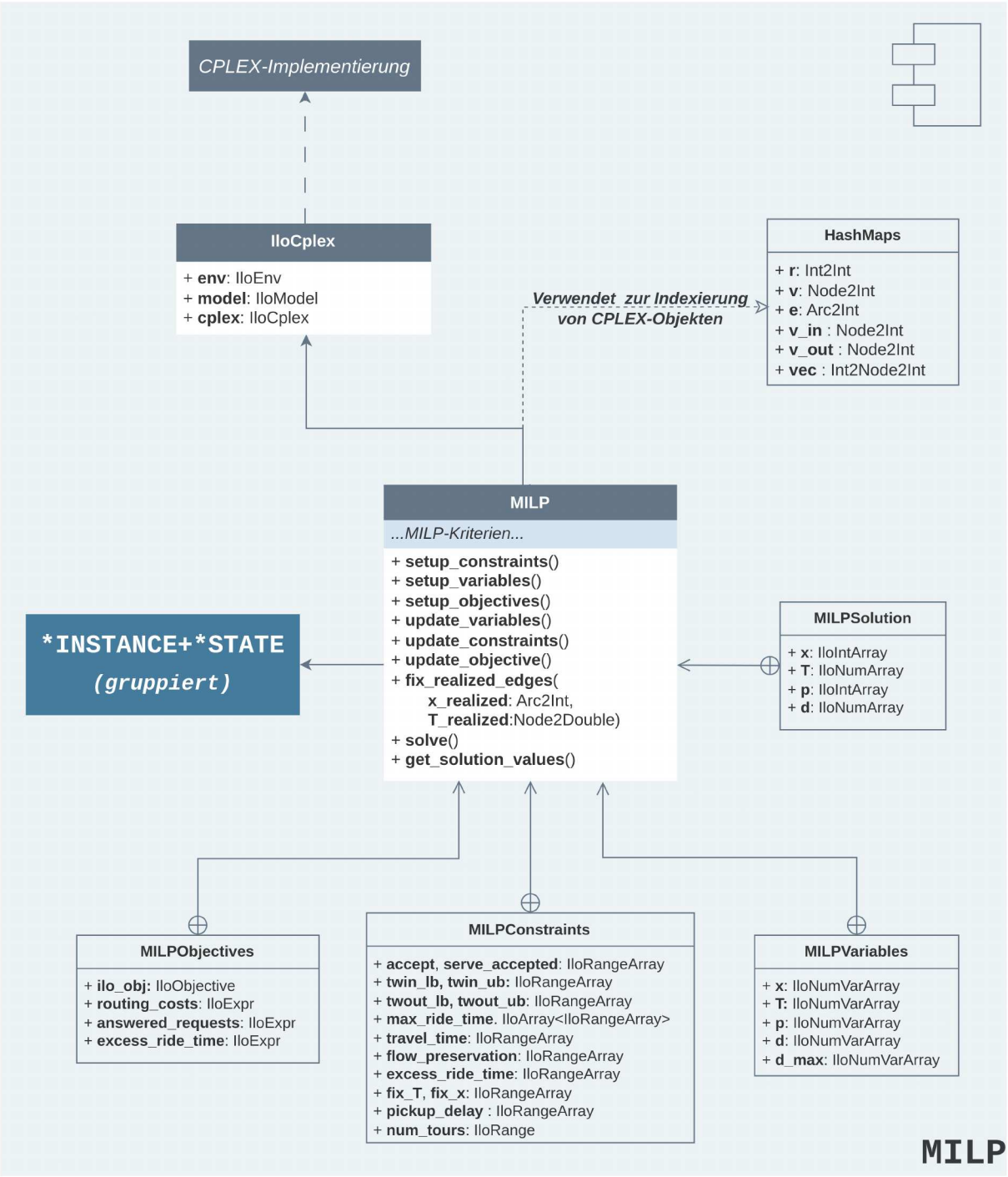


Abbildung 5: UML-Diagramm des MILP-Moduls (aus Gründen der Übersichtlichkeit ohne elastische Nebenbedingungen)

NODE	`std::array<int, S>`
ARC	`std::pair<NODE<S>, NODE<S>>`
ActiveNode	`std::pair<NODE<S>, NODE<S>>`
ActiveArc	`std::unordered_map<int, ARC<S>>`
FeasibilityVec	`std::vector<std::vector<std::array<bool,2>>>`
IncrementalCostVec	`std::vector<std::vector<std::array<bool,2>>>`
Arc2Bool	`std::unordered_map<ARC<S>, bool, HashFunction<S>>`
Node2Double	`std::unordered_map<ARC<S>, double, HashFunction<S>>`
Node2ArcVec	`std::unordered_map<NODE<S>, std::vector<ARC<S>>, HashFunction<S>>`
Int2Int	`std::unordered_map<int, int>`
Node2Int	`std::unordered_map<NODE<S>, uint64_t, HashFunction<S>>`
Arc2Int	`std::unordered_map<ARC<S>, uint64_t, HashFunction<S>>`
Int2Node2Int	`std::unordered_map<int, Node2Int>`
EdgeQueue	`std::priority_queue<TimedEdge<Q>, std::vector<TimedEdge<Q>>, TimeComparator>`
RequestQueue	`std::priority_queue<Request, std::vector<Request>, TimeComparator>`
Request	`struct {int nr, double time}`
TimedEdge<S>	`struct {ARC<S> arc, double time}`
TimeComparator	*Komparatorstruktur; ordnet zwei Anfragen oder Kanten nach verknüpfter Zeit ein*
***Type**	kurz für `std::shared_ptr<Type>`

Abbildung 6: Legende für UML-Diagramme

.3 Ablaufdiagramm Simulation

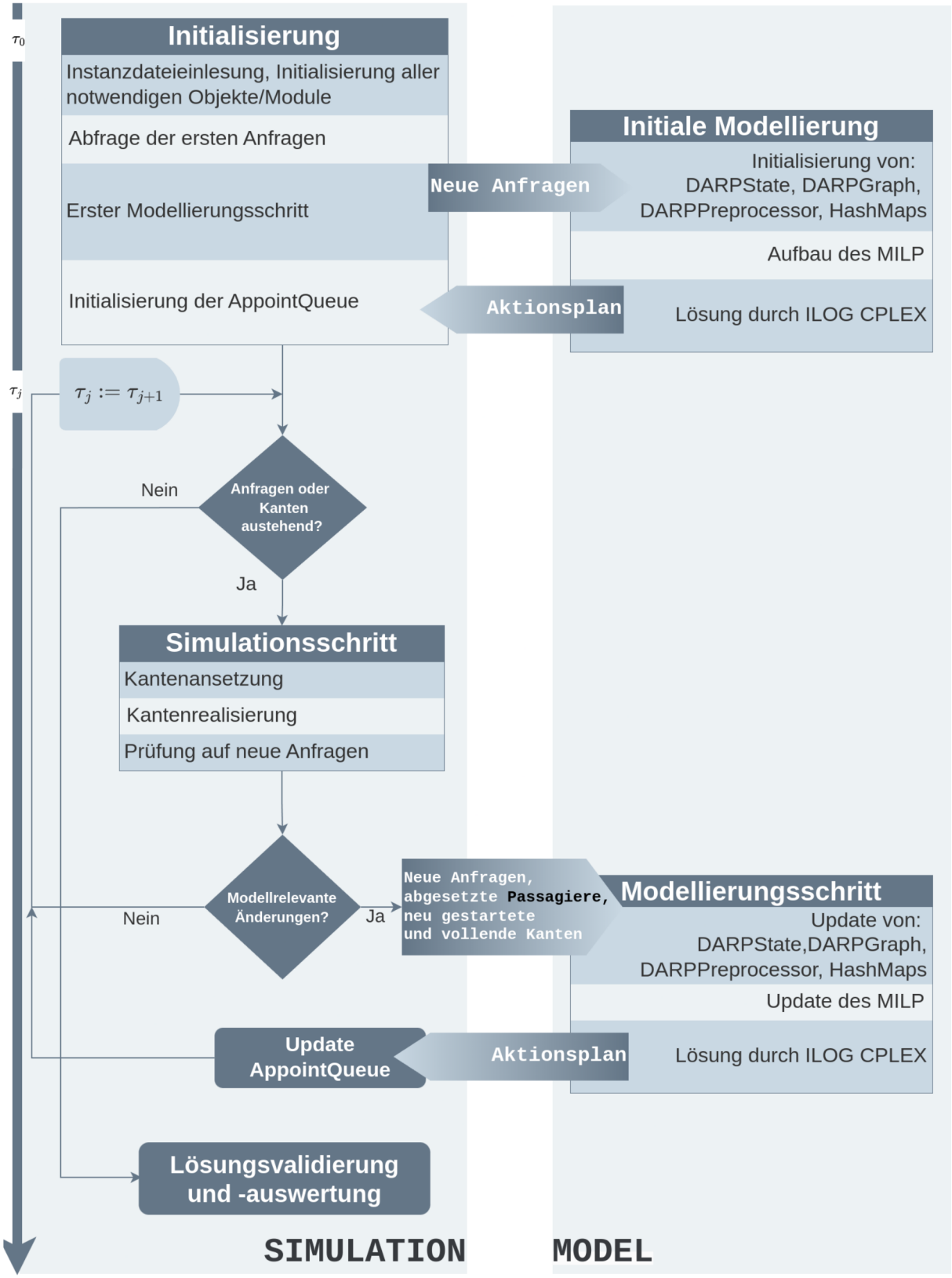


Abbildung 7: Visualisierung der Interaktion zwischen Simulation und Modellierungseinheit

.4 Screenshots

Abbildung 8: Allgemeiner Screenshot der Ausgabe eines Modellierungs- und mehrerer Simulationsschritte mit dem Ansetzen (*Vehicle has started...*) und dem Realisieren (*Vehicle arrived at...*) von Kanten bzw. Knoten sowie dem Eingang neuer Anfragen.
Die Ausgabe des Modellierungsschritts beschreibt für jedes Fahrzeug die Abfolge von Einsteigen(+) und Ausstiegen(-) in folgendem Format: [+|-][Anfrage_Nr]][Zeit in Minuten].
Erster und letzter Block entspricht dem Start bzw. Ende der Route am Depot.
Gelb hinterlegte Blöcke sind noch nicht realisiert, die Zeitangaben oder Abfolgen können sich also noch ändern.

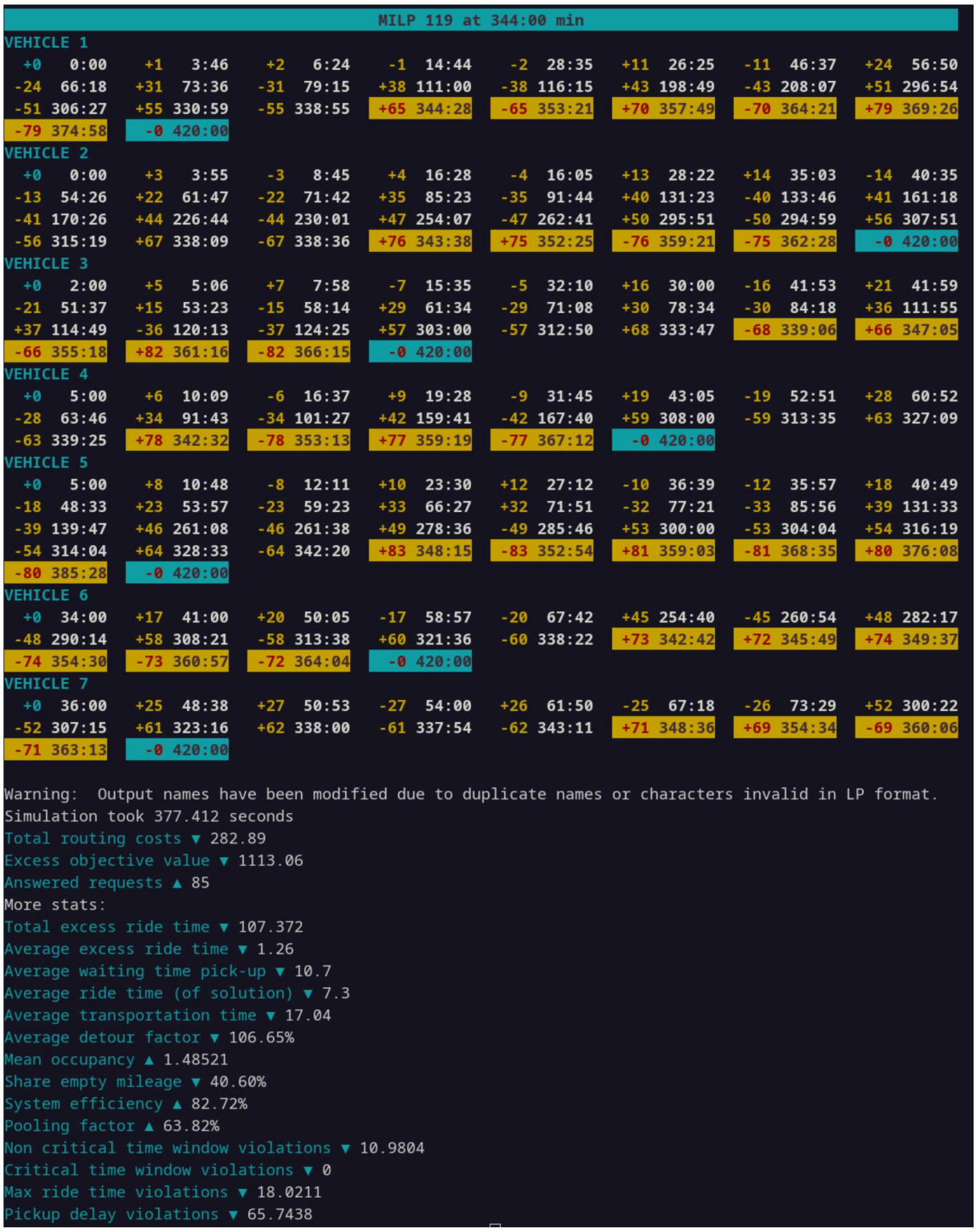


Abbildung 9: Erfolgreicher Durchlauf mit abschließender Anzeige der Kennzahlen.
Pfeile nach oben zeigen an, dass der zugehörige Wert vorzugsweise höher, Pfeile nach unten, dass der Wert vorzugsweise niedriger seien sollte.

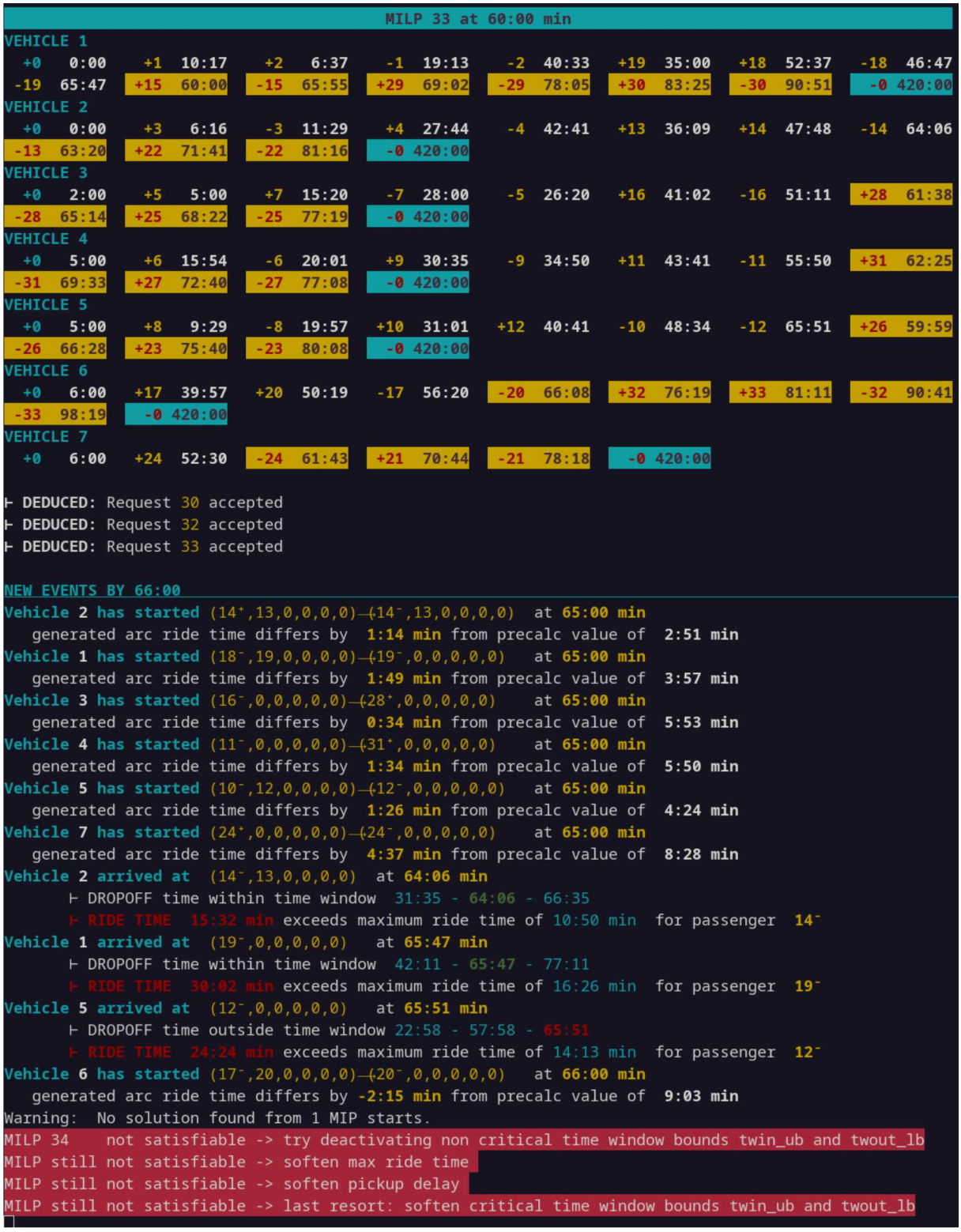
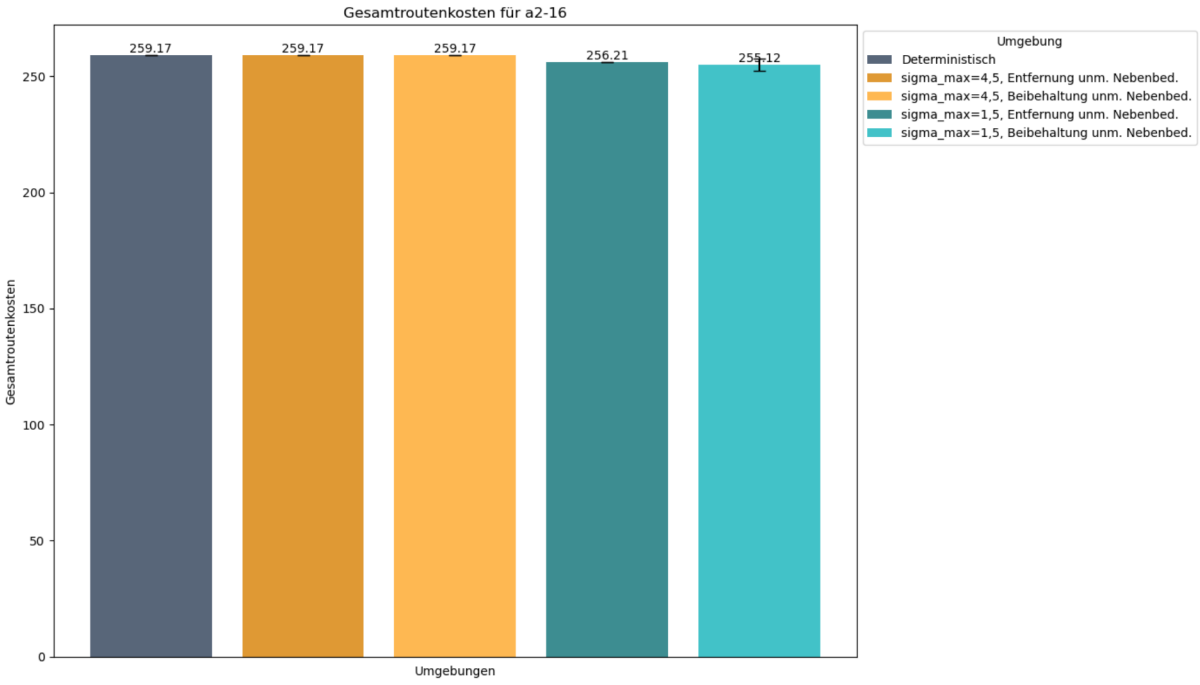
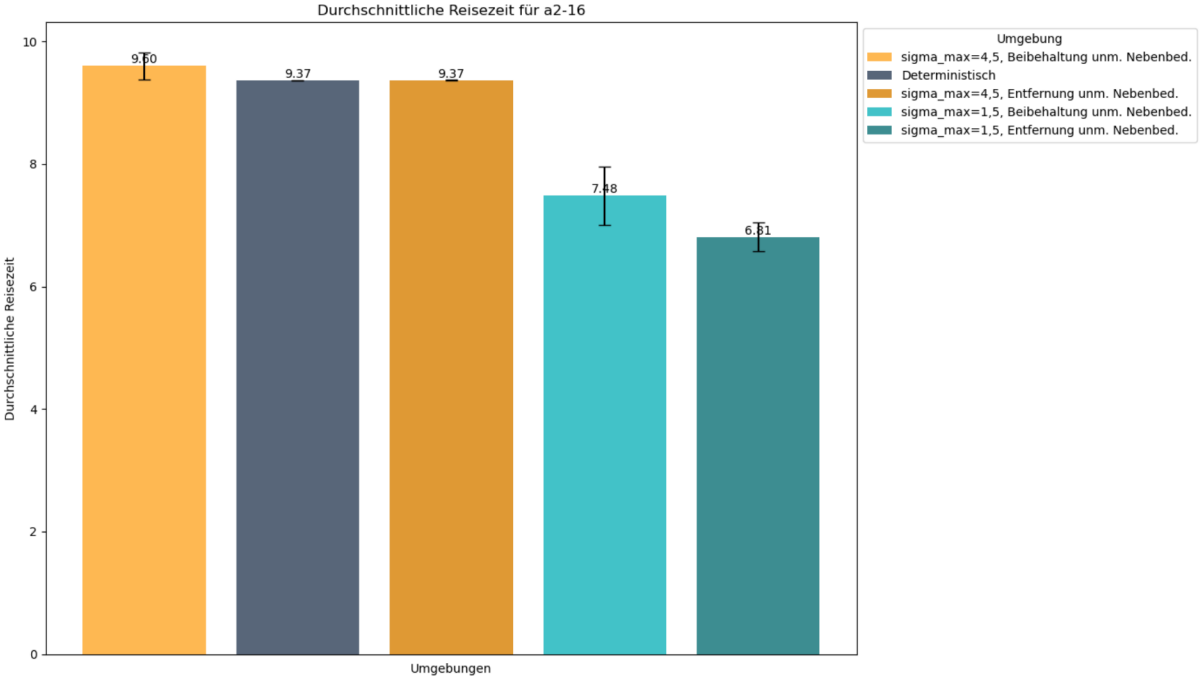


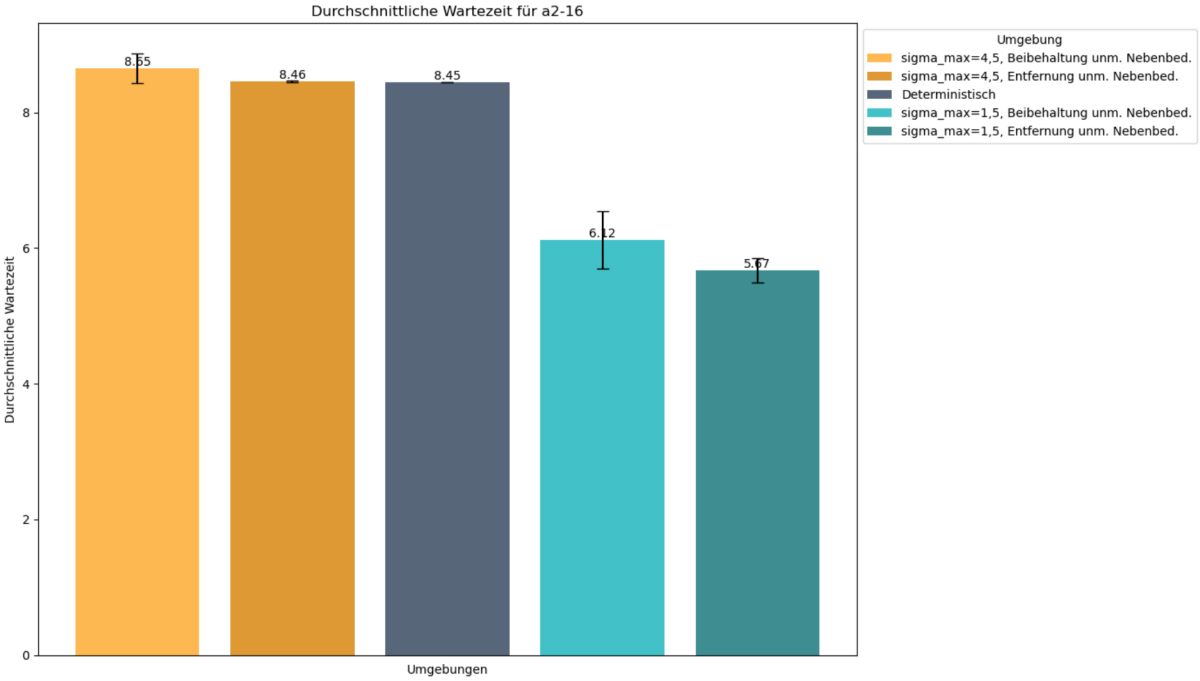
Abbildung 10: Vollständige Aufweichkaskade nach noch nicht lösbarem MILP - provoziert durch einen erhöhten Wert von $\sigma_{max} = 4,5$. Zu beachten sind auch die bereits im Vorfeld verletzten Zeitfenster und Maximalfahrzeiten (welche hauptsächlich vorigen Aufweichungen geschuldet sind).

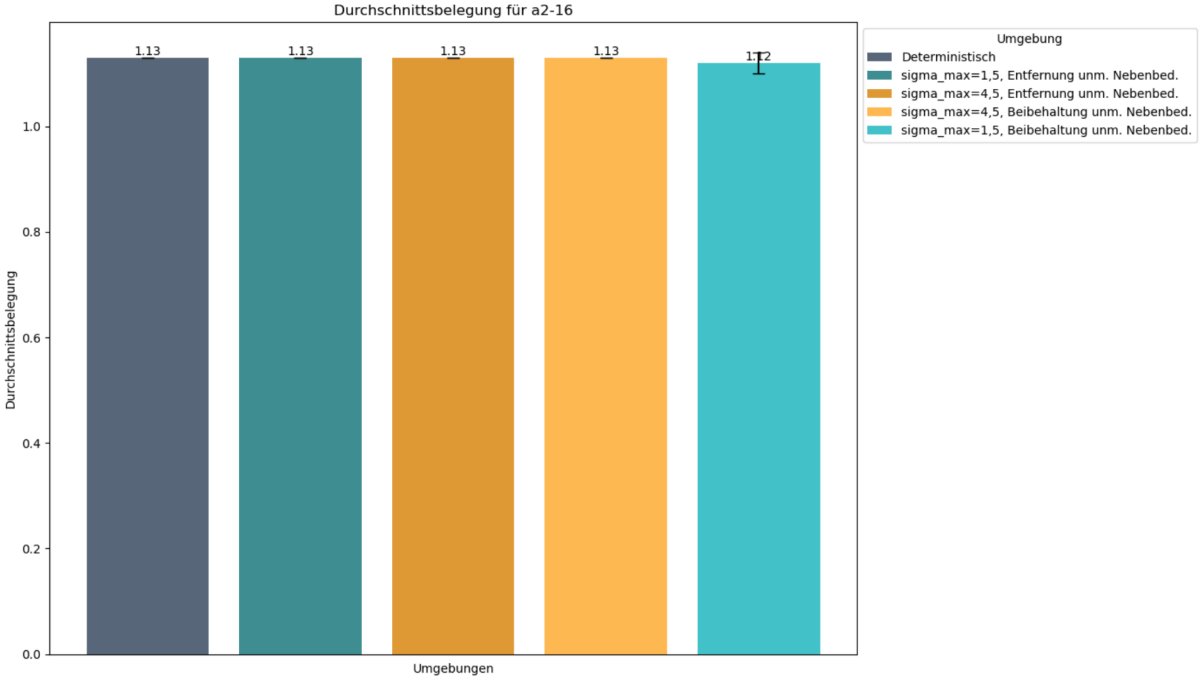
.5 Detaillierte Ergebnisse zu Kapitel 4

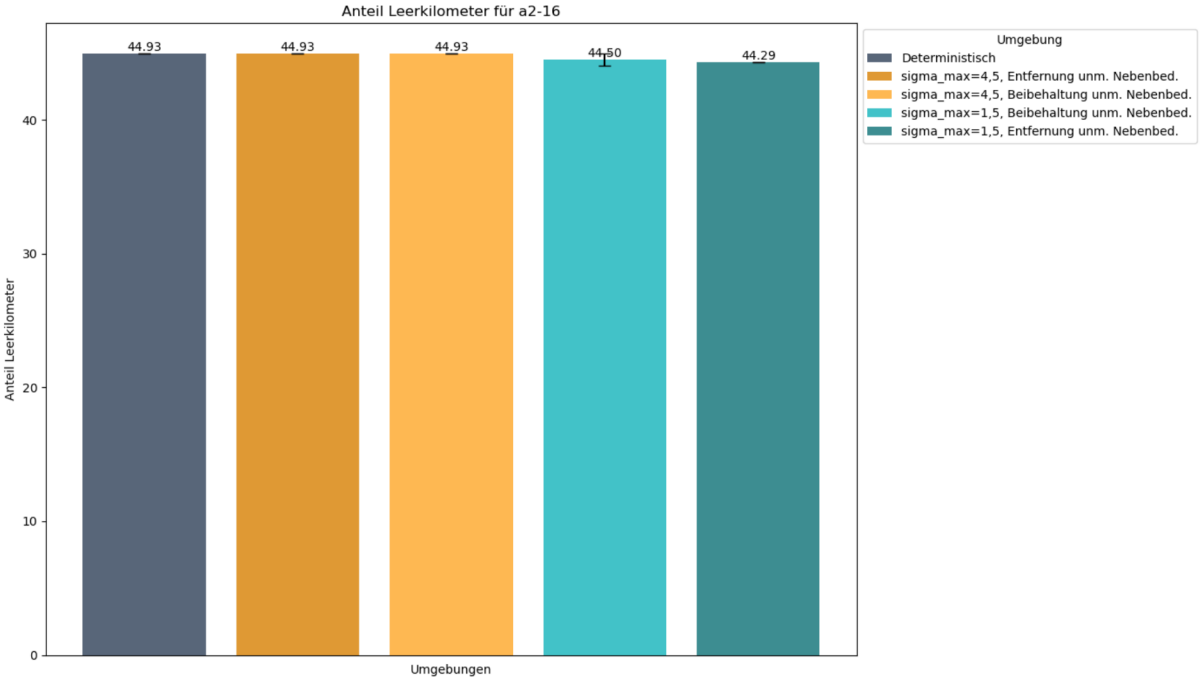
.5.1 Instanz *a2-16*

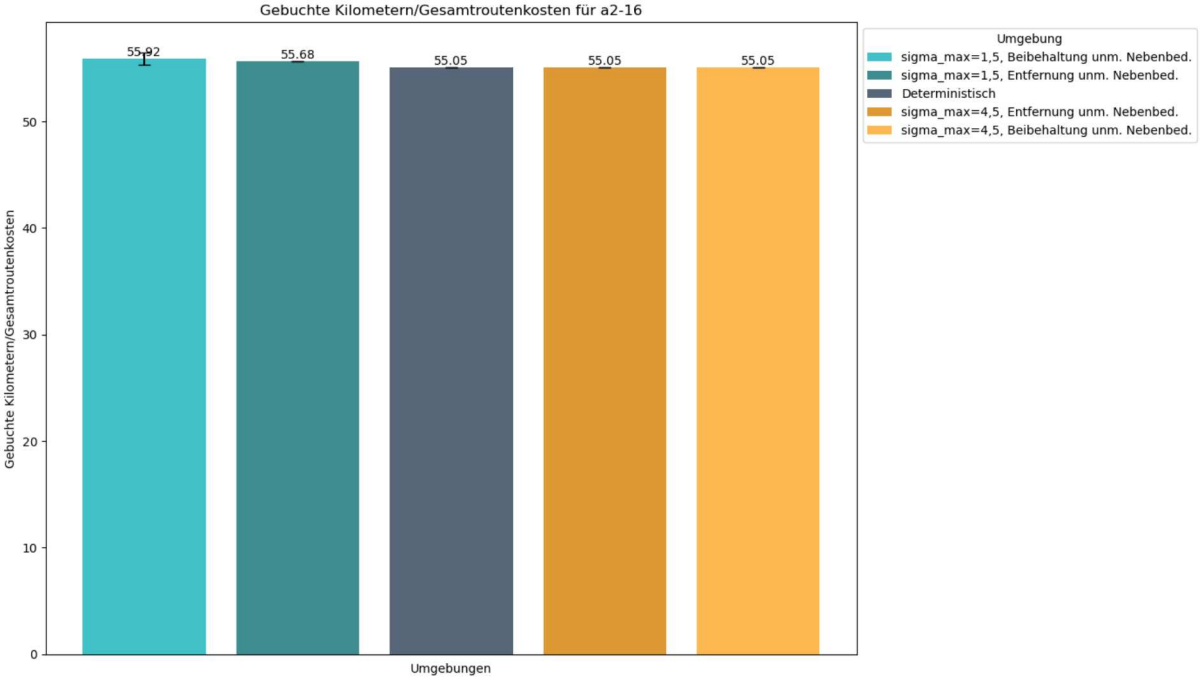


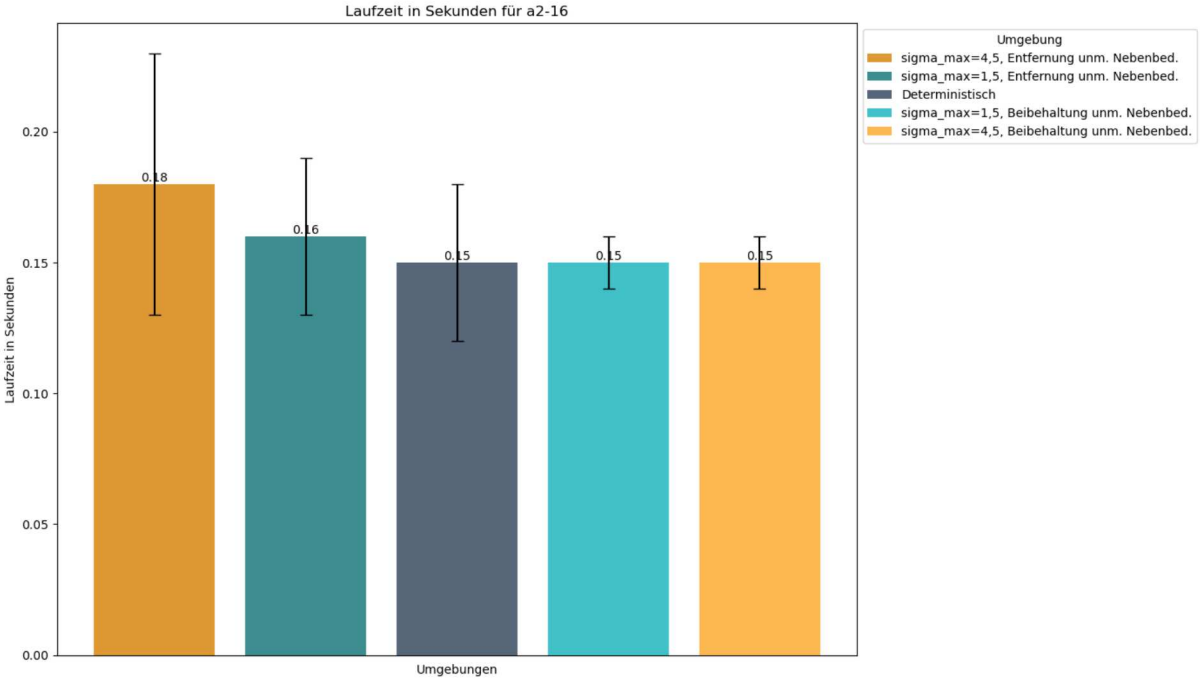

Durchschnittliche Reisezeit für a2-16
Durchschnittliche Reisezeit
Umgebungen
9.60
9.37
9.37
7.48
6.81
Umgebung
sigma_max=4,5, Beibehaltung unm. Nebenbed.
Deterministisch
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=1,5, Beibehaltung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.

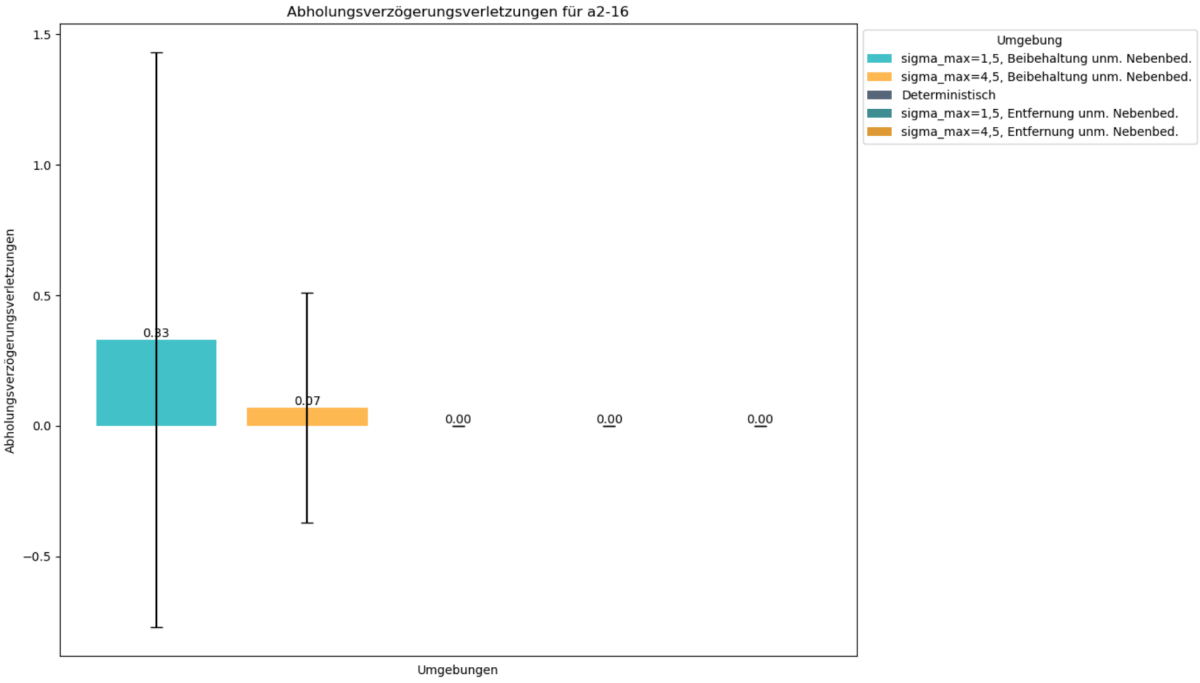

Durchschnittliche Wartezeit für a2-16
Durchschnittliche Wartezeit
Umgebungen
8.55
8.46
8.45
6.12
5.67
Umgebung
sigma_max=4,5, Beibehaltung unm. Nebenbed.
sigma_max=4,5, Entfernung unm. Nebenbed.
Deterministisch
sigma_max=1,5, Beibehaltung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.


Durchschnittsbelegung für a2-16
Durchschnittsbelegung
Umgebungen
1.13
1.13
1.13
1.13
1.12
0.0
0.2
0.4
0.6
0.8
1.0
Umgebung
Deterministisch
sigma_max=1,5, Entfernung unm. Nebenbed.
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=4,5, Beibehaltung unm. Nebenbed.
sigma_max=1,5, Beibehaltung unm. Nebenbed.

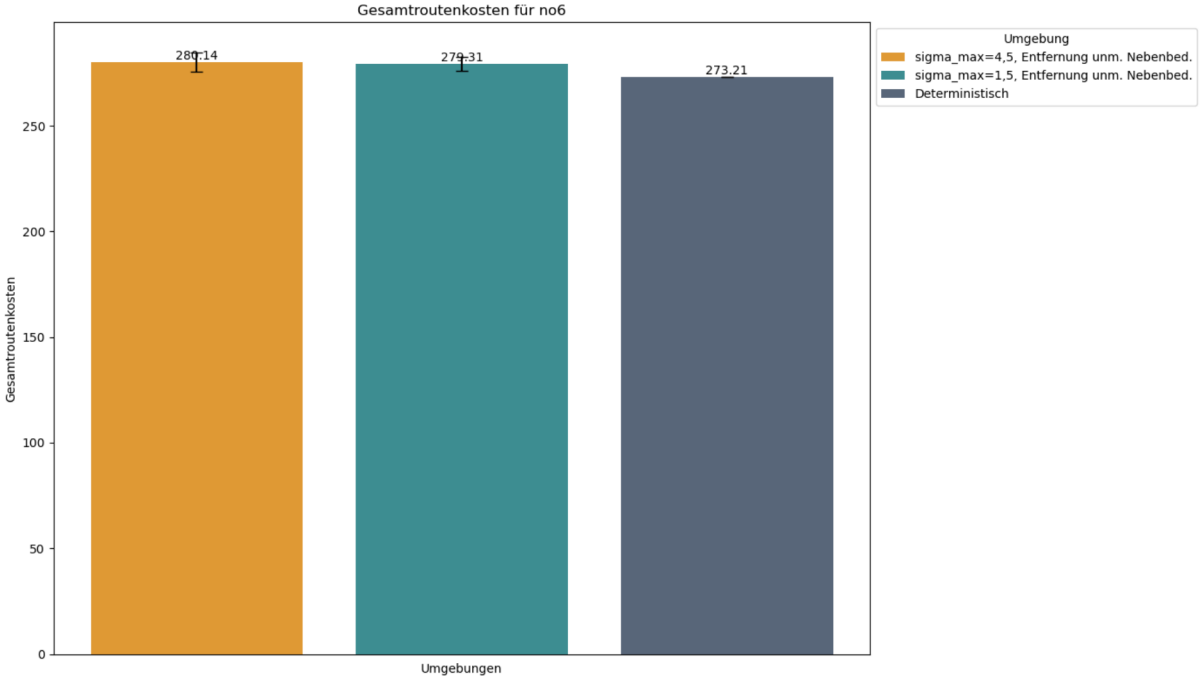

Anteil Leerkilometer für a2-16
Anteil Leerkilometer
Umgebungen
44.93
44.93
44.93
44.50
44.29
0
10
20
30
40
Umgebung
Deterministisch
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=4,5, Beibehaltung unm. Nebenbed.
sigma_max=1,5, Beibehaltung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.

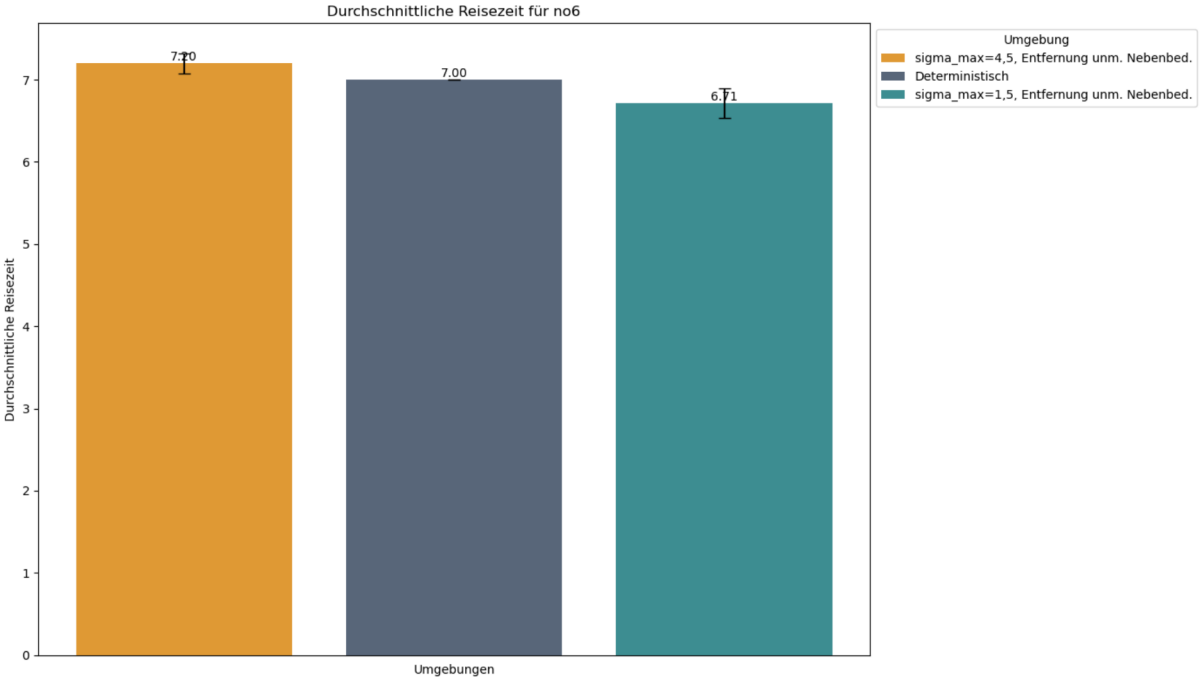

Gebuchte Kilometern/Gesamtroutenkosten für a2-16
Gebuchte Kilometern/Gesamtroutenkosten
Umgebungen
55.92
55.68
55.05
55.05
55.05
0
10
20
30
40
50
Umgebung
sigma_max=1,5, Beibehaltung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.
Deterministisch
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=4,5, Beibehaltung unm. Nebenbed.

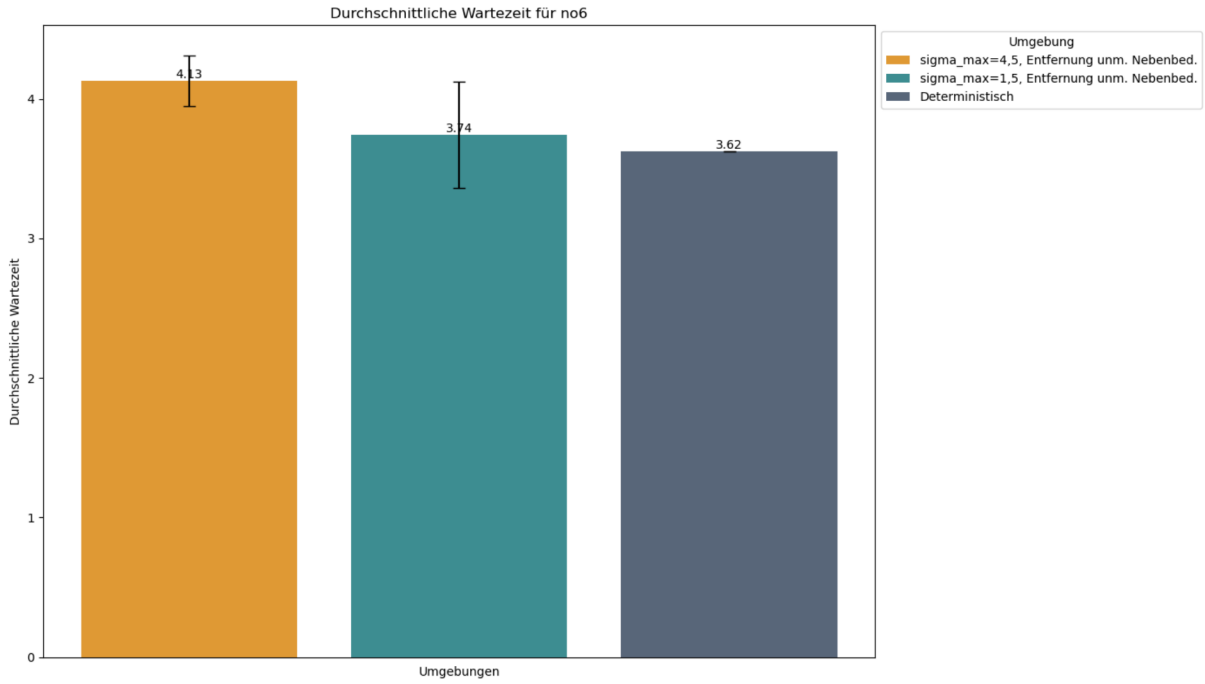

Laufzeit in Sekunden für a2-16
Laufzeit in Sekunden
Umgebungen
0.18
0.16
0.15
0.15
0.15
0.00
0.05
0.10
0.15
0.20
Umgebung
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.
Deterministisch
sigma_max=1,5, Beibehaltung unm. Nebenbed.
sigma_max=4,5, Beibehaltung unm. Nebenbed.

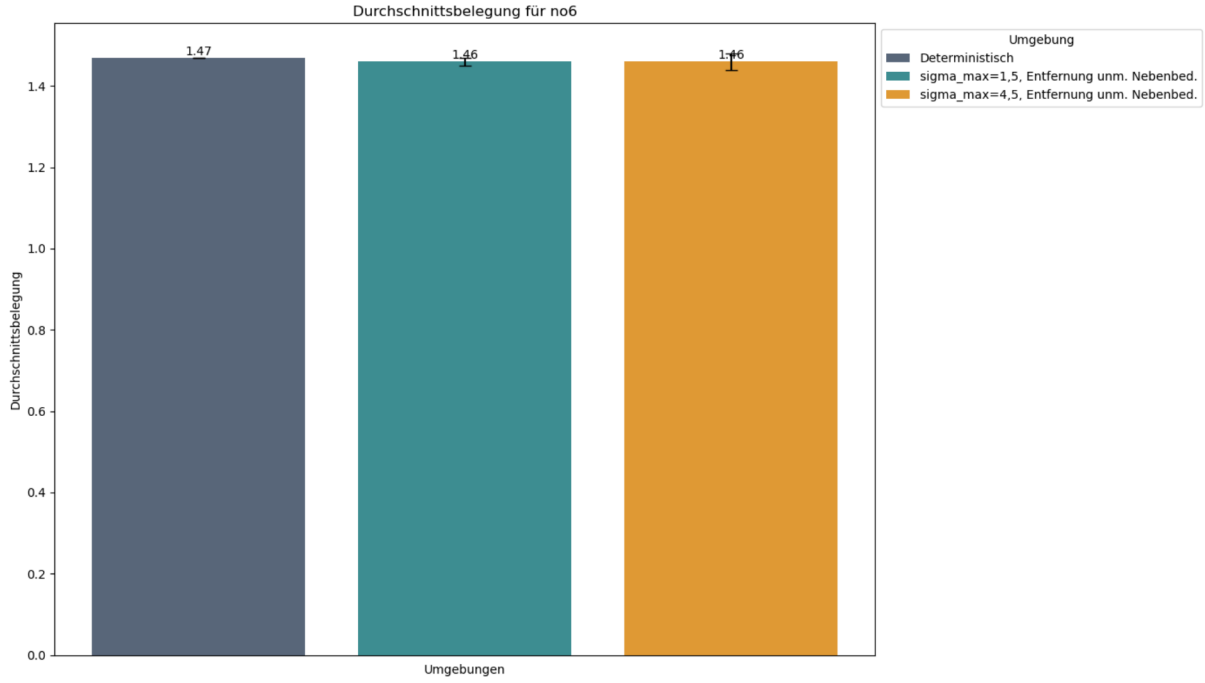

Abholungsverzögerungsverletzungen für a2-16
Abholungsverzögerungsverletzungen
Umgebungen
1.5
1.0
0.5
0.0
−0.5
0.33
0.07
0.00
0.00
0.00
Umgebung
sigma_max=1,5, Beibehaltung unm. Nebenbed.
sigma_max=4,5, Beibehaltung unm. Nebenbed.
Deterministisch
sigma_max=1,5, Entfernung unm. Nebenbed.
sigma_max=4,5, Entfernung unm. Nebenbed.

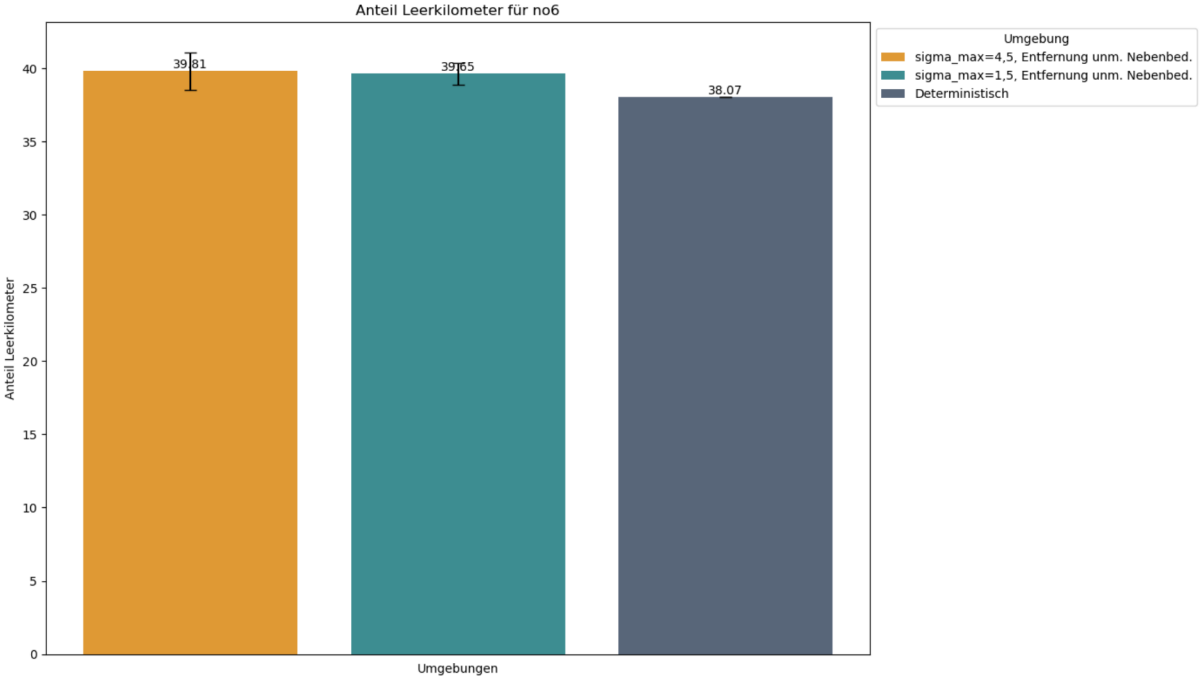
.5.2 Instanz *no_116_6*

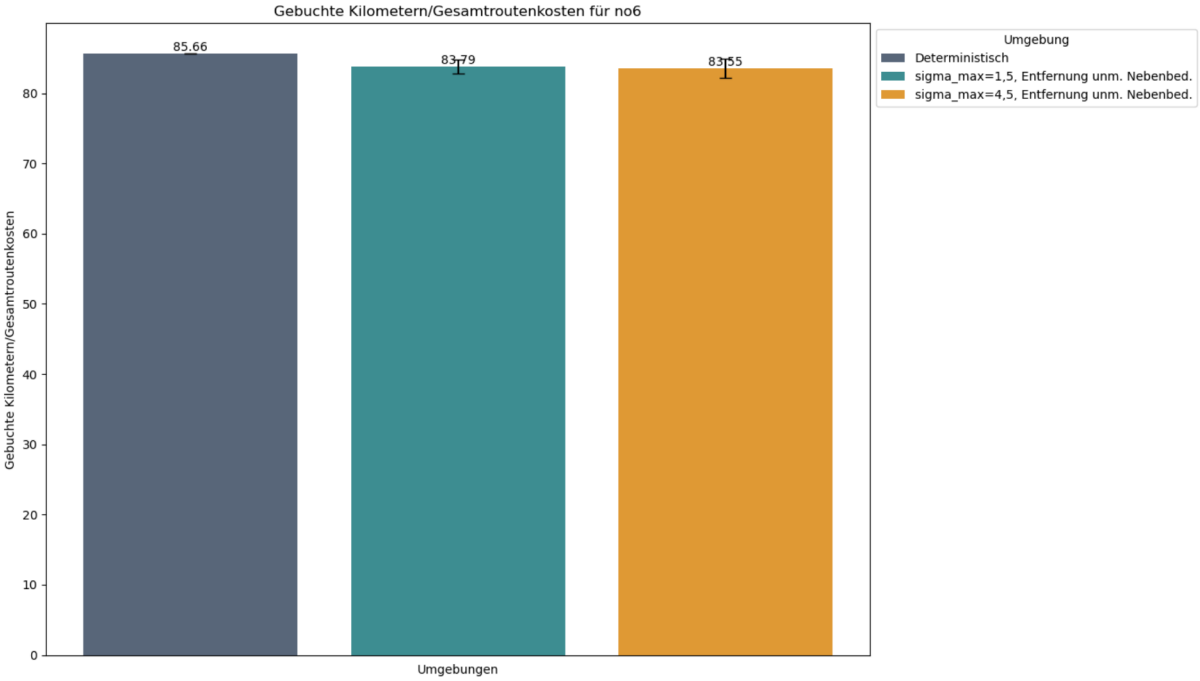

Gesamtroutenkosten für no6
Gesamtroutenkosten
Umgebungen
Umgebung
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.
Deterministisch
280.14
279.31
273.21
0
50
100
150
200
250

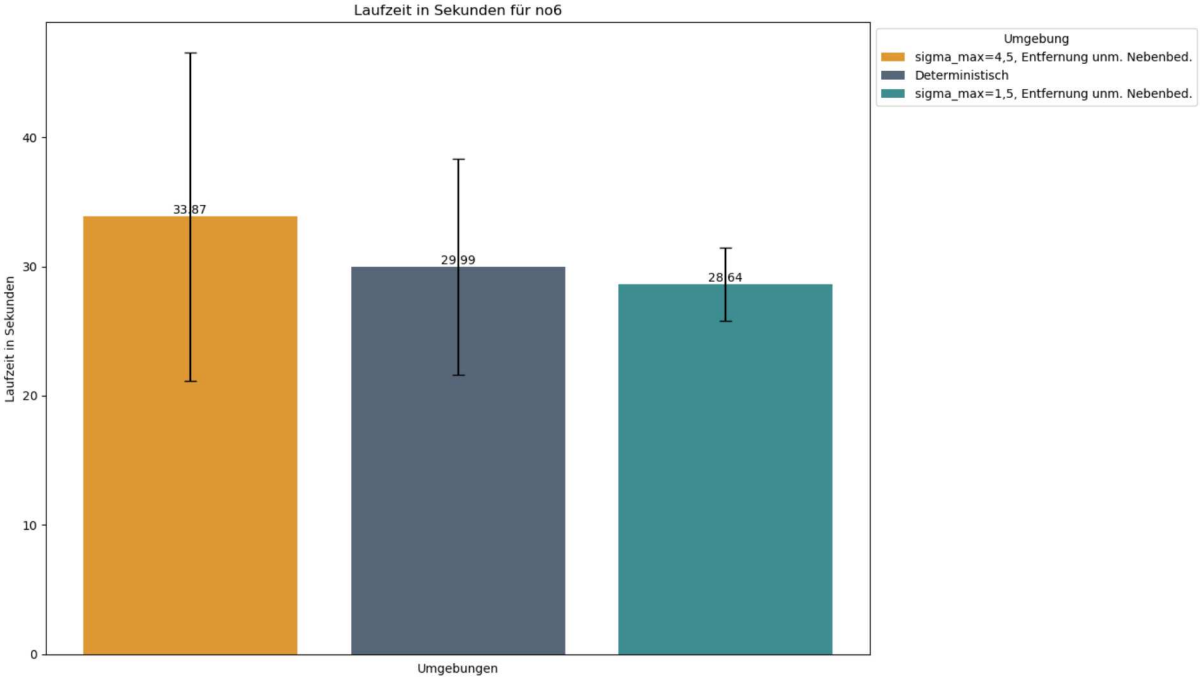

Durchschnittliche Reisezeit für no6
Durchschnittliche Reisezeit
Umgebungen
Umgebung
sigma_max=4,5, Entfernung unm. Nebenbed.
Deterministisch
sigma_max=1,5, Entfernung unm. Nebenbed.
7.00
0
1
2
3
4
5
6
7

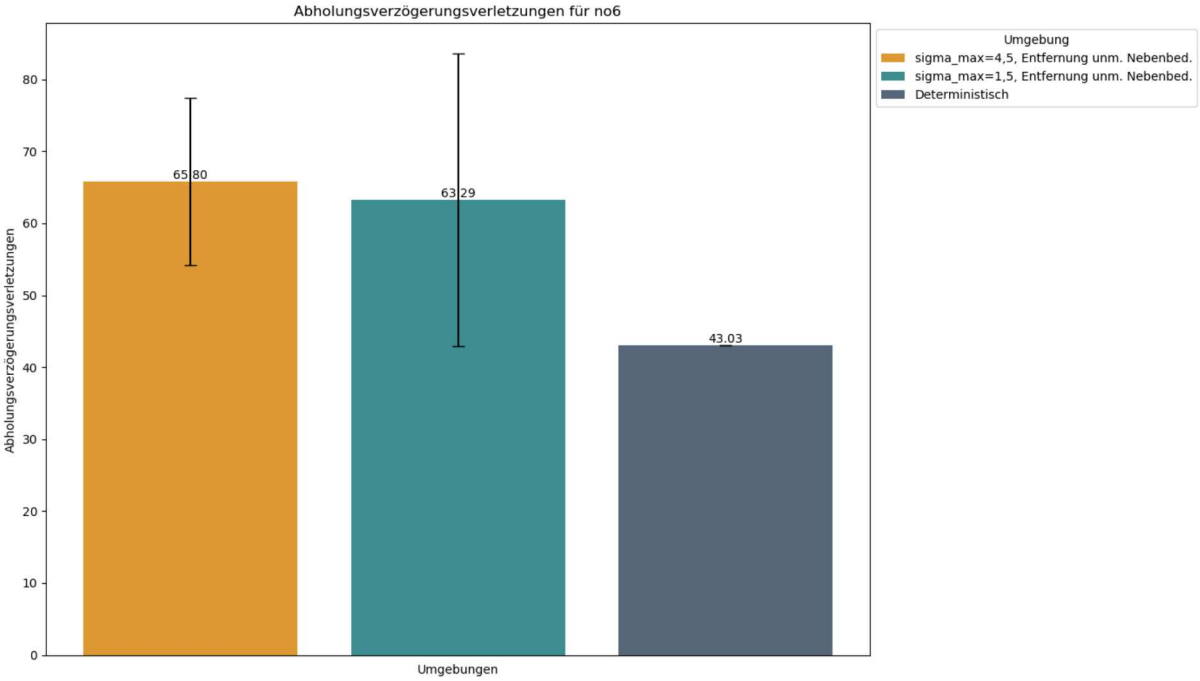

Durchschnittliche Wartezeit für no6
Durchschnittliche Wartezeit
Umgebungen
4.13
3.74
3.62
0
1
2
3
4
Umgebung
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.
Deterministisch

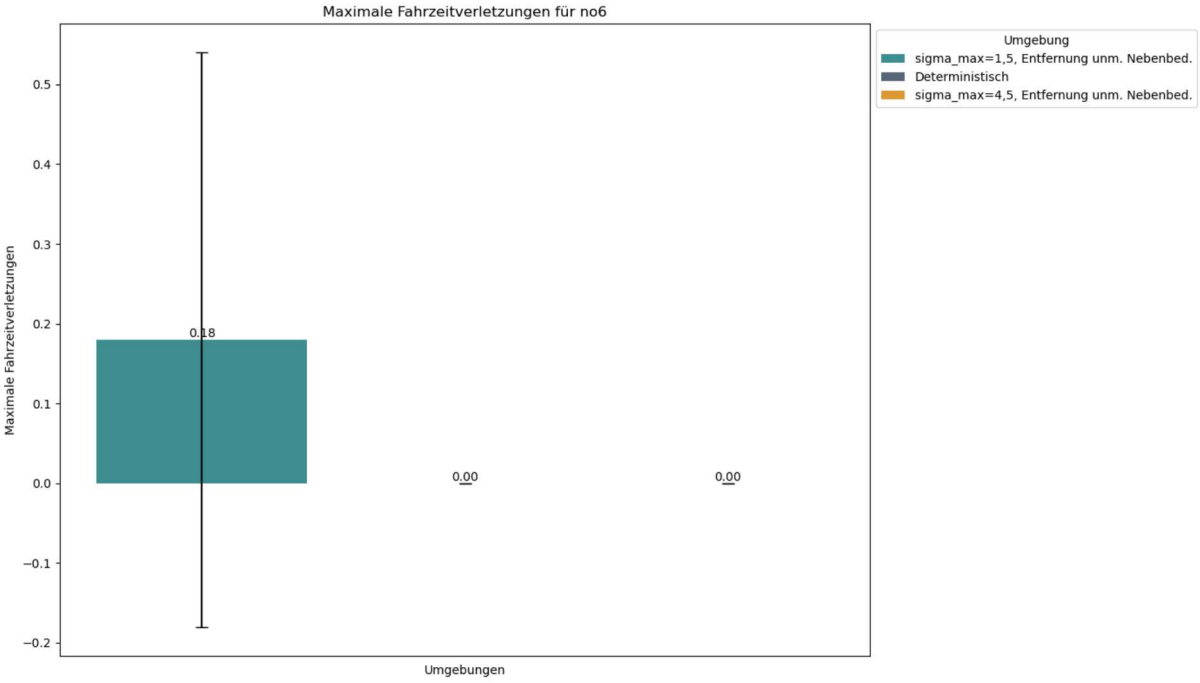

Durchschnittsbelegung für no6
Durchschnittsbelegung
Umgebungen
1.47
1.46
1.46
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Umgebung
Deterministisch
sigma_max=1,5, Entfernung unm. Nebenbed.
sigma_max=4,5, Entfernung unm. Nebenbed.


Anteil Leerkilometer für no6
Anteil Leerkilometer
Umgebungen
39.81
39.65
38.07
Umgebung
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.
Deterministisch


Gebuchte Kilometern/Gesamtroutenkosten für no6
Gebuchte Kilometern/Gesamtroutenkosten
Umgebungen
85.66
83.79
83.55
Umgebung
Deterministisch
sigma_max=1,5, Entfernung unm. Nebenbed.
sigma_max=4,5, Entfernung unm. Nebenbed.


Laufzeit in Sekunden für no6
Umgebung
sigma_max=4,5, Entfernung unm. Nebenbed.
Deterministisch
sigma_max=1,5, Entfernung unm. Nebenbed.
Laufzeit in Sekunden
0
10
20
30
40
33.87
29.99
28.64
Umgebungen


Abholungsverzögerungsverletzungen für no6
Umgebung
sigma_max=4,5, Entfernung unm. Nebenbed.
sigma_max=1,5, Entfernung unm. Nebenbed.
Deterministisch
Abholungsverzögerungsverletzungen
0
10
20
30
40
50
60
70
80
65.80
63.29
43.03
Umgebungen


Maximale Fahrzeitverletzungen für no6
Maximale Fahrzeitverletzungen
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
0.18
0.00
0.00
Umgebungen
Umgebung
sigma_max=1,5, Entfernung unm. Nebenbed.
Deterministisch
sigma_max=4,5, Entfernung unm. Nebenbed.

Literaturverzeichnis

[1] Büchel, B. ; Corman, F. : Review on Statistical Modeling of Travel Time Variability for Road-Based Public Transport. In: *Frontiers in Built Environment* 6 (2020). http://dx.doi.org/10.3389/fbuil.2020.00070. – DOI 10.3389/fbuil.2020.00070. – ISSN 2297–3362

[2] Chassaing, M. ; Fleury, G. ; Duhamel, C. ; Lacomme, P. : Determination of robust solutions for the DARP with variations in transportation time. In: *IFAC-PapersOnLine* 49 (2016), Nr. 12, 943-948. http://dx.doi.org/https://doi.org/10.1016/j.ifacol.2016.07.897. – DOI https://doi.org/10.1016/j.ifacol.2016.07.897. – ISSN 2405–8963. – 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016

[3] Cordeau, J.-F. : A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. In: *Operations Research* 54 (2006), 06, S. 573–586. http://dx.doi.org/10.1287/opre.1060.0283. – DOI 10.1287/opre.1060.0283

[4] Cordeau, J.-F. ; Laporte, G. : A tabu search heuristic for the static multi-vehicle dial-a-ride problem. In: *Transportation Research Part B: Methodological* 37 (2003), Nr. 6, 579-594. http://dx.doi.org/https://doi.org/10.1016/S0191-2615(02)00045-0. – DOI https://doi.org/10.1016/S0191–2615(02)00045–0. – ISSN 0191–2615

[5] Cordeau, J.-F. ; Laporte, G. : The dial-a-ride problem (DARP): Models and algorithms. In: *Annals OR* 153 (2007), 06, S. 29–46. http://dx.doi.org/10.1007/s10479-007-0170-8. – DOI 10.1007/s10479–007–0170–8

[6] Feng, X. ; Sun, H. ; Gross, B. ; Wu, J. ; Li, D. ; Yang, X. ; Lv, Y. ; Zhou, D. ; Gao, Z. ; Havlin, S. : Scaling of spatio-temporal variations of taxi travel routes. In: *New Journal of Physics* 24 (2022). http://dx.doi.org/10.1088/1367-2630/ac60e0. – DOI 10.1088/1367–2630/ac60e0

[7] Gaul, D. ; Klamroth, K. ; Stiglmayr, M. : Solving the Dynamic Dial-a-Ride Problem Using a Rolling-Horizon Event-Based Graph. 96 (2021), 8:1–8:16. http:

//nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:0030-drops-148776. – URN urn:nbn:de:0030–drops–148776. – ISBN 978–3–95977–213–6

[8] Gaul, D. ; Klamroth, K. ; Stiglmayr, M. : Event-based MILP models for ride-pooling applications. In: *European Journal of Operational Research* 301 (2022), Nr. 3, 1048-1063. http://dx.doi.org/https://doi.org/10.1016/j.ejor.2021.11.053. – DOI https://doi.org/10.1016/j.ejor.2021.11.053. – ISSN 0377–2217

[9] Kim, J. ; Mahmassani, H. ; Vovsha, P. ; Stogios, Y. ; Dong, J. : Scenario-Based Approach to Analysis of Travel Time Reliability with Traffic Simulation Models. In: *Transportation Research Record* 2391 (2013), S. 56 – 68. http://dx.doi.org/10.3141/2391-06. – DOI 10.3141/2391–06

[10] Mahmassani, H. ; Hou, T. ; Dong, J. : Characterizing Travel Time Variability in Vehicular Traffic Networks. In: *Transportation Research Record* 2315 (2012), S. 141 – 152. http://dx.doi.org/10.3141/2315-15. – DOI 10.3141/2315–15

[11] Masson, R. ; Lehuédé, F. ; Péton, O. : The Dial-A-Ride Problem with Transfers. In: *Computers and Operations Research* 41 (2014), 01, S. 12–23. http://dx.doi.org/10.1016/j.cor.2013.07.020. – DOI 10.1016/j.cor.2013.07.020

[12] Molenbruch, Y. ; Braekers, K. ; Caris, A. : Typology and literature review for dial-a-ride problems. In: *Annals of Operations Research* 259 (2017), S. 295 – 325. http://dx.doi.org/10.1007/s10479-017-2525-0. – DOI 10.1007/s10479–017–2525–0

[13] Oyola, J. ; Arntzen, H. ; Woodruff, D. L.: The stochastic vehicle routing problem, a literature review, part I: models. In: *EURO Journal on Transportation and Logistics* 7 (2018), Nr. 3, 193-221. http://dx.doi.org/https://doi.org/10.1007/s13676-016-0100-5. – DOI https://doi.org/10.1007/s13676–016–0100–5. – ISSN 2192–4376

[14] Parragh, S. N. ; Doerner, K. ; Hartl, R. : Variable neighborhood search for the dial-a-ride problem. In: *Comput. Oper. Res.* 37 (2010), S. 1129–1138. http://dx.doi.org/10.1016/j.cor.2009.10.003. – DOI 10.1016/j.cor.2009.10.003

[15] Peer, S. ; Koopmans, C. ; Verhoef, E. : Predicting Travel Time Variability for Cost-Benefit Analysis. In: *SSRN Electronic Journal* (2010), 01. http://dx.doi.org/10.2139/ssrn.1646370. – DOI 10.2139/ssrn.1646370

[16] Psaraftis, H. N.: A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. In: *Transportation Science* 14

(1980), Nr. 2, 130-154. http://dx.doi.org/10.1287/trsc.14.2.130. – DOI 10.1287/trsc.14.2.130

[17] Ricard, L. ; Desaulniers, G. ; Lodi, A. ; Rousseau, L.-M. : *Predicting the probability distribution of bus travel time to move towards reliable planning of public transport services*. https://arxiv.org/abs/2102.02292. Version: 2021

[18] Rietveld, P. ; Zwart, B. ; Wee, B. van ; Hoorn, T. van d.: On the relationship between travel time and travel distance of commuters. In: *The Annals of Regional Science* 33 (1999), S. 269–287. http://dx.doi.org/10.1007/S001680050105. – DOI 10.1007/S001680050105

[19] Schilde, M. ; Doerner, K. ; Hartl, R. : Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. In: *European Journal of Operational Research* 238 (2014), S. 18 – 30. http://dx.doi.org/10.1016/j.ejor.2014.03.005. – DOI 10.1016/j.ejor.2014.03.005

[20] Sohrabi, S. ; Ziarati, K. ; Keshtkaran, M. : Revised eight-step feasibility checking procedure with linear time complexity for the Dial-a-Ride Problem (DARP). In: *Computers and Operations Research* 164 (2024), 106530. http://dx.doi.org/https://doi.org/10.1016/j.cor.2024.106530. – DOI https://doi.org/10.1016/j.cor.2024.106530. – ISSN 0305–0548

[21] Susilawati, S. ; Taylor, M. ; Somenahalli, S. : Distributions of travel time variability on urban roads. In: *Journal of Advanced Transportation* 47 (2013), 12. http://dx.doi.org/10.1002/atr.192. – DOI 10.1002/atr.192

[22] Tsapakis, I. ; Cheng, T. ; Bolbol, A. : Impact of weather conditions on macroscopic urban travel times. In: *Journal of Transport Geography* 28 (2013), 204-211. http://dx.doi.org/https://doi.org/10.1016/j.jtrangeo.2012.11.003. – DOI https://doi.org/10.1016/j.jtrangeo.2012.11.003. – ISSN 0966–6923

[23] Vansteenwegen, P. ; Melis, L. ; Aktaş, D. ; Montenegro, B. D. G. ; Sartori Vieira, F. ; Sörensen, K. : A survey on demand-responsive public bus systems. In: *Transportation Research Part C: Emerging Technologies* 137 (2022), 103573. http://dx.doi.org/https://doi.org/10.1016/j.trc.2022.103573. – DOI https://doi.org/10.1016/j.trc.2022.103573. – ISSN 0968–090X

[24] Xiang, Z. ; Chu, C. ; Chen, H. : The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. In: *Eur. J. Oper. Res.* 185 (2008), S. 534–551. http://dx.doi.org/10.1016/j.ejor.2007.01.007. – DOI 10.1016/j.ejor.2007.01.007