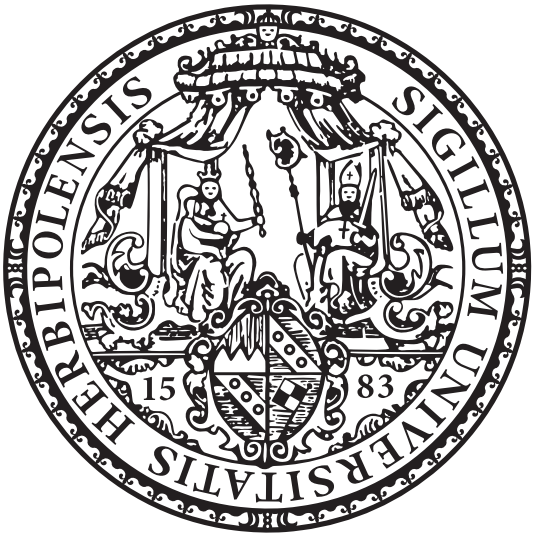
Bachelorarbeit

Vergleich von einer anfragebasierten und fahrplanbasierten Minibuslinie durch Simulation

Leo Puppe

Abgabedatum:	8. August 2023
Betreuer:	Prof. Dr. Marie Schmidt
	Kendra Reiter, M. Sc.



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Optimierung unter Ressourcenbeschränkungen

Zusammenfassung

In den vergangenen Jahren wurde viel an nachfrageorientierten (demand-responsive) Personentransportsystemen geforscht, allerdings ging es dabei meistens um Haus-zu-Haus Transportsysteme. Nachdem auf manchen Buslinien mit geringer Nachfrage bisher Minibusse im taktbasierten Betrieb eingesetzt werden, vergleicht diese Arbeit den taktbasierten Betrieb auf einer Minibuslinie mit einem nachfrageorientierten Betrieb, mit dem Ziel, dadurch effizientere Transporte zu ermöglichen. Dazu wird zunächst eine Simulation definiert, in der verschiedene Betriebsarten, sogenannte Transport-Strategien, getestet werden können. Anschließend wird eine anfragebasierte Transport-Strategie beschrieben, die festlegt, wie die Minibusse auf Anfragen reagieren und wann sie wo auf der Buslinie fahren sollen. Im Anschluss werden die taktbasierte und die anfragebasierte Transport-Strategie in der Simulation mit den gleichen Nachfragesituationen getestet und die Ergebnisse hinsichtlich verschiedener Kriterien ausgewertet. Dabei ergibt sich, dass die anfragebasierte Transport-Strategie vor allem bei geringer Nachfrage hinsichtlich jedem untersuchten Kriterium bessere Ergebnisse erzielt.

Abstract

In recent years, much research has been done on demand-responsive passenger transportation systems, but most of this research has focused on door-to-door transportation systems. Since some bus routes with low demand have so far used minibuses in clock-based operation, this paper compares clock-based operation on a minibus route with demand-responsive operation with the aim that this will enable more efficient transportation. To this end, a simulation is first defined in which different modes of operation, called transportation strategies, can be tested. Then, a demand-based transportation strategy is described, which determines how the minibuses respond to requests and when they should travel where on the bus route. Subsequently, the clock-based and the request-based transport strategies are tested in the simulation with the same demand situations and the results are evaluated with respect to different criteria. The results show that in the case of low demand the request-based transport strategy achieves better results with respect to each criterion examined.

Inhaltsverzeichnis

1. Einleitung

Im Zuge der Bekämpfung des Klimawandels soll die Emission von Treibhausgasen deutlich gesenkt werden. Dazu lohnt es sich, den Straßenverkehr zu betrachten: Er alleine war im Jahr 2021 für 18,8% der Treibhausgasemissionen Deutschlands verantwortlich [GG23]. Um die Emissionen des Verkehrs zu mindern, schlagen vom Umweltbundesamt beauftragte Forscher unter anderem vor, den privaten Personenverkehr teilweise auf den öffentlichen zu verlagern [RHB+23]. Damit sich mehr Personen entscheiden, mit dem öffentlichen Verkehr zu fahren, ist es notwendig, ein attraktives Angebot für sie zu schaffen. Gleichzeitig muss berücksichtigt werden, dass sich ein nicht benutztes Verkehrsangebot schädlich auf die Treibhausgasemissionen auswirken kann.

Im ländlichen Raum, aber auch in Stadtrandgebieten sind konventionelle Busse oft nicht ausgelastet. Bereits heute werden auf derartig nachfrageschwachen Buslinien teilweise Minibusse eingesetzt. Dabei handelt es sich um Fahrzeuge mit einer im Vergleich zu normalen Omnibussen geringen Kapazität. Oft werden diese Linien fahrplan- bzw. taktbasiert betrieben.

In den letzten Jahren hat die Idee des anfragebasierten (on demand) ÖPNV-Betriebs in der Literatur viel Aufmerksamkeit bekommen. Allerdings wurden überwiegend Haus-zu-Haus-Systeme vorgeschlagen und untersucht. Nach dem Wissensstand des Autors wurde bisher noch kein System betrachtet, welches Fahrten entlang einer festen Buslinie anfragebasiert plant und durchführt und bei dem kleine Busse eingesetzt werden. Daher sind dem Autor auch noch keine guten Methoden für die Situation bekannt. Deshalb soll in dieser Bachelorarbeit untersucht werden, ob bereits der Betrieb mit einer einfachen, anfragebasierten Greedy-Strategie zum Planen der Busfahrten bei einer festen Linie und dem Einsatz von Minibussen Vorteile gegenüber eines taktbasierten Betriebs hat.

Dabei sollen einerseits mögliche Vorteile für die Passagiere und andererseits mögliche Vorteile für das Unternehmen, welches die Linie betreibt, betrachtet werden. Zu den potentiellen Vorteilen für die Fahrgäste zählen zum einen eine Wartezeitverkürzung und zum anderen eine Verkürzung der Zeit, die sie im Bus verbringen. Darüber hinaus ist es für die Passagiere vorteilhaft, wenn sie häufiger die Möglichkeit haben, mit dem Bus zu fahren. Dadurch steigt die Attraktivität eines öffentlichen Nahverkehrsangebot, wodurch mehr potentielle Fahrgäste den Bus benutzen. Für das Unternehmen, das die Linie betreibt, und für die Umwelt ist es besser, möglichst kurz mit Bussen zu fahren, da fahrende Busse mehr Treibstoffkosten und -emissionen produzieren. Außerdem ist eine hohe Auslastung gut, da das Unternehmen dann gegebenenfalls höhere Einnahmen erzielt.

Der Vergleich der beiden Betriebsarten (im weiteren Verlauf auch als "Transport-Strategien" bezeichnet) soll durch Simulationen stattfinden. Dadurch ist es möglich, beide Transport-Strategien unter den selben, fest definierten Bedingungen zu testen. Das hat gegenüber eines realen Tests den Vorteil, dass Situationen reproduzierbar sind,

wodurch man (zumindest in der Simulation) herausfinden kann, warum bestimmte Sachverhalte passieren.

Abschließend wird nun die Aufgabenstellung genauer beschrieben. Gegeben sei eine Buslinie mit mehreren Haltestellen, auf der Minibusse fahren können. Außerdem gibt es Fahrtanfragen von Passagieren. Eine Anfrage besteht aus einem Start- und einem Zielort, wobei dafür nur Haltestellen der Linie in Frage kommen. Darüber hinaus wird jede Anfrage zu einem bestimmten Zeitpunkt gestellt, hat einen weiteren Zeitpunkt als gewünschte Abfahrtszeit, eine maximale Wartezeit sowie eine späteste Ankunftszeit am Zielort. Das zu optimierende Zielkriterium besteht darin, dass möglichst viele Nachfragen mit möglichst kurzen Warte- und Fahrzeiten bedient werden. Insbesondere soll die anfragebasierte Strategie hinsichtlich dieses Problems mit der taktbasierten verglichen werden, um herauszufinden, ob Verbesserungen möglich sind.

2. Literaturübersicht

In diesem Kapitel soll die vorliegende Arbeit in die bereits existierende wissenschaftliche Literatur eingeordnet werden.

Über die letzten Jahre wurde eine Großzahl an unterschiedlichen anfragebasierten Systemen vorgeschlagen. Alleine die Übersichtsarbeit von Vansteenwegen et al. [VMA+22] nennt insgesamt mehr als 100 weitere Arbeiten, die solche anfragebasierte Systeme vorschlagen. Dabei klassifizieren sie anfragebasierte Systeme hauptsächlich nach dem Zeitraum, bis wann Anfragen gestellt werden können. Wenn alle Anfragen bekannt sein müssen, bevor ein Bus im System fährt, optimiert das System die Fahrten statisch ("static"). Wenn neue Anfragen nur durch Busse bedient werden können, die noch nicht losgefahren sind, optimiert das anfragebasierte System die Fahrten dynamisch offline ("dynamic offline"). Wenn neue Anfragen auch durch Busse bedient werden können, die zu dem Zeitpunkt, zu dem die Anfrage gestellt wurde, bereits fahren, zählt das System als ein dynamisch online ("dynamic online") optimierendes System [VMA+22]. Um letzteren Fall handelt es sich in dieser Arbeit. Die Anfragen dürfen beliebig knapp vor dem Fahrtwunsch gestellt werden und können von jedem beliebigen Bus bedient werden. Bei den weiteren von Vansteenwegen et al. vorgestellten Kategorien handelt es sich in dieser Arbeit um ein "many-to-many" System, da die Start- und Zielhaltestellen der Anfragen uneingeschränkt sind. Da alle Haltestellen vorgegeben sind, ist die hier vorgestellte anfragebasierte Strategie "semi-flexible" [VMA+22].

Ein Großteil der von Vansteenwegen et al. vorgestellten Arbeiten untersucht Transport-Systeme, welche entweder Haus-zu-Haus-Fahrten anbieten oder an festgelegten Haltestellen anhalten, die über eine größere (nicht linienförmige) Fläche verteilt sind. Nur ein kleinerer Teil der Arbeiten behält bei ihren Transport-Systemen die linienartige Struktur bei [VMA+22]. Da sich diese Arbeit mit einem anfragebasierten Transport-System auf Basis einer Buslinie beschäftigt, werden im Folgenden Arbeiten mit einem ähnlichen Transport-System zusammengefasst.

Crainic et al. beschreiben ein solches nachfragebasiertes Transport-System: Auf einer kreisförmigen Route fährt ein Bus und hält dabei stets an bestimmten wichtigen Stationen. Zwischen allen Stationen darf der Bus, wenn Anfragen das wünschen und seine Zeitbeschränkungen es zulassen, Umwege fahren und dadurch an beliebigen Straßenkreuzungen halten [CMNG05]. Allerdings vergleichen sie ihr neues Transport-System nicht mit herkömmlichen, sondern untersuchen verschiedene Methoden, wie effizient ermittelt werden kann, zwischen welchen Haltestellen der Bus welche Umwege fahren soll.

Pei et al. untersuchen eine flexible Buslinie, die nur an Stationen anhält, wenn das für Fahrtanfragen der Passagiere notwendig ist, und insbesondere vorzeitig umkehrt, wenn Haltestellen am Streckenende wegen fehlender Fahrtanfragen nicht mehr bedient werden müssen. Sie vergleichen die flexible Buslinie auf einer realen Strecke mit einer konventio-

nellen Buslinie hinsichtlich der Warte- und Transportzeiten sowie der Summe der beiden, der sogenannten Reisezeit. So kommen sie zu dem Ergebnis, dass sie bei einem geringen Passagieraufkommen und bei einer kleinen Anzahl an Bussen bessere Reisezeiten für die Passagiere erzielen als das konventionelle Transport-System. Es wird jedoch nicht analysiert, warum es zu diesem Verhalten kommt. Außerdem treffen sie implizit die Annahme, dass Passagiere beliebig lange an einer Haltestelle warten können. Im Gegensatz zu dieser Arbeit verwenden sie Busse ohne Kapazitätsbeschränkung [PLO19].

3. Methodik

Das folgende Kapitel befasst sich mit der Herangehensweise, wie die taktbasierte Minibuslinie und die anfragebasierte Minibuslinie miteinander verglichen werden. Zunächst wird das allgemeine Problem, Passagiere entlang einer festgelegten Linien zu transportieren, formal definiert. Danach wird die Simulation beschrieben, die zum Vergleich verwendet wird. Daraufhin folgt eine Definition der beiden Transport-Strategien und abschließend wird die Implementierung erläutert.

3.1. Formale Problemdefinition

3.1.1. Ausgangssituation

Formal betrachtet ist die Ausgangssituation dieser Arbeit folgende:

Gegeben sei eine nicht kreisförmige Linie mit k Haltestellen $H = (h_1, h_2, \ldots, h_k)$, die in einer festen Sequenz angeordnet sind und in beide Richtungen bedient werden können. Nur die benachbarten Haltestellen sind jeweils durch Fahrstrecken $E = \{\{h_i, h_j\} \mid 1 \leq i \leq k-1 \wedge j = i+1\}$ verbunden. Für den Netzgraph G gilt also $G = (H, E)$. Die Distanz δ und die Fahrtrichtung ϑ zwischen zwei Haltestellen h_i, h_j ist definiert durch die Differenz ihrer Indizes in H.

$$\delta \colon H^2 \to \mathbb{N}_0, (h_i, h_j) \mapsto |i - j|$$

$$\Theta = \{\text{keine}, \text{hinauf}, \text{hinunter}\}$$

$$\vartheta \colon H^2 \to \Theta, (h_i, h_j) \mapsto \begin{cases} \text{hinauf} & \text{falls } i < j \\ \text{hinunter} & \text{falls } i > j \\ \text{keine} & \text{sonst} \end{cases}$$

Busse dürfen nur entlang der Fahrstrecken fahren. Daraus folgt insbesondere, dass Busse, wenn sie an einer Haltestelle nicht anhalten, trotzdem keine Abkürzung fahren dürfen und an der Haltestelle vorbeifahren müssen. Die Fahrzeit v zwischen allen Haltestellen und auch die minimale Standzeit s bei jeder Station, wenn an ihr gehalten wird, ist gleich. Wenn an einer Station nicht gehalten wird, entfällt nur die Standzeit und keine Fahrzeit. Weiter gibt es m Fahrtanfragen $R = (r_1, r_2, \ldots, r_m)$, die aufsteigend nach ihrer gewünschten Abfahrtszeit sortiert sind. Jede Fahrtanfrage $r_i = (o_{r_i}, d_{r_i}, a_{r_i}, g_{r_i}, w_{r_i}, l_{r_i})$ besteht aus einer Start- $o_{r_i} \in H$ und einer Zielhaltestelle $d_{r_i} \in H$, wobei $o_{r_i} \neq d_{r_i}$ gilt. Darüber hinaus hat jede Anfrage einen Zeitpunkt a_{r_i}, zu dem sie gestellt wird, die eben erwähnte gewünschte Abfahrtszeit g_{r_i}, die Zeit, bis zu der der Passagier maximal an der

Starthaltestelle wartet w_{r_i} und eine späteste Ankunftszeit l_{r_i} an der Zielhaltestelle. Für eine Fahrtanfrage r gilt dabei $a_r \leq g_r \leq w_r \leq l_r$. Um die Fahrtanfragen zu bedienen, stehen n Busse $B = \{b_1, b_2, \ldots, b_n\}$ bereit, die jeweils eine Kapazität von c Sitzplätzen haben.

Jeder Bus b hat zu jedem Zeitpunkt t einen Aktionsplan ϱ mit $z_{b,t}$ Aktionen:

$$\varrho(b,t) = \left(p_{b,t}^{(1)}, p_{b,t}^{(2)}, \ldots, p_{b,t}^{(z_{b,t})}\right)$$

Jede Aktion $p_{b,t}^{(i)} = (\gamma_{b,t}^{(i)}, \sigma_{b,t}^{(i)}, \chi_{b,t}^{(i)}, \Psi_{b,t}^{(i)})$ besteht aus einem Typ $\gamma_{b,t}^{(i)} \in \{\text{abfahrt}, \text{ankunft}\}$, einer Haltestelle $\sigma_{b,t}^{(i)}$, einer Zeit $\chi_{b,t}^{(i)} \in \mathbb{N}_0$ und einer Menge an Fahrtanfragen $\Psi_{b,t}^{(i)}$, die aber auch unbenutzt sein kann.

$$\Psi_{b,t}^{(i)} \in \left(2^R \cup \{\text{unbenutzt}\}\right)$$

Wenn $\Psi_{b,t}^{(i)} = \text{unbenutzt}$ bei einer Abfahrtsaktion gilt, bedeutet das, dass dieser Wert nicht verwendet wird. Das heißt, dass alle Passagiere in den Bus einsteigen bis er voll ist und zwar in der Reihenfolge, die durch die Indizes in R definiert ist. Andernfalls versuchen nur diejenigen Passagiere einzusteigen, die in der Menge $\Psi_{b,t}^{(i)}$ enthalten sind. Insbesondere folgt daraus, dass wenn $\Psi_{b,t}^{(i)} = \emptyset$ gilt, kein einziger Passagier einsteigen wird. Bei Ankunftsaktionen gilt immer $\Psi_{b,t}^{(i)} = \text{unbenutzt}$.

Auch wenn die Aktionspläne durch die Transport-Strategien festgelegt werden, gelten für sie einige allgemeine Bedingungen. Im Anhang in Abschnitt A.1 werden diese formell beschrieben.

Alle Busse starten an der Station h_1, daher enthält der Aktionsplan von jedem Bus b für jeden Zeitpunkt $t \in \mathbb{N}_0$ an erster Stelle das Element $p_{b,t}^{(1)} = (\text{ankunft}, \text{h}_1, 0, \text{unbenutzt})$.

Schließlich können noch Zustände für Fahrtanfragen und Busse definiert werden. Zu jedem Zeitpunkt t kann man einer Fahrtanfrage r einen Zustand $\phi_R(r,t) \in \Phi_R$ zuordnen. Eine Fahrtanfrage ist unbekannt, wenn die angegebene Zeit t vor dem Zeitpunkt a_r liegt, zu dem die Anfrage gestellt wird. Sie gilt als bekannt, wenn die angegebene Zeit nach a_r und vor dem Zeitpunkt der gewünschten Abfahrtszeit g_r liegt. Wenn der Zeitpunkt nach g_r liegt und vor der Zeit, bis zu der die Anfrage maximal wartet, gilt sie als wartend, sofern der zugehörige Passagier noch nicht zu dieser Zeit in einem Bus sitzt. (Die Funktion $\omega(b,t)$ gibt die Menge der Fahrtanfragen von Passagieren an, die zu einem bestimmten Zeitpunkt t in einem Bus b sitzen, und ist im Anhang in Abschnitt A.2 mathematisch beschrieben.) Während der Passagier einer Anfrage im Bus sitzt, gilt die Anfrage als wirdbearbeitet. Schließlich ist eine Fahrtanfrage abgeschlossen, wenn der zugehörige Passagier irgendwann zuvor in einem Bus saß. Als abgelehnt wird r nur dann bezeichnet, wenn der Zeitpunkt t nach dem Zeitpunkt des maximalen Wartens w_r liegt

und der Passagier der Fahrtanfrage nie in einem Bus saß. Formal heißt das:

$$\Phi_R = \{\text{unbekannt}, \text{bekannt}, \text{wartend}, \text{wirdbearbeitet}, \text{abgeschlossen}, \text{abgelehnt}\}$$

$$\phi_R \colon R \times \mathbb{N}_0 \to \Phi_R, (r,t) \mapsto \begin{cases} \text{unbekannt} & \text{falls } t < a_r \\ \text{bekannt} & \text{falls } a_r \leq t < g_r \\ \text{wartend} & \text{falls } g_r \leq t \leq w_r \wedge \forall b \in B \colon r \notin \omega(b,t) \\ \text{wirdbearbeitet} & \text{falls } \exists b \in B \colon r \in \omega(b,t) \\ \text{abgeschlossen} & \text{falls } \forall b \in B \colon r \notin \omega(b,t) \\ & \quad \wedge \exists t_{\text{alt}} < t \colon \exists b \in B \colon r \in \omega(b, t_{\text{alt}}) \\ \text{abgelehnt} & \text{falls } w_r < t \\ & \quad \wedge \forall t_{\text{alt}} \leq t \colon \forall b \in B \colon r \notin \omega(r, t_{\text{alt}}) \end{cases}$$

Zur besseren Verständlichkeit ist in Abbildung 3.1 ein Zustandsübergangsdiagramm zu sehen. Dabei ist zu beachten, dass die Zustände unbekannt, bekannt und wartend theoretisch übersprungen werden können, z.B. falls die Fahrtanfrage erst zum gewünschten Abfahrtszeitpunkt gestellt wird.

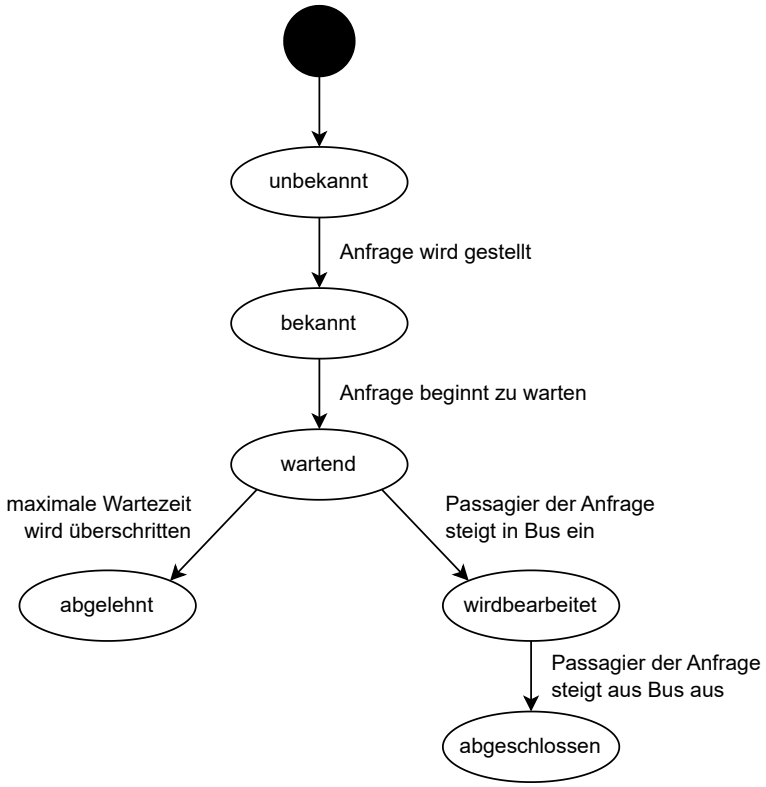


Abb. 3.1.: Zustandsübergangsdiagramm für Fahrtanfragen. Die ersten drei Zustände können theoretisch übersprungen werden.

Auch den Bussen b kann man zu jeder Zeit t einen Zustand ϕ_B zuordnen. Das geschieht

auf Basis der letzten Aktion $\left(\gamma_{b,t}^{(i)}, \sigma_{b,t}^{(i)}, \chi_{b,t}^{(i)}, \Psi_{b,t}^{(i)}\right) \in \varrho(b,t)$, die der Bus b zum Zeitpunkt t durchgeführt hat. :

$$\Phi_B = \{\text{fährt}, \text{steht}\}$$

$$\phi_B \colon B \times \mathbb{N}_0 \to \Phi_B, (b,t) \mapsto \begin{cases} \text{fährt} & \text{falls } \gamma_{b,t}^{(\tau(b,t))} = \text{abfahrt} \\ \text{steht} & \text{falls } \gamma_{b,t}^{(\tau(b,t))} = \text{ankunft} \end{cases}$$

Die verwendeten Variablen und Funktionen sind in den Tabellen 3.1 und 3.2 alphabetisch sortiert zusammengefasst.

3.1.2. Evaluationskriterien

Weiter soll auf die Evaluationskriterien eingegangen werden, unter denen die beiden Transport-Strategien "taktbasierter Minibus" und "anfragebasierter Minibus" verglichen werden. Dazu werden die Zustände aller Anfragen und aller Busse bis zum Endzeitpunkt der Simulation $t_{\max}$ untersucht.

Zunächst ist hier die durchschnittliche Wartezeit $\bar{t}_\text{w}$ aller Anfragen zu nennen. Sie sollte möglichst klein sein und ist folgendermaßen definiert:

$$R_\text{w} = \{r \in R \mid \phi_R(r, t_{\max}) \in \{\text{wartend}, \text{wirdbearbeitet}, \text{abgeschlossen}, \text{abgelehnt}\}\}$$

$$\bar{t}_\text{w} = \frac{\sum_{r \in R_\text{w}} t_\text{w}(r)}{|R_\text{w}|}$$

$$t_\text{w} \colon R \to \mathbb{N}_0, r \mapsto |\{t \in \mathbb{N}_0 \mid t < t_{\max} \wedge \phi_R(r,t) = \text{wartend}\}|$$

Hierbei enthält die durchschnittliche Wartezeit auch die Wartezeiten von allen Fahrtanfragen, die nie transportiert wurden. Der Grund dafür ist, dass eine Fahrtanfrage r erst nach ihrem maximalen Wartezeitpunkt w_r als abgelehnt gilt.

Auch die durchschnittliche Transport-Zeit $\bar{t}_\text{t}$ (also die Zeit, die ein Passagier von einer Fahrtanfrage in einem Bus sitzt) soll untersucht werden, wobei bei ihr ebenfalls kleine Werte besser sind.

$$R_\text{t} = \{r \in R \mid \phi_R(r, t_{\max}) \in \{\text{wirdbearbeitet}, \text{abgeschlossen}\}\}$$

$$\bar{t}_\text{t} = \frac{\sum_{r \in R_\text{t}} t_\text{t}(r)}{|R_\text{t}|}$$

$$t_\text{t} \colon R \to \mathbb{N}_0, r \mapsto |\{t \in \mathbb{N}_0 \mid t < t_{\max} \wedge \phi_R(r,t) = \text{wirdbearbeitet}\}|$$

Weniger als eigenes Evaluationskriterium sondern mehr um die Transport-Zeit $\bar{t}_\text{t}$ besser interpretieren zu können, soll zusätzlich die durchschnittliche Fahrzeit pro Anfrage $\bar{t}_\text{f}$ erhoben werden. Bei der Fahrzeit $t_\text{f}(r)$ handelt es sich um die Zeit, welche die Anfrage r im Bus verbringt und der Bus gleichzeitig fährt. Sie ist folglich proportional zu der Streckenlänge, die eine Fahrtanfrage zu fahren wünscht. Definiert ist sie wie folgt:

$$\bar{t}_\text{f} = \frac{\sum_{r \in R_\text{t}} t_\text{f}(r)}{|R_\text{t}|}$$

$$t_\text{f} \colon R \to \mathbb{N}_0, r \mapsto |\{t \in \mathbb{N}_0 \mid t < t_{\max} \wedge \phi_R(r,t) = \text{wirdbearbeitet} \wedge \exists b \in B \colon r \in \omega(b,t) \wedge \phi_B(b,t) = \text{fährt}\}|$$

Neben diesen Anfrage-bezogenen Zeiten gibt es auch eine Bus-bezogene Zeit: die gesamte Fahrzeit t_b. Analog zu den vorherigen Zeiten ist sie besser, wenn sie geringer ist, und ist definiert als:

$$t_\mathrm{b} = \sum_{b \in B} t_\mathrm{bb}(b)$$
$$t_\mathrm{bb} \colon B \to \mathbb{N}_0, b \mapsto |\{t \in \mathbb{N}_0 \,|\, t < t_\mathrm{max} \wedge \phi_B(b,t) = \text{fährt}\}|$$

Weiterhin soll die Mitnahmerate q_m untersucht werden. Es ist vorteilhaft, wenn sie höher ist.

$$R_\mathrm{v} = \{r \in R \,|\, \phi_R(r, t_\mathrm{max}) \in \{\text{wirdbearbeitet}, \text{abgeschlossen}, \text{abgelehnt}\}\}$$
$$q_\mathrm{m} = \frac{|R_\mathrm{t}|}{|R_\mathrm{v}|}$$

Die Auslastung der Busse q_a ist ebenfalls eine relevante Messgröße, die verglichen werden soll. $t_\mathrm{br}(b)$ entspricht dabei der summierten Fahrzeit von allen Fahrtanfragen für einen Bus b. Die Auslastung sollte möglichst hoch sein.

$$q_\mathrm{a} = \frac{\sum_{b \in B} t_\mathrm{br}(b)}{c \cdot \sum_{b \in B} t_\mathrm{b}(b)}$$
$$t_\mathrm{br} \colon B \to \mathbb{N}_0, b \mapsto |\{(t,r) \in \mathbb{N}_0 \times R \,|\, t < t_\mathrm{max} \wedge \phi_B(b,t) = \text{fährt} \wedge r \in \omega(b,t)\}|$$

3.2. Simulation

Im Folgenden wird die Simulation beschrieben, die basierend auf der Ausgangssituation entworfen wurde und mit deren Hilfe die im Anschluss vorgestellten Transport-Strategien getestet werden können.

Die verwendete Simulation berechnet auf Basis der Strecke, der Fahrtanfragen, der Busse und insbesondere der Transport-Strategie, wo sich Fahrtanfragen bzw. Busse zu welcher Zeit befinden. Dazu teilt sie die Zeit in Zeitschritte, sogenannte Ticks, auf und berechnet den Ort bzw. den Zustand von jedem Bus und jeder Fahrtanfrage in jedem Schritt. Abschließend berechnet sie die in Abschnitt 3.1.2 vorgestellten Variablen, mit denen die Transport-Strategie evaluiert werden soll.

Wie das konkret passiert, ist in einem Aktivitätsdiagramm in Abbildung 3.2 dargestellt und wird nun erläutert. Die Zahlen in den runden Klammern geben dabei den entsprechenden Schritt im Aktivitätsdiagramm an. (1) Zu Beginn wird die Simulation initialisiert. Das heißt, dass die Transport-Strategie über allgemeine Simulationsparameter (wie z.B. die Anzahl der Busse oder die Route) informiert wird. Anschließend beginnt der eigentliche Zyklus der Simulation. (2) Solange die maximale Simulationsdauer noch nicht erreicht wurde, wird ein weiterer Zeitschritt simuliert. (3) Wenn ein weiterer Tick simuliert werden soll, wird zunächst der Zustand der Fahrtanfragen aktualisiert, die in diesem Zeitschritt bekannt werden. Daraufhin wird außerdem der Zustand der Fahrtanfragen angepasst, die in diesem Tick mit dem Warten beginnen, also deren gewünschte

Tab. 3.1.: Variablen in alphabetischer Reihenfolge

Variable	Erklärung
a_r	Zeitpunkt, zu dem die Fahrtanfrage r gestellt wird
b	Bus
c	Transportkapazität eines beliebigen Busses
d_r	Zielhaltestelle einer Fahrtanfrage r
g_r	gewünschter Abfahrtszeitpunkt der Anfrage r
h	Haltestelle
k	Anzahl der Haltestellen
l_r	späteste Ankunftszeit einer Fahrtanfrage r
m	Anzahl der Fahrtanfragen
n	Anzahl der Busses
o_r	Starthaltestelle einer Fahrtanfrage r
$p_{b,t}^{(i)}$	i-te Aktion eines Busses b im Aktionsplan vom Zeitpunkt t
r	Fahrtanfrage
s	minimale Standzeit an einer beliebigen Station
v	Fahrzeit zwischen benachbarten Haltestellen
w_r	Zeitpunkt, bis zu dem eine Fahrtanfrage r maximal wartet
$z_{b,t}$	Anzahl der Aktionen im Aktionsplan vom Zeitpunkt t des Busses b
γ	Typ einer Aktion
σ	Haltestelle einer Aktion
χ	Zeitpunkt einer Aktion
Ψ	Betroffene Fahrtanfragen einer Aktion

Tab. 3.2.: Funktionen in alphabetischer Reihenfolge

Funktion	Erklärung
$\phi(b,t)$	Zustand eines Busses b zum Zeitpunkt t
$\phi(r,t)$	Zustand einer Fahrtanfrage r zum Zeitpunkt t
$\vartheta(b,t)$	Fahrtrichtung eines Busses b zum Zeitpunkt t
$\vartheta(r)$	Fahrtrichtung einer Fahrtanfrage r
$\vartheta(h_1,h_2)$	Fahrtrichtung zwischen zwei Haltestellen h_1, h_2
$\epsilon(b,t)$	Index der nächsten Aktion eines Busses b zum Zeitpunkt t
$\tau(b,t)$	Index der letzten Aktion eines Busses b zum Zeitpunkt t
$\varrho(b,t)$	Aktionsplan eines Busses b (vergangene und geplante Aktionen) zum Zeitpunkt t
$\omega(b,t)$	Menge der Fahrtanfragen, die ein Bus b zum Zeitpunkt t befördert
$\delta(h_1,h_2)$	Abstand zwischen zwei Haltestellen h_1, h_2
$\xi(b,t)$	Menge an Haltestellen, die ein Bus in der Zukunft eingeplant hat

Wartezeit zum aktuellen Zeitpunkt ist. (4) Anschließend darf die Transport-Strategie die Aktionspläne aller Busse aktualisieren. Dazu wird sie über den aktuellen Zustand der Simulation unterrichtet. (5) Danach werden die Aktionspläne ausgeführt, wenn sie Aktionen für die aktuelle Zeit enthalten: (5.a.1) Bei einer Abfahrtsaktion ändert der Bus, wenn seine nächste Haltestelle das erfordert, seine Fahrtrichtung. Dabei dürfen sich jedoch keine Fahrgäste im Bus befinden. Wenn das der Fall sein sollte, würde die Simulation an dieser Stelle abbrechen. (5.a.2) Anschließend lässt der Bus die Passagiere von bestimmten Fahrtanfragen einsteigen. Um welche Fahrtanfragen es sich dabei handelt, ist abhängig von der Transport-Strategie. Für alle Passagiere, die in einen Bus eingestiegen sind, wird ihre Fahrtanfrage in den Zustand "wirdbearbeitet" versetzt. (5.a.3) Schließlich startet der Bus seine Fahrt in Richtung der Haltestelle, die in seinem Aktionsplan durch die nächste Ankunftsaktion festgelegt ist, indem er seinen Zustand zu "fährt" ändert. (5.b.1) Wenn eine Ankunftsaktion ausgeführt wird, hält der Bus erst an, indem er seinen Zustand zu "steht" ändert. (5.b.2) Anschließend steigen alle Passagiere der Fahrtanfragen aus, die sich im Bus befinden und an dieser Haltestelle aussteigen möchten. Der Zustand der Fahrtanfragen, deren Passagiere ausgestiegen sind, ändert sich hierbei zu "abgeschlossen". (6) Danach werden die Fahrtanfragen von allen Passagieren, die bereits an einer Haltestelle warten und deren maximale Wartezeit ausgelaufen ist, aufgegeben. Das heißt, ihr Zustand wird zu "abgelehnt" geändert. (7) Schließlich wird die Zeitvariable um eins erhöht, damit im nächsten Durchlauf des Zyklus der nächste Zeitschritt simuliert werden kann. (2) Nach einem Durchlauf des Zyklus wird erneut überprüft, ob das Simulationsende erreicht wurde. (8) Wenn das der Fall ist, werden die Werte unter denen die Transport-Strategien verglichen werden sollen, ermittelt.

3.2.1. Darstellung von Zuständen innerhalb der Simulation

Um einen Zustand der Simulation zu graphisch zu beschreiben, werden in dieser Arbeit häufig Strecke-Zeit-Diagramme wie in Abbildung 3.3 verwendet. Bei diesen ist an der horizontalen Achse immer die Strecke eingezeichnet. Für jede Haltestelle gibt es einen vertikalen Strich. Bei der vertikalen Achse handelt es sich um die Zeitachse. Je nach Auflösung sind alle zwei bis zehn Zeitschritte eine horizontale Linie, um eine bessere Übersicht zu ermöglichen. Die aktuelle Zeit ist durch eine horizontale, gestrichelte, rote Linie eingezeichnet (im Bild bei $t = 104$). Im Diagramm werden normalerweise zwei Dinge dargestellt: die Aktionspläne der Busse und ausgewählte Fahrtanfragen.

Die Aktionspläne der Busse werden durch bunte Polylinien (im Bild blau, lila und rot) in das Diagramm eingezeichnet. Dabei entspricht der Teil der Polylinie, der oberhalb der aktuellen Zeit ist, der bereits befahrenen Strecke. Unterhalb der rot gestrichelten Zeitlinie handelt es sich im Diagramm um geplante Fahrten. Jede Ecke einer solchen Linie entspricht einer Aktion. Vertikale Liniensegmente sind Standzeiten und schräge Segmente entsprechen Fahrten. Die Zahlen neben den Liniensegmenten entsprechen der Anzahl der Passagiere, die auf diesem Segment unterwegs waren bzw. transportiert werden sollen. Wenn eine solche Polylinie vor dem unteren Bildrand endet (wie etwa die rote) bedeutet das, dass der Aktionsplan des entsprechenden Busses zu diesem Zeitpunkt endet.

Manche Fahrtanfragen werden, um bestimmte Umstände besser zu veranschaulichen,

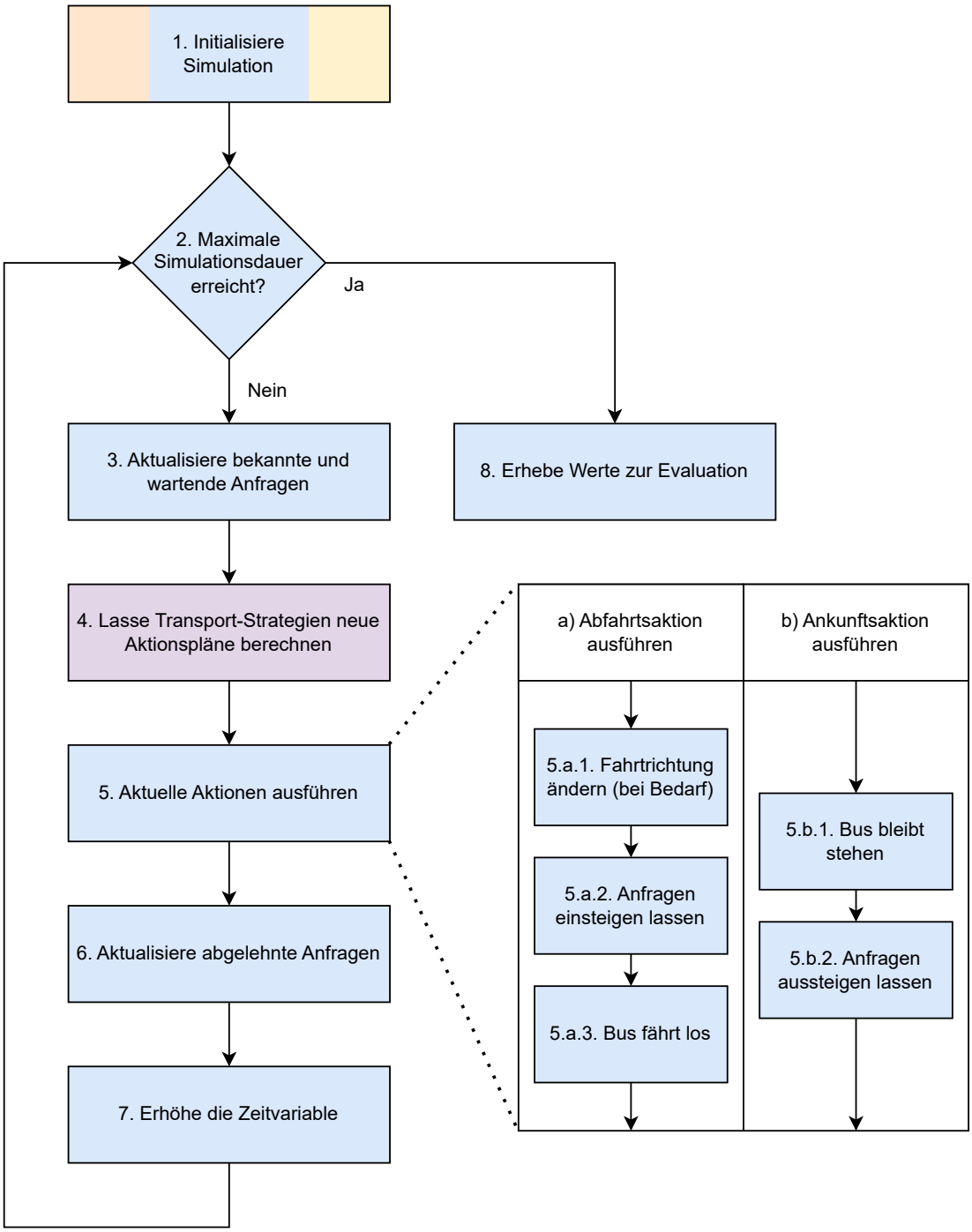


Abb. 3.2.: Der Simulationszyklus. Die Farben geben an, welches Modul der Implementierung einen Schritt durchführt (siehe auch Abbildung 3.10). (Orange = Streckengenerator, Gelb = Fahrtanfragengenerator, Blau = Programmkern, Lila = Transport-Strategie)

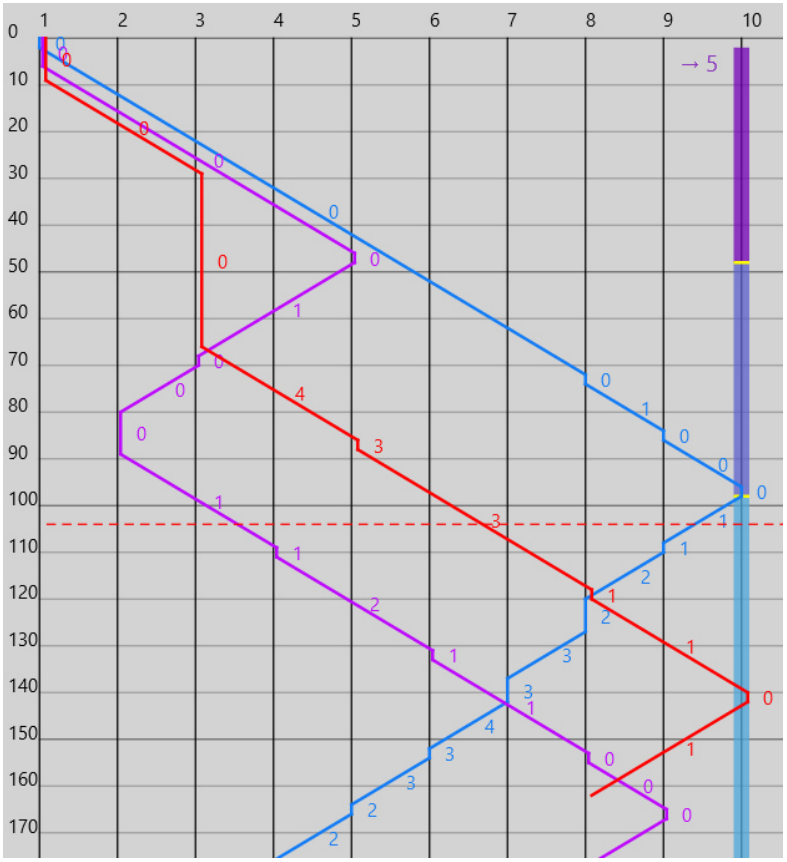


Abb. 3.3.: Ein Strecke-Zeit-Diagramm.

als Kasten bzw. dicke Linie auf eine Haltestellenlinie gezeichnet. Im Bild ist das an der Haltestelle 10 der Fall. Die Haltestelle, auf deren vertikale Linie die Anfrage r zu sehen sind, ist dann die Starthaltestelle o_r. Meistens steht am Anfang des Kastens die Zielhaltestelle d_r der Anfrage als Pfeil gefolgt von der entsprechenden Haltestellennummer. Der Kasten, der eine Anfrage visualisiert, ist lila eingefärbt, während die Anfrage bekannt ist, aber noch nicht wartet (also zwischen a_r und g_r). Der Zeitraum, den die Anfrage maximal wartet, ist auf dem Kasten dunkelblau eingefärbt (also zwischen g_r und w_r). Zuletzt ist die Zeit bis zu der die Anfrage spätestens befördert sein muss, hellblau eingefärbt (also die Zeit zwischen w_r und l_r). Damit die Grenzen zwischen den unterschiedlich gefärbten Teilen besser hervorgehoben werden, ist zum Zeitpunkt g_r und zum Zeitpunkt w_r jeweils eine gelbe horizontale Linie auf dem Kasten zu sehen.

3.3. Transport-Strategien

Es wird nun beschrieben, was eine Transport-Strategie im Allgemeinen ist. Anschließend werden die beiden Transport-Strategien vorgestellt, die in der vorliegenden Arbeit miteinander verglichen werden sollen.

Eine Transport-Strategie legt fest, wann und nach welchen Regeln die Aktionspläne der Busse aktualisiert werden. Dazu darf sie bei jedem Bus zukünftige Aktionen löschen, ändern oder neue hinzufügen. Allerdings muss sie verschiedene Regeln einhalten:

1. Die Zeitspanne zwischen einer Abfahrtsaktion an einer Haltestelle und einer Ankunftsaktion an einer anderen Haltestelle muss immer exakt der dafür benötigten Fahrzeit entsprechen.

2. Die Zeit zwischen einer Ankunftsaktion und der folgenden Abfahrtsaktion muss mindestens der minimalen Wartezeit entsprechen.

3. Ein Aktionsplan muss immer mit einer Ankunftsaktion enden.

4. Alle Passagiere, die zur aktuellen Zeit im Bus sitzen, müssen weiterhin zu ihrer Zielhaltestelle befördert werden, ohne dass der Bus dafür wenden muss.

5. Alle Passagiere, die zur aktuellen Zeit im Bus sitzen, müssen weiterhin zu ihrer spätesten Ankunftszeit an ihre Zielhaltestelle gebracht werden.

6. Falls der Bus gerade fährt, darf durch den neuen Aktionsplan nicht die Fahrtrichtung des Busses geändert werden.

Darüber hinaus bestimmt sie, welche Fahrtanfragen bei einer Abfahrtsaktion an einer Haltestelle tatsächlich bedient werden sollen.

3.3.1. Taktbasierter Minibus

Die Transport-Strategie “taktbasierter Minibus” entspricht einem koventionellen Linienbus. Alle zur Verfügung stehenden Minibusse fahren dabei mit der kleinstmöglichen

Taktzeit die komplette Strecke ab und halten an jeder Station. Dabei steigen Passagiere in den Minibus ein, wenn sie bereits an der Haltestelle warten, er in Richtung ihrer Zielhaltestelle fährt und noch freier Platz verfügbar ist.

Dazu werden die Startzeitpunkte aller Busse an der ersten Haltestelle so berechnet, dass sie ein Taktfahrplan ergeben. Anschließend wird der Aktionsplan eines Busses, immer wenn er leer ist, so geändert, dass er (ggf. ab seinem Startzeitpunkt) die komplette Strecke erst in die eine Richtung und dann in die andere Richtung fährt, wobei er an jeder Haltestelle hält und stets die vorgegebene Mindestwartezeit wartet.

3.3.2. Anfragebasierter Minibus

Die Transport-Strategie "anfragebasierter Minibus" bezieht hingegen die konkreten Fahrtanfragen in die Erstellung der Aktionspläne mit ein. Bei dieser Strategie fahren Minibusse nur dann, wenn auch eine konkrete Nachfrage besteht. Eine weitere Grundidee ist, dass Fahrtanfragen gebündelt von möglichst wenigen Bussen abgearbeitet werden sollen. Dazu sollen Fahrtanfragen gezielt von Bussen bedient werden, die die Strecke (zumindest teilweise) sowieso fahren müssen. Nur wenn das nicht klappt, soll ein Minibus den neuen Passagier transportieren, der dafür eine komplett neue Fahrt in Kauf nehmen muss. Um Verspätungen für andere Fahrten zu vermeiden, soll dieser andere Minibus die neue Fahrt nur akzeptieren, wenn er ohnehin gerade wartet.

Nachdem bisher für das Problem keine Transport-Strategien untersucht wurden, bei denen Fahrten auf Basis von Fahrtanfragen komplett neu geplant wurden, wurde versucht, das Problem mit einem einfachen gierigen Algorithmus zu lösen.

Jedes Mal, wenn eine Fahrtanfrage gestellt wird, wird diese entweder einem Minibus zugeordnet oder abgelehnt. Wenn sie einem Bus zugewiesen wurde, wird sie nie mehr zu einem anderen zugewiesen. In diesem Fall wird die zugehörige Fahrt in den Aktionsplan des ausgewählten Minibusses integriert. Passagiere steigen bei dieser Strategie nur in einen Bus ein, wenn ihre Fahrtanfrage dem Bus zugeordnet wurde.

Konkret kann das im Algorithmus 1 nachvollzogen werden und wird nun näher beschrieben.

Genaue Definition der anfragebasierten Strategie

Schritt 1: Busse, die direkt fahren können, ermitteln

Wenn eine Fahrtanfrage r gestellt wird, also zum Zeitpunkt a_r, wird zunächst die nachgefragte Fahrtrichtung ermittelt. Anschließend werden alle Busse in zwei Gruppen eingeteilt: Alle Busse, die wenden müssen, um den Passagier der Anfrage mitzunehmen, kommen in B_w und alle anderen, also diejenigen, die direkt die Fahrtanfrage einfügen können in B_d. Da jeder Bus, der nicht in die gleiche Richtung wie die neue Anfrage fährt, wenden muss, um die neue Anfrage mitzunehmen, sind diese alle in B_w. Für die Busse, die in die gleiche Richtung fahren, wie die angefragte Fahrtrichtung, gibt es verschiedene Fälle, die in Abbildung 3.4 beispielhaft dargestellt sind. Die Pfeile in der Grafik entsprechen den bereits vorhandenen (zukünftigen) Aktionsplänen der Busse. Alle Busse, die an der Starthaltestelle vorbeikommen ohne davor zu wenden, sind in B_d. "Vorbeikommen"

meint in diesem Fall alle Busse, die an der Starthaltestelle anhalten oder vorbeifahren und anschließend in Fahrtrichtung der neuen Fahrtanfrage weiterfahren. Alle anderen sind ebenfalls in B_w.

Algorithmus 1: AnfrageZuBusZuweisen(Anfrage r, int t)

Eingabe: neue Anfrage r, aktuelle Zeit t
Ausgabe: wenn Zuweisung möglich (Zugewiesener Bus, neuer Aktionsplan), sonst **null**

```
1  B_d = GebeDirektFahrendeBusse(r, t)
2  B_w = B \ B_d
3  C = []                                              /* C ist eine Liste */
4  foreach b ∈ B_d do
5      P_neu = null                  /* P_neu soll ein neuer Aktionsplan werden */
6      if b ∈ B_d then
7          P_neu = IntegriereAnfrageBeimFahren(b, r, t)
8      else
9          P_neu = IntegriereAnfrageBeimWarten(b, r, t)
10     t_warten = GebeWartezeit(P_neu, r)
11     Hinzufügen(C, (b, P_neu, t_warten))
12 SortiereAufsteigend(C, Nach(t_warten))
13 foreach (b, P_neu, t_warten) ∈ C do
14     if IstMachbar(P_neu) then
15         return (b, P_neu)
16 return null
```

Zur exakten Definition ist die Aufteilung der Busse in Pseudocode in Algorithmus 2 beschrieben. Zur Veranschaulichung wird in einigen Fällen im Folgenden auf Busse aus dem Beispiel in Abbildung 3.4 verwiesen. Für jeden Bus wird seine aktuelle (bzw. wenn er steht nächste) Fahrtrichtung berechnet. Alle Busse, die zum gewünschten Abfahrtszeitpunkt g_r in die falsche Richtung fahren, werden aussortiert (Zeile 9). Anschließend muss für jeden Bus b der Aktionsplan durchgegangen werden. Dazu wird jeweils ein Paar aus zwei aufeinanderfolgenden Aktionen $(p_{\text{vorherig}}, p_{\text{nächste}})$ des alten Aktionsplans betrachtet. Das erste betrachtete Paar enthält die letzte zum Zeitpunkt t bereits ausgeführte Aktion und die erste Aktion, die noch nicht ausgeführt wurde. Falls die nächste Aktion eine Abfahrtsaktion an der Starthaltestelle o_r ist, hält der Bus im Moment bei o_r und wird daher in B_d aufgenommen (Zeile 15) (Bus b_3). Wenn beim Durchgehen der Aktionspaare ein Paar gefunden wird, bei dem die nächste Aktion eine Ankunft ist und der Bus dabei in die falsche Richtung fährt, wird der Bus wenden bevor er bei o_r vorbeifährt und kommt damit nicht in B_d (Zeile 20) (Busse b_4, b_5). Wenn ein Paar aus einer Abfahrts- und einer Ankunftsaktion gefunden wird, zwischen deren Haltestellen o_r liegt, kann der Bus an sich zu B_d hinzugefügt werden (Zeile 26) (mglw. Busse b_1, b_2). Allerdings muss der Sonderfall berücksichtigt werden, dass der Bus zum Zeitpunkt t

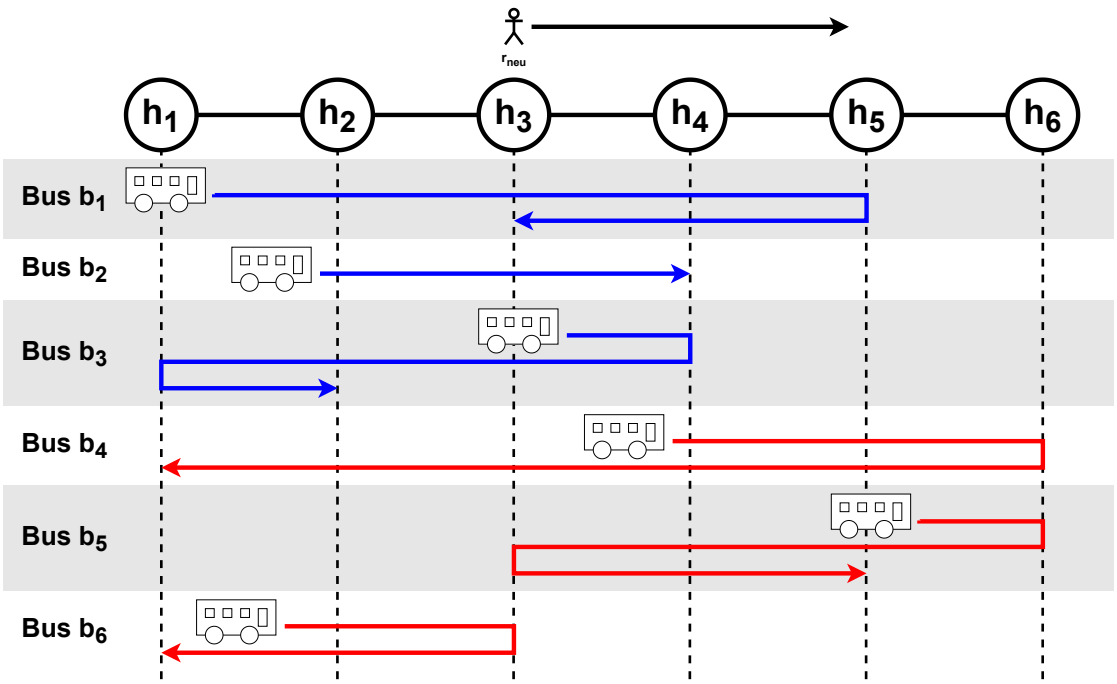


Abb. 3.4.: Ermittlung, ob Busse nach dem gewünschten Abfahrtszeitpunkt $g_{r_{\text{neu}}}$ der Anfrage r_{neu} direkt an der Starthaltestelle $o_{r_{\text{neu}}}$ vorbeikommen. In der Grafik sind alle Busse mit blauen Pfeilen in B_d und alle mit roten Pfeilen in B_w.

bereits an der Haltestelle vorbeigefahren sein könnte, da die letzte Aktion p_{vorherig} in der Vergangenheit liegen kann. In diesem Sonderfall, wird der Bus nicht mehr, ohne zu wenden, an der Haltestelle o_r vorbeikommen und damit kommt er nicht in B_d (Zeile 23). Schließlich kann es auch vorkommen, dass ein Paar aus Abfahrts- und Ankunftsaktion die Starthaltestelle o_r als Ankunftshaltestelle hat. Dann muss überprüft werden, ob der Bus nach diesem Halt weiter in Richtung der Zielhaltestelle fährt oder ob der Bus an der Starthaltestelle wendet/endet. In ersterem Fall kann der Bus in B_d aufgenommen werden (Zeile 32) (mglw. Busse b_1, b_2) in letzterem Fall nicht (Zeile 30) (Bus b_6).

Schritt 2: Aktionspläne für alle Busse ermitteln

Im Anschluss wird für alle Busse berechnet, wann sie frühstmöglich bei der Starthaltestelle ankommen können. Dazu wird testweise ein neuer Aktionsplan für jeden Bus ermittelt, bei der die neue Fahrtanfrage integriert ist. Das passiert auf unterschiedliche Weise, je nachdem, ob der betrachtete Bus b in der Menge B_d oder B_w ist. Außerdem hängt das von dem bisherigen Aktionsplan des jeweiligen Busses ab.

Schritt 2a: Aktionspläne für Busse, die direkt fahren können, ermitteln

Wie man für die Busse $b \in B_d$ zu einem neuen Aktionsplan kommt, der die neue Fahrtanfrage r berücksichtigt, wird im Folgenden erst grob und danach ausführlich erläutert und findet sich in Algorithmus 3.

Zunächst werden alle Aktionen, die vor der gewünschten Abfahrtszeit g_r liegen, in den neuen Aktionsplan kopiert (Zeile 4). Beim Iterieren über die restlichen Aktionen des alten Aktionsplans durchläuft der Algorithmus fünf verschiedene Phasen. Alle Aktionen, die auf der geplanten Strecke des Busses passieren, bevor der Bus an der Starthaltestelle o_r vorbeikommt, können aus dem alten Aktionsplan übernommen werden (Zeile 5). Anschließend muss die Starthaltestelle in den neuen Aktionsplan P_{neu} eingefügt werden. Je nachdem, ob dafür ein neuer Halt eingeplant werden muss, kann es dabei zu Verzögerungen kommen (Zeile 6). Danach müssen alle Aktionen, die zwischen der Start- und der Zielhaltestelle von r liegen, kopiert und gegebenenfalls verzögert werden. Die Verzögerung kann sich dabei reduzieren, falls an einer Haltestelle länger als die minimale Standzeit gehalten wird (Zeile 7). Daraufhin muss die Zielhaltestelle eingefügt werden. Hierbei kann es sowohl zu weiteren Verzögerungen kommen, wenn neue Fahr- oder Standzeiten eingeplant werden, als auch zu einer Verzögerungsreduktion, falls an der Zielhaltestelle bereits länger als die minimale Standzeit gehalten wird (Zeile 8). Schließlich werden alle weiteren Aktionen nach der Zielhaltestelle kopiert und gegebenenfalls verspätet eingefügt. Wenn möglich wird auch hier die Verzögerung reduziert (Zeile 9).

Auch dieser Algorithmus lässt sich exakt definieren. Dazu werden nun die verwendeten Funktion genau beschrieben.

Algorithmus 4 kopiert alle Aktionen in den neuen Aktionsplan bevor die Starthaltestelle eingefügt werden soll. Damit bestimmt er insbesondere die Position im Aktionsplan, bei der der Halt an der Starthaltestelle eingeplant werden soll. Zu Beginn des Aufrufs ist der Index i durch Algorithmus 3 so gewählt, dass die vorherige Aktion p_{vorherig} vor

Algorithmus 2: GebeDirektFahrendeBusse(Anfrage r, int t)

Eingabe: neue Anfrage r, aktuelle Zeit t

Ausgabe: Liste der direkt fahrenden Busse B_d

```
1  B_d = []                                              /* B_d ist eine Liste */
2  if t = 0 then
3      return B_d          /* Positionen der Busse sind noch unbekannt */
4  foreach b ∈ B do
5      P_alt = ϱ(b, t − 1)
6      i = GebeAktuellenAktionsplanIndex(P_alt, g_r)
7      f = GebeNächsteFahrtrichtung(P_alt, i)
8      if f ≠ ϑ(r) then
9          continue
10     while i < |P_alt| do
11         (γ_vorherig, σ_vorherig, χ_vorherig, Ψ_vorherig) = p_vorherig = P_alt[i]
12         (γ_nächste, σ_nächste, χ_nächste, Ψ_nächste) = p_nächste = P_alt[i + 1]
13         if γ_nächste = abfahrt then
14             if σ_nächste = o_r then
                   /* b steht bereits an der Starthaltestelle           */
15                 Hinzufügen(B_d, b)
16                 continue foreach
17         else
18             f = GebeNächsteFahrtrichtung(P_alt, i)
19             if f ≠ ϑ(r) then
                   /* b kehrt vor Starthaltestelle um                   */
20                 continue foreach
21             if LiegtDazwischen(o_r, σ_vorherig, σ_nächste) then
22                 if χ_vorherig + v · δ(σ_vorherig, o_r) < t then
                       /* b ist eben an Starthaltestelle vorbeigefahren */
23                     continue foreach
24                 else
                       /* b fährt noch an Starthaltestelle vorbei       */
25                     Hinzufügen(B_d, b)
26                     continue foreach
27             if σ_nächste = o_r then
28                 f = GebeNächsteFahrtrichtung(P_alt, i + 1)
29                 if f ≠ ϑ(r) then
                       /* b kehrt an Starthaltestelle um oder endet     */
30                     continue foreach
31                 else
                       /* b hält noch an Starthaltestelle               */
32                     Hinzufügen(B_d, b)
33                     continue foreach
34         i = i + 1
       /* Wenn dieser Punkt erreicht wird, endet der Aktionsplan von b
          vor der Starthaltestelle.                                     */
35 return B_d
```

Algorithmus 3: IntegriereAnfrageBeimFahren(Bus b, Anfrage r, int t)

Eingabe: Bus b, neue Anfrage r, aktuelle Zeit t

Ausgabe: Neuer Aktionsplan mit integrierter Anfrage P_{neu}

1 $P_{\text{neu}} = []$ `/* P_neu ist eine Liste */`
2 $P_{\text{alt}} = \varrho(b, t-1)$
3 $i =$ `GebeAktuellenAktionsplanIndex`(P_{alt}, g_r)
4 `Hinzufügen`$(P_{\text{neu}}, P_{\text{alt}}[1 : i+1])$ `/* Kopiere die ersten i Aktionen */`
5 $(P_{\text{neu}}, i) =$ `KopiereAktionenVorStart`$(P_{\text{alt}}, P_{\text{neu}}, i, r)$
6 $(P_{\text{neu}}, i, t_{\text{Verzögerung}}) =$ `FügeStartEin`$(P_{\text{alt}}, P_{\text{neu}}, i, r)$
7 $(P_{\text{neu}}, i, t_{\text{Verzögerung}}) =$ `VerzögereAktionenZwSUndZ`$(P_{\text{alt}}, P_{\text{neu}}, i, t_{\text{Verzögerung}}, r)$
8 $(P_{\text{neu}}, i, t_{\text{Verzögerung}}) =$ `FügeZielEin`$(P_{\text{alt}}, P_{\text{neu}}, i, t_{\text{Verzögerung}}, r)$
9 $P_{\text{neu}} =$ `VerzögereAktionenNachZiel`$(P_{\text{alt}}, P_{\text{neu}}, i, t_{\text{Verzögerung}})$
10 **return** P_{neu}

der gewünschten Abfahrtszeit liegt und die nächste Aktion $p_{\text{nächste}}$ zu oder nach der gewünschten Abfahrtszeit geplant ist. Mit $p_{\text{nächste}}$ beginnend, wird jede folgende Aktion untersucht. Wenn sie eine Abfahrtsaktion ist und als zugehörige Haltestelle die Starthaltestelle o_r der Anfrage r enthält, wurde die Position im Aktionsplan gefunden, bei der die Starthaltestelle liegt (Zeile 6). Wenn $p_{\text{nächste}}$ eine Ankunftsaktion ist und zwischen der Haltestelle der letzten Aktion σ_{vorherig} und der Haltestelle der nächsten Aktion $\sigma_{\text{nächste}}$ die Starthaltestelle o_r liegt, wurde die Starthaltestelle ebenfalls entdeckt. Wenn beide Bedingungen nicht erfüllt sind, wird die Aktion aus dem alten Aktionsplan in den neuen Aktionsplan kopiert (Zeile 10) und anschließend mit der nächsten Aktion fortgefahren. Sobald eine Bedingung erfüllt ist, wird mit dem Kopieren abgebrochen und das Einfügen der Starthaltestelle wird dem Algorithmus 5 überlassen. Dadurch, dass diese Funktion nur für Aktionspläne von Bussen $b \in B_d$ aufgerufen wird, ist sichergestellt, dass, wenn die Funktion beendet wird, die neuen/geänderten Aktionen für die Starthaltestelle o_r als nächstes eingefügt werden können.

In Abbildung 3.5 ist beispielhaft dargestellt, wie sich der Index i bei einem beliebigen Aktionsplan durch den Aufruf des soeben beschriebenen Algorithmus ändert, wenn eine neue Fahrtanfrage r_4 mit $o_{r_4} = h_5$, $d_{r_4} = h_8$, $a_{r_4} = 10$ und $g_{r_4} = 20$ eingefügt werden soll. Die Aktionen für den Halt bei der Starthaltestelle werden im neuen Aktionsplan zwischen den Aktionen mit Index 6 und Index 7 eingeordnet.

Nachdem der Index im alten Aktionsplan gefunden wurde, bei dem die neuen/geänderten Aktionen für die Starthaltestelle einsortiert werden sollen, muss das auch umgesetzt werden. Hierbei gibt es zwei Varianten: Entweder ist vorgesehen, dass der Bus sowieso an der Starthaltestelle anhält unabhängig von der neuen Fahrtanfrage r, oder der Bus muss einen neuen Zwischenhalt einlegen. In Abbildung 3.6 sind beide Varianten beispielhaft für eine Anfrage r mit $o_r = 2$, $d_r = 5$, $a_r = 10$ und $g_r = 15$ dargestellt. In Abbildung 3.6a hält der Bus bisher nicht an der Starthaltestelle o_r. Um den Zwischenhalt hinzuzufügen müssen die nachfolgenden Aktionen um die minimale Standzeit s herausgezögert werden. Wenn der Bus wie in Abbildung 3.6b sowieso hält, ist das nicht notwendig.

Algorithmus 4: KopiereAktionenVorStart(Liste P_{alt}, Liste P_{neu}, int i, Anfrage r)

Eingabe: Alter Aktionsplan P_{alt}, neuer Aktionsplan P_{neu}, Index i, einzufügende Anfrage r

Ausgabe: Neuer Aktionsplan P_{neu}, Index i

```
1  while i < |P_alt| do
2      (γ_vorherig, σ_vorherig, χ_vorherig, Ψ_vorherig) = p_vorherig = P_alt[i]
3      (γ_nächste, σ_nächste, χ_nächste, Ψ_nächste) = p_nächste = P_alt[i + 1]
4      if γ_nächste = abfahrt then
5          if σ_nächste = o_r then
6              break
7      else
8          if LiegtDazwischen(o_r, σ_vorherig, σ_nächste) then
9              break
10     Hinzufügen(P_neu, p_nächste)
11     i = i + 1
12 return (P_neu, i)
```

$i = 2$

Index	Typ γ	Haltestelle σ	Zeit χ	Anfragen Ψ
1	ankunft	h_1	0	unbenutzt
2	abfahrt	h_1	2	$\emptyset$
3	ankunft	h_3	22	unbenutzt
4	abfahrt	h_3	24	$\{r_1\}$
5	ankunft	h_4	34	unbenutzt
6	abfahrt	h_4	36	$\{r_2, r_3\}$
7	ankunft	h_6	56	unbenutzt
8	abfahrt	h_6	58	$\emptyset$
9	ankunft	h_7	68	unbenutzt

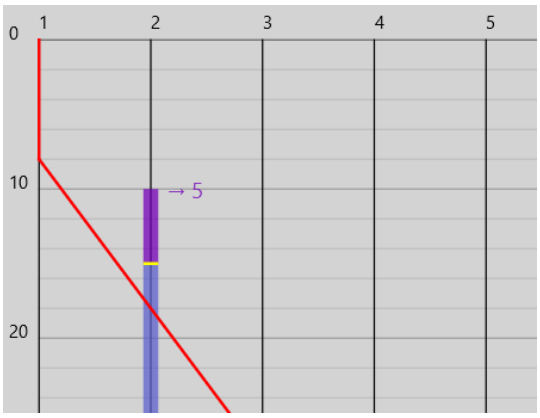
(a) Vor dem Aufruf

$i = 6$

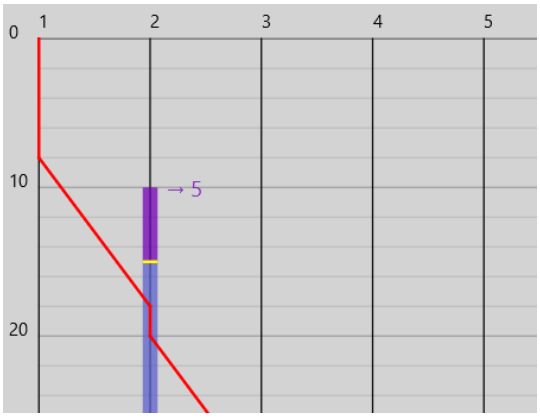
Index	Typ γ	Haltestelle σ	Zeit χ	Anfragen Ψ
1	ankunft	h_1	0	unbenutzt
2	abfahrt	h_1	2	$\emptyset$
3	ankunft	h_3	22	unbenutzt
4	abfahrt	h_3	24	$\{r_1\}$
5	ankunft	h_4	34	unbenutzt
6	abfahrt	h_4	36	$\{r_2, r_3\}$
7	ankunft	h_6	56	unbenutzt
8	abfahrt	h_6	58	$\emptyset$
9	ankunft	h_7	68	unbenutzt

(b) Nach dem Aufruf

Abb. 3.5.: Visualisierung der Änderung des Index i auf dem alten Aktionsplan P_{alt} durch den Aufruf von Algorithmus 4. Es sei angenommen, dass die Funktion zum Zeitpunkt $t = 10$ mit der neuen Anfrage r_4 mit $o_{r_4} = h_5$, $d_{r_4} = h_8$, $a_{r_4} = 10$ und $g_{r_4} = 20$ aufgerufen wurde.



(a) Der Bus fährt an der Starthaltestelle vorbei, ohne zu halten.



(b) Der Bus hält sowieso an der Starthaltestelle.

Abb. 3.6.: Zwei Möglichkeiten, wie der Aktionsplan eines Busses im Vergleich zur Starthaltestelle einer Anfrage verlaufen kann. Entlang der horizontalen Achse ist die Strecke dargestellt, entlang der vertikalen Achse die Zeit. Die rote Linie entspricht dem geplanten Aktionsplan eines Busses. Violett markiert ist der Zeitraum, in dem eine Fahrtanfrage von Haltestelle 2 zu Haltestelle 5 bekannt ist. Der blaue Zeitraum stellt die Zeit dar, den die Fahrtanfrage maximal wartet.

Genau beschrieben wird das von Algorithmus 5. Wenn die nächste Aktion eine Abfahrtsaktion ist, handelt es sich auf Grund des zuvor beschriebenen Algorithmus 4 um die Abfahrtsaktion an der Starthaltestelle o_r der neuen Anfrage r. In diesem Fall muss die nächste Aktion so modifiziert werden, dass die neue Anfrage r mit zu den Fahrtanfragen gehören, deren Passagiere bei dieser Aktion in den Bus einsteigen sollen (Zeile 5). Selbstverständlich entsteht dadurch keine Verzögerung. Im anderen Fall, also wenn die nächste Aktion eine Ankunftsaktion ist, hat der Bus bisher nicht vorgehabt, an o_r zu halten. Dann muss ein neuer Zwischenhalt bestehend aus einer Ankunfts- und einer Abfahrtsaktion eingeplant werden und zwar genau nach der Fahrzeit $v \cdot \delta(\sigma_{\text{vorherig}}, o_r)$. Dabei entsteht durch die minimale Standzeit eine Zeitverzögerung von s (Zeile 9). Falls die gewünschte Abfahrtszeit g_r nach der geplanten Abfahrt bei o_r wäre, muss die Verzögerung $t_{\text{Verzögerung}}$ entsprechend verlängert werden (Zeile 11). Dieser Fall ist nur auf Grund eines Fehlers im Algorithmus 2 möglich und wird in der Diskussion in Kapitel 5.1.4 beschrieben. Es ist zu beachten, dass die nächste Aktion $p_{\text{nächste}}$ aus dem alten Aktionsplan noch nicht verarbeitet wurde, weshalb der Index i nicht hochgezählt wird.

Nachdem sichergestellt wurde, dass die Starthaltestelle o_r der neuen Anfrage r im neuen Aktionsplan enthalten ist, kann es sein, dass einige andere Aktionen folgen, bevor die Aktionen für die Zielhaltestelle hinzugefügt werden müssen. Diese Aktionen zwischen Start- und Zielhaltestelle von r werden zunächst nur kopiert. Es kann jedoch vorkommen, dass sie verzögert werden müssen. Ein Beispiel hierfür findet sich in Abbildung 3.7. Dabei wird in einen alten Aktionsplan in Abbildung 3.7a eine Fahrtanfrage r mit $o_r = 2$, $d_r = 5$, $a_r = 8$ und $g_r = 10$ integriert. Durch das Einfügen der Starthaltestelle verschiebt sich die voraussichtliche Ankunft des Busses bei Haltestelle 3 um die minimale Standzeit

Algorithmus 5: FügeStartEin(Liste P_{alt}, Liste P_{neu}, int i, Anfrage r)

Eingabe: Alter Aktionsplan P_{alt}, neuer Aktionsplan P_{neu}, Index i, einzufügende Anfrage r

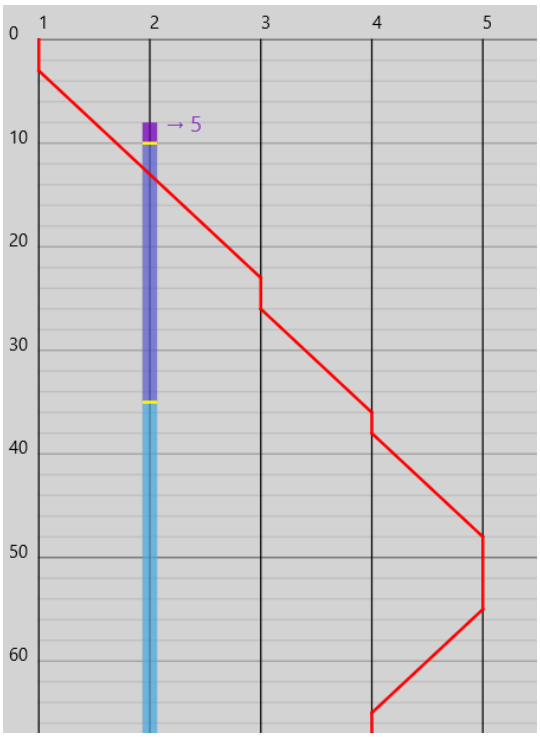
Ausgabe: Neuer Aktionsplan P_{neu}, Index i, Verzögerungszeit $t_{\text{Verzögerung}}$

```
1  t_Verzögerung = 0
2  (γ_vorherig, σ_vorherig, χ_vorherig, Ψ_vorherig) = p_vorherig = P_alt[i]
3  (γ_nächste, σ_nächste, χ_nächste, Ψ_nächste) = p_nächste = P_alt[i + 1]
4  if γ_nächste = abfahrt then
5  |  Hinzufügen(P_neu, (γ_nächste, σ_nächste, χ_nächste, Ψ_nächste ∪ {r}))
6  |  i = i + 1
7  else
8  |  Hinzufügen(P_neu, (ankunft, o_r, χ_vorherig + v · δ(σ_vorherig, o_r), unbenutzt))
9  |  t_Verzögerung = s
10 |  if χ_vorherig + v · δ(σ_vorherig, o_r) + t_Verzögerung < g_r then
11 |  |  t_Verzögerung = g_r − χ_vorherig − v · δ(σ_vorherig, o_r)
12 |  Hinzufügen(P_neu, (abfahrt, o_r, χ_vorherig + v · δ(σ_vorherig, o_r) + t_Verzögerung, {r}))
13 return (P_neu, i, t_Verzögerung)
```

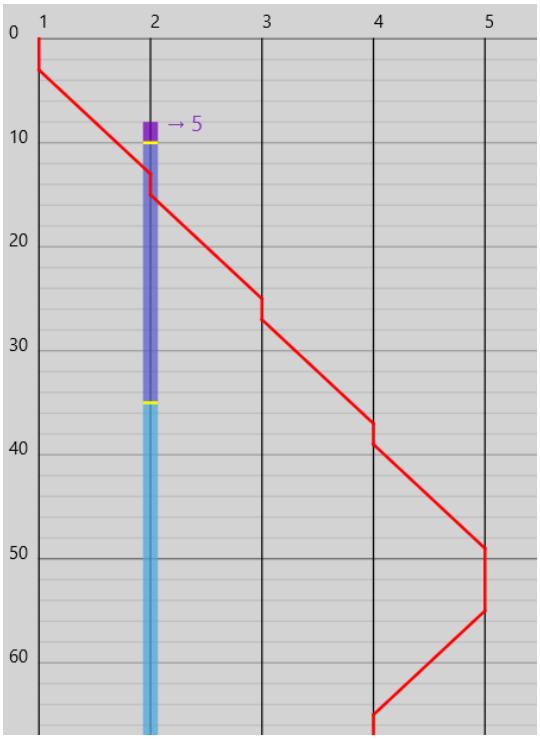
$s = 2$ von $\chi = 23$ zu $\chi = 25$ im neuen Aktionsplan in Abbildung 3.7b. Die Abfahrt bei Haltestelle 3 verschiebt sich allerdings nur noch um einen Zeitschritt nach hinten, da die Standzeit bei Haltestelle 3 einen Zeitschritt länger war als mindestens erforderlich ist und diese Standzeit folglich gekürzt werden kann.

Das Kopieren und Verzögern der Aktionen zwischen Start- und Zielhaltestelle ist in Algorithmus 6 definiert. Das Grundprinzip ist das gleiche wie bei der Funktion, die die Aktionen vor der Starthaltestelle kopiert: Es werden solange Aktionen kopiert, bis ein bestimmtes Abbruchkriterium erfüllt ist. Nur wird diesmal die Verzögerung auf jede Aktion, die kopiert wird, angewandt (Zeile 14). Falls an einer Station länger als die minimale Standzeit s gehalten wird, kann die Verzögerung reduziert werden (Zeile 8). Bei dieser Funktion gibt es vier Abbruchkriterien: Handelt es sich bei der nächsten Aktion um eine Abfahrtsaktion und ist die Fahrtrichtung danach nicht mehr gleich der Fahrtrichtung der neuen Fahrtanfrage r, wird der Kopiervorgang unterbrochen (Zeile 7). Auch wenn die nächste Aktion eine Ankunftsaktion an der Zielhaltestelle d_r ist, wird die Schleife vorzeitig verlassen (Zeile 11). Außerdem wird die Schleife auch beendet, wenn die nächste Aktion eine Ankunft ist und die Zielhaltestelle d_r zwischen der vorherigen und der nächsten Haltestelle liegt (Zeile 13). Schließlich gibt es noch den Fall, dass die Schleife durch die Schleifenbedingung endet (Zeile 1). Dann endet der Aktionsplan des Busses, bevor er an der Zielhaltestelle vorbeigekommen ist.

Anschließend muss die Zielhaltestelle eingefügt werden. Dabei gibt es vier verschiedene Situationen, die bestimmen, wie das passieren muss. Drei der vier werden im Folgenden an Beispielen in Abbildung 3.8 erklärt. Dabei ist die Ausgangssituation (Abbildung 3.8a) immer die selbe: Es gibt einen Bus (in den Abbildungen die rote, durchgezogene Linie),



(a) Vor dem Einfügen der Starthaltestelle



(b) Nach dem Einfügen der Starthaltestelle

Abb. 3.7.: Durch das Einfügen einer Starthaltestelle werden spätere Aktionen verzögert.

Algorithmus 6: VerzögereAktionenZwSUndZ(Liste P_{alt}, Liste P_{neu}, int i, int $t_{\text{Verzögerung}}$, Anfrage r)

Eingabe: Alter Aktionsplan P_{alt}, neuer Aktionsplan P_{neu}, Index i, Verzögerungszeit $t_{\text{Verzögerung}}$, einzufügende Anfrage r
Ausgabe: Neuer Aktionsplan P_{neu}, Index i, Verzögerungszeit $t_{\text{Verzögerung}}$

while $i < |P_{\text{alt}}|$ **do**
  $(\gamma_{\text{vorherig}}, \sigma_{\text{vorherig}}, \chi_{\text{vorherig}}, \Psi_{\text{vorherig}}) = p_{\text{vorherig}} = P_{\text{alt}}[i]$
  $(\gamma_{\text{nächste}}, \sigma_{\text{nächste}}, \chi_{\text{nächste}}, \Psi_{\text{nächste}}) = p_{\text{nächste}} = P_{\text{alt}}[i+1]$
  if $\gamma_{\text{nächste}} =$ abfahrt **then**
    $f =$ `GebeNächsteFahrtrichtung`(P_{alt}, i)
    if $f \neq \vartheta(r)$ **then**
      break
    $t_{\text{Verzögerung}} = \max(0, t_{\text{Verzögerung}} - (\chi_{\text{nächste}} - \chi_{\text{vorherig}} - s))$
  else
    if $d_r = \sigma_{\text{nächste}}$ **then**
      break
    if `LiegtDazwischen`$(d_r, \sigma_{\text{vorherig}}, \sigma_{\text{nächste}})$ **then**
      break
  `Hinzufügen`$(P_{\text{neu}}, (\gamma_{\text{nächste}}, \sigma_{\text{nächste}}, \chi_{\text{nächste}} + t_{\text{Verzögerung}}, \Psi_{\text{nächste}}))$
  $i = i + 1$
return $(P_{\text{neu}}, i, t_{\text{Verzögerung}})$

dessen Aktionsplan einige Aktionen nach der aktuellen Zeit $t = 2$ eingeplant hat. Zum Zeitpunkt $t = a_r = 5$ wird eine neue Fahrtanfrage r an der Haltestelle $o_r = 2$ gestellt. Je nachdem, welche Zielhaltestelle man wählt, entstehen drei unterschiedliche Situationen.

1. Ist das Ziel d_r die Haltestelle 3, so muss ein neuer Zwischenhalt während einer bereits geplanten Fahrt (also der Fahrt von Haltestelle 2 zu Haltestelle 4) eingefügt werden. Das kann in Abbildung 3.8b nachvollzogen werden.

2. Wenn die Zielhaltestelle d_r der neuen Fahrtanfrage Haltestelle 4 ist, hält der Bus sowieso an der Zielhaltestelle und es muss nichts geändert werden. Das sieht man in Abbildung 3.8c.

3. Wenn die Anfrage als Ziel Haltestelle 5 hat, wendet der Bus bereits bevor er an der Haltestelle vorbeikommt. Dann muss die entsprechende Fahrt bis Haltestelle 5 verlängert werden und anschließend eine Rückfahrt zur vorherigen Wendehaltestelle eingeplant werden. Das ist in Abbildung 3.8d dargestellt.

4. Der vierte Fall beim Einfügen der Zielhaltestelle ist, dass der Aktionsplan des Busses endet, bevor er an der Zielhaltestelle vorbeikommt. Er ist nicht in der Abbildung 3.8 enthalten. In diesem Fall kann die Fahrt einfach bis zur Zielhaltestelle verlängert werden. Das entspricht im Prinzip dem Fall 3 aus Abbildung 3.8d nur mit dem Unterschied, dass die Rückfahrt zur Haltestelle 4 entfallen würde.

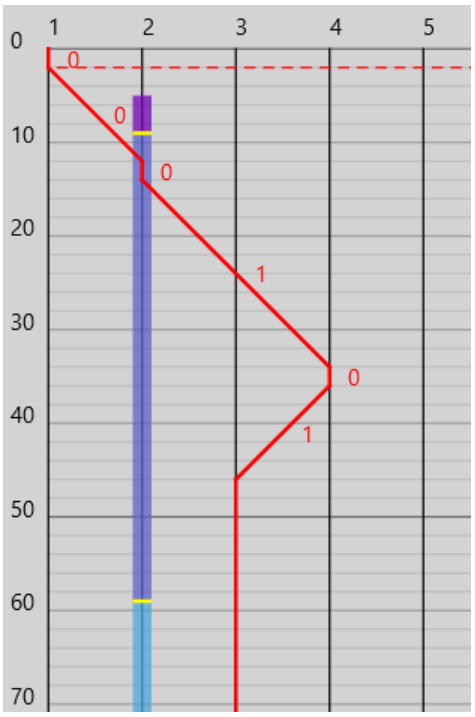
Welche Aktionen exakt in jedem Fall eingefügt werden, kann in Algorithmus 7 nachgelesen werden.

Wie der Abbildung 3.8 deutlich entnommen werden kann, kann es durch das Einfügen der Zielhaltestelle d_r zu Verzögerungen bei den nachfolgenden Aktionen kommen. Auch die Verzögerungen, die durch das Einfügen der Starthaltestelle o_r aufgekommen sind, können sich noch auf die Aktionen nach dem Halt bei der Zielhaltestelle d_r auswirken. Daher können die Aktionen nach dem Halt bei der Zielhaltestelle kopiert werden, allerdings müssen die Zeiten entsprechend angepasst werden. Das ist in Algorithmus 8 beschrieben. Wie bereits bei der Funktion, die die Aktionen zwischen dem Halt bei der Starthaltestelle und dem Halt bei der Zielhaltestelle verzögert hat, können auch hier Standzeiten, die länger als die minimale Standzeit s sind, dazu genutzt werden, die Verzögerung abzubauen (Zeile 5).

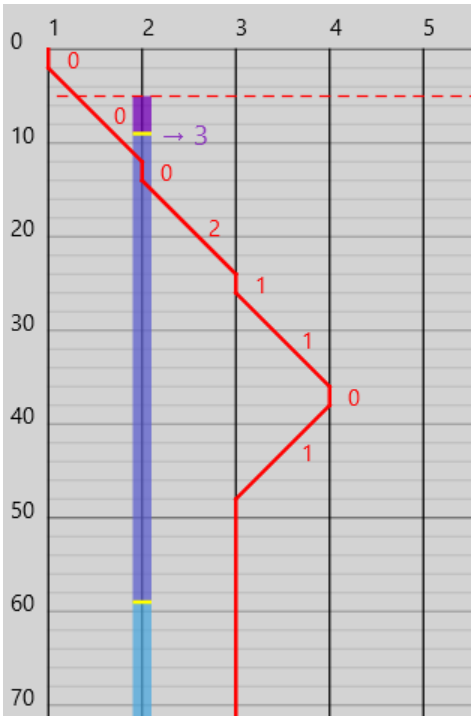
Damit ist die Erstellung der Aktionspläne für Busse $b \in B_d$, die ohne zu wenden an der Starthaltestelle vorbeikommen, vollständig definiert. Für alle anderen Busse $b \in B_w = B \setminus B_d$ gibt es einen eigenen Algorithmus, der nachfolgend dargelegt wird.

Schritt 2b: Aktionspläne für Busse, die wenden müssen, ermitteln

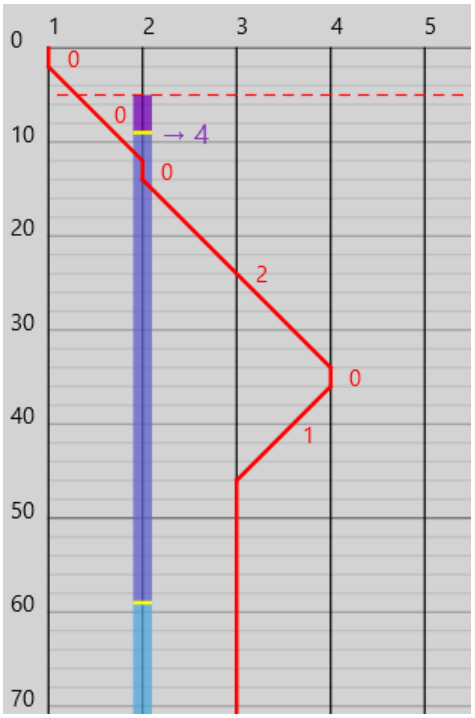
Wie man für einen Bus $b \in B_w$ zu einem neuen Aktionsplan kommt, der die neue Fahrtanfrage r berücksichtigt, wird nun grob beschrieben und ist in Algorithmus 9 definiert. Für eine ausführliche Definition aller verwendeter Methoden sei auf die Algorithmen 10, 11, 12 und 8 verwiesen.



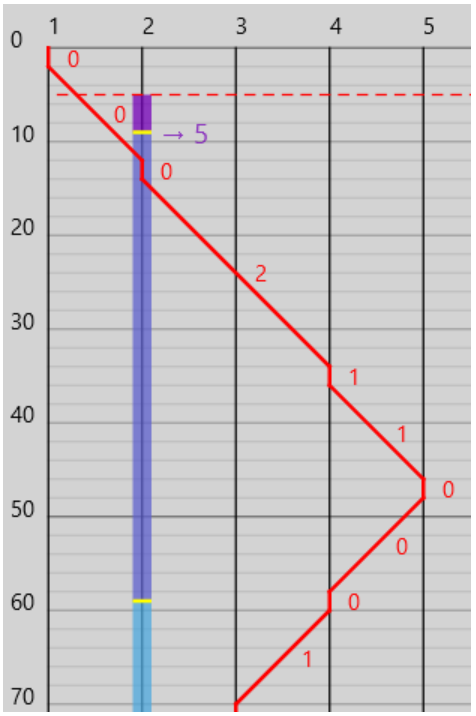
(a) Ausgangssituation



(b) Fall 1: Einfügen von r mit $d_r = 3$ führt zu einem neuen Zwischenhalt



(c) Fall 2: Einfügen von r mit $d_r = 4$ verändert den Aktionsplan nicht



(d) Fall 3: Einfügen von r mit $d_r = 5$ führt zu einer Fahrtverlängerung

Abb. 3.8.: Verschiedene Varianten, wie die Zielhaltestelle von einer neuen Anfrage r mit $o_r = 2$ in den Aktionsplan eines Busses integriert werden können.

Algorithmus 7: FügeZielEin(Liste P_{alt}, Liste P_{neu}, int i, int $t_{\text{Verzögerung}}$, Anfrage r)

Eingabe: Alter Aktionsplan P_{alt}, neuer Aktionsplan P_{neu}, Index i, Verzögerungszeit $t_{\text{Verzögerung}}$, einzufügende Anfrage r
Ausgabe: Neuer Aktionsplan P_{neu}, Index i, Verzögerungszeit $t_{\text{Verzögerung}}$

1 $(\gamma_{\text{vorherig}}, \sigma_{\text{vorherig}}, \chi_{\text{vorherig}}, \Psi_{\text{vorherig}}) = p_{\text{vorherig}} = P_{\text{alt}}[i]$
2 **if** $i = |P_{\text{alt}}|$ **then**
  `/* Fall 4: Nach Ende des Aktionsplans einfügen */`
3 $t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + s$
4 `Hinzufügen`$(P_{\text{neu}}, (\text{abfahrt}, \sigma_{\text{vorherig}}, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \emptyset))$
5 $t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + v \cdot \delta(\sigma_{\text{vorherig}}, d_r)$
6 `Hinzufügen`$(P_{\text{neu}}, (\text{ankunft}, d_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \text{unbenutzt}))$
7 **else**
8 $(\gamma_{\text{nächste}}, \sigma_{\text{nächste}}, \chi_{\text{nächste}}, \Psi_{\text{nächste}}) = p_{\text{nächste}} = P_{\text{alt}}[i+1]$
9 **if** $\gamma_{\text{nächste}} = \text{abfahrt}$ **then**
    `/* Fall 3: Nach Wendehaltestelle einfügen */`
10 $t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + s$
11 `Hinzufügen`$(P_{\text{neu}}, (\text{abfahrt}, \sigma_{\text{vorherig}}, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \emptyset))$
12 $t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + v \cdot \delta(\sigma_{\text{vorherig}}, d_r)$
13 `Hinzufügen`$(P_{\text{neu}}, (\text{ankunft}, d_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \text{unbenutzt}))$
14 $t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + s$
15 `Hinzufügen`$(P_{\text{neu}}, (\text{abfahrt}, d_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \emptyset))$
16 $t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + v \cdot \delta(d_r, \sigma_{\text{nächste}})$
17 `Hinzufügen`$(P_{\text{neu}}, (\text{ankunft}, \sigma_{\text{nächste}}, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \text{unbenutzt}))$
18 **else**
19 **if** $\sigma_{\text{nächste}} = d_r$ **then**
    `/* Fall 2: Zielhaltestelle wird sowieso bedient */`
20 `Hinzufügen`$(P_{\text{neu}}, (\gamma_{\text{nächste}}, \sigma_{\text{nächste}}, \chi_{\text{nächste}} + t_{\text{Verzögerung}}, \Psi_{\text{nächste}}))$
21 $i = i + 1$
22 **else**
    `/* Fall 1: Einfügen eines Zwischenhalts */`
23 `Hinzufügen`$(P_{\text{neu}}, (\text{ankunft}, d_r, \chi_{\text{vorherig}} + v \cdot \delta(\sigma_{\text{vorherig}}, d_r) + t_{\text{Verzögerung}}, \text{unbenutzt}))$
24 $t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + s$
25 `Hinzufügen`$(P_{\text{neu}}, (\text{abfahrt}, d_r, \chi_{\text{vorherig}} + v \cdot \delta(\sigma_{\text{vorherig}}, d_r) + t_{\text{Verzögerung}}, \emptyset))$
26 **return** $(P_{\text{neu}}, i, t_{\text{Verzögerung}})$

Algorithmus 8: VerzögereAktionenNachZiel(Liste P_{alt}, Liste P_{neu}, int i, int $t_{\text{Verzögerung}}$)

Eingabe: Alter Aktionsplan P_{alt}, neuer Aktionsplan P_{neu}, Index i, Verzögerungszeit $t_{\text{Verzögerung}}$
Ausgabe: Neuer Aktionsplan P_{neu}

1 **while** $i < |P_{\text{alt}}|$ **do**
2 $\quad(\gamma_{\text{vorherig}}, \sigma_{\text{vorherig}}, \chi_{\text{vorherig}}, \Psi_{\text{vorherig}}) = p_{\text{vorherig}} = P_{\text{alt}}[i]$
3 $\quad(\gamma_{\text{nächste}}, \sigma_{\text{nächste}}, \chi_{\text{nächste}}, \Psi_{\text{nächste}}) = p_{\text{nächste}} = P_{\text{alt}}[i+1]$
4 $\quad$**if** $\gamma_{\text{nächste}} = \text{abfahrt}$ **then**
5 $\quad\quad t_{\text{Verzögerung}} = \max(0, t_{\text{Verzögerung}} - (\chi_{\text{nächste}} - \chi_{\text{vorherig}} - s))$
6 $\quad$`Hinzufügen`$(P_{\text{neu}}, (\gamma_{\text{nächste}}, \sigma_{\text{nächste}}, \chi_{\text{nächste}} + t_{\text{Verzögerung}}, \Psi_{\text{nächste}}))$
7 **return** P_{neu}

Ähnlich wie bei der Funktion, die eine Anfrage während der Fahrt integriert, werden zunächst alle bereits vergangenen Aktionen aus dem alten Aktionsplan P_{alt} in den neuen P_{neu} kopiert (Zeile 4 in Algorithmus 9). Der Unterschied ist nur, dass dies nicht für alle Aktionen bis zur gewünschten Abfahrtszeit g_r der neuen Anfrage r gemacht wird, sondern nur für alle Aktionen vor der aktuellen Zeit t. Anschließend werden alle Fahrtanfragen ermittelt, deren Passagiere zum Zeitpunkt t im Bus b sitzen, und in einer Liste ω_{geplant} gespeichert (Zeile 5). Die Liste ω_{geplant} wird von den folgenden Kopier-Funktionen jeweils aktualisiert, sodass immer bekannt ist, welche Anfragen bei welchem Index i des Aktionsplans gerade mit dem Bus transportiert werden. Danach werden alle Aktionen bis zum gewünschten Abfahrtszeitpunkt g_r kopiert, wobei ω_{geplant} so aktualisiert wird, dass es alle Fahrtanfragen enthält, die zum gewünschten Abfahrtszeitpunkt g_r transportiert werden sollen (Zeile 6). Ab den Aktionen, die nach g_r geplant sind, wird bei jedem Halt überprüft, ob der Bus länger als mindestens notwendig hält und falls dem so ist, ob er dann leer ist. Bis diese Bedingungen erfüllt sind, werden alle Aktionen jeweils kopiert und auf die Liste der geplanten Fahrgäste ω_{geplant} angewandt, sodass diese nach jeder kopierten Aktion immer die Passagiere enthält, die voraussichtlich im Bus sitzen sollen (Zeile 7). Im Anschluss wird die für die Anfrage r notwendige Fahrt (inklusive leerer Anfahrt und leerer Rückfahrt) in den Aktionsplan eingefügt (Zeile 8). Da die Fahrt länger dauern könnte als die Wartezeit an der Haltestelle ist, müssen die nachfolgenden Aktionen gegebenenfalls verzögert werden (Zeile 9).

Nachdem das Kopieren der Aktionen bereits ähnlich und sehr ausführlich in Schritt 2a beschrieben wurde, wird das an dieser Stelle übersprungen. Der Genauigkeit halber kann die Funktion, die die Aktionen vor der gewünschten Abfahrtszeit kopiert, in Algorithmus 10 gefunden werden. Die Funktion, die die Aktionen kopiert, bevor der Bus leer ist und länger als die minimale Standzeit s hält, ist sehr ähnlich und hat hauptsächlich eine andere Abbruchbedingung. Sie findet sich in Algorithmus 11. Die Funktion, die den Fahrgastwechsel ausführt, ändert nur die Liste der Fahrgäste, ohne dabei zeitliche Bedingungen zu überprüfen. Sie ist im Anhang in Abschnitt B.3 definiert.

Für das Einfügen der Fahrt für die neue Anfrage r gibt es verschiedene Fälle, die

Algorithmus 9: IntegriereAnfrageBeimWarten(Bus b, Anfrage r, int t)

Eingabe: Bus b, neue Anfrage r, aktuelle Zeit t
Ausgabe: Neuer Aktionsplan mit integrierter Anfrage P_{neu}

```
1  P_neu = []                                          /* P_neu ist eine Liste */
2  P_alt = ϱ(b, t − 1)
3  i = GebeAktuellenAktionsplanIndex(P_alt, t)
4  Hinzufügen(P_neu, P_alt[1 : i + 1])          /* Kopiere die ersten i Aktionen */
5  ω_geplant = ω(b, t)                                 /* ω_geplant ist eine Liste */
6  (P_neu, i, ω_geplant) = KopAkVorGewAbfahrt(P_alt, P_neu, i, ω_geplant, r)
7  (P_neu, i, ω_geplant) = KopAkVorLeerWarten(P_alt, P_neu, i, ω_geplant, r)
8  (P_neu, i, t_Verzögerung) = FügeFahrtEin(P_alt, P_neu, i, r, t)
9  P_neu = VerzögereAktionenNachZiel(P_alt, P_neu, i, t_Verzögerung)
10 return P_neu
```

Algorithmus 10: KopAkVorGewAbfahrt(Liste P_{alt}, Liste P_{neu}, int i, Liste ω_{geplant}, Anfrage r)

Eingabe: Alter Aktionsplan P_{alt}, neuer Aktionsplan P_{neu}, Index i, bei Index geplant transportierte Fahrtanfragen ω_{geplant}, einzufügende Anfrage r
Ausgabe: Neuer Aktionsplan P_{neu}, Index i, bei Index geplant transportierte Fahrtanfragen ω_{geplant}

```
1 while i < |P_alt| do
2     (γ_vorherig, σ_vorherig, χ_vorherig, Ψ_vorherig) = p_vorherig = P_alt[i]
3     (γ_nächste, σ_nächste, χ_nächste, Ψ_nächste) = p_nächste = P_alt[i + 1]
4     if χ_nächste ≥ g_r then
5         break
6     Hinzufügen(P_neu, p_nächste)
7     FührеFahrgastwechselAus(ω_geplant, p_nächste)
8     i = i + 1
9 return (P_neu, i, ω_geplant)
```

Algorithmus 11: KopAkVorLeerWarten(Liste P_{alt}, Liste P_{neu}, int i, Liste ω_{geplant}, Anfrage r)

Eingabe: Alter Aktionsplan P_{alt}, neuer Aktionsplan P_{neu}, Index i, bei Index geplant transportierte Fahrtanfragen ω_{geplant}, einzufügende Anfrage r
Ausgabe: Neuer Aktionsplan P_{neu}, Index i, bei Index geplant transportierte Fahrtanfragen ω_{geplant}

```
1  while i < |P_alt| do
2      (γ_vorherig, σ_vorherig, χ_vorherig, Ψ_vorherig) = p_vorherig = P_alt[i]
3      (γ_nächste, σ_nächste, χ_nächste, Ψ_nächste) = p_nächste = P_alt[i + 1]
4      if γ_nächste = abfahrt then
5          if χ_nächste − χ_vorherig > s then
6              if |ω_geplant| = 0 then
7                  break
8      Hinzufügen(P_neu, p_nächste)
9      FührеFahrgastwechselAus(ω_geplant, p_nächste)
10     i = i + 1
11 return (P_neu, i, ω_geplant)
```

nachfolgend dargestellt werden. Die Funktion, durch die die neue Fahrt eingefügt wird, ist in Algorithmus 12 zu finden.

Durch die beiden vorhergehenden Funktionen ist sichergestellt, dass wenn die neue Fahrt eingefügt werden soll, die vorherige Aktion

$$p_{\text{vorherig}} = (\text{ankunft}, \sigma, \chi_{\text{vorherig}}, \text{unbenutzt})$$

eine Ankunftsaktion ist. Allerdings kann es sein, dass der Zeitpunkt der Ankunftsaktion bereits vor der aktuellen Zeit t liegt, also $\chi_{\text{vorherig}} < t$ gilt. Zunächst muss ein Halt bei der Starthaltestelle von r eingeplant werden. Dafür gibt es zwei Fälle:

1. $\sigma = o_r$:

   Im ersten Fall ist die Haltestelle σ, bei der der Bus wartet, die gleiche Haltestelle, wie die, an der die neue Fahrtanfrage r starten möchte. Da der Bus mindestens die minimale Standzeit s abwarten muss, bevor er weiterfährt, kann die Abfahrtsaktion an der Starthaltestelle frühstens zum Zeitpunkt $\chi_{\text{vorherig}} + s$ eingefügt werden. Es kann aber auch vorkommen, dass die gewünschte Abfahrtszeit nach diesem frühstmöglichen Zeitpunkt liegt, also $\chi_{\text{vorherig}} + s < g_r$ gilt. Dazu wird eine erste Verzögerung $t_{\text{Verzögerung,s}} = \max(s, g_r - \chi_{\text{vorherig}})$ notiert. Anschließend wird die folgende Abfahrtsaktion eingefügt:

   $$(\text{abfahrt}, \sigma, \chi_{\text{vorherig}} + t_{\text{Verzögerung,s}}, g_r), \{r\})$$

2. $\sigma \neq o_r$:

Algorithmus 12: FügeFahrtEin(Liste P_{alt}, Liste P_{neu}, int i, Anfrage r, int t)

Eingabe: Alter Aktionsplan P_{alt}, neuer Aktionsplan P_{neu}, Index i, einzufügende Anfrage r, aktuelle Zeit t

Ausgabe: Neuer Aktionsplan P_{neu}, Index i, Verzögerungszeit $t_{\text{Verzögerung}}$

1 $(\gamma_{\text{vorherig}}, \sigma_{\text{vorherig}}, \chi_{\text{vorherig}}, \Psi_{\text{vorherig}}) = p_{\text{vorherig}} = P_{\text{alt}}[i]$
2 $t_{\text{Verzögerung}} = s$
3 **if** $\chi_{\text{vorherig}} + t_{\text{Verzögerung}} < t$ **then**
4 $\quad t_{\text{Verzögerung}} = t - \chi_{\text{vorherig}}$
`/* Anfahrt zur Starthaltestelle der Anfrage` r `*/`
5 **if** $\sigma_{\text{vorherig}} \neq o_r$ **then**
6 $\quad$ `Hinzufügen`$(P_{\text{neu}}, (\text{abfahrt}, \sigma_{\text{vorherig}}, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \emptyset))$
7 $\quad t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + v \cdot \delta(\sigma_{\text{vorherig}}, o_r)$
8 $\quad$ `Hinzufügen`$(P_{\text{neu}}, (\text{ankunft}, o_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \text{unbenutzt}))$
9 $\quad t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + s$
`/* Fahrt der Anfrage` r `*/`
10 **if** $\chi_{\text{vorherig}} + t_{\text{Verzögerung}} < g_r$ **then**
11 $\quad t_{\text{Verzögerung}} = g_r - \chi_{\text{vorherig}}$
12 `Hinzufügen`$(P_{\text{neu}}, (\text{abfahrt}, o_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \{r\}))$
13 $t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + v \cdot \delta(o_r, d_r)$
14 `Hinzufügen`$(P_{\text{neu}}, (\text{ankunft}, d_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \text{unbenutzt}))$
`/* Rückfahrt von Zielhaltestelle der Anfrage` r `*/`
15 **if** $i < |P_{\text{alt}}|$ **then**
16 $\quad (\gamma_{\text{nächste}}, \sigma_{\text{nächste}}, \chi_{\text{nächste}}, \Psi_{\text{nächste}}) = p_{\text{nächste}} = P_{\text{alt}}[i+1]$
17 $\quad$ **if** $d_r = \sigma_{\text{nächste}}$ **then**
18 $\quad\quad t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + s$
19 $\quad\quad$ `Hinzufügen`$(P_{\text{neu}}, (\text{abfahrt}, d_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \emptyset))$
20 $\quad\quad t_{\text{Verzögerung}} = t_{\text{Verzögerung}} + v \cdot \delta(d_r, \sigma_{\text{nächste}})$
21 $\quad\quad$ `Hinzufügen`$(P_{\text{neu}}, (\text{ankunft}, \sigma_{\text{nächste}}, \chi_{\text{vorherig}} + t_{\text{Verzögerung}}, \text{unbenutzt}))$
22 **return** $(P_{\text{neu}}, i, t_{\text{Verzögerung}})$

Im wesentlich häufigeren Fall ist die Haltestelle σ, bei der der Bus wartet, eine andere Haltestelle als die Starthaltestelle. In diesem Fall müssen drei Aktionen hinzugefügt werden, um einen Halt an der Starthaltestelle zu ermöglichen. Die erste ist eine Abfahrtsaktion. Sie kann frühstens eingefügt werden, nachdem die minimale Standzeit s abgewartet wurde. Allerdings muss bei ihr sichergestellt werden, dass sie nicht in der Vergangenheit, also vor dem aktuellen Zeitpunkt t eingefügt wird. Folglich wird als erste Verzögerung $t_{\text{Verzögerung,s1}} = \max(s, t - \chi_{\text{vorherig}})$ festgehalten. Als erste Aktion wird die folgende eingeplant:

$$(\text{abfahrt}, \sigma, \chi_{\text{vorherig}} + t_{\text{Verzögerung,s1}}, \emptyset)$$

Weiter muss die Ankunftsaktion bei der Starthaltestelle o_r nach der Fahrzeit eingefügt werden. Dazu wird die Verzögerung $t_{\text{Verzögerung,s2}} = t_{\text{Verzögerung,s1}} + v \cdot \delta(\sigma, o_r)$ vermerkt. Es ergibt sich die folgende Aktion:

$$(\text{ankunft}, o_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung,s2}}, \text{unbenutzt})$$

An der Starthaltestelle angekommen, muss mindestens die minimale Standzeit s gewartet werden. Falls die daraus resultierende Abfahrtszeit vor der gewünschten Abfahrtszeit g_r liegt, wird die Abfahrt länger verzögert. Formal erhält man daher eine Verzögerung von $t_{\text{Verzögerung,s}} = \max(t_{\text{Verzögerung,s2}} + s, g_r - \chi_{\text{vorherig}})$. Das führt zu dieser Abfahrtsaktion:

$$(\text{abfahrt}, \sigma, \chi_{\text{vorherig}} + t_{\text{Verzögerung,s}}, g_r), \{r\})$$

Falls es eine Aktion nach dem Warten gibt, wird diese mit

$$p_{\text{nächste}} = (\text{abfahrt}, \sigma, \chi_{\text{nächste}}, \Psi_{\text{nächste}})$$

bezeichnet. Beim Einfügen der Zielhaltestelle gibt es wieder zwei Fälle:

1. $p_{\text{nächste}}$ existiert nicht $\vee$ $\sigma = d_r$:

   In diesem Fall muss der Bus nicht zurück zur Haltestelle σ fahren, weil es keine weiteren Aktionen in seinem Aktionsplan gibt oder weil die nächste Abfahrtsaktion an der Haltestelle d_r stattfindet. Für die Fahrt von o_r nach d_r benötigt der Bus eine Zeit von $v \cdot \delta(o_r, d_r)$, weshalb eine Verzögerung von $t_{\text{Verzögerung,z}} = t_{\text{Verzögerung,s}} + v \cdot \delta(o_r, d_r)$ festgehalten wird. Daraus folgt diese Ankunftsaktion:

   $$(\text{ankunft}, \sigma, \chi_{\text{vorherig}} + t_{\text{Verzögerung,z}}, \text{unbenutzt})$$

2. $p_{\text{nächste}}$ existiert $\wedge$ $\sigma \neq d_r$:

   Im Fall, dass der Bus zurück zur Haltestelle σ fahren muss, müssen ähnlich wie bei der Starthaltestelle drei Aktionen eingefügt werden. Zunächst muss die Ankunftsaktion bei der Zielhaltestelle d_r eingefügt werden. Dafür fällt analog zum anderen

Fall eine Verzögerung von $t_{\text{Verzögerung,z1}} = t_{\text{Verzögerung,s}} + v \cdot \delta(o_r, d_r)$ an. Eingefügt wird anschließend diese Aktion:

$$(\text{ankunft}, d_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung,z1}}, \text{unbenutzt})$$

Danach soll der Bus so schnell wie möglich wieder zurück fahren. Daher wird die Abfahrtsaktion gleich nach der minimalen Standzeit s eingefügt. Die Verzögerung berechnet sich also mit $t_{\text{Verzögerung,z2}} = t_{\text{Verzögerung,z1}} + s$. Konkret lautet die Abfahrtsaktion, die dadurch eingefügt wird:

$$(\text{abfahrt}, d_r, \chi_{\text{vorherig}} + t_{\text{Verzögerung,z2}}, \emptyset)$$

Zuletzt muss noch die Ankunftsaktion bei der Haltestelle σ eingefügt werden. Zur bisherigen Verzögerung wird also noch die Fahrzeit hinzuaddiert, wodurch man auf eine finale Verzögerung von $t_{\text{Verzögerung,z}} = t_{\text{Verzögerung,z2}} + v \cdot \delta(d_r, \sigma)$ kommt. Die Ankunftsaktion die folglich eingeplant wird, ist:

$$(\text{ankunft}, \sigma, \chi_{\text{vorherig}} + t_{\text{Verzögerung,z}}, \text{unbenutzt})$$

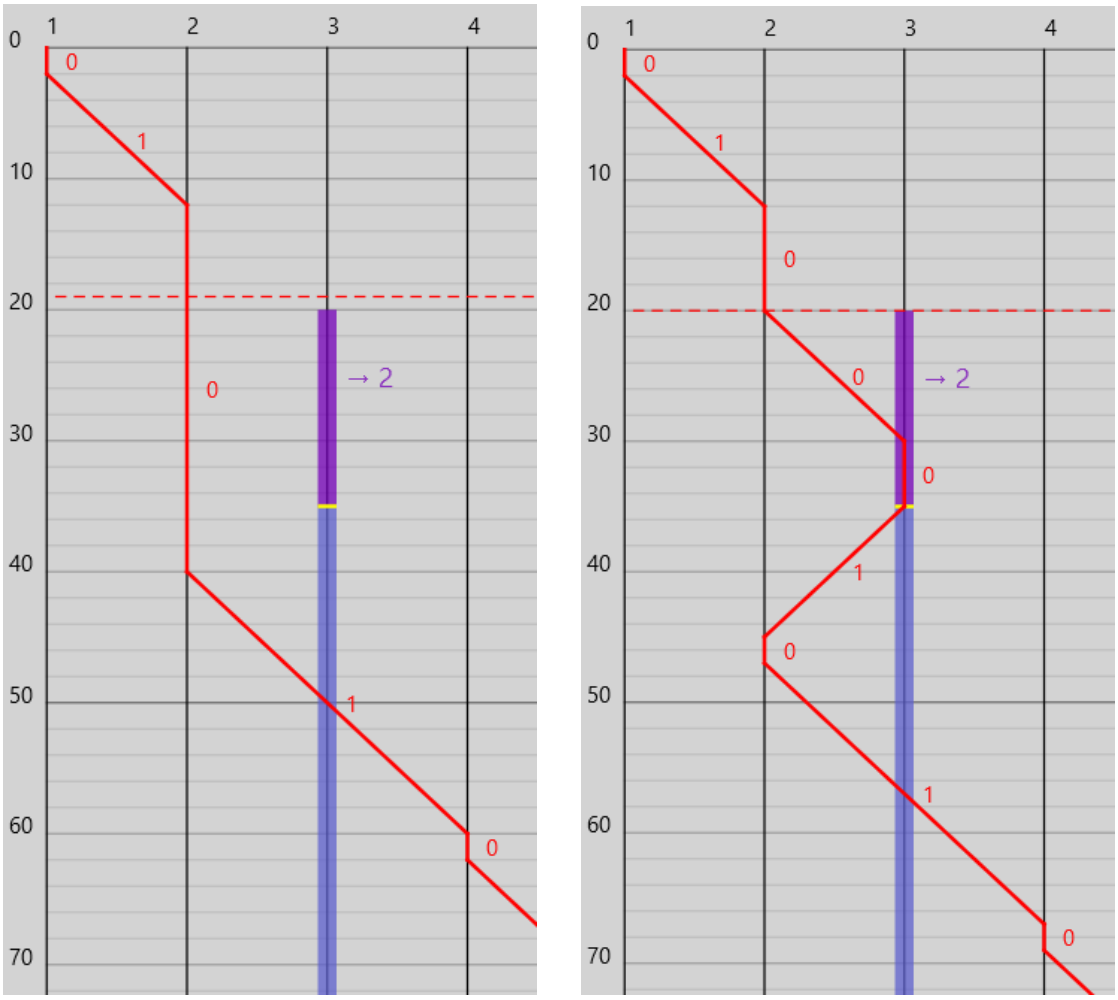
Wenn es noch weitere Aktionen gibt, nachdem der Bus die neue Fahrtanfrage bedient hat, müssen diese unter Umständen verzögert werden. Das passiert auf die gleiche Art und Weise wie im Schritt 2a mit dem Algorithmus 8.

Um das Einfügen der neuen Fahrtanfrage nochmal zu veranschaulichen, wird im Folgenden ein Beispiel gezeigt. Es kann in Abbildung 3.9 betrachtet werden. Angenommen ein Bus (in der Abbildung die rote, durchgezogene Linie) wartet an der Haltestelle $\sigma = 2$ vom Zeitpunkt $\chi_{\text{vorherig}} = 12$ bis zum Zeitpunkt $\chi_{\text{nächste}} = 40$ und hat danach noch weitere Aktionen. Nun wird zum Zeitpunkt $t = 20$ eine neue Fahrtanfrage r mit $o_r = 3$, $d_r = 2$, $a_r = 20$ und $g_r = 35$ gestellt. Die Starthaltestelle o_r der neuen Anfrage ist nicht die selbe Haltestelle, wie die, an der der Bus wartet. Deshalb handelt es sich beim Einfügen des Halts bei der Starthaltestelle um Fall 2. Der Bus muss zu der Starthaltestelle erst hinfahren, wobei er frühstens zum aktuellen Zeitpunkt t starten kann. An der Starthaltestelle angekommen, muss der Bus bis zum gewünschten Abfahrtszeitpunkt g_r warten, um den neuen Passagier mitzunehmen. Anschließend wird die Zielhaltestelle entsprechend Fall 1 eingefügt. Nach dem Einfügen des Halts an der Zielhaltestelle $d_r = \sigma = 2$ hat der Bus eine Verzögerung von $t_{\text{Verzögerung,z}} = 33$. Diese kann, dadurch dass der Bus an der Haltestelle 28 Zeitschritte wartet, auf $33 - 28 = 5$ Zeitschritte reduziert werden. Folglich müssen alle nachfolgenden Aktionen um höchstens 5 Zeitschritte verzögert werden.

Schritt 3: Wartezeit der neuen Anfrage bestimmten

Nachdem für die neue Fahrtanfrage r für jeden Bus $b \in B$ ein neuer Aktionsplan erstellt wurde, wird für jeden Aktionsplan jeweils die Wartezeit der neuen Fahrtanfrage bestimmt. Das geht im Vergleich zum vorherigen Schritt einfach, weshalb auf einen Algorithmus verzichtet wird und die Wartezeit lediglich mathematisch definiert wird.

In jedem für die Anfrage r neu erstellten Aktionsplan P_{neu} gibt es eine Abfahrtsaktion $p = (\text{abfahrt}, \sigma, \chi, \Psi)$, für die gilt, dass $r \in \Psi$ ist. Die Wartezeit für einen neuen



(a) Vor dem Einfügen von r ($t = 19$)

(b) Nach dem Einfügen von r ($t = 20$)

Abb. 3.9.: Einfügen einer neuen Anfrage r mit $o_r = 3$, $d_r = 2$, $a_r = 20$, $g_r = 35$, während der rote Bus wartet.

Aktionsplan ist definiert als die Differenz zwischem dem geplante Zeitpunkt χ dieser Abfahrtsaktion und der gewünschten Abfahrtszeit g_r:

$$t_{\text{warten}} = \chi - g_r$$

Schritt 4: Neue Aktionspläne nach Wartezeit sortieren

Anschließend werden die Tripel, bestehend aus dem Bus b, dem zu diesem Bus gehörenden neuen Aktionsplan P_{neu} und die dabei aufkommende Wartezeit t_{warten}, in einer Liste C gesammelt und aufsteigend nach der Wartezeit sortiert.

Schritt 5: Ersten machbaren Aktionsplan umsetzen

Schließlich wird die Liste C der Tripel aus Bus, neuem Aktionsplan und der Wartezeit für die neue Anfrage (beginnend bei dem Tripel mit der kleinsten Wartezeit) solange durchsucht, bis ein realisierbarer Aktionsplan gefunden wird, der keine Kapazitätsgrenze überschreitet und der keine zeitlichen Bedingungen verletzt. Wenn einer gefunden wird, wird dieser umgesetzt, d.h. der Aktionsplan des Busses wird entsprechend geändert. Wenn kein realisierbarer Aktionsplan gefunden wird, gilt die Fahrtanfrage r als vorläufig abgelehnt, da kein weiteres Mal versucht wird, sie in einen Aktionsplan zu integrieren. (Gemäß der Definition der Ausgangssituation in Abschnitt 3.1.1 durchläuft sie trotzdem den Zustand wartend und kommt erst danach in den Zustand abgelehnt.)

Ob ein Aktionsplan zeitliche Bedingungen verletzt, ist folgendermaßen definiert. Sei $R_{P_{\text{neu}}}$ die Menge aller Fahrtanfragen, die in Aktionen von einem Aktionsplan P_{neu} enthalten sind:

$$R_{P_{\text{neu}}} = \{r \in R \mid \exists p = (\gamma, \sigma, \chi, \Psi) \in P_{\text{neu}} \colon r \in \Psi\}$$

Dann muss gelten, dass für alle diese Anfragen bei der entsprechenden Einstiegsaktion p_{s} die gewünschte Abfahrtszeit und die maximale Wartezeit respektiert werden und an der entsprechenden Ausstiegsstation p_{z} die späteste Ankunftszeit eingehalten werden:

$$\begin{aligned} &\forall r \in R_{P_{\text{neu}}} \colon \\ \exists p_{\text{s}} = (\gamma_{\text{s}}, \sigma_{\text{s}}, \chi_{\text{s}}, \Psi_{\text{s}}) \in P_{\text{neu}} \colon &\gamma_{\text{s}} = \text{abfahrt} \land o_r = \sigma_{\text{s}} \land r \in \Psi_{\text{s}} \land g_r \leq \chi_{\text{s}} \leq w_r \\ &\land \exists p_{\text{z}} = (\gamma_{\text{z}}, \sigma_{\text{z}}, \chi_{\text{z}}, \Psi_{\text{z}}) \in P_{\text{neu}} \colon d_r = \sigma_{\text{z}} \land \chi_{\text{s}} < \chi_{\text{z}} \leq l_r \end{aligned}$$

Darüber hinaus muss für einen neuen Aktionsplan P_{neu} von einem Bus b die Kapazitätsbedingung nach allen zukünftigen Aktionen überprüft werden, d.h. es muss sichergestellt werden, dass niemals mehr als c Fahrtanfragen mit dem Bus b transportiert werden. Weil es mathematisch einfacher zu beschreiben geht und für die vergangenen Aktionen immer erfüllt ist, wird die Kapazitätsbedingung nachfolgend für alle Aktionen des Aktionsplan überprüft. (Der entsprechende Algorithmus macht das natürlich nur für die zukünftigen Aktionen.) Sei R_i die Menge der Fahrtanfragen, die ein Bus laut dem Aktionsplan P_{neu} nach der i-ten Aktion $p^{(i)} = (\gamma^{(i)}, \sigma^{(i)}, \chi^{(i)}, \Psi^{(i)})$ des Plans transportiert.

Es gilt

$$R_1 = \emptyset$$
$$R_i = \begin{cases} R_{i-1} \cup \Psi^{(i)} & \text{falls } \gamma^{(i)} = \text{abfahrt} \\ \{r \in R_{i-1} | r \neq \sigma^{(i)}\} & \text{sonst} \end{cases} \quad \forall 2 \leq i \leq |P_{\text{neu}}|$$

Damit die Kapazitätsbedingung erfüllt ist, muss folglich gelten, dass

$$\forall 1 \leq i \leq |P_{\text{neu}}| \colon |R_i| \leq c$$

3.4. Implementierung

Nachdem das untersuchte Problem nun sehr detailliert beschrieben wurde, wird sich im Folgenden kurz der Implementierung der Simulation gewidmet.

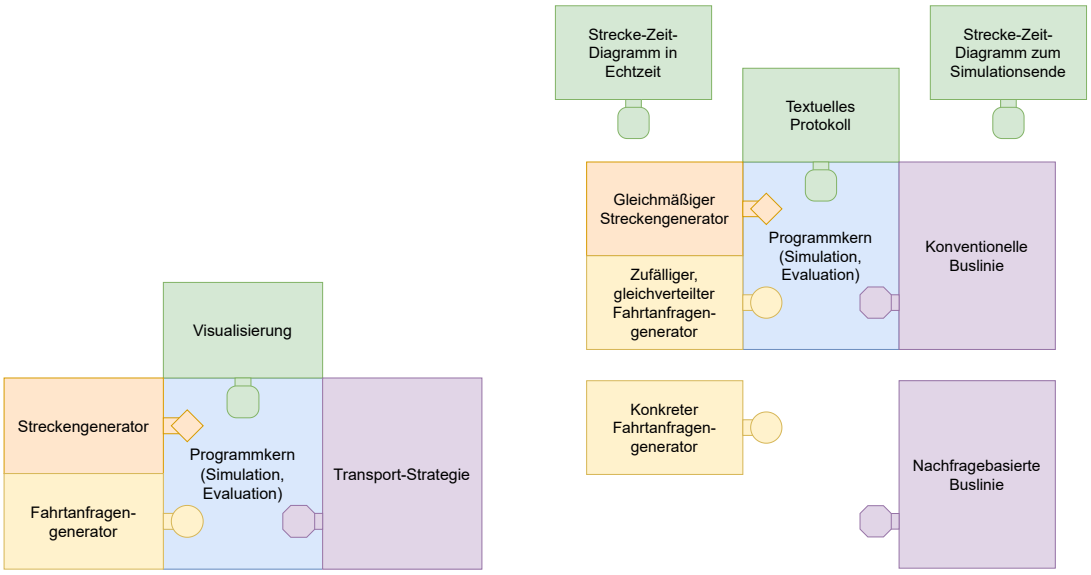
Die Implementierung erfolgte in Java. Dazu wurde die Java Version 20 verwendet. Um sicherzustellen, dass die Simulation korrekt abläuft, wurde die Simulation in insgesamt fünf Module aufgespalten, die in Abbildung 3.10a zu sehen sind. Dafür wurde jedes Modul einzeln zusammen mit dem Programmkern auf Fehler überprüft. Ein weiterer Vorteil der modularen Programmstruktur ist die einfache Wiederverwendbarkeit und Austauschbarkeit, falls zum Beispiel andere Transport-Strategien untersucht werden sollen. So gibt es für bestimmte Modultypen mehrere konkrete Module. Abbildung 3.10b zeigt einige konkrete Versionen des jeweiligen Moduls. Jedes Modul außer das Visualisierungsmodul muss vorhanden sein, damit ein Szenario simuliert werden kann. In den folgenden Abschnitten werden die einzelnen Module näher beschrieben.

3.4.1. Programmkern

Der Programmkern dient dazu, die Simulation durchzuführen und anschließend zu evaluieren. Außerdem kontrolliert der Programmkern, dass alle Eingaben von den anderen Modulen korrekt sind. Dafür lässt er sich zunächst von den beiden Generator-Modulen die Strecke und anschließend alle Fahrtanfragen generieren. Nachdem die Strecke und die Fahrtanfragen auf Fehler überprüft wurden, führt er die Simulation schrittweise aus. In jedem Simulationsschritt gibt er dem Transport-Strategie-Modul die Möglichkeit, die zukünftigen Aktionspläne der Busse zu ändern, wobei er prüft, dass die Transport-Strategie gültige Aktionspläne erzeugt. Am Ende eines Simulationsschrittes informiert der Programmkern das Visualisierungsmodul über alles, was in diesem Simulationsschritt passiert ist, falls es ein solches Modul gibt. Nachdem die Simulation abgeschlossen ist, berechnet er die Statistiken, die zur Evaluation verwendet werden.

3.4.2. Streckengenerator

Der Streckengenerator ist das kleinste Modul. Mit ihm kann eine neue Strecke, also ein Tupel an Haltestellen inklusive Fahrzeiten erzeugt werden. Der Streckengenerator bestimmt dabei die Anzahl und die Fahrzeiten zwischen den Haltestellen.



(a) Allgemeine Modultypen

(b) Auswahl an verfügbaren Modulen

Abb. 3.10.: Die modulare Programmstruktur, die zur Implementierung verwendet wurde. Jedes Modul außer dem Programmkern kann ausgetauscht werden.

3.4.3. Fahrtanfragengenerator

Der Fahrtanfragengenerator erzeugt zum Beginn der Simulation alle Fahrtanfragen. Je nach dem, welcher Generator verwendet wird, können die Fahrtanfragen konkret auf Basis einer Datei oder mit unterschiedlichen Wahrscheinlichkeitsverteilungen (z.B. gleichverteilt) erstellt werden.

3.4.4. Transport-Strategie

Das Modul “Transport-Strategie” kann in jedem Simulationsschritt die zukünftigen Aktionspläne der Busse ändern. Hierzu wird dem Modul eine schreibgeschützte Ansicht auf den aktuellen Simulationszustand übergeben. Durch die Schnittstelle zwischen dem Programmkern-Modul und dem Transport-Strategie-Modul ist sichergestellt, dass die Transport-Strategie ausschließlich die Informationen erhält, die zum aktuellen Zeitpunkt bekannt sind und insbesondere kein Zugriff auf noch unbekannte Abfragen möglich ist. Da eine Änderung eines Aktionsplans anschließend von dem Programmkern auf Zulässigkeit überprüft wird, kann eine Transport-Strategie zum Beispiel keine Aktionspläne erstellen, bei denen ein Bus in kürzerer Zeit als der Fahrzeit zu einer anderen Haltestelle gelangen kann.

Für diese Arbeit wurde das Modul “Transport-Strategie” auf zwei unterschiedliche Weisen implementiert: Einmal wurde eine konventionelle Buslinie beschrieben und im zweiten Fall die oben definierte anfragebasierte Buslinie.

3.4.5. Visualisierung

Das Visualisierungsmodul ermöglicht das Anzeigen des gesamten Simulationsablaufs. Dazu wurde zum einen eine Implementierung umgesetzt, die alle Aktionen der Simulation in einem textuellen Protokoll speichert. Darüber hinaus gibt es eine Implementierung, welche die zuvor vorgestellten Strecke-Zeit-Diagramme automatisch für einen Simulationsablauf visualisieren kann. Die Strecke-Zeit-Diagramme können auch, während die Simulation läuft, erstellt werden, wobei dann nicht nur die bereits ausgeführten Aktionen dargestellt werden sondern auch die bereits geplanten zukünftigen sichtbar sind. Ein Screenshot aus dem Visualisierungsprogramm findet sich in Abbildung 3.11. Dazu wird die Standard Benutzeroberflächen-Bibliothek JavaFX verwendet.

Der vollständige Quellcode kann unter diesem Link[1] eingesehen werden.

[1] https://gitlab2.informatik.uni-wuerzburg.de/s411716/line-based-minibus-simulation

Abb. 3.11.: Das Visualisierungsprogramm, das den Simulationszustand während der Simulation in Form eines Strecke-Zeit-Diagramms ausgeben kann. Entlang der x-Achse sind die Haltestellen durch Linien eingezeichnet, entlang der y-Achse die Zeit in Simulationsschritten. Die rot gestrichelte, horizontale Linie stellt den aktuellen Zeitpunkt dar. Die bunten Linien entsprechen den Fahrten der Busse, wobei die Fahrten unterhalb der gestrichelten Linie nur geplant und noch nicht durchgeführt worden sind. Wenn man mit der Maus auf einer Fahrtlinie schwebt, kann man in einem Tooltip Informationen über die geplanten, bzw. tatsächlich transportierten Fahrgäste einsehen.

4. Ergebnisse

In diesem Kapitel sollen die Ergebnisse verschiedener Untersuchungen beschrieben werden. Im nachfolgenden Kapitel werden diese diskutiert.

4.1. Wahl der festen Parameter

Einige Parameter wurden für alle Simulationen gleich gewählt. Diese sollen im Folgenden beschrieben werden.

Zuerst wurde festgelegt, dass ein Zeitschritt in den Simulationen einer Zehntelminute, also sechs Sekunden, entspricht. Diese Entscheidung wurde aus dem Grund getroffen, dass dadurch einerseits eine einfache Umrechnung in Minuten ermöglicht wird, die auch im Kopf erfolgen kann, und andererseits die Möglichkeit besteht, Bruchteile von Minuten darzustellen.

Anschließend wurden die Fahr- und Standzeiten der Busse bestimmt. Um einigermaßen realisitische Zeiten zu verwenden, wurden auf zwei Fahrten mit konventionellen Linienbussen im Stadtgebiet Würzburg die Fahr- und Standzeiten gestoppt. Beide Messungen erfolgten an einem Werktag zwischen 15 und 16 Uhr, wobei es kein außergewöhnlich hohes Verkehrsaufkommen gab. Die gemessenen Standzeiten zusammen mit den jeweils ein- und ausgestiegenen Fahrgästen sind in Tabelle 4.1 aufgeführt. Bei einer Haltestelle kam ein Bezahlvorgang beim Busfahrer vor, wodurch die Standzeit deutlich erhöht wurde. Ohne diese Haltestelle beträgt die durchschnittliche Standzeit 16,91 Sekunden. Da bei Minibussen von einem geringeren Fahrgastwechsel auszugehen ist, wurde die minimale Standzeit s der Simulation auf den nächstkleineren darstellbaren Wert von 12 Sekunden ($s = 2$) festgelegt.

Für die Fahrzeiten sind die gemessenen Werte in Tabelle 4.2 abgebildet. Dass es mehr Fahrzeiten als Standzeiten gibt, ist damit zu begründen, dass die beiden Busse nicht an jeder Haltestelle gehalten haben. Im Durchschnitt ergab sich eine Fahrzeit von 55,81 Sekunden, wobei diese auf Grund von Ampeln und Kreuzungen stark schwankt. Für die Simulation wird die nächst größere Fahrzeit von 60 Sekunden, $v = 10$ gewählt.

Für die zu untersuchende Route wurde festgelegt, dass diese 10 Haltestellen ($|H| = 10$) haben soll. Dabei werden Minibusse mit einer maximalen Transport-Kapazität von vier Passagieren ($c = 4$) eingesetzt.

Auch für die Fahrtanfragegeneratoren sind einige Parameter unabhängig von der konkreten Simulation fixiert worden. So wird der Zeitpunkt a_r, zu dem eine Anfrage r gestellt wird, für jede Anfrage gleichverteilt über den Simulationszeitraum gewählt. Der gewünschte Abfahrtszeitpunkt g_r einer Anfrage ist zwischen 0 (inklusive) und 250 (exklusive) Ticks später, wobei auch diese Differenz gleichverteilt erzeugt wird. Die maximale Zeit w_r, bis zu der eine Anfrage wartet, ist 50 Zeitschritte nach der gewünschten

Tab. 4.1.: Gemessene Standzeiten in Sekunden zweier Stadtbusse in Würzburg

Haltestelle	Standzeit	Anzahl an ein- und aussteigenden Fahrgästen
1	19	4
2	16	5
3	19	6
4	21	8
5	17	5
6	24	6
7	46	3 (mit Bezahlvorgang)
8	9	1
9	14	3
10	17	5
11	16	4
12	14	3
Durchschnitt	19,33	
	16,91	(ohne Haltestelle 7)

Tab. 4.2.: Gemessene Fahrzeiten in Sekunden zweier Stadtbusse in Würzburg

Strecke	Fahrzeit	Meter (laut Google Maps)
1	77	450
2	55	400
3	67	400
4	49	240
5	52	400
6	33	240
7	45	300
8	76	300
9	63	300
10	36	300
11	21	240
12	29	400
13	88	240
14	66	400
15	51	450
16	85	350
Durchschnitt	55,81	

Abfahrtszeit. Es gilt also für jede Anfrage r $g_r + 50 = w_r$. Die späteste Ankunftszeit l_r für jede Anfrage r ist proportional zur gewünschten Streckenlänge fixiert auf $l_r = g_r + 4v \cdot \delta(o_r, d_r)$. Damit entspricht die maximal akzeptierte Fahrzeit der vierfachen Fahrzeit, wenn die gewünschte Strecke ohne Zwischenhalt befahren wird.

Als weitere Simulationsparameter wurde eine Simulationslänge von $t_{\max} = 1000$ Zeitschritten gewählt, was 100 Minuten entspricht. Damit die Ergebnisse nicht von bestimmten Verteilungen der Fahrtanfragen abhängen, wurde jede Simulation 100 Mal durchgeführt. Die Ergebnisse in den folgenden Abschnitten sind also Durchschnittswerte über die 100 Simulationsläufe.

Zuletzt wurde festgelegt, dass alle Busse an der Haltestelle h_1 starten.

4.2. Variable Parameter

Während die zuvor beschriebenen Parameter bei allen Untersuchungen nicht variiert werden, werden nun diejenigen Parameter vorgestellt, für die unterschiedliche Werte untersucht werden sollen.

Da diese Arbeit hauptsächlich untersuchen möchte, in welchen Nachfragesituationen welche Transport-Strategie zu besseren Ergebnissen gelangt, ist der wichtigste variable Parameter die Anzahl der Anfragen $|R|$, die in dem Zeitraum von 1000 Ticks bzw. 100 Minuten gestellt werden. Für diesen Parameter werden alle Werte zwischen 10 Anfragen in 100 Minuten (6 Anfragen pro Stunde) und 300 Anfragen in 100 Minuten (180 Anfragen pro Stunde) mit einer Schrittweite von 10 Anfragen getestet.

Weiterhin soll auch die Anzahl der Minibusse geändert werden. Hierfür sollen Szenarien mit ein bis drei zum Einsatz kommenden Bussen untersucht werden.

Im hauptsächlich betrachteten Fall werden die Start- und die Zielhaltestelle o_r, d_r gleichverteilt generiert. Schließlich wurde noch eine andere Art der Verteilung dieser untersucht. In vielen Fällen ist es aber so, dass es auf einer Buslinie wichtige Haltestellen gibt, an denen auf Grund von Umsteigemöglichkeiten ein großer Teil der Passagiere ein- und aussteigen. In dieser Arbeit wird allerdings nur der Fall betrachtet, in denen es eine einzige wichtige Station (h_1) am Anfang der Linie gibt. Dabei soll die Auswirkungen untersucht werden, wenn 90% der Anfragen ein Start- oder Ziel an der wichtigen Haltestelle haben.

Zusammengefasst werden die folgenden drei Parameter variiert:

1. Anzahl der Anfragen
2. Anzahl der Busse
3. Verteilung der Anfragen (Räumlich gleichverteilt bzw. eine wichtige Haltestelle)

4.3. Räumlich gleich verteilte Fahrtanfragen

4.3.1. Mitnahmerate

Für räumlich gleichverteilte Fahrtanfragen finden sich die Ergebnisse für die Mitnahmerate q_{m} in Abbildung 4.1. Generell gilt, dass je mehr Busse eingesetzt werden, desto höher ist die Mitnahmerate. Außerdem sinkt die Mitnahmerate bei beiden Strategien ab einer bestimmten Menge an Fahrtanfragen, je mehr Fahrtanfragen gestellt werden. Weiter ist festzuhalten, dass die vorgestellte anfragebasierte Transport-Strategie für eine kleine Anzahl an Fahrtanfragen eine höhere Mitnahmerate aufweist als die konventionell taktbasierte Transport-Strategie. Einzige Ausnahme davon sind die Werte für zehn Fahrtanfragen bei fünf Bussen. In diesem Fall liegt die Mitnahmerate der taktbasierten Strategie leicht über derjenigen der anfragebasierten Strategie. Die Anzahl der Fahrtanfragen, ab denen eine taktbasierte Strategie bessere Werte bei der Mitnahmerate erzeugt, sind 150 Fahrtanfragen, wenn ein Bus eingesetzt wird, 130 Fahrtanfragen bei drei Bussen und 100 Anfragen, wenn fünf Busse eingesetzt werden. Auffällig ist, dass es bei der anfragebasierten Strategie ein Maximum gibt, welches zumindest im Fall von drei bzw. fünf Bussen nicht bei der geringsten Anzahl an Fahrtanfragen liegt. Zuletzt kann man erkennen, dass die Mitnahmerate bei der konventionellen taktbasierten Strategie ab einer bestimmten Anzahl an Fahrtanfragen merklich schneller sinkt.

4.3.2. Wartezeit pro Anfrage

In den Diagrammen zur durchschnittlichen Wartezeit pro Fahrtanfrage $\bar{t}_{\mathrm{w}}$ in Abbildung 4.2 sind neben Werten für die taktbasierte und die anfragebasierte Transport-Strategie weitere Werte für die anfragebasierte Strategie zu sehen. Bei diesen handelt es sich um die durchschnittliche Wartezeit von allen akzeptierten Fahrtanfragen. Während gemäß der Problemdefinition in Abschnitt 3.1.1 eine Anfrage erst dann als abgelehnt gilt, wenn die maximale Wartezeit überschritten wird, kann bei der anfragebasierten Strategie direkt nach dem Stellen einer neuen Fahrtanfrage angegeben werden, ob diese transportiert werden kann. Daher entfällt für einen Passagier, dessen Fahrtanfrage abgelehnt wird, auch die Wartezeit. Dadurch kommt die graue Kurve im Diagramm zustande.

Bei den Ergebnissen zur Wartezeit erkennt man zunächst, dass sämtliche Werte für Wartezeiten geringer werden, je mehr Busse im Einsatz sind. Grob gesagt gilt, dass je mehr Fahrtanfragen gestellt werden, desto höher wird die durchschnittliche Wartezeit. Eine deutliche Ausnahme hiervon bildet die anfragebasierte Strategie bei weniger als 50 Fahrtanfragen mit fünf Bussen. Bei der konventionellen taktbasierten Strategie tritt dieser Effekt deutlich schwächer auf und ist erst ab einem bestimmten Schwellwert an Fahrtanfragen spürbar. Zuvor ist die durchschnittliche Wartezeit mehr oder weniger konstant. Dieser Schwellwert ist 230 Fahrtanfragen bei einem Bus, 120 Fahrtanfragen bei drei Bussen und 90 Fahrtanfragen bei fünf Bussen. Ab 160, 200 oder 200 Fahrtanfragen bei ein, drei oder fünf eingesetzten Bussen erzielt die anfragebasierte Transport-Strategie schlechtere Wartezeit-Werte als die taktbasierte. Bereinigt von den abgelehnten Fahrtanfragen erzielt die anfragebasierte Strategie immer deutlich kürzere Wartezeiten als die konventionell taktbasierte.

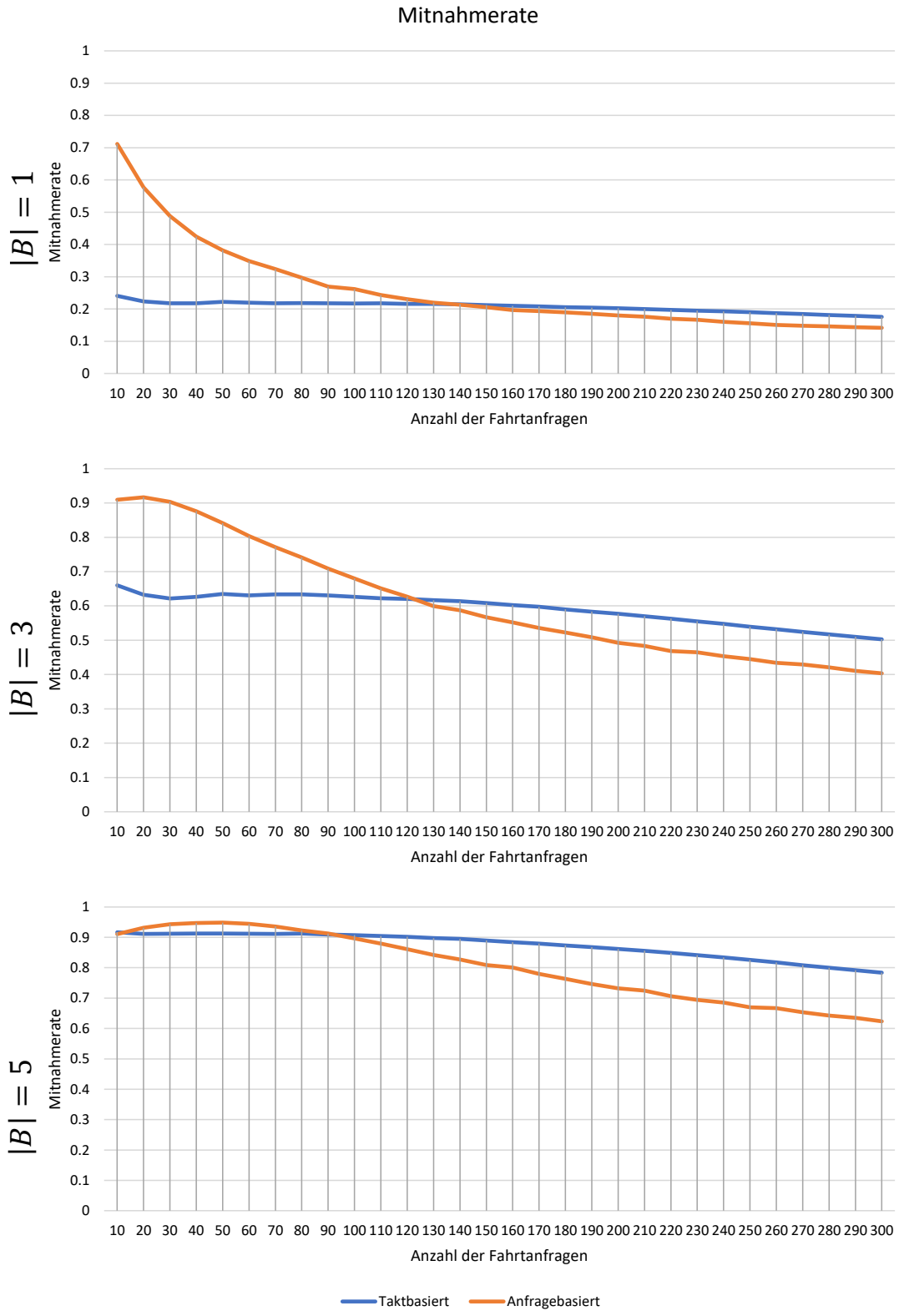


Abb. 4.1.: Mitnahmerate in Abhängigkeit der Anzahl der Anfragen und der Anzahl der Busse

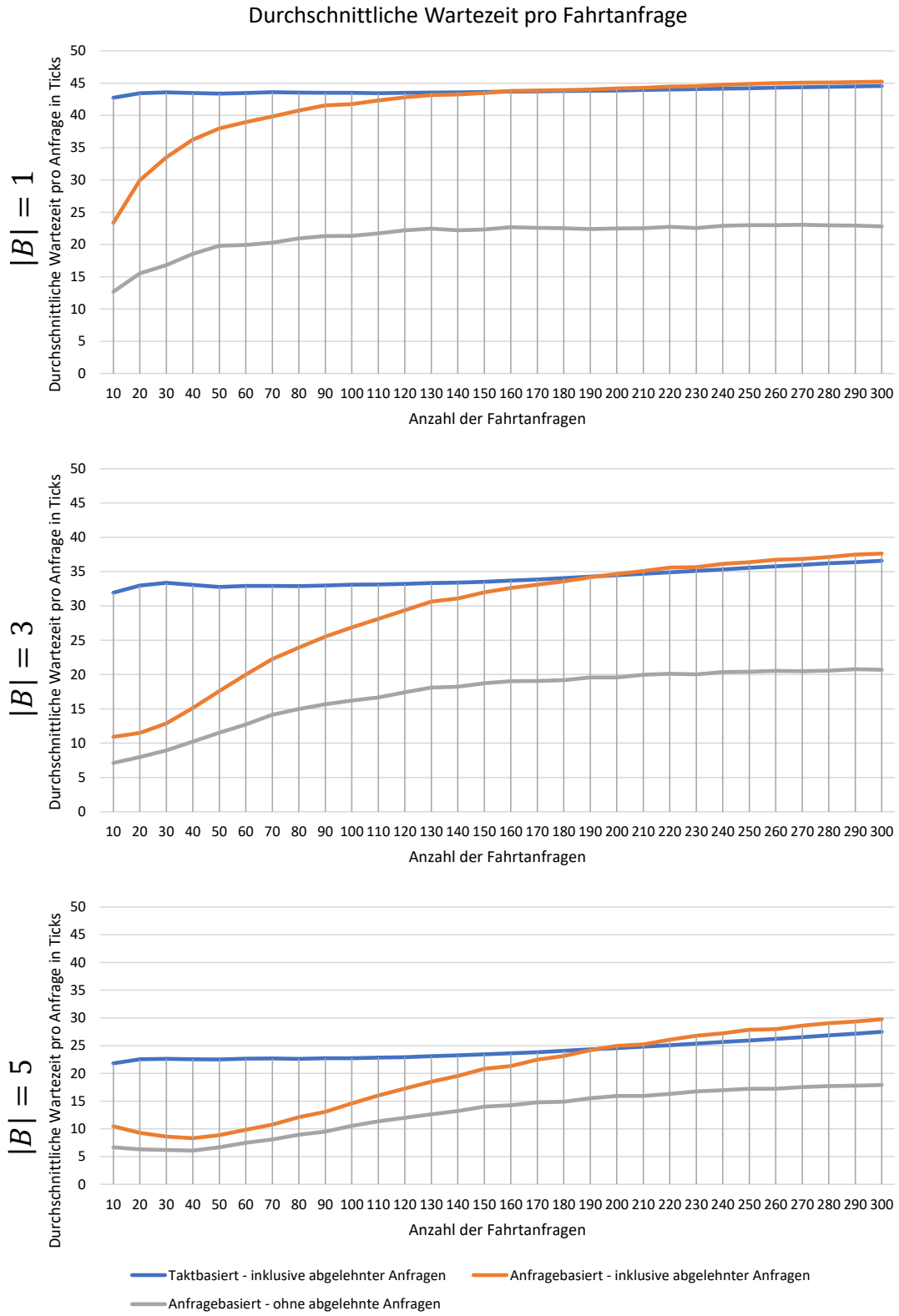


Abb. 4.2.: Wartezeit pro Anfrage in Abhängigkeit der Anzahl der Anfragen und der Anzahl der Busse

4.3.3. Fahrzeit pro Anfrage

Eine Auswertung der Fahrzeit pro Anfrage $\bar{t}_\mathrm{f}$ für das Szenario mit den räumlich gleichverteilten Fahrtanfragen findet sich in Abbildung 4.3. Auffällig ist, dass die Fahrzeit pro Anfrage bei der taktbasierten Strategie fast immer größere Werte annimmt als bei der anfragebasierten Strategie. Die einzigen Ausnahmen sind Nachfrageszenarien mit 10-40 Fahrtanfragen, die von einem Bus bedient werden. Darüber hinaus ist bei der taktbasierten Strategie erkennbar, dass die durchschnittlichen Fahrzeiten pro Anfrage leicht ansteigen, je mehr Fahrtanfragen gestellt werden, wohingegen bei der anfragebasierten Transport-Strategie die durchschnittlichen Fahrzeiten pro Anfrage abnehmen, je mehr Fahrtanfragen gestellt werden.

4.3.4. Transportzeit pro Anfrage

Für das Szenario mit den räumlich gleichverteilten Fahrtanfragen finden sich die Ergebnisse für die Transportzeit $\bar{t}_\mathrm{t}$ in Abbildung 4.4. Erkennbar ist, dass unabhängig, ob ein, drei oder fünf Busse eingesetzt werden, die Transportzeit für die anfragebasierte Strategie geringer ist als die für die taktbasierte Transport-Strategie. Bei der taktbasierten Strategie fällt auf, dass diese geringfügig ansteigt, je mehr Fahrtanfragen gestellt werden. Bei der anfragebasierten Strategie bemerkt man einen leichten Anstieg in der Transportzeit bis zu ca. 140 Fahrtanfragen. Nach diesem Maximum sinken die Werte für die durchschnittliche Transportzeit bei einem anfragebasierten System marginal.

4.3.5. Auslastung

Die Ergebnisse zur Auslastung der Busse q_a für den Fall der räumlich gleichverteilten Fahrtanfragen können in Abbildung 4.5 betrachtet werden. Bei der taktbasierten Strategie sieht man unabhängig von der Anzahl der Busse zunächst einen linearen Anstieg der Busauslastung. Dieser endet bei ein, drei oder fünf Bussen ca. bei 120 Fahrtanfragen. Anschließend flacht die Kurve für die taktbasierte Strategie unterschiedlich stark ab. Wenn ein Bus im Einsatz ist, ergibt sich bei 300 Fahrtanfragen eine Auslastung von 55,5%. Bei drei Bussen beträgt die durchschnittliche Auslastung 58,8% und bei fünf Bussen 55,4%. Für die anfragebasierte Strategie gilt prinzipiell das gleiche. Allerdings unterliegen die Werte einer gewissen Schwankung, weshalb es schwieriger ist, die Anzahl der Fahrtanfragen herauszufinden, ab der das Abflachen auftritt. Für einen Bus tritt dieser Effekt bei ca. 160 Anfragen auf, bei drei Bussen bei 200 Anfragen und bei fünf Bussen ist kein solcher Punkt für die betrachten Anzahlen der Fahrtanfragen zu finden. Festzuhalten ist, dass die Auslastung der anfragebasierten Minibusse für ein, drei bzw. fünf Bussen bei einem 6,1, 6,1 bzw. 7,4 Mal höheren Wert für zehn Fahrtanfragen startet als die Auslastung bei taktbasierten Minibussen. Jedoch steigt die Auslastung der anfragebasierten Minibusse unabhängig von der Anzahl der eingesetzten Bussen langsamer an als die der taktbasierten Busse, weshalb es bei einem eingesetzten Bus ab 120 Fahrtanfragen, bei drei Bussen ab 100 Fahrtanfragen und bei fünf Bussen ab 110 Anfragen zu einer höheren Auslastung der taktbasierten Busse kommt.

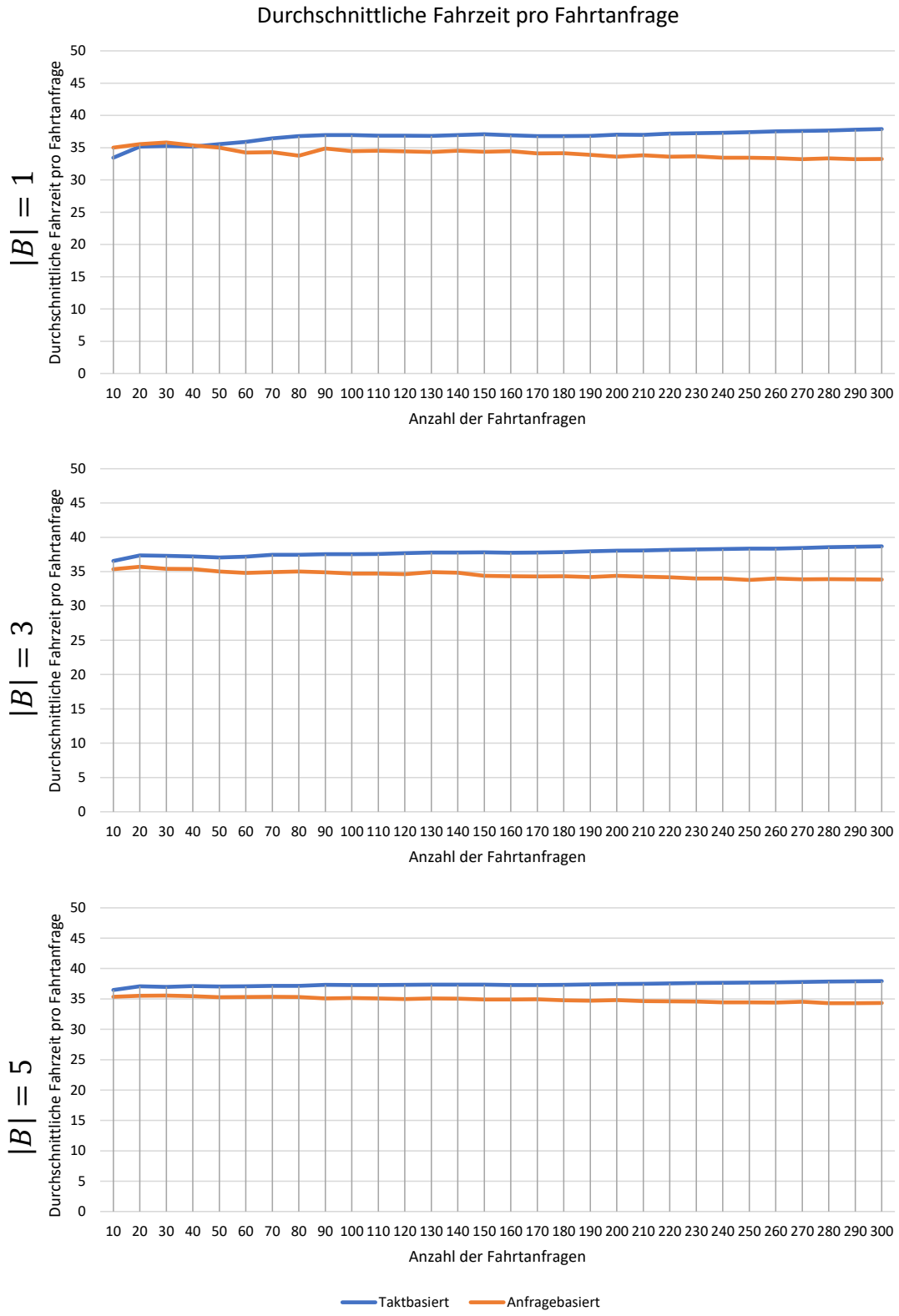


Abb. 4.3.: "Echte" Fahrzeit pro Anfrage in Abhängigkeit der Anzahl der Anfragen und der Anzahl der Busse

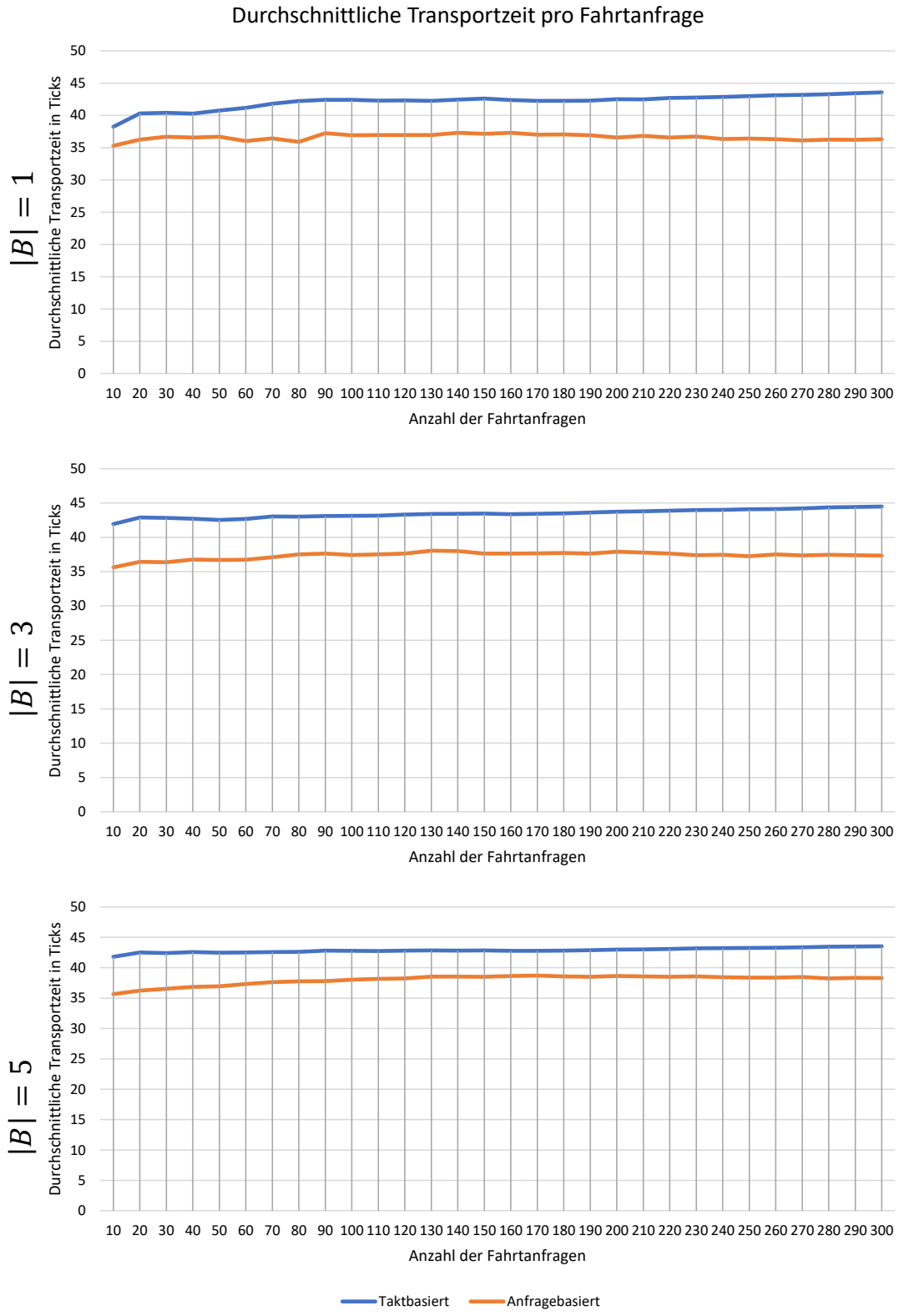


Abb. 4.4.: Transportzeit pro Anfrage in Abhängigkeit der Anzahl der Anfragen und der Anzahl der Busse

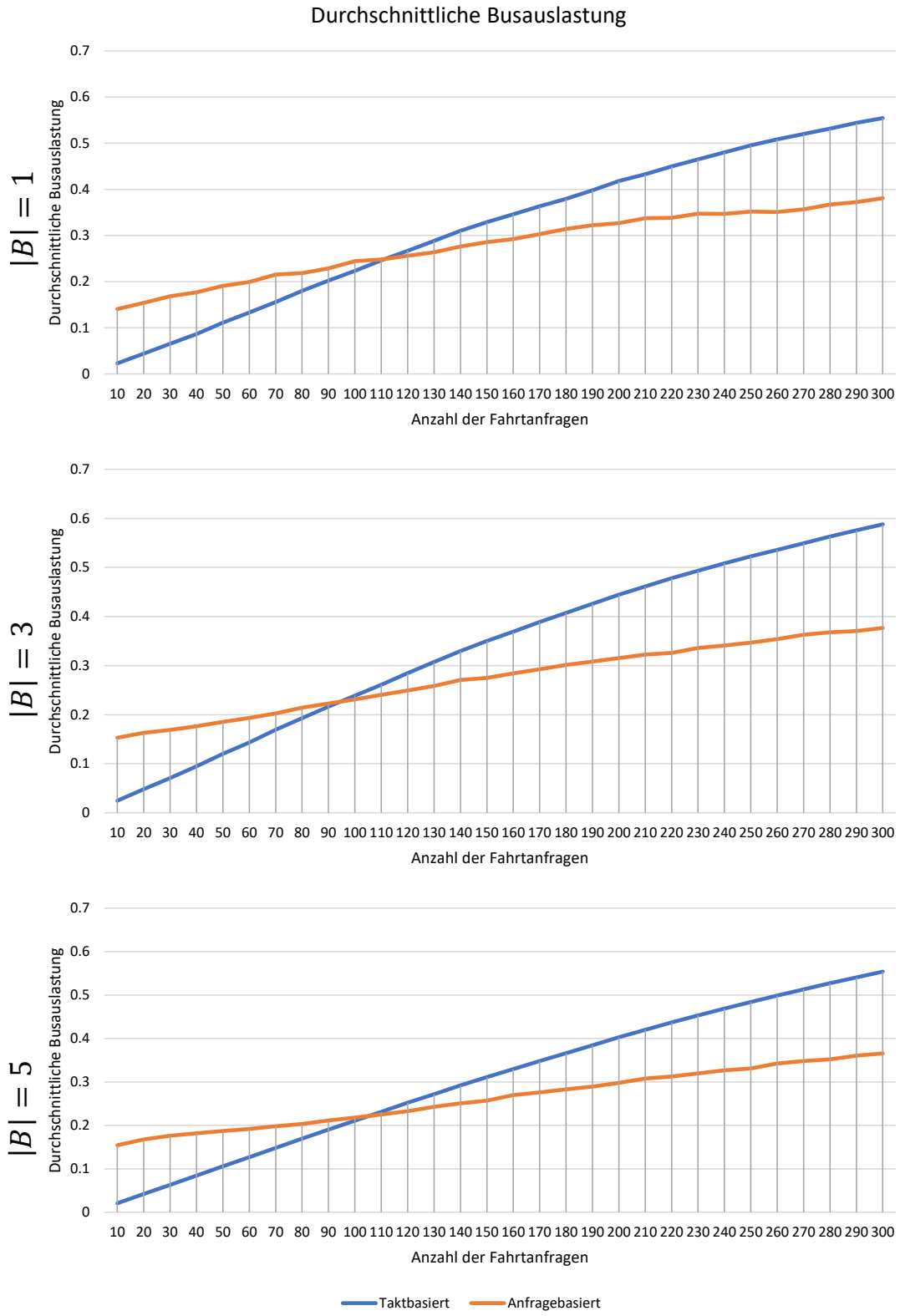


Abb. 4.5.: Busauslastung in Abhängigkeit der Anzahl der Anfragen und der Anzahl der Busse

4.3.6. Busfahrzeit insgesamt

Abbildung 4.6 stellt die Ergebnisse zur gesamten Busfahrzeit t_b für alle räumlich gleichverteilten Nachfragesituationen und beide Transport-Strategien dar. Für die koventionell taktbasierte Strategie sind diese unabhängig von der Anzahl der Fahrtanfragen konstant. Bei der anfragebasierten Transport-Strategie wird die Busfahrzeit durch die Anzahl der Fahrtanfragen beeinflusst. Je mehr Fahrtanfragen kommen, desto mehr fahren die Busse im anfragebasierten System. Beginnend bei einer Gesamtfahrzeit von ca. 416 Ticks bei einem Bus, 507 Zeitschritten bei drei Bussen und 502 Schritten bei fünf Bussen, steigt die insgesamt gefahrene Zeit stark und asymptotisch an. Ab 60 Fahrtanfragen bei einem Bus, 80 Anfragen bei drei Bussen und 130 Anfragen bei fünf Bussen ist die Busfahrzeit von allen Bussen im anfragebasierten System geringfügig höher als diejenige im taktbasierten System.

4.4. Fahrtanfragen mit wichtiger Haltestelle

Tatsächlich unterscheiden sich die Ergebnisse, wenn bei 90% der Anfragen die erste Haltestelle h_1 entweder Start- oder Zielhaltestelle ist, qualitativ nicht von den Ergebnissen im räumlich gleich verteilten Szenario. Daher wurde auf eine erneute Auflistung der Ergebnisse an dieser Stelle verzichtet.

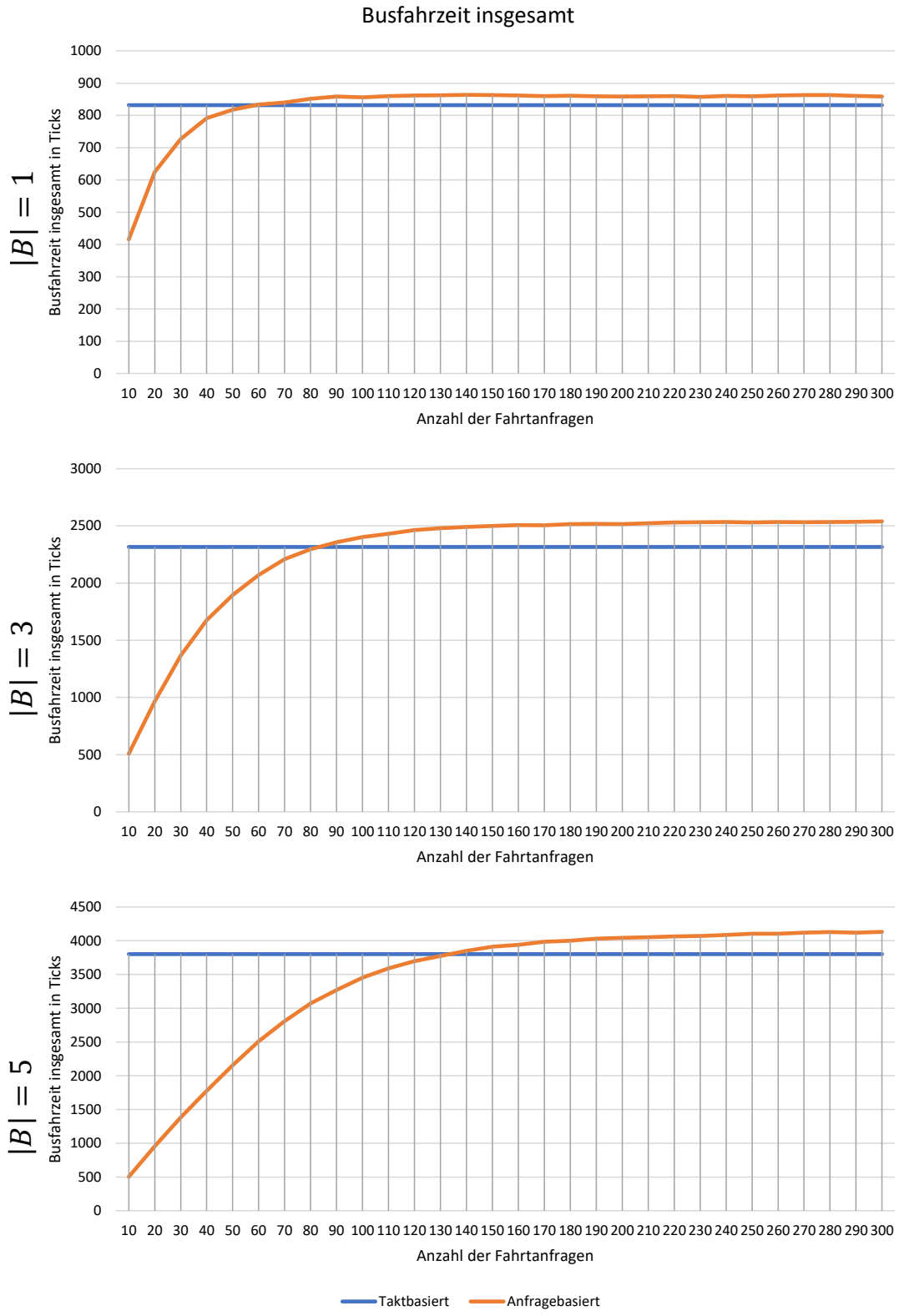


Abb. 4.6.: Busfahrzeit insgesamt in Abhängigkeit der Anzahl der Anfragen und der Anzahl der Busse

5. Diskussion

Im Folgenden werden die zuvor beschriebenen Ergebnisse untersucht. Dazu werden Hypothesen und Begründungen gegeben, warum bestimmtes Verhalten passiert.

5.1. Räumlich gleich verteilte Fahrtanfragen

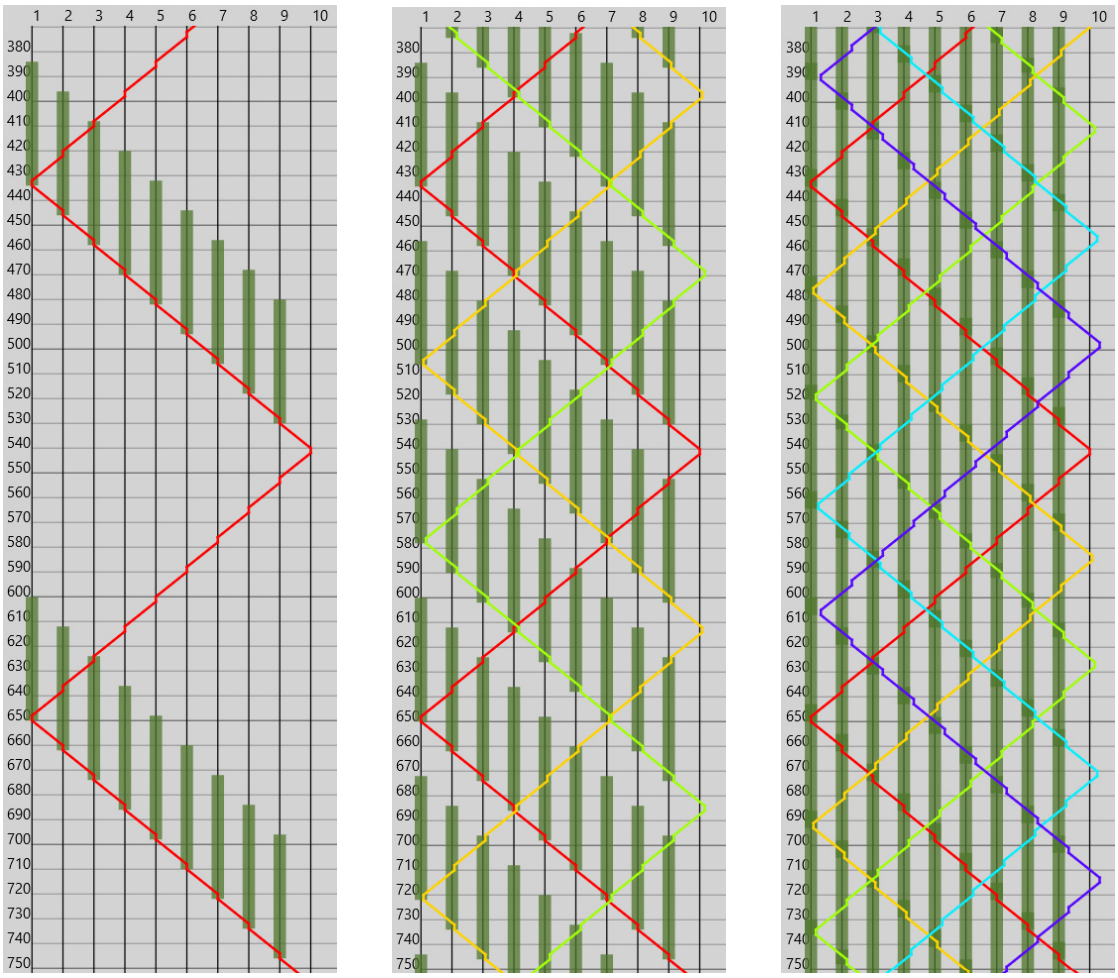
5.1.1. Mitnahmerate

Zunächst sollen die Ergebnisse für die räumlich gleichverteilten Anfragen aus Abbildung 4.1 diskutiert werden.

Dass die Mitnahmerate ansteigt, wenn mehr Busse eingesetzt werden, ist einfach nachzuvollziehen. Einerseits haben mehr Busse mehr Zeit, in der sie Anfragen transportieren können. Andererseits steigt bei einer taktbasierten Strategie immer und bei einer anfragebasierten Strategie meistens auch die zeitliche Abdeckung, zu der Anfragen an Haltestellen abgeholt werden können. Das ist in Abbildung 5.1 für das taktbasierte System zu sehen. In den durch grüne Balken hervorgehobenen Zeiträumen können Anfragen an der jeweiligen Haltestelle in Richtung Haltestellen mit größerem Index mitgenommen werden. Es ist gut erkennbar, dass die grün markierten Flächen mit der Anzahl der Busse ansteigt und bei $|B| = 5$ eine Anfrage zu jedem Zeitpunkt transportiert werden kann. Für die taktbasierte Transport-Strategie lässt sich die Abdeckung q_d, also der Anteil der Zeit, zu dem eine Anfrage mitgenommen werden kann, einfach rechnerisch bestimmen. Dafür benötigt man nur die Umlaufzeit t_u und die für alle Anfragen $r \in R$ einheitliche maximale Wartezeit $w_r - g_r = 50$:

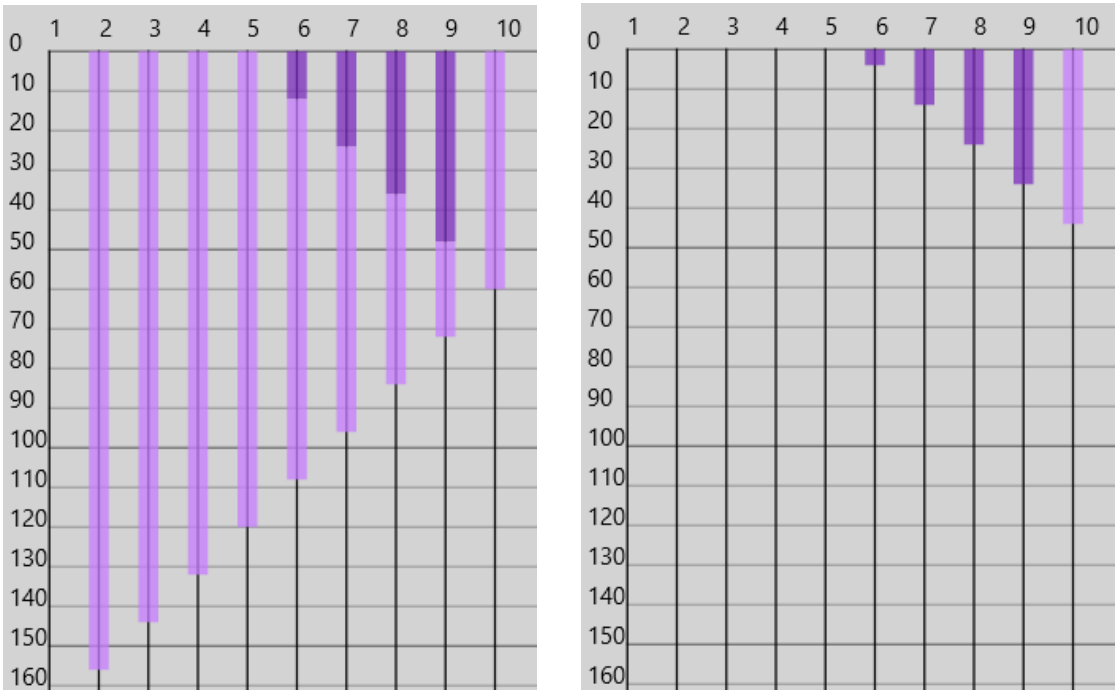
$$q_\mathrm{d} = \frac{|B| \cdot \frac{\sum_{r \in R} w_r - g_r}{|R|}}{t_\mathrm{u}} = \frac{|B| \cdot \frac{\sum_{r \in R} w_r - g_r}{|R|}}{2 \cdot (|H| - 1) \cdot (v + s)} = \frac{|B| \cdot 50}{216}$$

Für $|B| = 1$ ergibt das $q_\mathrm{d} \approx 0{,}23$, für $|B| = 3$ kommt $q_\mathrm{d} \approx 0{,}69$ heraus und für $|B| = 5$ erhält man eine Abdeckung von $q_\mathrm{d} \approx 1{,}16$. Dass die Mitnahmeraten für die taktbasierte Strategie auch bei einer geringen Anfragenzahl von 10 Anfragen unterhalb der Abdeckung bzw. merklich unter 100% liegen, erklärt sich unter anderem damit, dass zu Beginn der Simulation die Busse alle an Haltestelle h_1 starten und es damit unmöglich ist, bestimmte Anfragen zu transportieren. In Abbildung 5.2a sind die Kombinationen aus Starthaltestelle o_r und gewünschter Abfahrtszeit g_r von einer Anfrage r eingezeichnet, die mit taktbasierten Bussen niemals transportiert werden können. Ein weiterer Grund dafür, dass die Mitnahmerate unter der Abdeckung bzw. unter 100% liegt, sind Fahrtanfragen für sehr kurze Strecken. Insbesondere bei Fahrtanfragen r mit $\delta(o_r, d_r) = 1$ kommt es vor, dass die späteste Ankunftszeit $l_r = g_r + 4 \cdot v \cdot \delta(o_r, d_r)$ überschritten wird.



(a) $|B| = 1$ (b) $|B| = 3$ (c) $|B| = 5$

Abb. 5.1.: Abdeckung der Busse. Innerhalb der grünen Balken muss der gewünschte Abfahrtszeitpunkt g_r einer Anfrage liegen, damit die Anfrage mit der taktbasierten Buslinie zu einer Haltestelle mit einer höheren Nummer fahren kann.



(a) Bei der taktbasierten Strategie

(b) Bei der anfragebasierten Strategie

Abb. 5.2.: Bereiche, innerhalb der der gewünschte Abfahrtszeitpunkt g_r und Abfahrtsort o_r einer Anfrage r nicht liegen dürfen, wenn sie transportiert werden müssen. Dunkelviolett eingefärbt bedeutet, dass Anfragen in beide Fahrtrichtungen nicht bedient werden können. Hellviolett eingefärbt bedeutet, dass Anfragen nicht in Richtung eines kleineren Indizes bedient werden können.

Die anfragebasierte Transport-Strategie liefert für kleine Mengen an Fahrtanfragen bessere Ergebnisse. In den Fällen mit $|B| = 1$ und $|B| = 3$ erklärt sich das vorwiegend damit, dass bei der taktbasierten Strategie keine vollständige zeitliche Abdeckung q_{d} existiert. Dadurch, dass die anfragebasierte Strategie gezielt zu Fahrtanfragen hinfährt, erzielt diese eine deutlich bessere Abdeckung. Im Fall $|B| = 5$ wurde vermutet, dass dieser Effekt dadurch zustande kommt, dass bei der anfragebasierten Strategie deutlich weniger Anfragekombinationen aus Abfahrtszeitpunkt und Abfahrtsort unmöglich zu bedienen sind als bei der taktbasierten Transport-Strategie. Hier sei nochmal auf Abbildung 5.2 verwiesen, in der die unmöglichen Kombinationen grafisch dargestellt sind. Um das zu überprüfen, wurden in Tabelle 5.1 zehn zufälligen Nachfrageszenarien bei 50 Fahrtanfragen und $|B| = 5$ Bussen untersucht, wie viele Anfragen in den markierten Zeiträumen zu Beginn abgelehnt wurden $|R_{\mathrm{ab}}|$ und wie viele Anfragen ansonsten abgelehnt wurden $|R_{\mathrm{as}}|$. Es ist sehr deutlich zu erkennen, dass bei der anfragebasierten Transport-Strategie weniger Anfragen zu Beginn abgelehnt werden.

Tab. 5.1.: Anzahl der abgelehnten Anfragen bei 10 gleichverteilten Nachfrageinstanzen mit 50 Fahrtanfragen und fünf Bussen. Aufgegliedert nach denjenigen, die zum Simulationsbeginn nicht bedienbar sind $|R_{\mathrm{ab}}|$, und allen sonstigen $|R_{\mathrm{as}}|$.

	Taktbasiert			Anfragebasiert		
Instanz	$\lvert R_{\mathrm{ab}}\rvert$	$\lvert R_{\mathrm{as}}\rvert$	$\lvert R_{\mathrm{ab}}\rvert + \lvert R_{\mathrm{as}}\rvert$	$\lvert R_{\mathrm{ab}}\rvert$	$\lvert R_{\mathrm{as}}\rvert$	$\lvert R_{\mathrm{ab}}\rvert + \lvert R_{\mathrm{as}}\rvert$
1	3	1	4	0	3	3
2	2	5	7	0	2	2
3	0	2	2	0	2	2
4	2	4	6	0	2	2
5	1	6	7	0	1	1
6	0	3	3	0	4	4
7	3	2	5	0	4	4
8	0	0	0	0	4	4
9	3	2	5	0	1	1
10	2	3	5	1	3	4
Summe	16	28	44	1	26	27

Die beiden Spitzen in der Mitnahmerate der anfragebasierten Strategie bei $|B| = 3$ bzw. $|B| = 5$ Bussen und 20 bzw. 50 Fahrtanfragen sind darauf zurückzuführen, dass erst bei dieser Menge an Fahrtanfragen häufig alle zur Verfügung stehenden Busse eingesetzt werden. Dadurch, dass die Busse nur fahren, wenn sie eine Anfrage bedienen (wollen), kommt es bei wenigen Fahrtanfragen häufig vor, dass Busse an der Haltestelle h_1 stehen bleiben. Wenn dann eine Fahrtanfrage an einer weit entfernten Haltestelle gestellt wird, sind sie nicht in der Lage, im vorgegebenen Zeitfenster zu dieser Haltestelle zu fahren, weshalb sie weiter an der Haltestelle h_1 stehen bleiben. Wenn sich die aktiven Busse ebenfalls in der Nähe von h_1 aufhalten, kann kein einziger Bus die Anfrage bedienen. Wenn dagegen mehr Fahrtanfragen gestellt werden, ist die Wahrscheinlichkeit höher, dass alle Busse eingesetzt werden, womit auch die Wahrscheinlichkeit steigt, dass Busse

in den von h_1 weit entfernten Strecken fahren. Auf Grund des Phänomens, dass die Busse zu schlecht über die Strecke verteilt sein könnten, gibt es auch keine derartige Spitze für $|B| = 1$.

Bei der taktbasierten Strategie lässt sich die merkliche Abnahme der Mitnahmerate ab einer bestimmten Menge der Fahrtanfragen dadurch erklären, dass die Busse immer häufiger voll ausgelastet fahren und damit mehr und mehr Anfragen abgelehnt werden, weil die maximale Anzahl an Fahrgästen erreicht wurde.

Bei der anfragebasierten Strategie gibt es bereits ein anderes Problem deutlich bevor ein Bus voll ist. Bei steigender Anzahl der Fahrtanfragen gilt: Ab dem Maximum der Mitnahmerate haben immer mehr Busse während der gesamten Wartezeit einer neuen Fahrtanfrage bereits einen Aktionsplan für diese Zeit. Daher kann ein Bus dann nur noch Fahrtanfragen aufnehmen, deren Starthaltestellen sowieso auf seiner Strecke liegen. Das wird durch eine in diesem Fall ungünstige Definition der anfragebasierten Transport-Strategie aus dieser Bachelorarbeit erschwert.

Ob eine Haltestelle "auf der Strecke" eines Busses liegt, wird zu dem Zeitpunkt a_r entschieden, zu dem die Anfrage gestellt wird. Dabei wird überprüft, ob ein Bus zum gewünschten Abfahrtszeitpunkt g_r in die gleiche Richtung wie die Fahrtanfrage weiterfahren wird und ohne zu wenden entsprechend seines Aktionsplans an der Starthaltestelle vorbeikommen wird. Je näher eine Haltestelle an einem beliebigen Ende der Linie ist, desto unwahrscheinlicher ist es, dass ein Bus diese Bedingung erfüllt. Insbesondere wenn die Starthaltestelle einer neuen Anfrage am Rand liegt, in diesem Fall also $o_r = h_1$ oder $o_r = h_{10}$ gilt, ist es sehr schwierig, diese Bedingung zu erfüllen. Nur wenn der gewünschte Abfahrtszeitpunkt g_r genau während der Standzeit des Busses an der Randhaltestelle ist, würde diese Bedingung erfüllt werden. In Abbildung 5.3 findet sich ein Beispiel, um das zu veranschaulichen. Die gewünschte Abfahrtszeit g_r (im Bild durch einen gelben waagrechten Strich dargestellt) liegt zu einem Zeitpunkt, zu dem der rote Bus noch in die falsche Richtung fährt. Daher wird die Anfrage r abgelehnt, obwohl die Fahrtanfrage ansonsten ideal in den Aktionsplan des roten Busses integriert werden könnte.

Das Problem, dass Haltestellen am Rand nicht "auf der Strecke" von einem Bus liegen, ist weniger schlimm bei einer geringeren Anzahl der Fahrtanfragen. Dann fährt einfach ein neuer Bus, der zum gewünschten Abfahrtszeitpunkt länger als die minimale Standzeit steht, zu der Anfrage am Rand.

Insgesamt bleibt festzuhalten, dass die in dieser Arbeit vorgestellte anfragebasierte Transport-Strategie vor allem dann zu einer (merklich) besseren Mitnahmerate führt, wenn die Anzahl der Fahrtanfragen gering ist und die zur Verfügung stehenden Busse nicht ausreichen, um taktbasiert eine vollständige zeitliche Abdeckung zu erzeugen.

5.1.2. Wartezeit pro Anfrage

Als Nächstes sollen die Ergebnisse der Wartezeit pro Anfrage für die räumlich gleichverteilte Nachfragesituation aus Abbildung 4.2 interpretiert werden.

Für das Resultat, dass die Wartezeiten geringer werden, wenn mehr Busse eingesetzt werden, sind verschiedene Gründe verantwortlich. Bei den Werten, bei denen die Wartezeit von den abgelehnten Anfragen mitgezählt wird, sorgt vor allem die höhere

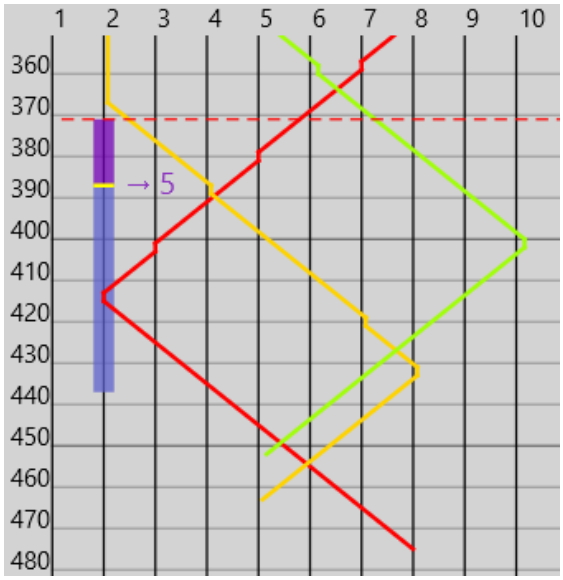


Abb. 5.3.: Fahrtanfrage r mit $o_r = 2$, $d_r = 5$, $a_r = 371$, $g_r = 387$, $w_r = 437$ und $l_r = 507$ zum Zeitpunkt $t = a_r = 371$ (dargestellt als lila/blauer Balken). Obwohl das problemlos möglich wäre, wird sie nicht zum roten Bus hinzugefügt, sondern abgelehnt, da der Bus zum Zeitpunkt $g_r = 387$ in die falsche Richtung fährt.

Mitnahmerate bei mehr Bussen für kleinere Ergebnisse. Dass die Wartezeit pro Anfrage bei mehr Fahrtanfragen steigt, lässt sich ebenfalls damit begründen, dass die Mitnahmerate dann sinkt. Auch dass die durchschnittliche Wartezeit bei großen Mengen an Fahrtanfragen für die anfragebasierte Strategie höher ist als diejenige der taktbasierten Strategie, ist durch die geringere Mitnahmerate der anfragebasierten Strategie in diesen Fällen erklärbar.

Bei dem anfragebasierten System kommt hinzu, dass mit mehr Bussen mehr Fahrtwünsche gleichzeitig mit einer sehr geringen Wartezeit erfüllt werden können. Dieser Effekt beeinflusst natürlich die Werte der anfragebasierten Strategie ohne die abgelehnten Anfragen genauso.

Die in dieser Arbeit vorgestellte anfragebasierte Strategie optimiert immer die Wartezeit der neu einzufügenden Anfrage. Dabei kann sie neue Anfragen auf zwei verschiedene Arten dem Aktionsplan eines Busses hinzufügen: Entweder sie integriert die Anfrage auf einer Strecke, die der Bus sowieso fährt, oder sie erzeugt eine komplett neue Fahrstrecke für die neue Anfrage. Letzterer Fall führt zu einer deutlich kürzeren Wartezeit für die neue Anfrage, da nur die Bedingungen von einer einzigen Fahrtanfrage hierfür erfüllt werden müssen. Wenn eine Anfrage früh genug gestellt wird, ist es in letzterem Fall häufig möglich, die Anfrage ohne Wartezeit zu erfüllen. Wie schon bei der Mitnahmerate beschrieben wurde, ist dieser Fall immer seltener möglich, je mehr Fahrtanfragen gestellt werden. Denn meistens hat ein Bus dann zum gewünschten Abfahrtszeitpunkt bereits Aktionen eingeplant. Im anderen Fall, wenn eine Anfrage auf einer bereits geplanten Strecke eingefügt wird, kann die Wartezeit normalerweise nicht optimiert werden.

Daraus folgt der schnelle Anstieg der Wartezeit, wenn mehr Anfragen gestellt werden.

Dass das Optimum bei der anfragebasierten Strategie mit $|B| = 5$ Bussen nicht bei der geringsten Anzahl der Fahrtanfragen liegt, ist wie bei der Mitnahmerate damit zu erklären, dass erst bei 40 Fahrtanfragen häufig alle zur Verfügung stehenden Busse auch genutzt werden.

Zusammenfassend bleibt auch hier festzuhalten, dass die hier vorgeschlagene Strategie nur für eine geringe Anzahl an Fahrtanfragen kürzere Wartezeiten ergibt als die taktbasierte.

Pei et al. haben eine leicht unterschiedliche Situation untersucht: Sie haben eine reale Linie mit neun Haltestellen, deren Umlaufzeit bei taktbasierten Bussen ca. doppelt so hoch war wie die hier verwendete betrachtet. Dabei haben sie die taktbasierte Strategie mit einer eigenen flexiblen Transport-Strategie, die auf konkrete Nachfrage eingeht, durch Simulation verglichen. Allerdings geht aus ihrer Arbeit nicht eindeutig hervor, ob Fahrtanfragen abgelehnt werden können. Bei 30 Anfragen pro Stunde und zwei eingesetzten Bussen erzielten sie mit ihrer flexiblen Strategie eine Verkürzung der Wartezeit um 21,12% gegenüber der taktbasierten Strategie. Bei 90 Anfragen pro Stunde lag die Verkürzung nur bei 0,09% [PLO19]. Zumindest qualitativ entspricht das den Ergebnissen dieser Arbeit: Wenige Fahrtanfragen pro Stunde führen zu einer relevanten Wartezeitverkürzung wohingegen bei vielen Fahrtanfragen kaum eine Verbesserung stattfindet.

5.1.3. Fahrzeit pro Anfrage

Weiter sollen die Ergebnisse zur Fahrzeit pro Anfrage aus der Abbildung 4.3 in der räumlich gleichverteilten Nachfragesituation diskutiert werden.

Der Erwartungswert der Fahrzeit einer Anfrage lässt sich folgendermaßen aus der erwarteten Anzahl an Stationen $E[\delta]$, die eine Fahrtanfrage fahren möchte, berechnen:

$$E[v \cdot \delta] = v \cdot E[\delta] = v \cdot \frac{\sum_{i=1}^{|H|-1} i \cdot (|H| - i) \cdot 2}{|H| \cdot (|H| - 1)} = \frac{110}{3} \approx 36{,}7$$

Die durchschnittliche Fahrzeit pro transportierter Fahrtanfrage entspricht grob also immer diesem Erwartungswert.

Wenn die Starthaltestelle und die Zielhaltestelle der Anfragen jeweils gleichverteilt generiert werden, wollen auf den mittleren Streckenabschnitten deutlich mehr Passagiere fahren als auf den äußeren. Dadurch kommt es bei der taktbasierten Strategie bei steigender Fahrtanfragenzahl zuerst auf den mittleren Streckenabschnitten zu voll ausgelasteten Fahrten. Aus diesem Grund können bei den mittleren Haltestellen dann keine weiteren Fahrtanfragen einsteigen. Fahrtanfragen, die in der Mitte starten, können per Definition keine Strecken beantragen, die länger als die Hälfte der Gesamtstrecke sind. Daher sinkt die Wahrscheinlichkeit, dass Fahrtanfragen akzeptiert werden, die nur die halbe Strecke oder weniger fahren wollen. Die Wahrscheinlichkeit, dass Anfragen akzeptiert werden, die die gesamte Streckenlänge befahren wollen, bleibt jedoch gleich. Deshalb steigt bei zunehmenden Fahrtanfragen die durchschnittliche Fahrzeit der transportierten Anfragen im taktbasierten System leicht an.

Bei der hier vorgestellten anfragebasierten Strategie passiert das Gegenteil. Wie bereits bei der Diskussion der Mitnahmerate beschrieben wurde und in Abbildung 5.3 beispielhaft gezeigt wurde, werden Fahrtanfragen, deren Starthaltestellen weiter außen von der Linie liegen, seltener akzeptiert, je mehr Fahrtanfragen insgesamt gestellt werden. Dadurch sinkt mit wachsender Menge der Fahrtanfragen die Wahrscheinlichkeit, dass für eine akzeptierte Anfrage r gilt, dass $\delta(o_r, d_r) = |H| - 1$. Das führt insgesamt dazu, dass die durchschnittliche Fahrzeit pro akzeptierter Anfrage sinkt.

5.1.4. Transportzeit pro Anfrage

Im Folgenden sollen für räumlich gleichverteilte Anfragen die Ergebnisse zur Transportzeit pro Anfrage, also der Zeit, die ein Passagier durchschnittlich in einem Bus verbringt, interpretiert werden. Sie sind in Abbildung 4.4 zu finden und wurden bereits in Kapitel 4.3.4 beschrieben.

Die taktbasierte Strategie erzielt in allen Fällen eine höhere Transportzeit als die anfragebasierte Strategie. Das lässt sich unter anderem darauf zurückführen, dass die durchschnittliche Fahrzeit pro Anfrage in fast allen Fällen bei der taktbasierten Strategie höher war als bei dem anfragebasierten System.

Betrachtet man die Differenz zwischen der durchschnittlichen Transportzeit und der durchschnittlichen Fahrzeit, erhält man die durchschnittliche Standzeit pro Anfrage. Dabei handelt es sich um die durchschnittliche Zeit, die ein Passagier im Bus sitzt, während der Bus hält. Sie wurde in Abbildung 5.4 visualisiert.

Ein weiterer Grund dafür, dass die anfragebasierte Strategie bessere durchschnittliche Transportzeiten pro Anfrage erzeugt, ist folglich, dass die durchschnittlichen Standzeiten pro Anfrage immer mindestens 30% geringer sind als bei der taktbasierten Strategie.

So lassen sich auch die Maxima bei der Transportzeit der anfragebasierten Strategie erklären. Denn je mehr Fahrtanfragen gestellt werden, desto häufiger müssen Zwischenhalte bei der anfragebasierten Strategie eingeplant werden. Daher steigt die entsprechende durchschnittliche Standzeit. Da die Standzeit ab einer bestimmten Anzahl der Fahrtanfragen schwächer steigt als die Fahrzeit sinkt, überwiegt ab 140 Fahrtanfragen das Sinken der durchschnittlichen Fahrzeit.

Außerdem enthält die anfragebasierte Strategie bei der Auswahl der direkt fahrenden Busse in Algorithmus 2 einen kleinen Fehler, der sich häufiger auswirkt, je mehr Anfragen gestellt werden. Durch diesen Fehler kommt es manchmal zu längeren Standzeiten für Busse, wenn eine Starthaltestelle einer neuen Anfrage in eine bereits geplante Fahrstrecke eingefügt wird, da der Bus auf den Passagier wartet.

An einem Beispiel in Abbildung 5.5 soll der Fehler erklärt werden: Zum Zeitpunkt $t = a_r = 164$ wird eine neue Fahrtanfrage r mit $o_r = 8$, $d_r = 4$, $a_r = 164$, $g_r = 220$, $w_r = 270$ und $l_r = 380$ gestellt. Sie ist in der Abbildung als lila/blauer Balken an Haltestelle 8 visualisiert. Vor dem Einfügen der neuen Anfrage in Abbildung 5.5a soll der hellblaue Bus zum gewünschten Abfahrtszeitpunkt $g_r = 220$ (in der Abbildung der gelbe Strich) von Haltestelle 9 zu Haltestelle 1 fahren und dabei eine entsprechende andere Fahrtanfrage $r_{\text{alt}} = (9, 1, \dots)$ transportieren. Starthaltestelle $o_r = 8$ liegt zwischen Haltestelle 9 und Haltestelle 1. Da der Algorithmus 2 nur überprüft, ob der Bus zum

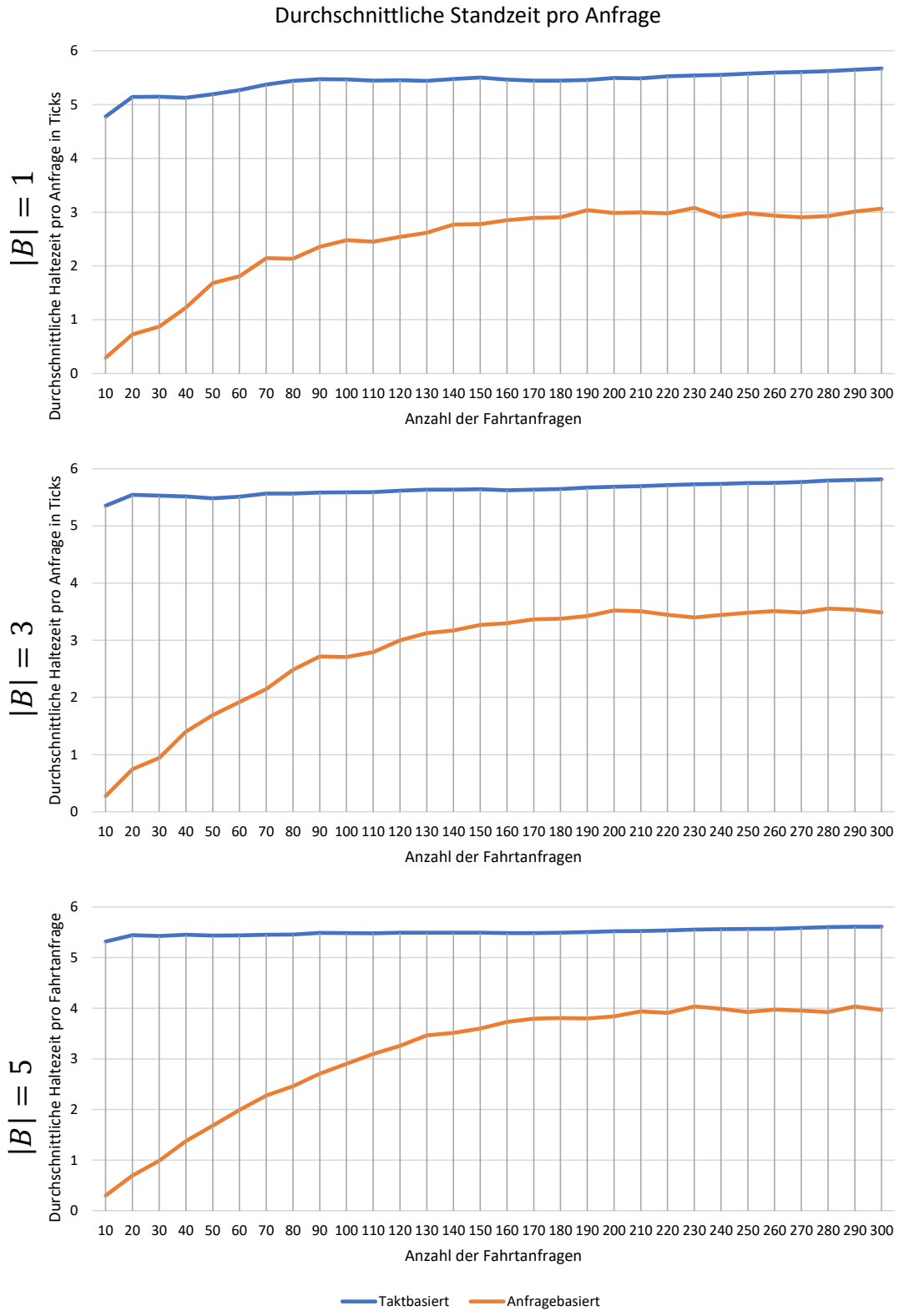
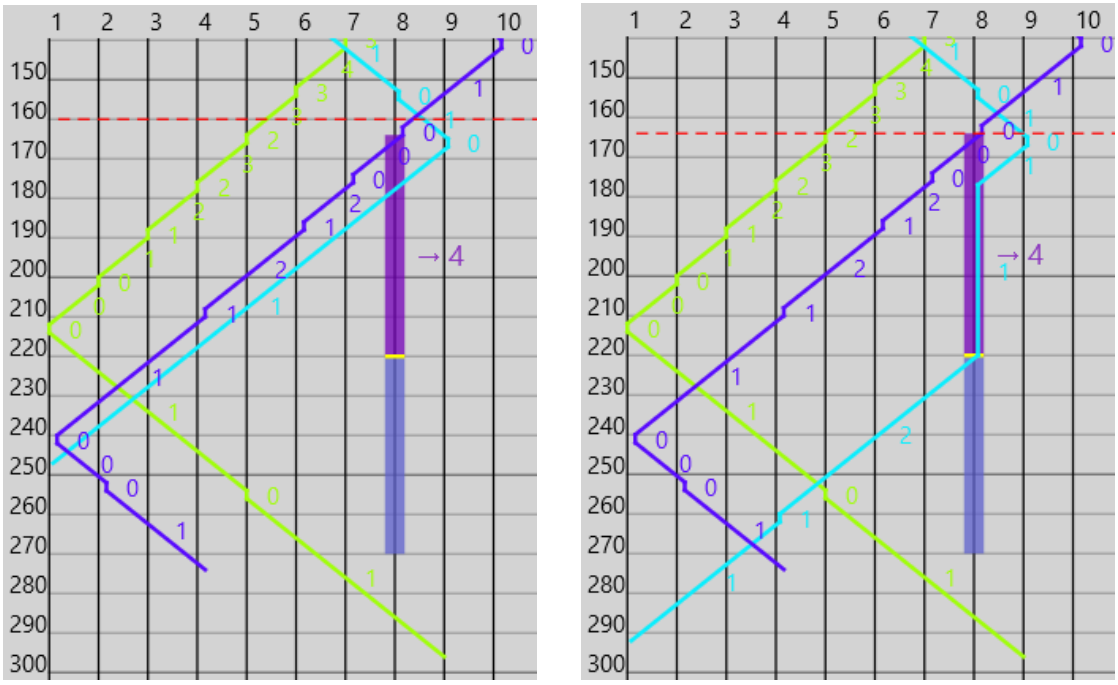


Abb. 5.4.: Differenz zwischen der durchschnittlichen Transportzeit und der durchschnittlichen Fahrzeit pro Anfrage.

aktuellen Zeitpunkt t an der Station 8 vorbeigefahren ist und nicht ob er das auch zum Zeitpunkt g_r ist, wird ein Zwischenhalt eingefügt, um die neue Anfrage r mitzunehmen. Dadurch, dass der gewünschte Abfahrtszeitpunkt $g_r = 220$ nach dem Zeitpunkt ist, an dem der hellblaue Bus an der Haltestelle abfahren würde ($\chi_{\text{theoretisch}} = 179$), muss der Bus nun auf den neuen Fahrgast warten, wie in Abbildung 5.5b zu sehen ist. Das führt zu einer ungewollten Transport- und Standzeitverlängerung von 41 Ticks für den Fahrgast r_{alt}. Bei einer weiteren Evaluation der hier vorgestellten, anfragebasierten Strategie sollte dieser Fall nicht als Auf-der-Strecke-Einfügen gehandhabt werden.



(a) Vor dem Einfügen von r ($t = 160$)

(b) Nach dem Einfügen von r ($t = 164$)

Abb. 5.5.: Zum Zeitpunkt $a_r = 164$ wird eine neue Anfrage r mit $o_r = 8$, $d_r = 4$, $a_r = 164$, $g_r = 220$, $w_r = 270$ und $l_r = 380$ in den Aktionsplan des hellblauen Busses eingefügt. Das ist unerwünscht, da die Transportzeit der Fahrtanfrage, die bereits vom hellblauen Bus transportiert wird, deutlich erhöht wird.

Da dieser Fehler größere Auswirkungen hat, je mehr Passagiere während der verlängerten Standzeit im Bus sitzen, und bei mehr Fahrtanfragen die Wahrscheinlichkeit steigt, dass er auftritt, führt er zu einer weiteren Standzeitverlängerung, je mehr Fahrtanfragen gestellt werden.

Es bleibt trotzdem festzuhalten, dass die hier vorgestellte anfragebasierte Strategie in allen Fällen mindestens eine 30% kürzere Standzeit hat als die taktbasierte Strategie. Dadurch sorgt die anfragebasierte Strategie auch für eine Transportzeitreduktion neben der Tatsache, dass die durchschnittliche Fahrzeit von akzeptierten Anfragen im anfragebasierten System kürzer ist als diejenige im taktbasierten System. Durch beide Effekte zusammen ist die Transportzeitreduktion in den betrachteten Fällen nur geringfügig

abhängig von der Menge der gestellten Anfragen.

Im Gegensatz dazu war die Transportzeitverkürzung bei der flexiblen Transport-Strategie von Pei et al. deutlich abhängig von der Anzahl der gestellten Fahrtanfragen. Sie untersuchten eine reale Buslinie mit neun Haltestellen, deren taktbasierte Umlaufzeit ca. doppelt so hoch war, wie die von dieser Arbeit simulierte. Sie simulierten die normale Buslinie und die flexible Buslinie jeweils mit zwei Bussen. Dabei kamen sie zum Ergebnis, dass bei 30 Fahrtanfragen pro Stunde die Transportzeit um 13,31% reduziert werden konnte. Bei 90 Fahrtanfragen pro Stunde erzielten sie jedoch nur eine Reduktion um 0,16%. Leider nannten sie keine Hypothesen oder Gründe für das unterschiedliche Verhalten ihrer flexiblen Strategie je nach Anzahl der Fahrtanfragen [PLO19]. Da aus der Arbeit nicht hervorgeht, dass die Busse eine Kapazitätsgrenze haben, geht der Autor dieser Arbeit davon aus, dass bei mehr Fahrtanfragen die Busse der flexiblen Transport-Strategie an fast jeder Haltestelle anhalten müssen, wodurch kaum Transportzeitverkürzungen erzielt werden können.

5.1.5. Auslastung

Nun sollen die Ergebnisse zur Auslastung der Busse bei räumlich gleichverteilten Anfragen aus Abbildung 4.5 diskutiert werden.

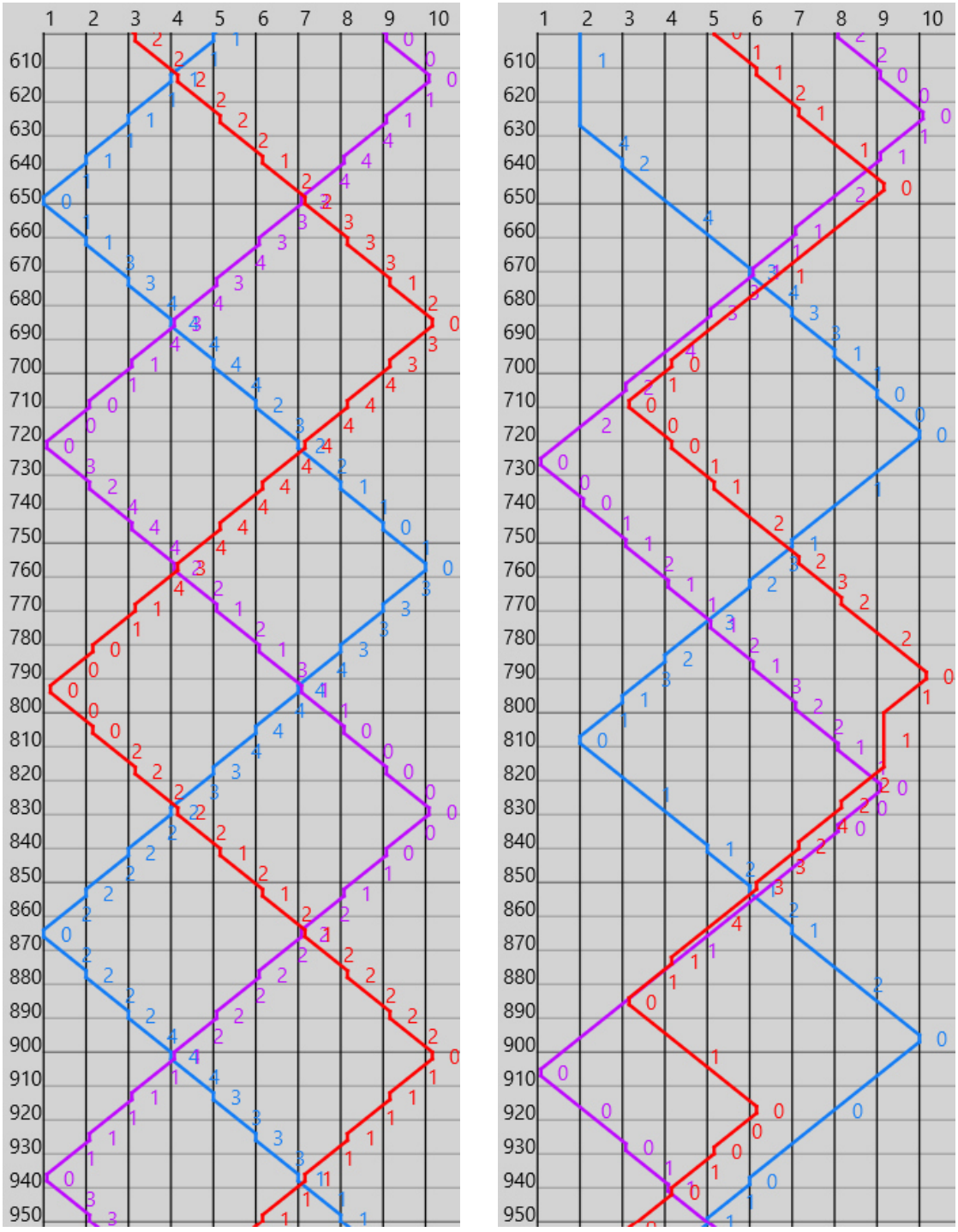
Der lineare Anstieg der Busauslastung bei der taktbasierten Strategie bis zu einem bestimmten Schwellwert der Fahrtanfragen lässt sich damit erklären, dass zunächst jede Fahrtanfrage bedient wird. Dadurch, dass jede Fahrtanfrage im Durchschnitt eine bestimmte Strecke fährt, wird die Busauslastung im Durchschnitt mit jeder weiteren Fahrtanfrage um einen bestimmten Wert erhöht.

Bei mehr Fahrtanfragen kommt es häufiger vor, dass Busse voll sind, wodurch einzelne Anfragen nicht bedient werden können. Dadurch ist das Abflachen der Auslastung bei größeren Fahrtanfragenzahlen zu erklären.

Die durchschnittliche Busauslastung der Busse der anfragebasierten Strategie beginnt bei einem deutlich höheren Wert als die der Busse der taktbasierten Strategie. Das ist darauf zurückzuführen, dass die anfragebasierten Busse bei wenigen Fahrtanfragen gezielt die Fahrtanfragen transportieren und, wenn es keine Anfragen gibt, einfach stehen bleiben. Die taktbasierten Busse fahren im Gegensatz dazu die ganze Zeit größtenteils leer die Strecke entlang. Dadurch kommt es bei der anfragebasierten Strategie pro transportierter Anfrage zu weniger und kürzeren Leerfahrten, was die bessere Auslastung bei wenigen Fahrtanfragen zur Folge hat.

Dass die Auslastung bei der anfragebasierten Strategie bei großen Fahrtanfragenzahlen kleiner ist als die Auslastung der taktbasierten Strategie, hat verschiedene Gründe. Sie sollen beispielhaft anhand der gefahrenen Strecke der Busse einer taktbasierten und einer anfragebasierten Strategie in Abbildung 5.6 aufgezeigt werden.

Ein Phänomen, das bei der anfragebasierten Strategie hin und wieder auftritt, ist, dass mehrere Busse zu beinahe der selben Zeit die selbe Strecke fahren. Das kann im Beispiel zwischen den Zeitschritten 650 und 710 bzw. 820 und 885 bei dem roten und dem lila Bus gesehen werden. Dadurch teilen sich die Fahrtanfragen, die im taktbasierten Fall durch einen einzigen Bus transportiert werden, auf zwei Busse auf, was meistens dazu



(a) Taktbasierte Strategie

(b) Anfragebasierte Strategie

Abb. 5.6.: Gefahrene Strecke mit Passagierzahlen von $|B| = 3$ Bussen bei den gleichen 300 Anfragen in 1000 Simulationsschritten zwischen $t = 600$ und $t = 950$.

führt, dass beide Busse schlechter ausgelastet sind.

Außerdem führt das dazu, dass die Abdeckung bei der anfragebasierten Strategie sinkt, wodurch insgesamt weniger Fahrtanfragen transportiert werden. Dadurch wird die Auslastung ebenfalls gesenkt, da die Busse bei einer ähnlichen Gesamtfahrzeit deutlich weniger Anfragen bedienen können. Im Beispiel kann das bei den Fahrten von den Haltestellen mit hohen Nummern zu den Haltestellen mit niedrigen Nummern gesehen werden. Zum Zeitpunkt $t = 729$ fährt der blaue Bus an Haltestelle 8 vorbei in Richtung Haltestelle 3. Danach fährt bis zum Zeitpunkt $t = 816$ überhaupt kein Bus in die beschriebene Fahrtrichtung. Dadurch entgehen der anfragebasierten Strategie Anfragen, die im taktbasierten Fall von der blauen Buslinie im entsprechenden Zeitraum transportiert werden.

Die Gründe dafür, dass bei der anfragebasierten Strategie mehrere Busse gleichzeitig die gleiche Strecke fahren, sind einerseits die Zielfunktion und andererseits der Umstand, dass die hier vorgestellte Strategie greedy optimiert. Die Fahrgäste, die ab dem Zeitpunkt $t = 816$ in den roten Bus einsteigen, könnten größtenteils auch mit dem lila Bus vier Zeitschritte später fahren. Allerdings wäre ihre Wartezeit dann minimal länger, weshalb der rote Bus zuvor eingeplant wird. Umgekehrt könnte der lila Bus die Fahrt beginnend bei $t = 823$ an Haltestelle 9 erst zum Zeitpunkt $t = 860$ fahren. Dadurch müsste er zwar höchstwahrscheinlich auf den einen transportierten Fahrgast verzichten, aber die Wahrscheinlichkeit ist groß, dass er dann mehr Passagiere transportieren würde.

Ein weiteres Phänomen, welches bei der hier vorgestellten anfragebasierten Strategie auftritt, ist, dass die Busse, direkt nachdem sie gewendet haben, sehr wenige Fahrtanfragen transportieren. Wie das zustande kommt, wurde bereits bei der Diskussion der Mitnahmerate in Abschnitt 5.1.1 beschrieben. Das fällt auch im Beispiel auf: Auf den Strecken, direkt nachdem ein Bus gewendet hat, ist die Auslastung der Busse im taktbasierten Fall im Durchschnitt 42,5% während sie im anfragebasierten Fall nur 12,5% ist.

Insgesamt ist hinsichtlich der Auslastung der Busse festzuhalten, dass die hier vorgestellte anfragebasierte Strategie nur bei einer geringen Anzahl der Fahrtanfragen bessere Ergebnis produziert und insbesondere durch die für dieses Kriterium ungünstige Zielfunktion bzw. den Greedy-Ansatz schlechtere Ergebnisse bei hoher Nachfrage erzielt.

5.1.6. Busfahrzeit insgesamt

Abschließend sollen die Ergebnisse zur Busfahrzeit insgesamt bei räumlich gleichverteilten Anfragen diskutiert werden. Sie sind in Abbildung 4.6 zu finden.

Dass die Busfahrzeit aller Busse insgesamt bei der taktbasierten Strategie unabhängig von der Anzahl der Fahrtanfragen immer gleich bleibt, ist selbstverständlich darauf zurückzuführen, dass die Busse stets ihren taktbasierten Fahrplan einhalten ohne zu berücksichtigen, wie hoch das Fahrgastaufkommen ist.

Bei der anfragebasierten Strategie ist das Gegenteil der Fall: Da die Busse nur bei Bedarf fahren, steigt die Fahrzeit insgesamt mit der Menge der Fahrtanfragen. Bei geringen Anfragezahlen ist dieser Anstieg sehr stark, da für jede neue Anfrage eine zusätzliche Fahrt inklusive der zugehörigen Anfahrt eingeplant werden muss. Je mehr Anfragen

gestellt werden, desto häufiger können Anfragen “auf der Strecke” in den Aktionsplan eines Busses integriert werden, wodurch weniger neue Fahrzeit erzeugt wird. Ab einem bestimmten Punkt ist die gesamte Busfahrzeit der anfragebasierten Busse höher als die der taktbasierten. Das lässt sich darauf zurückführen, dass die anfragebasierten Busse seltener anhalten und die entsprechende Zeit weiterfahren können. Bei $|B| = 1$ Bus lässt sich die längere Gesamtfahrzeit mit einer höheren Mitnahmerate bei bis zu 130 Fahrtanfragen rechtfertigen, für $|B| = 3$ Busse gilt das immerhin für bis zu 120 Fahrtanfragen.

5.2. Fahrtanfragen mit wichtiger Haltestelle

Dass die Ergebnisse für die anfragebasierte Strategie im Vergleich zur taktbasierten Strategie im Fall mit einer wichtigen Haltestelle nicht deutlich besser sind als im gleichverteilten Fall, lässt sich darauf zurückführen, dass die anfragebasierte Strategie die falschen Fahrtanfragen bündelt. So wie die anfragebasierte Strategie in dieser Arbeit definiert wurde, bündelt sie Fahrten, die zumindest teilweise auf der gleichen Strecke liegen. Das ist jedoch für alle Fahrtanfragen, die an der wichtigen Haltestelle h_1 starten, der Fall. Dadurch gilt, dass wenn in einem Bus eine einzige Anfrage r ist, deren Zielhaltestelle d_r am anderen Ende der Linie liegt, der Bus bis zur entsprechend weit entfernten Haltestelle fahren muss. Da das häufig passiert, kommen alle Busse bei der anfragebasierten Strategie deutlich seltener an die wichtige Haltestelle h_1 als das bei einer besseren Bündelung von gleich langen Fahrten möglich wäre.

6. Fazit

Das Ziel dieser Arbeit war, beim Einsatz von Minibussen auf einer festen Buslinie eine taktbasierte Transport-Strategie mit einer selbst entwickelten anfragebasierten Transport-Strategie zu vergleichen. Dazu wurde eine anfragebasierte Transport-Strategie definiert, die mit Hilfe eines Greedy-Algorithmus die Wartezeiten der Fahrgäste möglichst gering halten soll. Anschließend wurde eine Simulation entwickelt, in der beide Strategien unter den gleichen Bedingungen getestet wurden. Danach wurden die Simulationsläufe von beiden Transport-Strategien hinsichtlich der Mitnahmerate, der Wartezeit pro Anfrage, der Transportzeit pro Anfrage sowie der Busauslastung und der gesamten Busfahrzeit ausgewertet. Dabei hat sich herausgestellt, dass die entwickelte, anfragebasierte Transport-Strategie bei allen Kriterien außer der Transportzeit nur bei einer geringen Anzahl an Fahrtanfragen vorteilhaft ist. Die durchschnittliche Transportzeit wird sogar sowohl bei geringer als auch bei hoher Nachfrage reduziert.

Hinsichtlich aller untersuchten Kriterien können für $|B| = 1$ verfügbaren Minibus bis zu 60 Fahrtanfragen in 100 Minuten besser mit der anfragebasierten Strategie als mit der taktbasierten Strategie transportiert werden. Nimmt man eine längere Gesamtfahrzeit für den Bus in Kauf, erzielt die anfragebasierte Strategie sogar für bis zu 110 Fahrtanfragen in 100 Minuten bessere Ergebnisse als die taktbasierte Transport-Strategie. Bei $|B| = 3$ verfügbaren Minibussen ist die anfragebasierte Strategie bei bis zu 80 Fahrtanfragen in 100 Minuten hinsichtlich aller untersuchten Kriterien gegenüber der taktbasierten Strategie vorzuziehen. Aus Sicht eines Fahrgasts, also wenn man nur die Mitnahmerate, Wartezeit und Transportzeit betrachtet, sind die Ergebnisse der anfragebasierten Strategie sogar bei bis zu 120 Fahrtanfragen in 100 Minuten besser. Wenn $|B| = 5$ Minibusse zur Verfügung stehen, gilt, dass die anfragebasierte Strategie für 20 bis 80 Fahrtanfragen in 100 Minuten bei allen betrachteten Kriterien bessere Ergebnisse hervorbringt. Bei 10 Fahrtanfragen gelangt die taktbasierte Strategie bei der Mitnahmerate zu leicht besseren Ergebnissen.

Damit wurde gezeigt, dass auf Minibuslinien mit geringer Nachfrage selbst einfache anfragebasierte Strategien bereits bessere Ergebnisse produzieren können als eine taktbasierte Strategie.

Zumindest qualitativ entspricht das Gesamtresultat dieser Arbeit somit dem Ergebnis der Arbeit von Pei et al., die eine andere flexible (anfrageorientierte) Transport-Strategie mit einer taktbasierten auf einer festen Buslinie verglichen [PLO19].

Trotz der guten Ergebnisse gibt es noch verschiedene Möglichkeiten, wie die Evaluation und die entwickelte anfragebasierte Transport-Strategie verbessert werden können.

Bei der Auswertung und Diskussion der Ergebnisse dieser Arbeit hat sich herausgestellt, dass es sinnvoll ist, die Evaluationskriterien erst zu erheben, wenn sich die Simulation in einen stabilen Zustand eingependelt hat. Dadurch werden Verzerrungen

der Ergebnisse durch Effekte, die nur zum Beginn der Simulation passieren, vermieden. Beispielsweise würde sich dadurch die zum Simulationsbeginn geringe Abdeckung der taktbasierten Strategie weniger auf die Ergebnisse der Mitnahmerate auswirken. Daher sollte bei einer zukünftigen Evaluation nur ein festgelegter Zeitraum evaluiert werden, der nicht zum Beginn oder zum Ende der Simulation liegt. Ein Problem dabei ist allerdings, festzulegen, wie mit Anfragen umgegangen werden soll, die auf den Grenzen des festgelegten Zeitraums liegen. Andererseits konnte so gezeigt werden, dass die anfragebasierte Strategie ein besseres Anlaufverhalten hat, was in der Realität wichtig ist, da viele Systeme nicht durchgehend in Betrieb sind.

Weiter könnte man noch mehr Parameter bei dem System variieren und zum Beispiel untersuchen, wie sich das Verhältnis zwischen der minimalen Standzeit und der Fahrzeit zwischen den Haltestellen auf die Ergebnisse auswirkt. Auch spannend wäre zu überprüfen, ob andere Minibuskapazitäten zu anderen Ergebnissen führen. Außerdem ist es interessant andere Nachfrageverteilungen zu betrachten. Pei et al. haben beispielsweise eine Nachfrageverteilung untersucht, bei der besonders viele Fahrtwünsche in eine Fahrtrichtung gestellt wurden [PLO19].

Bei der entwickelten anfragebezogenen Transport-Strategie handelt es sich um eine zumindest auf abstrakter Ebene einfache Greedy-Strategie. Für jede neue Fahrtanfrage bestimmt sie, wie die Anfrage in den bestehenden Aktionsplan eines Busses integriert wird, sodass die geringste Wartezeit für besagte Anfrage entsteht. Statt der geringsten Wartezeit könnte man zukünftig andere Zielfunktionen verwenden, wie zum Beispiel die frühste Ankunftszeit der neuen Anfrage, um so die Transportzeit weiter zu verkürzen. Eine andere Möglichkeit wäre, die neue Anfrage so in einen Aktionsplan einzufügen, dass die gesamte Verzögerung von allen betroffenen Anfrage so gering wie möglich gehalten wird. Schließlich könnte man bei jeder neuen Anfrage alle Aktionspläne zusammen neu berechnen und dabei versuchen gezielt ähnliche Strecken in einem Fahrzeug zu bündeln.

Literaturverzeichnis

[CMNG05] Theodor Gabriel Crainic, Federico Malucelli, Maddalena Nonato und François Guertin: Meta-Heuristics for a Class of Demand-Responsive Transit Systems. 2005.

[GG23] Dirk Günther und Patrick Gniffke: *Berichterstattung unter der Klimarahmenkonvention der Vereinten Nationen und dem Kyoto-Protokoll 2023*. Umweltbundesamt, 2023.

[PLO19] Mingyang Pei, Peiqun Lin und Junfeng Ou: Real-Time Optimal Scheduling Model for Transit System with Flexible Bus Line Length. *Transportation Research Record*, 2019.

[RHB+23] Julia Repenning, Ralph Harthan, Ruth Blanck, Hannes Böttcher, Sibylle Braungardt, Veit Bürger, Vanessa Cook, Lukas Emele, Katharina Göckeler, Wolf Kristian Görz, Florian Hacker, Wolfram Jörß Klaus Hennenberg, Peter Kasten, Konstantin Kreye, Sylvie Ludig, Felix Chr. Matthes, Lorenz Moosmann, Christian Nissen, Judith Reise, Margarethe Scheffler, Katja Schumacher, Kirsten Wiegmann, Alexander Zerrahn, Heike Brugger, Tobias Fleiter, Tim Mandel, Matthias Rehfeldt, Clemens Rohde, Jana Deurer und Jan Steinbach: *Klimaschutzinstrumente-Szenario 2030 (KIS-2030) zur Erreichung der Klimaschutzziele 2030*. Umweltbundesamt, 2023.

[VMA+22] Pieter Vansteenwegen, Lissa Melis, Dilay Aktaş, Bryan Galarza Montenegro, Fábio Sartori Vieira und Kenneth Sörensen: A survey on demand-responsive public bus systemes. 2022.

A. Ausführlichere formale Problemdefinition

A.1. Allgemeine Bedingungen für Aktionspläne

Die Aktionspläne werden zwar durch die benutzte Transport-Strategie festgelegt, dennoch gelten für sie einige allgemeine Bedingungen. Im Folgenden werden diese alle vorgestellt, außer diejenigen, die bereits transportierte Passagiere betreffen.

Für alle Aktionen $p_{b,t}^{(i)}$ eines Aktionsplans $\varrho(b,t)$ gilt, dass der Zeitpunkt einer Aktion $\chi_{b,t}^{(i)}$ einmalig innerhalb des Aktionsplan sein muss und dass alle Aktionen chronologisch sortiert sind:

$$\forall \left(p_{b,t}^{(i)}, p_{b,t}^{(j)}\right) \in (\varrho(b,t))^2 : \left(i < j \Leftrightarrow \chi_{b,t}^{(i)} < \chi_{b,t}^{(j)}\right) \wedge \left(i = j \Leftrightarrow \chi_{b,t}^{(i)} = \chi_{b,t}^{(j)}\right)$$

Außerdem müssen sich Ankunfts- und Abfahrtsaktionen in jedem Aktionsplan abwechseln. Die erste und die letzte Aktion im Aktionsplan müssen eine Ankunftsaktion sein.

$$\begin{aligned} \forall 1 \leq i \leq z_{b,t} - 1 \colon \gamma_{b,t}^{(i)} &\neq \gamma_{b,t}^{(i+1)} \\ \gamma_{b,t}^{(1)} &= \text{ankunft} \\ \gamma_{b,t}^{(z_{b,t})} &= \text{ankunft} \end{aligned}$$

Für die Haltestellen $\sigma_{b,t}^{(i)}$ gilt, dass, wenn eine Abfahrtsaktion auf eine Ankunftsaktion folgt, beide Stationen gleich sein müssen. Im umgekehrten Fall gilt, dass beide Stationen ungleich sein müssen.

$$\forall 1 \leq i \leq z_{b,t} - 1 \colon \gamma_{b,t}^{(i)} = \text{ankunft} \Leftrightarrow \sigma_{b,t}^{(i)} = \sigma_{b,t}^{(i+1)}$$

Auch für die Zeitdifferenzen zwischen zwei Aktionen gibt es Regeln. Zwischen einer Abfahrts- und einer Ankunftsaktion muss genau so viel Fahrzeit vergehen wie die Fahrzeit für die Strecke es vorsieht. Zwischen einer Ankunfts- und eine Abfahrtsaktion muss eine Zeitspanne liegen, die größer gleich der minimalen Standzeit s ist.

$$\begin{aligned} &\forall 1 \leq i \leq z_{b,t} - 1 \colon \gamma_{b,t}^{(i)} = \text{ankunft} \Rightarrow \chi_{b,t}^{(i+1)} - \chi_{b,t}^{(i)} \geq s \\ &\forall 1 \leq i \leq z_{b,t} - 1 \colon \gamma_{b,t}^{(i)} = \text{abfahrt} \Rightarrow \chi_{b,t}^{(i+1)} - \chi_{b,t}^{(i)} = \delta\left(\sigma_{b,t}^{(i)}, \sigma_{b,t}^{(i+1)}\right) \cdot v \end{aligned}$$

Die Funktion $\tau(b,t)$ berechnet für einen Bus b den größten Index i einer Aktion $p_{b,t}^{(i)}$ in einem Aktionsplan $\varrho(b,t)$, deren Zeitpunkt $\chi_{b,t}^{(i)}$ vor der Zeit t liegt. Diese Aktion

entspricht der letzten durchgeführten Aktion des Busses. (Im Sonderfall, dass $t = 0$ gilt, ist er 1).

$$\tau\colon B \times \mathbb{N}_0 \to \mathbb{N}, (b,t) \mapsto \max\left(\left\{1 \leq i \leq z_{b,t} \;\middle|\; \chi_{b,t}^{(i)} < t\right\} \cup \{1\}\right)$$

Die Funktion ϵ berechnet darauf aufbauend den Index der nächsten Aktion des Busses b.

$$\epsilon\colon B \times \mathbb{N}_0 \to \mathbb{N}, (b,t) \mapsto \tau(b,t) + 1$$

Die Transport-Strategie darf nur zukünftige Aktionen verändern. Daher müssen zu jedem Zeitpunkt t alle vergangenen Aktionen eines Busses b identisch zum Aktionsplan des Busses $\varrho(b, t-1)$ sein. Dementsprechend darf ein Aktionsplan mit zunehmender Zeit nur größer werden.

$$\forall t \in \mathbb{N}\colon\ |\varrho(b,t)| \geq |\varrho(b,t-1)|$$
$$\forall t \in \mathbb{N}\colon \forall 1 \leq i \leq \tau(b,t)\colon p_{b,t}^{(i)} = p_{b,t-1}^{(i)}$$

Die Fahrtrichtung ϑ eines Busses b zum Zeitpunkt t ist folgendermaßen definiert:

$$\vartheta\colon B \times \mathbb{N}, (b,t) \mapsto \vartheta\left(\sigma_{b,t}^{(\tau(b,t))}, \sigma_{b,t}^{(\epsilon(b,t))}\right)$$

Weiterhin ist es verboten, dass der Bus seine Fahrtrichtung während der Fahrt wechselt. Daher gilt zu jedem Zeitpunkt $t > 0$ für jeden Bus b: Wenn seine letzte Aktion eine Abfahrt war, muss die Fahrtrichtung die gleiche sein wie bei $t-1$ bzw. wenn der Bus bei $t-1$ noch nicht gefahren ist, muss die neue Fahrtrichtung der Fahrtrichtung nach der Abfahrt bei $t-1$ entsprechen.

$$\begin{aligned}\gamma_{b,t}^{(\tau(b,t))} = \text{abfahrt} \Rightarrow \Big(&\gamma_{b,t-1}^{(\tau(b,t-1))} = \text{abfahrt} \wedge \vartheta(b,t-1) = \vartheta(b,t)\Big) \\ \vee\ \Big(&\gamma_{b,t-1}^{(\tau(b,t-1))} = \text{keine} \wedge \vartheta\Big(\sigma_{b,t-1}^{(\tau(b,t-1))}, \sigma_{b,t-1}^{(\epsilon(b,t-1))}\Big) = \vartheta(b,t)\Big)\end{aligned}$$

Außerdem darf die nächste Ankunft nicht an eine Haltestelle vorgezogen werden, an der der Bus bereits vorbeigefahren ist. Wie viele Haltestellen der Bus seit seiner letzten Abfahrt bereits passiert hat, wird durch die seitdem verstrichene Fahrzeit $t - \chi_{b,t}^{(\tau(b,t))}$ berechnet, indem es durch die Fahrzeit pro Haltestelle v dividiert wird.

$$\gamma_{b,t}^{(\tau(b,t))} = \text{abfahrt} \Rightarrow \delta\left(\sigma_{b,t}^{(\tau(b,t))}, \sigma_{b,t}^{(\epsilon(b,t))}\right) \geq \frac{t - \chi_{b,t}^{(\tau(b,t))}}{v}$$

A.2. Mathematische Definition der aktuell transportierten Passagiere

Wie im Hauptteil der Arbeit erwähnt wurde, können die Fahrtanfragen, deren Passagiere zum Zeitpunkt t in einem Bus b sitzen, von einer Funktion ω berechnet werden. Dabei

definieren sich die Passagiere, die im Bus sitzen, durch das Komplement zweier Mengen. η gibt an, welche Passagiere bis zum Zeitpunkt t in den Bus b eingestiegen sind und κ ist die Menge der Passagiere, die aus dem Bus b zum Zeitpunkt t ausgestiegen sind. Zum Startzeitpunkt $t = 0$ sind alle Busse leer.

$$\omega\colon B \times \mathbb{N}_0 \to 2^R, (b,t) \mapsto \begin{cases} \eta(b,t) \setminus \kappa(b,t) & \text{falls } t > 0 \\ \emptyset & \text{sonst} \end{cases}$$

Die Menge der ausgestiegenen Fahrgäste zu einem bestimmten Zeitpunkt t aus einem Bus b, lässt sich rekursiv vergleichsweise einfach ausdrücken. Zum Startzeitpunkt $t = 0$ ist sie leer. Wenn der Bus direkt vor dem aktuellen Zeitpunkt an keiner Station angekommen ist, entspricht sie der Menge der ausgestiegenen Fahrgäste vom gleichen Bus b zum Zeitpunkt $t - 1$. Wenn der Bus direkt vor dem aktuellen Zeitpunkt an einer Station angekommen ist, werden alle Fahrgäste, die an dieser Station aussteigen wollen, zu der Menge hinzugefügt.

$$\kappa\colon B \times \mathbb{N}_0 \to 2^R, (b,t) \mapsto \begin{cases} \emptyset & \text{falls } t = 0 \\ \kappa(b,t-1) & \text{falls } t \neq 0 \\ & \wedge \left(\gamma_{b,t}^{(\tau(b,t))} \neq \text{ankunft} \right. \\ & \quad \left. \vee \chi_{b,t}^{(\tau(b,t))} \neq t-1\right) \\ \kappa(b,t-1) & \text{sonst} \\ \cup \left\{ r \in \omega(b,t-1) \;\middle|\; d_r = \sigma_{b,t}^{(\tau(b,t))} \right\} & \end{cases}$$

Die Menge der bisher eingestiegenen Fahrgäste berechnet sich auf den ersten Blick analog: Zum Startzeitpunkt $t = 0$ ist sie leer. Wenn der Bus b nicht zum letzten Zeitpunkt $t - 1$ an einer Station abgefahren ist, entspricht sie der gleichen Menge zum Zeitpunkt $t-1$. Wenn der Bus jedoch soeben an einer Station abgefahren ist, wird eine Hilfsfunktion η_g benötigt, um zu ermitteln, welche Passagiere in genau diesen Bus einsteigen.

$$\eta\colon B \times \mathbb{N}_0 \to 2^R, (b,t) \mapsto \begin{cases} \emptyset & \text{falls } t = 0 \\ \eta(b,t-1) & \text{falls } \gamma_{b,t}^{(\tau(b,t))} \neq \text{abfahrt} \\ & \quad \vee\, \chi_{b,t}^{(\tau(b,t))} \neq t-1 \\ \eta(b,t-1) & \text{sonst} \\ \cup \bigcup_{\{r \in R \,|\, \eta_g(t,b,r)=1\}} & \end{cases}$$

Die Funktion η_g ist eine binäre Hilfsfunktion, die 1 ausgibt, wenn der Passagier mit dem Index seiner Fahrtanfrage j in den Bus mit dem Index i exakt vor dem Zeitpunkt t eingestiegen ist. Ansonsten gibt sie 0 aus. Sie ist notwendig, da sichergestellt werden muss, dass ein Passagier r_j nicht in mehrere Busse gleichzeitig einsteigt, bzw. ein Bus nicht von beliebig vielen Passagieren gleichzeitig betreten werden kann. Um das zu evaluieren, wird erst überprüft, ob der Zeitpunkt $t = 0$ ist, in diesem Fall kann keiner einsteigen, da es zu diesem Zeitpunkt bei jedem Bus eine Ankunftsaktion gibt. Anschließend wird überprüft, ob der Bus zum letzten Zeitpunkt an der richtigen Haltestelle angehalten

hat. Wenn das der Fall ist, wird sichergestellt, dass der Bus zum richtigen Zielort fährt. Außerdem wird überprüft, ob der Passagier zeitlich überhaupt an der Haltestelle stehen und warten kann. Schließlich wird festgestellt, ob der Passagier nicht in einen früheren Bus eingestiegen ist oder der Bus voll ist. Wenn alle Überprüfungen zu einem negativen Ergebnis kommen, wird 1 ausgegeben.

$$\eta_g \colon \mathbb{N}_0 \times B \times R \to \{0,1\},$$

$$(t, b_i, r_j) \mapsto \begin{cases} 0 & \text{wenn } t = 0 \\ 0 & \text{sonst wenn } \gamma_{b_i,t}^{(\tau(b_i,t))} \neq \text{abfahrt} \\ 0 & \text{sonst wenn } \chi_{b_i,t}^{(\tau(b_i,t))} \neq t-1 \\ 0 & \text{sonst wenn } \sigma_{b_i,t}^{(\tau(b_i,t))} \neq o_{r_j} \\ 0 & \text{sonst wenn } d_{r_j} \notin \xi(b_i,t) \\ 0 & \text{sonst wenn } \Psi_{b,t}^{(i)} \neq \text{unbenutzt} \wedge r_j \notin \Psi_{b,t}^{(i)} \\ 0 & \text{sonst wenn } t < g_{r_j} \\ 0 & \text{sonst wenn } w_{r_j} < t \\ 0 & \text{sonst wenn } \exists 1 \leq t_{\text{alt}} < t \colon r_j \in \omega(b, t_{\text{alt}}) \\ 0 & \text{sonst wenn } \exists 1 \leq i_{\text{vor}} < i \colon \eta_g(t, b_{i_{\text{vor}}}, r_j) = 1 \\ 0 & \text{sonst wenn } c \leq \sum_{j_{\text{vor}}=1}^{j-1} \eta_g(t, b_i, r_{j_{\text{vor}}}) + |\omega(b_i, t-1)| \\ 1 & \text{sonst} \end{cases}$$

Dabei gibt die Hilfsfunktion ξ an, an welchen Stationen ein Bus basierend auf seinem Aktionsplan zum Zeitpunkt t voraussichtlich halten soll. Dazu wird ermittelt, bei welchen Haltestellen h es in der Zukunft Ankünfte gibt. Da ein Aktionsplan die Strecke über die nächste Wende hinaus plant, dürfen nur Haltestellen vor diesem Wendepunkt betrachtet werden.

$$\xi \colon B \times \mathbb{N}_0 \to 2^H,$$
$$(b,t) \mapsto \Big\{ h \in H \,\Big|\, \exists \left(\gamma_{b,t}^{(i)}, \sigma_{b,t}^{(i)}, \chi_{b,t}^{(i)}, \Psi_{b,t}^{(i)}\right) \in \varrho(b,t) \colon i > \tau(b,t) \wedge \gamma_{b,t}^{(i)} = \text{ankunft} \wedge \sigma_{b,t}^{(i)} = h$$
$$\wedge \forall \tau(b,t) < j \leq i \colon \left(\gamma_{b,t}^{(i)} \neq \text{ankunft} \vee \vartheta\left(\sigma_{b,t}^{(j-1)}, \sigma_{b,t}^{(j)}\right) = \vartheta\left(\sigma_{b,t}^{(i-1)}, \sigma_{b,t}^{(i)}\right)\right) \Big\}$$

B. Definition weiterer verwendeter Funktionen in Pseudocode

Insbesondere in Abschnitt 3.3.2 wurden einige Algorithmen und Verhaltensweisen durch Pseudocode definiert. Dabei wurden manche nicht wesentliche Funktionen als bereits definiert angenommen. Damit Klarheit besteht, was diese Funktionen exakt tun, werden sie im Folgenden ausgeführt.

B.1. Funktion zur Ermittlung der nächsten Fahrtrichtung

Algorithmus 13 soll entsprechend ihrem Namen die nächste Fahrtrichtung an einem bestimmten Index in einem Aktionsplan bestimmen.

Algorithmus 13: GebeNächsteFahrtrichtung(Liste P, int i)

Eingabe: Liste von Aktionen (Aktionsplan) P, Index in P nach welcher Aktion die Fahrtrichtung bestimmt werden soll i

Ausgabe: Eine Fahrtrichtung aus Θ

```
1 if |P| ≤ i then
2     return keine
3 P_vorherig = P[i]
4 P_nächste = P[i + 1]
5 f = ϑ(P_vorherig, P_nächste)
6 if f = keine then
7     return GebeNächsteFahrtrichtung(P, i + 1)
8 return f
```

B.2. Funktion zur Ermittlung eines Aktionsplanindex zu einer Zeit

Algorithmus 14 gibt den Index der letzten Aktion in der Aktionsliste aus, die ausgeführt wurde. Wenn bisher keine Aktion ausgeführt wurde, gibt er 0 aus.

Algorithmus 14: GebeAktuellenAktionsplanIndex(Liste P, int t)

Eingabe: Liste von Aktionen (Aktionsplan) P, Zeitpunkt t
Ausgabe: Index der Aktion, die zum Zeitpunkt t zuletzt ausgeführt wurde

```
1 i = 1
2 while i ≤ |P| do
3     (γ, σ, χ, Ψ) = p = P[i]
4     if t ≤ χ then
5         return i − 1
6     i = i + 1
7 return |P|
```

B.3. Funktion, die einen Fahrgastwechsel ausführt

Algorithmus 15 ändert eine Liste von Fahrgästen ω_{geplant} so wie, wenn die Aktion p ausgeführt werden würde. Dabei ignoriert sie die zeitlichen Bedingungen vollständig.

Algorithmus 15: FührеFahrgastwechselAus(Liste ω_{geplant}, Aktion $p = (\gamma, \sigma, \chi, \Psi)$)

Eingabe: Liste von Fahrgästen ω_{geplant}, auszuführende Aktion p
Ausgabe: Liste von Fahrgästen nach Fahrgastwechsel ω_{geplant}

```
1 if γ = abfahrt then
2     Hinzufügen(ω_geplant, Ψ)
3 else
4     ω_neu = []                          /* ω_neu ist eine Liste */
5     foreach r ∈ ω_geplant do
6         if d_r ≠ σ then
7             Hinzufügen(ω_neu, r)
8     ω_geplant = ω_neu
9 return ω_geplant
```

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Versicherung zur selbständigen Leistungserbringung

Titel der Bachelorarbeit

Vergleich von einer anfragebasierten und fahrplanbasierten Minibuslinie durch Simulation

Thema bereitgestellt von (Titel, Vorname, Nachname, Lehrstuhl):

Prof. Dr. Marie Schmidt, Lehrstuhl für Informatik I

Eingereicht durch (Vorname, Nachname, Matrikel):

Leo Puppe, 2603933

Ich versichere, dass ich die vorstehende schriftliche Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die benutzte Literatur sowie sonstige Hilfsquellen sind vollständig angegeben. Wörtlich oder dem Sinne nach dem Schrifttum oder dem Internet entnommene Stellen sind unter Angabe der Quelle kenntlich gemacht.

Weitere Personen waren an der geistigen Leistung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich nicht die Hilfe eines Ghostwriters oder einer Ghostwriting-Agentur in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar Geld oder geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Arbeit stehen.

☐ Mit dem Prüfungsleiter bzw. der Prüfungsleiterin wurde abgestimmt, dass für die Erstellung der vorgelegten schriftlichen Arbeit Chatbots (insbesondere ChatGPT) bzw. allgemein solche Programme, die anstelle meiner Person die Aufgabenstellung der Prüfung bzw. Teile derselben bearbeiten könnten, eingesetzt wurden. Die mittels Chatbots erstellten Passagen sind als solche gekennzeichnet.

Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit ist vollständig. Mir ist bewusst, dass nachträgliche Ergänzungen ausgeschlossen sind.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung zur Versicherung der selbstständigen Leistungserbringung rechtliche Folgen haben kann.

Gerbrunn, der 07.08.2023 Leo Puppe

Ort, Datum, Unterschrift