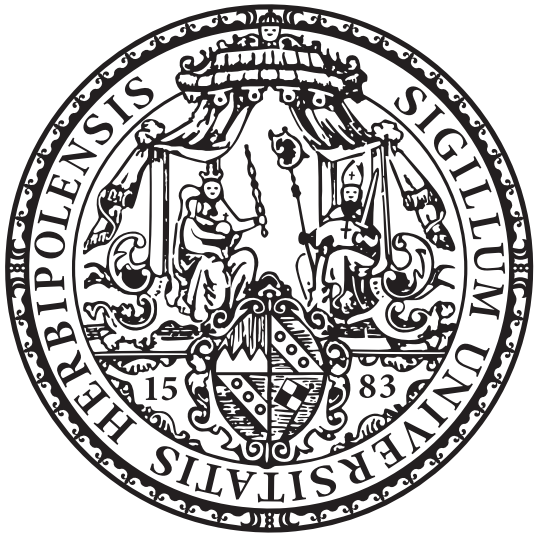
Bachelorarbeit

Simulation eines neuen Matching-Algorithmus´ für Studierenden-Ridesharing

Marie Holfelder

Abgabedatum: 26. Juli 2024
Betreuer: Prof. Dr. Marie Schmidt



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Abstract

Due to factors that include pollution and traffic congestion, it is advisable to look into reducing the number of cars on the road. For students, we could achieve that using ridesharing. Multiple commuters would where possible be grouped into a single car, belonging to one of them, for both the trip to university and the trip back separately. We suggest an algorithm to perform the task of grouping without allowing detours. In return, every passenger has to be within a certain distance of the driver's route and travel to the route themselves, in order to be picked up. We then simulate this algorithm using 14 datasets of, in our default case, roughly 6500 agents and some modified versions of those sets. The algorithm is implemented inside a framework for testing different types of campus mobility, that was written by Helena Fehler. The results show, that we can reduce the number of drivers to a third and halve the total distance travelled and therefore halve the emitted CO_2 for the standard values with a full set of agents, compared to the default case, where every agent uses their own car.

Inhaltsverzeichnis

1 Einleitung

Aktuell wird es - besonders durch steigende Preise und den Klimawandel - immer wichtiger für viele Menschen, für den Weg zur Arbeit Alternativen zu den klassischen öffentlichen Verkehrsmitteln oder zum Fahren mit dem eigenen Auto zu finden[SCZ20a]. Von allen Trips, die eine Person im Durchschnitt im Jahr 2022 in den USA zurückgelegt hat, fallen 18,4% auf Fahrten zum und vom Arbeitsplatz. Das macht 19,7% der pro Person insgesamt zurückgelegten Distanz aus[Laba]. Von diesen Fahrten wurde, nach Daten aus dem Jahr 2020, für etwa 84% alleine der private PKW genutzt[Labb]. Auch die Deutschen haben im Jahr 2022 etwa 65% der Strecken zur Arbeit im eigenen Auto zurückgelegt[Sta]. Das bietet einen großen Angriffspunkt, um Emissionen und Straßennutzung zu verringern. Grundsätzlich besteht auch die Möglichkeit, öffentliche Verkehrsmittel zu nutzen, was aber aktuell nicht für alle Pendler eine Möglichkeit ist. Für viele ist der zusätzliche Zeit- oder Geldaufwand nicht tragbar oder es besteht keine Anbindung an die öffentlichen Verkehrsmittel. Aber es gibt auch die Option, auf Ridesharing auszuweichen. Dabei befinden sich auf einer Fahrt mehrere Personen in einem Fahrzeug. Dafür muss keine zusätzliche Infrastruktur errichtet werden. Außerdem müssen, nach der hier genutzten Definition, auch keine fest angestellten Fahrer bezahlt werden. Beim Ridesharing teilen sich mehrere Personen ein einzelnes Fahrzeug, um zu ihrem Ziel zu gelangen.

Da sich in einem Fahrzeug mehr Personen befinden, wird die Gesamtanzahl an Autos auf der Straße verringert. Daher scheint Ridesharing zunächst ein geeigneter Ansatzpunkt zu sein, um sowohl Treibstoffkosten pro Person als auch die gesamten Emissionen zu senken, ohne zusätzliche Infrastruktur zu errichten. Durch das teilen eines Fahrzeugs für ein Fahrt teilen sich auch die Treibstoffkosten und der Einfluss auf den ökologischen Fußabdruck auf alle Personen im Fahrzeug auf. Eine Person, in unserem Fall der Besitzer des Autos, ist dabei der Fahrer und alle anderen sind Mitfahrer. [CS11]

Nun kommt aber die Frage auf, ob, wie und wie weit sich Werte wie die Menge an benötigten Fahrzeugen und das ausgestoßene CO_2 tatsächlich verringern lassen. Das untersuchen wir hier und vergleichen die Ergebnisse mit dem Fall, dass jeder alleine zur Universität fährt.

Ridesharing wird bisher auch privat organisiert unter Freunden und Familie genutzt, aber es gibt auch Apps, die das Ridesharing mit Fremden ermöglichen. Hier ist besonders BlaBlaCar sehr bekannt[Blab]. Die bekannten Apps sind aber häufig eher für einzelne Trips ausgelegt als für regelmäßiges Pendeln. Damit solche Apps - oder andere Möglichkeiten, an Ridesharing-Programmen teilzunehmen - für Nutzer attraktiv werden, müssen sie zuverlässig sein. Das bedeutet, dass Nutzer bei Anfragen auch meistens eine

Mitfahrgelegenheit oder Mitfahrer zugewiesen bekommen. Wenn mögliche Nutzer keine oder zu wenige Matches für ihre Trips finden, werden sie das Angebot vermutlich nicht weiter nutzen. Durch die hohe Menge an Anfragen, die ein Pendler im Gegensatz zu einem gelegentlichen Nutzer stellt, kommt es schneller vor, dass auch eine Anfrage nicht zugeordnet werden kann.

Mit der Zuverlässigkeit, sowie den Auswirkungen auf Metriken wie die gesamt zurückgelegte Strecke, die benötigte Zeit oder das ausgestoßene CO_2, eines solchen möglichen Angebots beschäftigen wir uns in dieser Arbeit. Hierfür nutzen wir auf Basis von Studenten der Universität Würzburg generierte Daten, die auf den realen Daten basieren, und stellen uns die Frage, wie viele Personen dieser Gruppe ein Ridesharing-Angebot nutzen müssten, damit ein Algorithmus relativ verlässlich Matches generieren kann. Insbesondere gehen wir auf Abweichungen in der Anzahl der generierten Matches ein und analysieren Unterschiede in anderen Metriken.

Im Gegensatz zu Matching-Algorithmen, die in existierenden Apps verwendet werden, wird hier davon ausgegangen, dass die Personen ihren Stundenplan für den Zeitraum eines Semesters erhalten und dadurch das Matching schon im Voraus stattfinden und später bei Ausfällen nur noch angepasst werden kann. Durch das kurzfristige Wesen anderer Algorithmen folgt, dass auch Pendler für jede Fahrt einzeln ein Match anfragen müssten, was hier durch die statische Betrachtung nicht der Fall ist. Dies erhöht auch den Aufwand für die Teilnehmer mehr als nötig, dass es meist für einen längeren Zeitraum im Voraus einen Plan gibt, der jede Woche oder sogar jeden Tag gleich ist.

Abschließend folgt eine Analyse der Ergebnisse sowie ein Fazit zu möglichen Erweiterungen.

2 Grundlagen

2.1 Abgrenzung des Begriffs

Ridesharing

In dieser Arbeit geht es um *Ridesharing*. Dies ist hier definiert wie eine Mitfahrgelegenheit. Dabei schließen sich mehrere Privatpersonen, die eine ähnliche Strecke zu einer ähnlichen Zeit zurücklegen müssen, zusammen, um sich zum Beispiel Fahrkosten zu teilen. Der Fahrer verfolgt hier kein finanzielles Interesse, sondern würde die Strecke auch ohne Mitfahrer zurücklegen und wird durch die Kompensation lediglich entlastet. [CS11] Dabei ist eine der Privatpersonen der Besitzer des Fahrzeugs und damit hier der Fahrer. Die Zuordnung von Personen zu Autos kann privat unter Freunden und Familie oder zum Beispiel über dedizierte Apps passieren[FDO+13]. Hierbei ist es grundsätzlich nicht relevant, welchem Zweck die Fahrt dient.

Es ist wichtig die verschiedenen Begriffe genau voneinander abzugrenzen, da viele der Begriffe ähnlich klingen und somit licht zu verwechseln sind, aber sehr unterschiedliche Gegebenheiten beschreiben. Andere Begriffe für Ridesharing sind im britischen Raum auch *car sharing* oder *liftsharing*, wobei *carsharing* und *car-sharing* ein anderes Konzept bezeichnen, das hier später in Abschnitt 2.1 beschrieben wird[CS11]. Dass hier ein Bindestrich, beziehungsweise ein Leerzeichen, die Bedeutung so weit verändert, unterstreicht, wie wichtig eine klare Abgrenzung des verwendeten Begriffes ist, um Missverständnisse zu vermeiden.

Man kann hier auch einige Unterkategorien definieren. So kann man in privates und organisiertes Ridesharing unterteilen. Diese Unterteilung wurde von Furuhata et al. [FDO+13] genutzt. Privates Ridesharing findet unter Freunden, Kollegen und Familie statt. Zum Beispiel können Freunde oder Kollegen, die sich gut kennen, gemeinsam zur Arbeit, Schule oder Uni fahren. Für manche Arten von Trips, wie zum Beispiel Fahrten zu Bars oder Clubs, wird dies schon oft genutzt. Das könnte daran liegen, dass durch den Alkoholkonsum für viele ein größerer Anreiz vorliegt, nicht selbst zu fahren. Durch das Ridesharing muss nur eine Person der Gruppe auf Alkohol verzichten. Allgemein hat privates Ridesharing den Vorteil, dass man die Gruppe kennt und sich flexibler absprechen kann. Allerdings fällt dadurch die Anzahl an möglichen Fahrern oder Mitfahrern deutlich kleiner aus. Organisiertes Ridesharing findet über dedizierte Apps oder Seiten statt. Hierbei können sich Menschen, die sich nicht kennen, zu Gruppen zusammenschließen. Sie müssen meist nur Start- und Zielort und die gewünschten Zeiten angeben und ein Algorithmus übernimmt das Matching. Der große Vorteil hierbei liegt in der Einfachheit

für den Nutzer, dass er nichts selbst planen und besprechen muss, sondern nur seine Fahrt einmal angeben, und in der größeren Menge an möglichen Matches. Als Nachteil ist hier zu nennen, dass man mögliche (Mit-)Fahrer nicht kennt, wodurch sich viele unwohl fühlen könnten. Zwischen diesen beiden Kategorien liegt eine Situation, in der sich Menschen, die sich nicht persönlich kennen, über nicht dedizierte Seiten wie Foren absprechen und so eine Gruppe für Ridesharing bilden. Dies vereint die Nachteile der fehlenden Einfachheit und der Tatsache, dass man die anderen Personen nicht kennt, mit dem Vorteil der größeren Zielgruppe, die über den eigenen Freundeskreis hinausgeht. [FDO+13]

Außerdem kann man eine Einteilung anhand der Häufigkeit der Nutzung vornehmen. Diese Aufteilung wurde ebenfalls von Furuhata et al.[FDO+13] vorgeschlagen. Pendler nutzen den Dienst für die Strecke zur Arbeit oder zurück bis zu mehrmals täglich, wobei bei gelegentlichem Ridesharing die gleiche Strecke deutlich weniger häufig genutzt wird und es oft um verschiedene Strecken geht. Außerdem ist besonders bei Pendlern die Flexibilität im Hinblick auf die Ankunfts- und Abfahrtszeit sehr klein und die Strecken sind im Durchschnitt kürzer. Zudem sind die Pläne bei Pendlern oft schon im Vorfeld für einen längeren Zeitraum bekannt und regelmäßig, wobei gelegentliche Nutzer entweder nur eine einzelne Reise weit im Voraus planen oder spontan nach Mitfahrgelegenheiten suchen. [FDO+13]

Eine andere Möglichkeit, um Ridesharing in mehrere Kategorien aufzuteilen, ist technischer und bezieht sich auf die beim extern organisierten Ridesharing genutzten Matching-Algorithmen. Diese können dynamisch sein oder nicht. Bei dynamischen Algorithmen erfolgt ein Matching, sobald eine Anfrage ankommt. Diese Version eignet sich besonders für gelegentliches Ridesharing. Da Pendler meist im Voraus schon einen Zeitplan vorliegen haben, ist es hier möglich, die Anfragen zu sammeln und dann ein Matching unter Berücksichtigung aller Anfragen durchzuführen, das dann aber auch bei Änderungen noch angepasst werden kann. Dadurch kann ein besseres Matching zustande kommen, da die Eigenschaft der Dynamik aufgrund der Gegebenheiten nicht in vollem Maße benötigt wird. Viele Artikel beschäftigen sich mit dynamischen Algorithmen[NR16] [PXZ+15] [SCZ20a], da in der Praxis ein zumindest teilweise dynamischen Ansatz nötig ist, um die Gegebenheiten abzubilden, wobei Sun et al.[SCZ20a] anmerken, dass jeder Algorithmus für das dynamische Ridesharing-Problem auch das statische lösen kann und, dass der dynamische Fall lösbar ist, indem ein Algorithmus für den statischen Fall bei jeder Änderung erneut genutzt wird. Daher beschäftigen sie sich gleichzeitig mit dem dynamischen und dem statischen Ansatz.

Auch die Idee, Unterkategorien anhand der zusammen gefahrenen Strecke festzulegen, ist bei Furuhata et al.[FDO+13] zu finden. So können die Teilnehmer die ganze Strecke zusammen fahren. Das kann man als *identisches Ridesharing*[FDO+13] bezeichnen. Außerdem besteht die Möglichkeit, auch Mitfahrer mit einzubeziehen, deren Ausgangs- und Zielort auf der Strecke des Fahrers liegen, was man als *einschließendes Ridesharing*[FDO+13] bezeichnen kann. Als Drittes besteht die Möglichkeit, das System so zu gestalten, dass Passagiere verschiedene Teile ihrer Strecke mit verschiedenen Fahrern zu-

rücklegen können. Hier liegt nur *partielles Ridesharing*[FDO+13] vor. In Fall vier, dem Ridesharing mit Umweg, nimmt ein Fahrer auch Umwege zum Ziel in Kauf, um einem Mitfahrer entgegenzukommen.[FDO+13] Im beispielhaften Straßennetz in Abbildung 2.1 könnten im ersten Fall nur Teilnehmer zusammen reisen, wenn sie beide zum Beispiel von A nach B, von A nach C oder von E nach F wollen. Im zweiten könnte auch der Fahrer von A nach B wollen und ein Mitfahrer zum Beispiel von A nach D oder von C nach D. In Fall drei könnte ein Mitfahrer die Strecke von E nach D auch zum Teil bei einem Fahrer zurücklegen, der von E nach F fährt. Der Rest der Strecke könnte dann mit einem Fahrer gefahren werden, der von A nach B will, wenn die Zeiten übereinstimmen. Im vierten Fall könnte auch ein Fahrer, der von A nach C fährt, einen Mitfahrer, der von F nach C kommen möchte, mitnehmen, indem er einen Umweg fährt. In dieser Arbeit

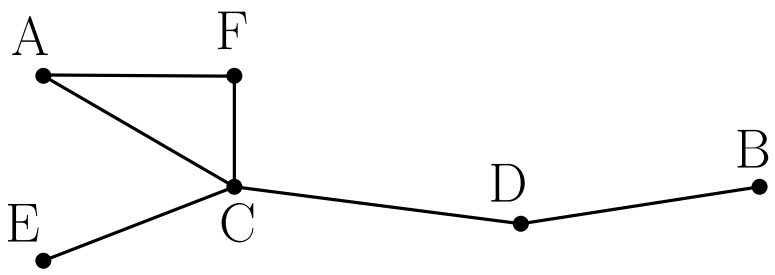


Abb. 2.1: Straßennetz Beispiel

werden wir uns auf das extern organisierte Ridesharing für Pendler (insbesondere Studenten) konzentrieren, das einen nicht dynamischen Matching-Algorithmus nutzt. Dabei müssen Ausgangs- und Zielort des Mitfahrers auf der Strecke des Fahrers liegen, können aber auch übereinstimmen.

Allerdings gibt es auch einige weitere ähnliche Begriffe, die leicht verwechselt werden können. Besonders wenn man im Internet nach Anbietern für Ridesharing sucht, werden oft auch eigentlich unpassende Ergebnisse für den Suchbegriff "Ridesharing" angezeigt [1]. Darunter fallen unter anderem Apps, um Gruppentaxis zu buchen. Diese fallen in andere Kategorien, die wir im Folgenden aufzählen werden. Zudem werden wir sie erklären und ihre Unterschiede zum Ridesharing aufzeigen. Dadurch soll verdeutlicht werden, um welches Konzept es in dieser Arbeit geht und um welche explizit nicht.

Carpooling

Der Begriff des *Carpoolings* wird oft synonym verwendet, aber auch teilweise als vollkommen separater Begriff abgegrenzt. Generell ist es eher eine Unterkategorie. Hier wird ein Auto von einer Gruppe Menschen genutzt, die gemeinsam die gleiche Strecke fahren[FDO+13]. Soweit ist es unserer Definition des Ridesharings sehr ähnlich. Allerdings wechseln sich beim Carpooling die Fahrer oft regelmäßig ab, wohingegen wir keinen besonderen Wert darauf legen, dass Teilnehmer beide Rollen gleichmäßig übernehmen.

[1] Suche in Google und Google Scholar, Suchbegriff: Ridesharing

Außerdem bleiben die Gruppen generell täglich die gleichen, sodass am Ende eine feststehende Gruppe mit regelmäßig wechselndem Fahrer entsteht.

Hier gibt es auch die Unterkategorie des *Casual Carpolings*. Dabei wird die Fahrt nicht vorher geplant, sondern Mitfahrer spontan auf der Reise aufgelesen. Das geschieht an bestimmten Orten wie Parkplätzen oder Bushaltestellen. [Kel07] [CS11]

Ridehailing

Abseits davon gibt es unter anderem das *Ridehailing*, manchmal auch *Ridesourcing* genannt, bei dem sich ein oder mehrere Personen ein Taxi bestellen. Hierbei besteht für den Fahrer der Anreiz darin, Geld zu verdienen. Der Fahrer wird entweder von den Mitfahrern direkt oder von einer Organisation, bei der er angestellt ist, dafür bezahlt, diese zu ihrem Ziel zu bringen.[Tir20] Dies wird bei mehreren Mitfahrern auch manchmal als Ridesharing bezeichnet[JCMW18]. Bei unserer Definition von Ridesharing hingegen besteht ein eigenes Interesse aller Personen, inklusive des Fahrers, die Strecke zurückzulegen und der Fahrer tut dies nicht, um Geld zu verdienen. Ob beim Ridehailing das Auto dem Fahrer gehört, ist irrelevant.

Carsharing

Zum anderen gibt es auch den ähnlichen Begriff des *Carsharings*. Dabei gehört ein Auto mehreren Personen und steht diesen jeweils nach Absprache für ihre privaten Fahrten zur Verfügung. Die Kosten für das Auto werden somit geteilt. Allerdings kann es dann natürlich nur eine Person gleichzeitig nutzen (außer das Ziel ist zufällig das gleiche, was aber dann auch wieder unter privates Ridesharing fällt). [SSW98]

Ridepooling

Ridepooling, auch als *Dial-a-ride* bekannt[LQTF24], trägt ebenfalls einen ähnlichen Namen, ist aber in der Definition etwas weiter entfernt. Auch dieses Konzept wird teilweise als Ridesharing bezeichnet.[LQTF24] Es liegt zwischen öffentlichen Verkehrsmitteln und Ridehailing. Der Fahrer muss eine vorgegebene Route abfahren, wie bei öffentlichen Verkehrsmitteln. Diese Route wird aus den Anfragen der Kunden zusammengesetzt. Die von diesen spezifizierten Start- und Zielorte müssen in die Route eingebunden werden. Ansonsten sind auch Abkürzungen möglich. [CL07]

2.2 Existierende Systeme

Viele Systeme sind regional begrenzt, aber es gibt auch internationale Angebote. Die meisten Systeme sind eher auf einzelne Fahrten ausgelegt, welche Pendler natürlich trotzdem nutzen können. Aber es gibt auch auf Pendler fokussierte Apps oder Webseiten.

Zuerst betrachten wir einige internationale Plattformen. Viele der Optionen, die bei einer Suche im Internet auftauchen, fallen nach unserer Definition eigentlich eher unter Ridehailing, werden aber oft als Ridesharing-Anbieter bezeichnet.

So zum Beispiel Via[VT]. Bei diesem Dienst gibt es dedizierte Fahrer, die die Passagiere befördern. Via selbst stellt hierbei die Plattform zur Verfügung und Firmen, Universitäten oder andere Einrichtungen, die Passagiere befördern wollen, beschäftigen die Fahrer. Allerdings ist es ihnen auch möglich, die zusätzlich von Via übernehmen zu lassen. Die Fahrer besitzen entweder selbst ein Fahrzeug, leihen es oder bekommen es auf anderem Wege zur Verfügung gestellt. Das Ziel der Fahrer ist es hierbei Profit zu machen, was im Gegensatz zu unserer Definition von Ridesharing steht. Das resultierende System wird vermutlich meist lokal genutzt, ist aber darin nicht im Allgemeinen beschränkt.

Die in Deutschland scheinbar am weitesten verbreitete App für Ridesharing nach unserer Definition ist BlaBlaCar[Blab], die alleine im Google Play Store schon über 50 Millionen Downloads hat[Blaa]. Mit der App können Personen als Fahrer eine Mitfahrgelegenheit anbieten, oder als Mitfahrer nach Fahrern suchen. Daraufhin kalkuliert die App eine Summe, die die Mitfahrer zahlen. Diese Summe soll nur einen Teil der Kosten des Fahrers decken. Sie wird so gewählt, dass der Fahrer keinen Profit machen kann.[Blad] Allerdings gibt es inzwischen weitere Kosten für Mitfahrer, die das Programm selbst finanzieren soll[Blac]. Laut Rezensionen führt die schon zu Problemen, wobei die App nur noch zum Vernetzen benutzt wird, die Bezahlung aber extern durchgeführt wird.

Das zeigt eine weitere Herausforderung für unabhängige Ridesharing-Dienste. Hier muss eine Balance gefunden werden, bei der sich die Dienste selbst finanzieren können, Nutzer aber nicht genug finanziell belastet werden, dass sie das Angebot nicht mehr nutzen.

Die App ist allerdings international nutzbar. So sind viele Städte in Europa und den USA als Start- oder Zielorte wählbar. Auch Fahrten, die Ländergrenzen überqueren, sind möglich.

Es gibt außerdem die ADAC Pendlernetz-App[ADA]. Diese ist wie aus dem Namen ableitbar auf Pendler ausgelegt. Sie bietet sogar die Option an, sich für eine Strecke als Fahrer, Mitfahrer oder flexibel Matches anzeigen zu lassen. Man gibt dann Start- und Zielort, sogar mit möglichen vorgeplanten Zwischenstopps, an. Außerdem kann man eine maximal tolerierte Zeit für Umwege angeben. Auch hier gibt es die Möglichkeit, die Fahrtkosten unter Fahrer und Mitfahrern aufzuteilen.

Ein weiteres in Deutschland verfügbares Angebot für Pendler ist das Pendlerportal[fWA]. Dort kann man entweder nach Fahrten suchen oder selbst welche einstellen. Dabei gibt man Start- und Zielort, so wie die gewünschte Zeit an. Es wird geraten, erst nach bestehenden Angeboten zu suchen, bevor man ein eigenes Angebot einstellt. Dabei ist das Ganze kostenlos, aber erst ab 18 Jahren uneingeschränkt nutzbar. Allerdings übernimmt die Plattform explizit keine Verantwortung, falls sich ein Teilnehmer gegenüber einem anderen falsch verhält. Dies könnte das Sicherheitsgefühl beeinträchtigen.

Als letztes Beispiel folgt eine weitere deutsche Plattform, die unter fahrgemeinschaft.de[rG] oder ride2go bekannt ist. Bei dieser kann man neue Angebote erstellen oder Angebote annehmen. Als Mitfahrer gibt man seinen Start- und Zielort sowie das Datum ein. Daraufhin wird eine Liste kompatibler Angebote angezeigt. Diese enthalten auch einige zusätzliche Informationen, wie mögliche Gepäckmitnahme oder auch ob das Rauchen unerwünscht ist. Es wird kein Matching durchgeführt. Der Nutzer sucht sich selbst ein Angebot aus.

2.3 Verwandte Arbeiten

Für jede Auseinandersetzung mit Ridesharing, mit dem Ziel der Optimierung oder Auswertung, wird eine Metrik benötigt, anhand derer das Ergebnis bewertet wird.

Hierfür schlagen Vincent Armant und Kenneth N. Brown[AB20] mathematisch ausformulierte Nebenbedingungen vor, mithilfe derer Ridematching für verschiedene Zielfunktionen optimiert werden kann. Diese testen sie dann für drei verschiedene Zielfunktionen: Maximierung der Anzahl an gematchten Mitfahrern, Maximierung der Anzahl an gematchten Nutzern, Minimierung der insgesamt von Autos zurückgelegten Strecke. Die Minimierung der zurückgelegten Strecke ist sehr ähnlich zur Minimierung des CO_2-Ausstoßes.

Agatz et al. [AESW11] beschäftigen sich nur mit der Minimierung der insgesamt gefahrenen Strecke, wohingegen sich Stiglic et al.[SASG16] mit der Maximierung der Anzahl an gematchten Nutzern befassen. Dabei betrachten sie den positiven Einfluss auf die Anzahl an Matches, den höhere Flexibilität der Teilnehmer erzeugt. Hier soll wiederum das Maß an zusätzlich benötigter Flexibilität, sowie die Anzahl an Nutzern, die flexibler sein müssen, minimiert werden.

Lyu et al. [GL23] schlagen einen Algorithmus vor, um den Gewinn einer Plattform, die Ridematching anbietet, zu maximieren und gleichzeitig Mitfahrer eher mit besser bewerteten Fahrern zu matchen. Im Gegenzug ist die Wartezeit etwas höher als die anderer Algorithmen, mit denen sie ihr Ergebnis vergleichen.

Dagegen betrachten Neda Masoud und R. Jayakrishnan [MJ17] das Problem aus der Perspektive der Mitfahrer. Dafür soll eine Funktion minimiert werden, die sich aus einer linearen Kombination von Fahrtzeit, Wartezeit und einer Zahl, die beschreibt, wie oft Passagiere umsteigen müssen, zusammensetzt.

De Carvalho et al.[dCG22] berücksichtigen einige verschiedene Metriken und berücksichtigen dabei sogar individuelle Präferenzen der Nutzer. Dabei betrachten sie ebenfalls Systeme, die nicht in unsere Klassifikation von Ridesharing fallen, da Fahrer und Plattform ihren Gewinn maximieren wollen. Das Ziel der Nutzer ist es, ihre Kosten zu minimieren. Die Berücksichtigung von Präferenzen soll die Zufriedenheit der Kunden erhöhen.

Nutzerzufriedenheit ist auch das Ziel, das Levinger et al. [LHA20] verfolgen. Hierfür

nutzen sie ein Deep Learning Model, das mit dem Ziel trainiert wurde, Matches zu erzeugen, die in einer möglichst hohen Gesamtzufriedenheit resultieren.

Durch die spontane Natur vieler Ridesharing-Services beschäftigt sich ein großer Teil der Artikel mit dynamischen Algorithmen. Generell können alle dynamischen Algorithmen auch das statische Problem lösen. Ein dynamischer Algorithmus wird dann benötigt, wenn Teilnehmer kurzfristig Anfragen stellen, die dann sofort gematcht werden. Dafür vergleichen Nourinejad und Roorda[NR16] einen heuristischen zentralisierten Algorithmus mit ihrem auf Agenten basierenden System. Sie zeigen hiermit, dass auch Ansätze, die nur lokal statt global die besten Entscheidungen treffen, trotz geringerem Rechenaufwand ähnlich erfolgreich sein können.

Einen etwas anderen Ansatz verfolgen Pelzer at al.[PXZ+15], die zunächst die möglichen Ursprungsorte in Gruppen einteilen. Am Ende wird das gesamte betrachtete Straßennetz so eingeteilt, dass nur innerhalb einer Gruppe nach Matches gesucht werden muss. Daraufhin verwenden sie einen Matching-Algorithmus, der ebenfalls auf Agenten basiert.

Heuristische Algorithmen sind auch eine häufiger vorkommende Lösung. Dabei ist die Lösung nicht unbedingt die beste Lösung, aber die Algorithmen sind dafür schneller. So verwenden zum Beispiel Hurimi et al.[MYSJ19] und Haferkamp et al.[SCZ20b] im Allgemeinen solche Algorithmen.

Der dritte prävalente Ansatz ist die ganzzahlige Optimierung. Dabei werden optimale Lösungen gefunden, indem das Problem als ganzzahliges Optimierungsproblem formuliert und mit IP-Solvern gelöst wird. Diese Lösung ist bei richtiger Formulierung eine optimale Lösung. Baldacci et al.[BMM04] nutzen einen solchen Ansatz mit zwei verschiedenen Formulierungen, um Matchings für Carpooling zu finden. Außerdem schlagen sie einen darauf basierenden heuristischen Algorithmus vor, der nicht exakt, aber schneller ist.

In ihrem Artikel über Peer-to-Peer Ridesharing [SCZ20a], das unserer hier genutzten Definition von Ridesharing entspricht, beschäftigen sich Sun et al. mit Lösungen für dynamische und statische Fälle von Ridesharing. Dabei ist ihr Ziel, die summierten Kosten für Fahrer und Mitfahrer im gesamten System zu minimieren. Die Algorithmen können aber abgewandelt auch mit anderen Metriken genutzt werden. Für das statische Problem schlagen sie zwei Algorithmen vor. Für den exakten Algorithmus generieren sie mögliche Routen und formulieren das Problem als ganzzahliges Programm, wie ein Mengenverpackungsproblem. Daraufhin lösen sie es mit einem IP-Solver. Der heuristische Ansatz nutzt dieselbe Idee, generiert aber nicht alle Möglichkeiten. Für den dynamischen Ansatz schlagen sie zwei verschiedene vor. Bei einer wird der statische Algorithmus genutzt, um die Lösung zu generieren, welche dann sofort angewendet wird. Für den anderen wird zuerst geprüft, ob eine Menge von Fahrern und Mitfahrern auch erst im nächsten Durchlauf betrachtet werden kann. Sonst wird die Lösung auch hier direkt umgesetzt.

In dieser Arbeit geht es ebenfalls um ein Modell, bei dem bereits in Vorfeld alle

Angaben bekannt sind, das aber auch ein kleines Maß an Dynamik bieten soll.

Liu et al. beschäftigen sich in ihrem Artikel [LTY+20] mit dem Carpooling für Pendler. Dabei stellen sie fest, dass höhere Flexibilität zwischen den Rollen des Fahrers und Mitfahrers für eine deutliche Verbesserung in der Anzahl an Matches sorgt. Ebenso führt eine länger in Kauf genommene zu Fuß zurückzulegende Distanz für Mitfahrer zu mehr Matches. Außerdem haben Anreize für Carpooling eine kleinere positive Auswirkung auf die Nutzung als zusätzliche Kosten für nicht-Nutzer. Wenn das Matching für das Carpooling ober einen Anbieter angeboten wird, dann wirken sich die Kosten für Mitfahrer auf die Anzahl an Matches aus.

Pan et al.[PLH19] haben eine Umgebung entwickelt, in der verschiedene Algorithmen getestet werden können. Dafür stellen sie einige Benchmarks zur Verfügung, damit verschiedene Szenarien getestet werden können. Diese enthalten verschiedene Straßennetze und Zahlen von Fahrern und Anfragen.

2.4 Fragestellung

Situation

Grundsätzlich soll es hier um Studierenden-Ridesharing gehen. Dabei werden wir fiktive Teilnehmer ein Ridesharing-Programm mit dem Zielort *Universität Würzburg* nutzen lassen. Nutzer geben einen Start- und einen Zielort an, sowie eine gewünschte Ankunfts- und Abfahrtszeit am Zielort. Der Startort kann frei gewählt werden und als Zielort wird einer der Universitätsstandorte in Würzburg ausgesucht.

Bei der Zeit gibt es eine Toleranz x von einigen Minuten, wodurch sich ein Zeitintervall von $2x$ ergibt. Es wird angenommen, dass auch eine Ankunftszeit von x Minuten früher oder später akzeptiert wird. Für die Abfahrtszeit ist nur ein bis zu $2x$ Minuten späterer Zeitpunkt akzeptabel, aber keine frühere Abfahrt. Das bedeutet, wenn ein Nutzer für eine Vorlesung von 10:15 bis 11:45 am Universitätsstandort sein muss, dann gibt dieser Nutzer bei einer Toleranz von $x = 15$ Minuten 10:00 als präferierte Ankunftszeit an. Wenn er frühestens um 11:45 wieder abfahren kann, so gibt er das als präferierte Abfahrtszeit an. Daraufhin wird nach anderen Agenten gesucht, die in den Zeiträumen 9:45-10:15 oder 11:45-12:15 ankommen beziehungsweise abfahren. Wenn ein solcher Agent gefunden wird, dann wird er dem ersten als mögliche Mitfahrgelegenheit zugeordnet. Eine solche Zuordnung bezeichnen wir als (zunächst als mögliches) *Match*. Nur, wenn auch ein mögliches Match für die Heimreise gefunden wird, kann eines für die Anreise gesucht werden. Matches können dann zustande kommen, wenn innerhalb des Zeitintervalls ein anderer Teilnehmer, wie in Abschnitt 2.1 unten erklärt, eine ähnliche Strecke zurücklegen möchte. Das bedeutet, dass sowohl Start- als auch Zielort des Mitfahrers auf der Strecke des Fahrers liegen oder mit denen des Fahrers übereinstimmen müssen. Der kleinste Abstand zwischen einem der festgelegten Uni-Standorte und der Strecke eines Fahrers, der diesen Standort nicht als Ziel hat, beträgt etwa 370 m. Daher gehen wir für den Standardfall davon aus, dass die Zielorte übereinstimmen.

Ziel

Die Idee hinter dem Algorithmus ist, dass möglichst viele Anfragen zu einem Match für den Nutzer führen, beziehungsweise dass möglichst wenige Fahrer alleine fahren sollen, wobei wir gleichzeitig auch die Anzahl der genutzten Fahrzeuge und damit die von allen Fahrzeugen insgesamt zurückgelegte Strecke verringern wollen, ohne Umwege der Fahrer zu erlauben. Durch einer Verringerung der gesamt gefahrenen Strecke, wird gleichzeitig auch die Menge an ausgestoßenem CO_2 verringert. Dieses Kriterium der wenigen einzelnen Fahrer haben wir für die Auswertung uns als eines der Ziele gewählt, da es aus Sicht der Nutzer eines Services das auffälligste ist, aber in der Literatur bisher weniger repräsentiert zu sein scheint. Dort werden häufiger Metriken betrachtet, die die Leistung des gesamten Systems betreffen, wie die gefahrene Distanz, die Fahrzeuganzahl oder der Kraftstoffverbrauch [AESW11][GL23]. Es fällt einem einzelnen Nutzer sehr deutlich auf, ob eine Anfrage dazu führt, dass das Auto geteilt wird. Gesamtheitliche Kriterien sind für sie hingegen nicht abschätzbar. Für einen potenziellen Service mit dynamischer Nachfrage ist es wichtig, dass es genug Nutzer gibt, da durch eine verringerte Nachfrage weniger Matches zustande kommen, wodurch sich die Anzahl an Nutzern wieder verringern könnte. Deshalb ist es zunächst wichtig zu sehen, wie viele Matches überhaupt zustande kommen, beziehungsweise welcher Prozentsatz an Anfragen nicht zu einem Match führt. Diese Auswertung bietet die Grundlage für ein mögliches Experiment mit sich dynamisch ändernder Nachfrage. Dafür könnte man festlegen, dass unzufiredene Nutzer den Service nicht mehr verwenden - oder ihn im Gegenteil, wenn sie zufrieden sind, sogar weiterempfehlen. Wir nehmen an, dass dafür ein hohes Maß an Zuverlässigkeit gegeben sein muss, was sich durch eine hohe Matchingrate zeigt. Da sich das nicht gut quantifizieren lässt, wählen wir hier frei einen Wert von 90% gematchten Anfragen. Wir möchten den Punkt, beziehungsweise die Anzahl an Nutzern, finden, an dem wir ein solches Ergebnis von etwa 90% gematchten Anfragen erreichen. Dabei ist es nicht relevant, wie viele Personen sich im Fahrzeug befinden. Da wir hierfür eine geringe Anzahl an einzelnen Fahrern vorliegen haben müssen, führt es für diese Auswertung zu besseren Ergebnissen, wenn zum Beispiel drei Passagiere auf drei Fahrzeuge verteilt sind, als wenn sich in einem Fahrzeug vier Agenten und in zwei jeweils nur ein Fahrer befinden, obwohl die gesamt zurückgelegte Strecke, sowie das gesamt ausgestoßene CO_2 und sogar der Durchschnitt der belegten Sitze weitestgehend gleich bleiben.

Aber wir werden trotzdem auch andere Aspekte auswerten, darunter CO_2-Ausstoß, beziehungsweise die gesamt gefahrene Strecke, und die benötigte Zeit, die kein primäres Ziel darstellen, sondern eher Nebeneffekte. Diese werden wir mit dem Fall vergleichen, in dem alle Teilnehmer im eigenen Auto fahren sowie mit dem Matching-Algorithmus für Ridesharing von Helena Fehler[Feh24]. Außerdem werden wir betrachten, wie viele Personen sich durchschnittlich, sowohl im gesamten Durchschnitt als auch im Maximum der jeweiligen Fahrt, in einem Auto befinden.

Annahmen und Eingeständnisse

Wir nehmen an, dass es möglich ist, einen Mitfahrer aufzulesen, wenn dessen Startort innerhalb einer gewissen Distanz zur Strecke des Fahrers liegt. Hierbei werden wir nicht berücksichtigen, ob Haltemöglichkeiten an der Strecke vorliegen.

Der Zielort eines Mitfahrers muss sich ebenfalls innerhalb eines gewissen Abstandes zur Strecke des Fahrers befinden oder mit dessen Ziel übereinstimmen, wie im Standardfall hier. Bei verschiedenen Zielorten wird für die Ankunfts- beziehungsweise Abfahrtszeit das Zeitintervall so verglichen, dass der Fahrer innerhalb des Zeitintervalls des Mitfahrers an dem Punkt ist, an dem er den Mitfahrer auflesen kann. Die Zeit, die der Mitfahrer benötigt, um vom Unistandort zu diesem Punkt zu kommen, wird dabei nicht beachtet. Dies liegt zum Teil daran, dass der für den Unistandort gewählte Ort nur ein einzelner Punkt ist, der auf einem Parkplatz liegt. So kann es durchaus sein, dass der Mitfahrer sich zufällig schon deutlich näher an der Strecke des Fahrers befindet, da die Unistandorte eine größere Fläche umfassen. Auch die benötigte Zeit, um eventuelle andere Mitfahrer zwischen diesem Punkt und dem Zielort aussteigen zu lassen, wird nicht beachtet.

Außerdem wird hier von Fahrzeugen mit fünf Sitzen ausgegangen, da diese am häufigsten anzutreffen sind. Damit können wir die gleichen Daten für alle Fahrzeuge nutzen, statt diese ebenfalls zu randomisieren.

Eine wichtige Annahme ist auch, dass alle Agenten nur die Fahrt zur Universität als Mitfahrer antreten würden, wenn auch für die Fahrt nach Hause eine Mitfahrgelegenheit zur Verfügung steht. So wird das Problem der verlorenen Studenten vermieden, das sich durch den Ridesharing-Matching-Algorithmus bei Helena Fehler ergeben hat.[Feh24] Dabei war ein Student gezwungen, eine alternative Transportmethode für die Heimreise zu nutzen. Aber auch ein vollständiges Ausschließen dieser Option bildet nicht die Realität ab, da es durchaus möglich ist, dass ein Student bereit ist, eine der Fahrten mit einem anderen Transportmittel zurückzulegen.

Offensichtlich müssen Fahrer immer für beide Strecken als Fahrer zugeordnet sein. Das gilt analog auch für die Mitfahrer.

Auch wird hier davon ausgegangen, dass jeder Student im Datensatz ein Auto besitzt, und damit die Möglichkeit hat, die Rolle des Fahrers oder eines Mitfahrers zu füllen. Dazu passend wird die Annahme getroffen, dass die Studenten keine Präferenz haben, sondern tatsächlich ohne Auswirkungen frei zuteilbar sind. Dabei werden die Agenten auch zufällig auf die Autos aufgeteilt. Allerdings ist davon auszugehen, dass so in der Realität Matches entstehen würden, die persönlichen Präferenzen der Agenten widersprechen. Das könnte zum Beispiel passieren, wenn ein Nichtraucher als Passagier einem Fahrer zugeteilt wird, der raucht.

Des Algorithmus löst eine statische Version des Ridesharingproblems. Das bedeutet, dass der Plan jedes Studenten im Voraus bekannt ist und das Matching nur einmal durchgeführt wird. In der Realität müsste man eher davon ausgehen, dass die Agenten zwar ihren Plan kennen, es aber immer zu kurzfristigen Änderungen kommen kann, die berücksichtigt werden müssen. Wenn ein Mitfahrer ausfällt, dann wirkt sich das nur auf

die Statistik aus, aber wenn ein Fahrer ausfällt oder ein zusätzlicher Agent hinzukommt, müssen zusätzlich Rollen angepasst werden.

Als Letztes wird vom hier gewählten Algorithmus die Strecke zwischen Start und Ziel nur einmal für den Hinweg berechnet. Hier könnte es allerdings sein, dass die Strecke für den Rückweg so weit abweicht, dass manche Matches nicht mehr zustande kommen würden. Abweichungen in der Route könnten zum Beispiel dann entstehen, wenn Abfahrten von größeren Straßen in der Route genutzt werden. Wenn dabei eine der Abfahrten gerade außerhalb der Reichweite ist, könnte die in die andere Richtung trotzdem nah genug sein. Da gleiche gilt für Routen mit Einbahnstraßen. Durch die Nutzung einer Einbahnstraße in eine Richtung könnten Matches zustande kommen, die in die andere Richtung nicht möglich sind. Da wir aber davon ausgehen, dass das nur in wenigen Einzelfällen passiert, nutzen wir hier nur die Route für eine Richtung, genauer die vom Wohnort zum Universitätsstandort. Für den Rückweg verwenden wir für Matches und Zeit- und Kilometerangaben dieselbe Route.

3 Methodik

Im Folgenden werden wir unser Vorgehen erläutern. Dabei werden wir zunächst auf das Framework[Feh24] eingehen, innerhalb dessen der Algorithmus läuft und danach den Algorithmus selbst genauer erklären.

3.1 Framework

Der Algorithmus, den wir implementieren, läuft innerhalb eines von Helena Fehler für ihre Masterarbeit entwickelten Frameworks zum Testen von verschiedenen Algorithmen für Ridesharing und ähnliche Probleme[Feh24]. Die verwendete Version ist hierbei nicht die aktuellste, sondern ist auf dem Stand vom 03.06.2024. Wir werden die für und wichtigen Aspekte dieses Frameworks in diesem Abschnitt erläutern.

Wir übernehmen die vorgegebene Klassenstruktur und Konfiguration. Die meisten Klassen sind für die eigene Implementierung nicht direkt relevant. In der Konfiguration finden sich Koordinaten für die Universitätsstandorte und ein Wert für die Distanz, die ein Agent maximal zu Fuß zurücklegen würde. Die Distanz wird hier so verwendet, dass sie die maximale Distanz darstellt, die ein Agent zu Fuß von seinem Wohnort zur Route eines anderen Agenten zurücklegen würde, um aufgesammelt zu werden. Die Distanz stellt dabei die einfache Strecke dar. Die Distanz für den Weg von der Strecke eines Fahrers bis zum Unistandort eines Mitfahrers wird gesondert betrachtet.

Dem Algorithmus wird eine Liste von Agent-Objekten übergeben, die jeweils einen Start- und Zielort als Koordinaten, sowie gewünschte Ankunfts- und Abfahrtszeiten mitbringen. Als Ausgabe wird eine Liste von Ride-Objekten benötigt, die jeweils einige Attribute, wie Ankunftszeit und Abfahrtszeit, Start- und Zielort, sowie beteiligte Agenten besitzen.

Dieser Algorithmus selbst wird in einer Klasse (*RidesharingStatic*) geschrieben, die eine Implementierung des Interfaces *Mobility_Mode* ist. Dieses hat einige zu implementierende Methoden. Die Methode *prepareMode()* soll den Algorithmus vorbereiten. Wir führen darin eine Methode aus, die für jeden Agenten A jeweils Listen für die Hin- und Rückfahrt erzeugt, die alle anderen Agenten enthalten, bei denen A als Passagier mitfahren könnte. Der eigentliche Matching Algorithmus wird dann auf Basis dieser Listen in *startMode()* durchgeführt, was aber eher für dynamische Algorithmen relevant ist. Weitere vorgegebene Methoden sind *returnRides()*, in der die fertigen Ride-Objekte zurückgegeben werden und *writeResultsToFile()*, wodurch das Ergebnis als .csv Dateien ausgegeben wird. Wir haben noch eine weitere Datei als Ausgabe hinzugefügt. Diese ist

ein einfaches Textdokument, das noch zusätzliche Informationen enthält.

Außerdem haben wir eine Veränderung zur Klasse *AgentManager*, die die Agenten aus einer Datei einliest, hinzugefügt, mit der wir ungefähr einen festgelegten Prozentsatz an Agenten einlesen können.

Zudem stehen im Framework einige Hilfsmethoden in der Klasse *CommonFunctionHelper* zur Verfügung. Wir haben die zwei dieser Methoden direkt genutzt. Eine der beiden Methoden (*isOverlapping(start Zeitraum1, ende Zeitraum1, start Zeitraum1, ende Zeitraum2)*) prüft, ob sich zwei Zeitintervalle schneiden. Diese nutzen wir, um Überschneidungen in den Ankunfts- und Abfahrtszeiten zu finden. Die andere Methode, *getSimpleBestGraphhopperPath(graphhopper, Koordinate 1, Koordinate 2)*, findet eine beste Fahrtroute zwischen zwei Punkten.

Für eine spätere Version des Frameworks werden mehr Optionen für die Ausgabe zur Verfügung stehen. Allerdings haben wir hier die Methode *writeResultsToFile()* genutzt, um die Ergebnisse so auszugeben, dass wir alle benötigten Informationen daraus ziehen können. Dafür haben wir als Basis die gleichnamige Methode aus der Klasse *RideSharing* genutzt und so abgeändert, dass auch Werte, die wir anders gespeichert haben, korrekt ausgegeben werden.

Das Framework nutzt GraphHopper[gh] zum Routen. Dabei nutzt GraphHopper Daten von OpenStreetMap. Mit GraphHopper können wir Routen finden, die entweder für einen Fußgänger oder für ein Auto berechnet werden.

Die Daten für Agenten und Anfragen. wurden auf der Basis von Daten von Studenten der Universität Würzburg generiert und es liegen mehrere Mengen von Agenten und Requests vor, die andere Verteilungen von Wohnorten und Zeiten besitzen, um zu vermeiden, dass ein einzelner Datensatz zufällig zu besseren oder schlechteren Ergebnissen für einen Algorithmus führt.

3.2 Mögliche Matches

Diese Methode, im Code mit *createPossibleLists()* benannt, erzeugt jeweils eine Liste für den Hin- und den Rückweg für jeden Agenten. Die Listen enthalten all diejenigen anderen Agenten, bei denen der Agent in die entsprechende Richtung mitfahren könnte. Diese Liste stellen dann die Basis für den Matching-Algorithmus dar.

Als Erstes berechnen wir für jeden Agenten B eine Route von seinem Wohnort zum Unistandort, suchen alle andere Unistandorte, die innerhalb einer festgelegten Distanz zu dieser Strecke liegen, und speichern sie als $dropoffs - Liste$ von B. Wenn diese Distanz null ist, so findet sich in der dropoffs-Liste nur der Zielort des Agenten selbst.

Um mögliche Mitfahrgelegenheiten für einen Agenten A zu finden, suchen wir zunächst alle Agenten B, in deren dropoffs-Liste sich der Ankunftsort von A befindet, und deren Ankunfts- oder Abfahrtszeitraum passend ist. Das bedeutet, wenn der Zielort derselbe ist, dass sich die Zeiträume überschneiden müssen. Wenn sich die Zielorte unterscheiden,

muss sich Agent B innerhalb eines Zeitraums von Agent A innerhalb der festgelegten Distanz zum Zielort von A befinden. Für den Standardfall wird die Distanz so klein gewählt, dass nur Agenten mit gleichen Zielorten gematcht werden können. Wenn das für einen Agenten B zu trifft, wird dieser Agent als mögliches Match für A für den Hin- beziehungsweise Rückweg gespeichert, je nachdem für welche Richtung sich die Zeiten überschneiden.

Daraufhin suchen wir für jeden Agenten A innerhalb dieser Listen nach Agenten B, deren Route nahe genug, am Wohnort von A verläuft, dass A die Route unter Einhaltung der maximalen Laufstrecke erreichen kann. Dafür betrachten wir nur Agenten, die weitere oder nur um wenig kürzere Strecken bis zu ihrem Ziel zurücklegen. Hier nutzen wir einen Wert von maximal $1,66$-mal der maximalen Laufstrecke, die eine Fahrstrecke von Agent B kürzer sein darf. Diese Distanz haben wir gewählt, da wir davon ausgehen, dass eine Route zwischen zwei Punkten mit dem Auto zwar weiter sein kann als zu Fuß, aber im Regelfall nicht unbeschränkt weiter sein wird, und dass ein Agent A somit um die Laufdistanz weiter entfernt wohnen kann und damit zum Wohnort von B läuft und von dort als Mitfahrer mit zur Universität fährt. In Abbildung 3.1 ist eine Situation zu sehen, in der das relevant ist. Dabei wohnt Agent B weiter vom Universitätsstandort entfernt als A, kann aber trotzdem bei A mitfahren, da das Kriterium der maximalen Laufstrecke trotzdem erfüllt ist. Die Fahrtstrecke von B ist allerdings weiter als die von A und die maximale Laufstrecke gemeinsam. Daher wählen wir einen Faktor, bei dem wir davon ausgehen, dass in den meisten Fällen, in denen diese Situation auftritt, die zusätzliche Fahrtstrecke kleiner ist, als das Produkt des Faktors und der maximalen Laufstrecke, statt direkt die Distanz der maximalen Laufstrecke zu nutzen.

Wenn das zutrifft, betrachten wir zunächst die Luftlinie zwischen dem Wohnort von A und der Route von B. Wenn die Route einen Kreis mit dem Radius der maximalen Laufdistanz schneidet, dann berechnen wir eine Route für die tatsächliche Laufdistanz. Nur wenn diese ebenfalls klein genug ist, fügen wir B zur entsprechenden Liste von möglichen Matches für den Hin- oder Rückweg hinzu. Die tatsächliche Berechnung ist zeitaufwändig, aber trotzdem nötig, da sich sonst deutlich zu große tatsächlich gelaufene Distanzen ergeben. Nur die Luftlinie zu nutzen, erhöht zwar in unseren Experimenten[1] die durchschnittlich in eine Richtung gelaufene Distanz nur um ca. 100 m, führt aber für einzelne Agenten zu gelaufenen Distanzen von bis zu 6 km, was wir vermeiden wollen.

Dieses Vorgehen wird genauer in Algorithmus 1 beschrieben. Am Ende werden die Listen von möglichen Mitfahrgelegenheiten nach Länge sortiert, wobei die Listen für die Rückfahrt bei gleicher Länge priorisiert werden. In Abbildung 3.2 ist zu sehen, wie eine Menge von Listen richtig sortiert wird, wobei die Reihenfolge innerhalb von Ansammlungen von Listen mit gleicher Richtung und Länge theoretisch frei wählbar ist. In unserem Algorithmus hängt sie von der Sortierung der Agenten ab. Das Ergebnis dieser Sortierung ist gleichzeitig die Eingabe für den Matching-Algorithmus.

In Tabelle 3.1 findet sich eine kurze Übersicht über die im Algorithmus verwende-

[1] Experiment außerhalb der ungenaueren Distanzberechnung mit Standardbedingungen aus Abschnitt 4.1 durchgeführt

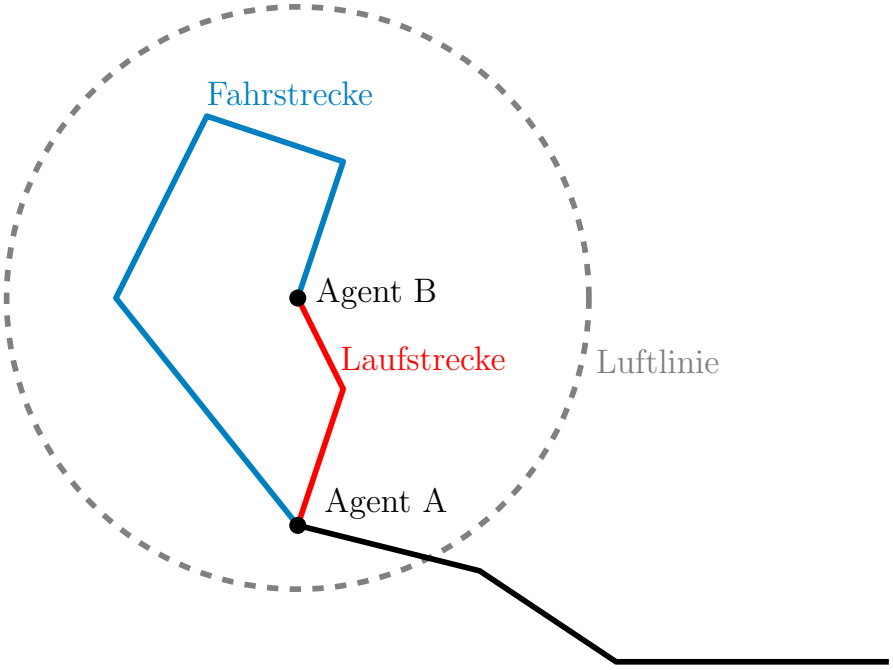


Abb. 3.1: Mögliche Situation

ten Hilfsmethoden und Variablen. Die Bezeichnungen entsprechen nicht zwingend den in der Implementierung verwendeten. Stattdessen wurden sie gewählt, um möglichst selbsterklärend zu sein. Die Tabelle beschreibt alle solchen in Algorithmen verwendeten Bezeichnungen. Hinweise darauf, wie diese Methoden und Variablem im Code implementiert sind, finden sich in Tabelle 3.2. Einige der im Pseudocode genutzten Hilfsmethoden dienen nur dessen Verständlichkeit und kommen im Code nicht vor. Die Funktionalität der Hilfsmethode ist dort an der entsprechenden Stelle direkt implementiert. Außerdem werden viele der Variablen, die grundsätzlich einfach als Attribute der Agenten zu speichern wären, (wie die Rolle als Fahrer oder Mitfahrer, Listen von möglichen Matches, der zugehörige Fahrer für die jeweilige Richtung oder die Mitfahrer) im Code, um die anderen Klassen des Frameworks[Feh24] möglichst intakt zu halten, als Map-Datenstruktur verwaltet, in denen das Attribut als Wert mit dem Agenten als Schlüssel gespeichert ist. Im Code finden sich noch viele weitere Map-Datenstrukturen, die allerdings nur dem Speichern von Daten für eine schnellere Ausgabe dienen. Diese Maps werden auch für den Matching-Algorithmus verwendet.

3.3 Matching

Hier war die erste Idee, dass weiter entfernte Studierende normalerweise mehr Passagiere auf dem Weg mitnehmen können und umgekehrt selbst von weniger anderen mitgenommen werden können. Wenn wir die Anfragen dieser Agenten zuerst verarbeiten und sie als

Tab. 3.1: Hilfsmethoden und Variablen

Methode/Variable	Erklärung
agent.home	Heimatort eines Agenten
agent.uni	Universitätsstandort, der Zielort des Agenten ist
agent.strecke	Strecke von Wohnort bis Universitätsstandort
agent.possUni	Universitätsstandorte, an denen ein Agent nahe genug vorbeikommt
agent. poss_fahrer_to	Liste mit möglichen Mitfahrgelegenheiten für einen Agenten agent zur Universität, mit _v angehängt vorläufige Liste
agent. poss_fahrer_back	Liste mit möglichen Mitfahrgelegenheiten für einen Agenten agent nach Hause, mit _v angehängt vorläufige Liste
agent. timeframe_to	• Zeitrahmen, in dem ein Agent am Ziel ankommen muss, • für Fahrer der Zeitrahmen, in dem die gesamte Gruppe ankommen kann
agent. timeframe_back	• Zeitrahmen, in dem ein Agent vom Ziel abfahren muss, • für Fahrer der Zeitrahmen, in dem die gesamte Gruppe abfahren kann
inWalkingDistance (Koordinate a, Koordinate b)	• berechnet die Distanz zwischen zwei Koordinaten und vergleicht das Ergebnis mit der maximal zu Fuß zu laufenden Strecke *acceptedWalkingDistance* • gibt **true** zurück, wenn die Distanz maximal den gleichen Wert hat
isOverlapping (Zeitintervall 1, Zeitintervall 2)	• vergleicht zwei Zeiträume • gibt **true** zurück, wenn sich die Zeiträume überschneiden
findEmptiestCar (Liste FahrerOptionen)	• findet Fahrer mit meisten freien Plätzen im Auto • vergleicht Anzahl freie Plätze in Autos der Fahrer • wenn gleiche Anzahl behält ersten Fahrer mit der Anzahl • gibt Fahrer zurück
overlap (Zeitintervall 1, Zeitintervall 2)	• findet das Zeitintervall, in dem sich die Intervalle schneiden • gibt neues Zeitintervall zurück

Tab. 3.2: Variablen im Code

Methode/Variable	Bezeichnung im Code
agent.home	agent.getHomePosition()
agent.uni	agent.getRequest().getDropoffLocation();
agent.strecke	routepoints.get(agent) in Map<Agent, List<Coordinate>> routepoints
agent.possUni	unilocations_possible.get(agent) in Map <Agent, List<Coordinate>> $unilocations_possible$
agent. poss_fahrer_to	poss.get(agent) in Map<Agent, List<Agent>> poss, mit _v angehängt poss_t.get(agent)
agent. poss_fahrer_back	possb.get(agent) in Map<Agent, List<Agent>> possb, mit _v angehängt possb_t.get(agent)
agent. timeframe_to	• für Agenten allgemein als zwei Zeitpunkte *agent.getRequest().getDepartureIntervalStart()* und *agent.getRequest().getDepartureIntervalEnd()* • für Fahrer in Liste in Map<Agent, List<LocalDateTime>> *driverTimes* mit den vier Zeiten für jeweils den Start und das Ende des Ankunfts- beziehungsweise Abfahrtsintervalls
agent. timeframe_back	wie agent. timeframe_to, aber für Agenten mit *agent.getRequest().getArrivalIntervalStart()* und *agent.getRequest().getArrivalIntervalEnd()*
inWalkingDistance (Koordinate a, Koordinate b)	Vergleich mit maximaler Distanz wird im Code direkt erledigt, Distanz wird zuerst als Luftline mit *Range (Koordinate a, Koordinate b)* berechnet, und dann, nur wenn die Distanz klein genug ist, mit *getBestWalkingPathToPoint(agent, Koordinate b).getDistance()* als Distanz zwischen a.getHomePosition() und Koordinate b verglichen, um Rechenzeit zu sparen
inUniDistance (Koordinate a, Koordinate b)	Distanz wird zuerst als Luftlinie mit *Range (Koordinate a, Koordinate b)* abgeschätzt und mit der maximal erlaubten Distanz verglichen, und dann, nur wenn die Distanz klein genug ist, mit getWalkingDistanceBetweenPoints(Koordinate a, Koordinate b) genau berechnet und verglichen, um Rechenzeit zu sparen
isOverlapping (Zeitintervall 1, Zeitintervall 2)	*CommonFunctionHelper.isOverlapping (Zeitintervall 1 start, Zeitintervall 1 ende, Zeitintervall 2 start, Zeitintervall 2 ende)*
findEmptiestCar (Liste Fahrer-Optionen)	direkt im Code gelöst durch Iterieren über mögliche Fahrer
overlap (Zeitintervall 1, Zeitintervall 2)	separat für Start und Ende eines Intervalls mit *getNewStartTime(Intervall 1 start, Intervall 2 start)* und *getNewEndTime(Intervall 1 ende, Intervall 2 ende)*

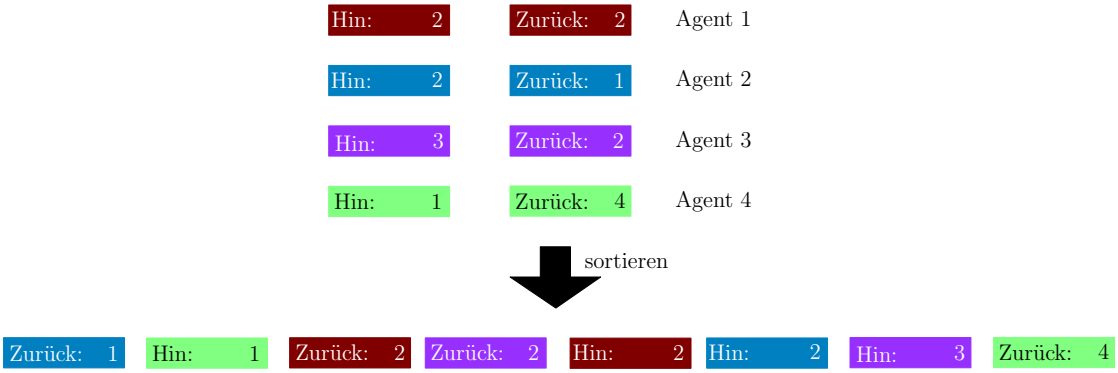


Abb. 3.2: Sortierte Listen

Fahrer einteilen, so gibt es für später betrachtete Agenten schon Fahrer, die sie mitnehmen können. Allerdings muss das nicht für alle weiter entfernt wohnenden Agenten der Fall sein. Stattdessen möchten wir mit denjenigen Agenten beginnen, die weniger mögliche Mitfahrgelegenheiten haben, da wir annehmen, dass für die mit mehr möglichen Mitfahrgelegenheiten die Chance höher ist, dass unter den möglichen Mitfahrgelegenheiten schon ein passender Fahrer zu finden ist. Auf der Basis dieser Annahme nutzen wir hier einen Greedy-Algorithmus, der der Reihe nach Agenten aus einer Liste zuordnet. Dabei ist es möglich, dass eine zufällig sehr (un-)günstige Reihenfolge das Ergebnis positiv (negativ) beeinflusst. Es wird in jedem Schritt versucht, den aktuellen Agenten als Mitfahrer einem Fahrer zuzuordnen. Wenn ein Agent schon fertig zugeordnet wurde, wird die Rolle nicht mehr abgeändert. Dafür beginnen wir beim Matching mit denjenigen Studierenden, die die wenigsten möglichen Mitfahrgelegenheiten haben. Wer keine Möglichkeit hat, mitzufahren, muss automatisch selbst fahren und steht somit den anderen sofort als Mitfahrgelegenheit zur Verfügung. So verringern wir auch die Chance, dass wir für später betrachtete Studenten keine Mitfahrgelegenheit finden, da die später betrachteten Studenten hier mehr Möglichkeiten haben. In dieser Reihenfolge betrachten wir nun jeden Agenten. Wenn keine Mitfahrgelegenheit bei einem anderen schon als Fahrer zugeordneten Agenten zur Verfügung steht, so wird der Agent selbst zum Fahrer. Wenn ein Fahrer zur Verfügung steht, wird der Agent als Passagier zugeordnet, und wenn es mehrere sind, wird er dem Fahrer zugeordnet, der aktuell die meisten freien Sitze hat. Und wenn sich auch dadurch keine eindeutige Lösung ergibt, wird der Fahrer gewählt, der zuerst betrachtet wurde. Hier bestünde die Möglichkeit, diesen Fall stattdessen genauer zu betrachten und anhand weiterer Kriterien zu wählen. Wir sind jedoch davon ausgegangen, dass der Unterschied nicht signifikant genug ist, um die Statistik merklich zu verändern. Trotzdem ist es eine zu erwähnende mögliche Erweiterung für den Algorithmus. Wenn ein Agent als Mitfahrer zugeordnet ist, muss das mögliche Zeitintervall für das Auto, beziehungsweise den Fahrer, so verändert werden, dass sowohl der Fahrer als auch der Mitfahrer im neuen Zeitintervall ankommen können.

Außerdem erzeugen wir die Ride-Objekte entgegen der Formulierung in Algorithmus 2 erst in einer weiteren Methode. Stattdessen speichern wir die vorläufigen Fahrten in

zwei Maps *Map <Agent, List<Agent>> drives_to* für den Hinweg und *Map <Agent, List<Agent>> drives_back* den Rückweg, wobei der Schlüssel der jeweilige Fahrer ist und der Wert die Liste von Passagieren. Diese Maps würden für unsere Zwecke ausreichen, aber um andere Funktionalitäten des Frameworks[Feh24] nutzen zu können, werden Ride-Objekte benötigt. Um die Kompatibilität für zukünftige Erweiterungen zu erhalten, erzeugen wir daher nach dem Matching-Algorithmus trotzdem Ride-Objekte und nutzen diese dann für die Ausgabe. Dafür nutzen wir die Maps und erzeugen für jeden Inhalt ein Ride-Objekt mit *new Ride (Coordinate heim, Coordinate uni, startZeit, ankunftsZeit, generisches Fahrzeug, Requesttype.DRIVETOUNI, List<Agent>)* beziehungsweise *new Ride (Coordinate uni, Coordinate heim, startZeit, ankunftsZeit, generisches Fahrzeug, Requesttype.DRIVEHOME, List<Agent>)* (wobei *Coordinate heim* und *Coordinate uni* den Start- und den Zielort des Fahrers beschreibt), dem wir jeweils eine Liste von Anhaltepunkten (*stop = new Stop()*) hinzufügen, die alle zusätzlichen Halte außerhalb von Start und Ziel enthält. Diesen geben wir jeweils einen Ort (stop.setStopCoordinate (Koordinate s)) und einen Haltegrund (*stop.setReasonForStopping (Stopreason.PICKUP)* oder *stop.setReasonForStopping (Stopreason.DROPOFF)*). Man kann die Ride-Objekte auch direkt im Matching-Algorithmus erzeugen, aber da wir für die korrekte Ausgabe der Ergebnisse die Haltepunkte in der korrekten Reihenfolge benötigen, haben wir uns entschieden sie erst danach zu erzeugen, um ein ständiges Zugreifen auf die Liste von Halten zum Umsortieren zu vermeiden.

Wenn dennoch zu viele Agenten spät im Matching-Algorithmus als einzelne Fahrer zugeordnet werden, besteht die Möglichkeit, das Matching durch einen weiteren Algorithmus zu verbessern, der dann nur einzelne Fahrer betrachten muss. Durch ein früheres Problem war dieser Algorithmus schon implementiert und ist auch immer noch im Code zu finden. Da die Verbesserung in der Anzahl der einzelnen Fahrer, die dieser zusätzliche Algorithmus bewirkt, nun aber im niedrigen einstelligen Bereich liegt, halten wir es nicht für notwendig, den, dem Matching-Algorithmus sehr ähnlichen, Pseudocode anzugeben, sondern erklären hier nur die Funktionsweise. Dabei ist die Zuordnung im Code dem Matching-AlgorithmusAlgorithmus 2 sehr ähnlich, betrachtet aber nur Fahrer A, die bisher in beide Richtungen alleine sind und sucht für diese, genauso wie der Matching-Algorithmus, ein Match für jede Richtung. Das heißt, wir suchen jeweils einen anderen Fahrer, der zum passenden Zeitpunkt zum passenden Ort fährt und Platz im Auto hat und fügen den Agenten dort als Mitfahrer hinzu. Diese Zuweisung als Mitfahrer funktioniert genauso, wie die im Matching-Algorithmus. Allerdings müssen wir vorher noch die Fahrten, die der Agent A aktuell als Fahrer anbietet, löschen. Aber im Gegensatz zum Matching-Algorithmus, der zwischen Agenten springt, werden direkt beide Richtungen für einen Agenten betrachtet und der Fahrer wird sofort zum Mitfahrer, sobald ein Match für beide Richtungen vorliegt. Auch gehen wir hier nicht mehr nach einer besonderen Reihenfolge vor.

In Algorithmus 2 wird der gesamte Matching-Algorithmus genauer gezeigt.

Algorithmus 1: MöglicheMatches(Agenten und requests)

Eingabe: Agenten a mit Requests
Ausgabe: Liste l mit möglichen Matches für jeden Agenten

```
for alle Agenten a do
    Strecke von a.home zu a.uni mit graphhopper berechnen
    a.strecke = Liste von Punkten i auf Strecke
for alle Agenten a do
    for alle Punkte i in a.strecke do
        for alle Universitätsstandorte u do
            if inUniDistance(i, u) then
                a.possUni+=u
for alle Agenten a do
    for alle anderen Agenten b do
        if b.possUni.containts(a.uni) AND
          isOverlapping(a.timeframe_to,b.timeframe_to) then
            a.poss_fahrer_to_t+=b
        if b.possUni.containts(a.uni) AND
          isOverlapping(a.timeframe_back,b.timeframe_back) then
            a.poss_fahrer_back_t+=b
for alle Agenten a do
    for Agenten b in a.poss_fahrer_to_t do
        for alle Punkte i in b.strecke do
            if inWalkingDistance(a.home, i) then
                a.poss_fahrer_to+=b
    for Agenten b in a.poss_fahrer_back_t do
        for alle Punkte i in b.strecke do
            if inWalkingDistance(a.home, i) then
                a.poss_fahrer_back+=b
Liste l
for alle Listen a.poss_fahrer_to do
    l+=a.poss_fahrer_to
for alle Listen a.poss_fahrer_back do
    l+=a.poss_fahrer_back
sortiere l nach Länge der Listen mit poss_fahrer_back zuerst für jede Länge
```

Algorithmus 2: MatchesFinden(Liste l)

Eingabe: Liste l mit Listen von mögliche Matches poss_fahrer_to bzw poss_fahrer_back, die jeweils einem Agenten zugeordnet sind, aufsteigend nach Länge dieser Listen sortiert

Ausgabe: Fertig zugeordnete rides

```
1  for alle poss in l mit zugehörigem Agent a
2      if poss vom Typ poss_fahrer_back then
3          if l.length==0 then
4              if a.Rolle==Mitfahrer then
5                  b = Agent, bei dem a als Mitfahrer für Hinfahrt zugordnet ist
6                  a aus aktueller Hinfahrt entfernen
7                  b.timeframe_to neu berechnen ohne a.timeframe_to
8              a.Rolle=Fahrer
9              neue Fahrt zurück mit Fahrer a
10         else
11             Liste optionen
12             for alle Agenten b in l do
13                 if b==Fahrer AND b hat Platz im Auto AND
                    isOverlapping(a.timeframe_back,b.timeframe_back) then
14                     optionen.add(b)
15             if optionen.length==0 then
16                 if a.Rolle==Mitfahrer then
17                     b = Agent, bei dem a als Mitfahrer für Hinfahrt zugordnet ist
18                     a aus aktueller Hinfahrt entfernen
19                     b.timeframe_to überarbeiten ohne a.timeframe_to
20                 a.Rolle=Fahrer
21                 neue Fahrt zurück mit Fahrer a
22             if optionen.length==1 then
23                 a.Rolle=Mitfahrer
24                 a zur Fahrt zurück mit Fahrer b hinzufügen
25                 b.timeframe_back=overlap(a.timeframe_back,b.timeframe_back)
26             if optionen.length>1 then
27                 a.Rolle=Mitfahrer
28                 b=findEmptiestCar(optionen)
29                 a zur Fahrt zurück mit Fahrer b hinzufügen
30                 b.timeframe_back=overlap(a.timeframe_back,b.timeframe_back)
```

```
22
23      if poss vom Typ poss_fahrer_to then
24          if l.length==0 then
25              if a.Rolle==Mitfahrer then
26                  b = Agent, bei dem a als Mitfahrer für Rückfahrt zugordnet ist
27                  a aus aktueller Rückfahrt entfernen
28                  b.timeframe_back überarbeiten ohne a.timeframe_back
29              a=Fahrer
30              neue Fahrt hin mit Fahrer a
31          else
32              Liste optionen
33              for alle Agenten b in l do
34                  if b==Fahrer AND b hat Platz im Auto AND
                     isOverlapping(a.timeframe_to,b.timeframe_to) then
35                      optionen.add(b)
36              if optionen.length==0 then
37                  if a.Rolle==Mitfahrer then
38                      b = Agent, bei dem a als Mitfahrer für Rückfahrt zugordnet ist
39                      a aus aktueller Rückfahrt entfernen
40                      b.timeframe_back überarbeiten ohne a.timeframe_back
41                  a=Fahrer
42                  neue Fahrt hin mit Fahrer a
43              if optionen.length==1 then
44                  a=rider
45                  a zur Fahrt hin mit Fahrer b hinzufügen
46                  b.timeframe_to=overlap(a.timeframe_to,b.timeframe_to)
47              if optionen.length>1 then
48                  a=rider
49                  b=findEmptiestCar(optionen)
50                  a zur Fahrt hin mit Fahrer b hinzufügen
51                  b.timeframe_to=overlap(a.timeframe_to,b.timeframe_to)
```

4 Analyse

4.1 Beschreibung der Experimente

Im Folgenden listen wir zunächst die Standardwerte aufn die genutzt werden, insofern es nicht anders angegeben ist. Danach beschreiben wir die verschiedenen Experimente, die wir durchführen wollen.

Standardwerte

Das Framework[Feh24] stellt 14 ähnliche Datensätze mit 12780 Agenten mit je einer Anfrage, die Zeiten und Zielort enthält, zur Verfügung. Darunter gibt es fünf verschiedene lokale Verteilungen von Agenten. Für die erste Verteilung gibt es zehn Datensätze, die sich in den von den Agenten präferierten Zeiten unterscheiden, für die anderen vier lokalen Verteilungen jeweils einen. Wir benennen die Datensätze im Folgenden, nach der lokalen und der zeitlichen Verteilung, mit 1_1 bis 1_10 beziehungsweise bis 5_1. Diese Agenten befinden sich alle in einer Distanz von weniger als 55 km zu ihrem jeweiligen Universitätstsandort. Die Anfragen beinhalten Zeiten die über einen Tag verteilt betrachtet werden.

Es sind einige Standardwerte festgelegt, die in jedem Experiment so gewählt wurden, insofern es nicht anders angegeben ist.

Wir betrachtet nur diejenigen Agenten deren Distanz zum jeweiligen Universitätstsandort zwischen $2,0$ und $25,0$ km liegt. Dabei erhalten wir bei voller Dichte Datensätze von etwa 6500 Agenten.

Passagiere laufen im Standardfall maximal 1200 m von ihrer Heimatposition zur Strecke eines anderen Agenten und sind nicht bereit zum Unistandort zu laufen. Das bedeutet, die Zielorte von Fahrer und Mitfahrern müssen übereinstimmen. Außerdem ist jedes Fahrzeug ein Auto mit fünf Sitzen, was heißt, dass jeder Fahrer maximal vier Passagiere transportieren kann.

Immer dann, wenn wir für ein Experiment einen Satz von Agenten mit geringerer Dichte, beziehungsweise mit weniger Agenten, benötigen, so nutzen wir Zufallszahlen, um diese auszuwählen. Wir wählen mit einem Zufallsgenerator ungefähr einen Prozentsatz an Agenten aus. Dafür generieren wir für jeden Agenten eine Zufallszahl zwischen 1 und 100. Wenn diese kleiner ist, als der festgelegte Wert für den Prozentsatz, so fügen wir diesen Agenten zum Datensatz hinzu. Die Ergebnisse sind rekreierbar, da der

Zufallsgenerator immer mit dem gleichen Seed[1] verwendet wird. Diese Änderung findet sich in der *AgentManager*-Klasse, die die Agenten aus einem json-Dokument einliest.

Um die Tabellen einheitlich zu gestalten und Platz zu sparen, benennen wir verschiedene Metriken mit Abkürzungen. Die Abkürzungen und deren Bedeutungen sind in Tabelle 4.1 zu finden.

Standardfälle

Zunächst führen wir den Algorithmus einmal mit jedem Datensatz als Grundlage aus. Auch den schon im Framework enthaltenen EverybodyDrives-Algorithmus[Feh24], der zum Vergleich Ergebnisse ohne Ridesharing liefert, führen wir für jede lokale Verteilung einmal aus.

Da die Annahme, dass weiter entfernt wohnende Agenten im Regelfall weniger mögliche Mitfahrgelegenheiten haben und umgekehrt, uns zur Idee für den Algorithmus gebracht hat, möchten wir hier testen, ob es tatsächlich stimmt, dass die Agenten, die als Fahrer zugeordnet wurden, im Durchschnitt höhere Distanzen zur Universität zurücklegen. Zudem möchten wir prüfen, wie groß die Differenzen in den Ergebnissen für verschiedene Metriken sind. Hier möchten wir unter anderem die Distanz und die durchschnittliche Anzahl an belegten Sitzplätzen mit verschiedenen Datensätzen untersuchen. Darauf aufbauend können wir abschätzen, ob sich das Ergebnis für einen echten Datensatz gut voraussagen lassen könnte. Wenn die Ergebnisse stark voneinander abweichen, ist dies nicht der Fall. Wir nehmen an, dass das Ergebnis für einen realen Datensatz ähnlich wäre, wenn die Unterschiede zwischen den Ergebnissen der Datensätze relativ klein sind. Daran können wir abschätzen, wie gut sich unsere Ergebnisse auf verschiedene ähnlich verteilte Datensätze übertragen lassen.

Experiment 1 - verschiedene Agentendichten

Für das erste Experiment nutzen wir den Datensatz 1_1. Dann verringern wir die Dichte schrittweise um 10%. Damit betrachten wir Anteile von 10% bis 100% des Datensatzes. Das soll eine Situation darstellen, in der weniger Studierende bereit sind, am System teilzunehmen.

Damit möchten wir feststellen, wie sich die Verhältnisse zwischen den Anzahlen von Fahrern, Mitfahrern und einzelnen Fahrern bei verschiedenen Dichten verändern. Hier suchen wir auch eine ungefähre Anzahl von Agenten, die an einem System teilnehmen müsste, damit 90% der Anfragen zu einem Match führen, beziehungsweise, damit insgesamt weniger als 10% der Genten alleine fahren.

[1] Seed: 1234

Experiment 2 - variable Laufstrecke

Danach führen wir einige weitere Experimente durch, bei denen die maximal akzeptierte Distanz zu Fuß variiert wird. Das gilt sowohl für die maximale Distanz vom Ausgangspunkt zur Strecke, als auch von der Strecke zum Zielort. Dieses Experiment führen wir nicht mit dem gesamten Datensatz durch, sondern mit etwa der Hälfte der Agenten des Datensatzes 3_1. Dabei erhalten wir einen Satz von 3298 Agenten.

Damit möchten wir feststellen, welche der beiden Distanzen einen größeren Einfluss auf das Ergebnis hat, beziehungsweise ob der Einfluss tatsächlich von den einzelnen Distanzen oder eher von der gesamten Strecke abhängt.

Experiment 3 - Vergleich

Daraufhin führen wir den Algorithmus mit den Standardwerten auf einem Datensatz aus, der aus 50% von Datensatz 1_5 besteht, und betrachten die Ergebnisse im Vergleich mit vorläufigen Ergebnissen eines Carpooling-Algorithmus', der von Jonas Bart im Zuge des Masterpraktikums entwickelt wird[Bar24]. Dafür nutzen wir außer diesem kleineren Datensatz noch den Satz 1_1. Bei dem Algorithmus von Jonas Barth werden die Agenten zu Clustern zusammengefasst und dann werden für die Cluster Routen berechnet, bei denen so viele Agenten wie möglich mit so wenigen Autos wie möglich transportiert werden können. Der Fahrer muss dabei am weitesten entfernt wohnen, um auf dem Weg andere Agenten aufzusammeln, und die Umwege sind zeitlich begrenzt. Der Algorithmus ist im gleichen Framework[Feh24] implementiert.

Dabei möchten wir betrachten, wie sich die gesamt von Fahrzeugen zurückgelegte Distanz ändert. Insbesondere darauf bezogen, wie sich die Umwege und die Anzahl der Fahrzeuge darauf auswirken. Außerdem interessiert uns hier die durchschnittliche Auslastung der Fahrzeuge, die sich direkt aus der Anzahl der Fahrer berechnen lässt, sowie die Anzahl an einzelnen Fahrern. Zuletzt ist es hier noch interessant zu prüfen, wie sich die durchschnittliche Zeit verändert, die ein Agent vom Startort bis zum Zielort benötigt.

Experiment 4 - Variable Sitzplatzanzahl

Hier möchten wir zusätzlich noch testen, wie sich eine Änderung der maximalen Sitzplatzanzahl auf die Anzahl von Fahrzeugen und einzelnen Fahrern auswirkt. Hier testen wir zwei Datensätze für Konfigurationen mit vier, fünf und sechs Plätzen im Auto. Hierfür nutzen wir die Datensätze 4_1 und 1_7.

Dabei wollen wir herausfinden, wie sich die Anzahl an Fahrern und einzelnen Fahrern dadurch verändert. Insbesondere möchten wir betrachten, ob es einen größeren Einfluss hat, die Sitzplatzanzahl zu reduzieren, als sie zu erhöhen.

Experiment 5 - kürzeste Laufdistanzen

Hierfür nutzen wir Datensatz 1_1, um zu testen, wie sich die gelaufenen Distanzen verändern, wenn wir statt die erste passende Laufstrecke zu akzeptieren, weiter suchen, ob wir noch eine kürzere finden. Dafür ändern wir die Methode, die nach möglichen Matches sucht (Algorithmus 1) so ab, dass die Schleife, in der wir testen, ob ein Punkt auf der Strecke eines Agenten B nahe genug am Ausgangsort eines Agenten A liegt, nicht mehr abgebrochen wird, sobald feststeht, dass der Agent eine mögliche Mitfahrgelegenheit bieten kann. Stattdessen untersuchen wir auch die andern Punkte, um möglichst kurze Routen zu finden.

Damit wollen wir untersuchen, wie genau unsere Ergebnisse für den Standardfall im Hinblick auf die gelaufenen Distanzen, beziehungsweise dafür benötigten Zeiten sind. Damit können wir abschätzen, um wie viel wir die Strecken ungefähr überschätzt haben.

Experiment 6 - verschiedene Distanzen der Agenten

Im Standardfall betrachten wir nur Agenten, die zwischen $2,0$ und $25,0$ km vom jeweiligen Universitätsstandort entfernt wohnen. Dabei ergeben sich, wie in Abbildung 4.8 zu sehen ist, aber viele Punkte für das Mitnehmen von Passagieren, die innerhalb von Würzburg liegen. Allerdings ist es interessant, auch andere Distanzen zu betrachten. Hierfür haben wir für den Datensatz 1_1 drei zusätzliche Radien getestet. So betrachten wir zuerst alle Agenten im Radius zwischen $5,0$ und $25,0$ km, dann die zwischen $5,0$ und $40,0$ km und zuletzt die $2,0$ und $40,0$ km.

Daraus möchten wir ableiten wie viele der Matches nur dadurch zustande kommen, dass innerhalb von Würzburg, wo die Agentendichte sehr hoch ist, noch ein passender Mitfahrer gefunden wird. Außerdem können wir an den Ergebnissen erkennen, wie viel sich dadurch ändert, dass wir weiter entfernte Agenten auch berücksichtigen.

4.2 Ergebnisse der Experimente

In diesem Abschnitt beschreiben wir, welche Metriken wir für die verschiedenen Experimente ausgewertet haben und welche Tabellen zu welchem Experiment gehören, beziehungsweise, wie diese zu lesen sind.

In Tabelle 4.2 und Tabelle 4.3 befinden sich die Ergebnisse des EverybodyDrives-Ansatzes sowie unseres Algorithmus', die auf Basis des Datensatzes 1_1 entstanden sind. Dabei beschreibt jede Zeile in Tabelle 4.3 den gleichen Datensatz wie die entsprechende Zeile in Tabelle 4.2. So beschreibt zum Beispiel die dritte Zeile in beiden Tabellen die Ergebnisse, die unser Algorithmus für den zweiten Datensatz liefert, die erste Zeile die Ergebnisse des EverybodyDrives-Ansatzes und die letzte Zeile zeigt den Durchschnitt aller Ergebnisse, die unser Algorithmus liefert. Die erste Spalte (Algo) benennt hier den genutzten Algorithmus und die zweite (dat) den entsprechenden Datensatz. Die anderen

Spalten enthalten die Ergebnisse. Zuerst ist hier die Anzahl an Fahrern, beziehungsweise damit die Anzahl an fahrenden Autos (v) zu finden. In der ersten Tabelle befinden sich sonst Ergebnisse, die mit der benötigten Zeit zu tun haben. Dabei ist zuerst die gesamte Zeit, die Agenten in Fahrzeugen verbringen, zu sehen (t_a_f_t), dann die Zeit, die sich alle Autos gesamt auf der Straße befinden (t_v_t), die gesamte Zeit, die alle Agenten inklusive Fußweg benötigen (t_a_t) und zuletzt die durchschnittliche Zeit, die im Fahrzeug verbracht wird (t_a_f_d), jeweils in Minuten, sowie einen Prozentsatz, der beschreibt, wie groß die im Fahrzeug verbrache Zeit im Gegensatz zum EverybodyDrives-Ansatz ist. Die letzte Spalte in Tabelle 4.2 zeigt, wie lange Agenten im Durchschnitt insgesamt unterwegs sind (+t_a_d), wieder als Prozentsatz, der sich aus der benötigten Zeit un der Zeit beim EverybodyDrives-Ansatz zusammensetzt. Das wird inklusive der für den Fußweg benötigten Zeit berechnet. In Tabelle 4.3 findet sich zunächst die Laufzeit des Algorithmus, danach die Distanz in Kilometern, die von Agenten (d_a_t) und Autos (d_v_t) jeweils insgesamt zurückgelegt wird. Die beiden nächsten Spalten beziehen sich nur auf die zu Fuß zurückgelegt Distanz. Die erste (d_f_t) beschreibt die gesamte zu Fuß zurückgelegte, die zweite die durchschnittliche nur von Mitfahrern in beide Richtungen zusammen zurückgelegte Laufstrecke (d_f_d), wieder in Kilometern. Die Fahrer sind hier nicht mit einbezogen. Die nächsten beiden Spalten zeigen, wie viele Fahrer auf dem Weg zum Zielort (v_e_h) beziehungsweise zurück (v_e_z) alleine gefahren sind. Der CO_2-Ausstoß (CO_2_t) bezieht sich auf den Gesamtausstoß aller Fahrzeuge. Die letzte Spalte beinhaltet die durchschnittliche Anzahl an Personen in einem Auto (s_d). Dabei werden alle Fahrzeuge berücksichtigt, auch die mit nur einem Fahrer und keinem Passagier.

Die beiden Tabellen 4.4 und 4.5 zeigen analog die Ergebnisse für die Agentenverteilungen 2_1 bis 5_1. Die Spalten haben hier die gleichen Bedeutungen. Zusätzlich gibt hier die dritte Spalte (a) die Anzahl an Agenten an.

Danach folgen Ergebnisse mit verschiedenen Dichten von Agenten, zu sehen in Tabelle 4.6. Beziehungsweise zeigt die Tabelle, wie sich die Anzahl an Agenten (a) auf die Menge an Matches auswirkt. Da uns hier besonders die Verhältnisse zwischen den gesamten Agenten, Fahrern (v) und einzelnen Fahrern interessieren (v_e_z für Rückweg, d_v_t für Hinweg), werden wir diese Werte angeben. Da die gesamt gefahrene Distanz (d_v_t) ein guter Anhaltspunkt für den CO_2-Ausstoß und die Kosten ist, werden wir diese ebenfalls eintragen. Für die verschiedenen Anzahlen von (einzelnen) Fahrern, geben wir die Werte zusätzlich als Anteil an der gesamten Menge von Agenten an. Dies ist durch den Prozentsatz in der jeweiligen Spalte dargestellt. Aus der gesamt von Fahrzeugen zurückgelegten Distanz berechnen wir außerdem die durchschnittliche Strecke pro Fahrer beziehungsweise Fahrzeug (d_v_d_v). In der letzten Spalte (s_d) findet sich noch die durchschnittliche Auslastung der Fahrzeuge, beziehungsweise die durchschnittliche Anzahl an belegten Sitzplätzen.

In Tabelle 4.7 zeigen wir die Unterschiede in der Anzahl an Matches bei verschiedenen Distanzen, die ein Agent maximal zu Fuß zurücklegen darf. Die Distanzen für die Strecke zwischen dem Ausgangsort eines Passagiers und der Strecke den Fahrers (Distanz

von Start), sowie der Strecke und dem Zielort, beziehunsweise dem Universitätsstandort (Distanz zu Ziel), befinden sich in der ersten beiden Spalten. Außerdem ist wieder nur die Anzahl an Fahrern (v), Mitfahrern und einzelnen Fahrern (v_e_h für Hinweg und v_e_z für Rückweg) relevant. Daher finden sich in der Tabelle diese Angaben, sowie die von Agenten benötigte Zeit, mit (t_a_t) und ohne (t_a_f_t) Fußweg und die gesamte von Fahrzeugen zurückgelegte Distanz (d_v_t).

Die Tabelle 4.8 enthält die Ergebnisse, die unser Algorithmus liefert, verglichen mit dem Algorithmus von Jonas Barth[Bar24] und dem EverybodyDrives-Ansatz. Hierbei betrachtet eine Gruppe von drei Spalten immer den gleichen Datensatz mit einer bestimmten Anzahl an Agenten (a). Die summierte Distanz (d_v_t) und Zeit (t_v_t) bezieht sich jeweils auf die von allen Fahrzeugen zusammen zurückgelegte Strecke. Die durchschnittliche Zeit in eine Richtung (t_a_d) bezieht sich auf die einzelnen Agenten und beinhaltet für unseren Algorithmus die Zeit, die ein Agent benötigt, um zur Strecke zu laufen. Außerdem geben wir die durchschnittliche Auslastung der Fahrzeuge (s_d) an. Die Agenten, die alleine gefahren sind (v_e_t), werden hier für beide Richtungen summiert und die Laufzeit des Algorithmus wird ebenfalls mit aufgeführt.

In Tabelle 4.9 und Tabelle 4.10 finden sich noch die Ergebnisse des vierten Experiments. Hier geben wir unter anderem die Anzahl an Fahrern (v) an. Für die einzelnen Fahrer betrachten wir die Anzahlen für die Hinfahrt (v_e_h) und die Rückfahrt (v_e_z), sowie die Summe dieser (v_e_t). Hier geben wir für die ersten drei dieser Spalten jeweils auch deren Anteil an der gesamten Menge von Agenten an. Für die andere (v_e_t) beschreibt der Prozentsatz den Anteil an gesamten Anfragen, also der doppelten Anzahl der Agenten. Außerdem nutzen wir wieder die von Autos gesamt zurückgelegte Strecke (d_v_t) als Indikator für die Auswirkungen auf die Umwelt. Als Letztes findet sich in den Tabellen die jeweilige durchschnittliche Anzahl an Personen in den Fahrzeugen (s_d).

Anschließend haben wir noch für einen kleinen Datensatz (30% von 1_1) sowie den vollständigen Satz 1_1 noch getestet, was passiert, wenn wir nicht die erste Laufroute zu einem betrachteten Punkt auf der Route nutzen, die kurz genug ist, sondern dann weiter suchen, ob wir noch eine kürzere Route finden. Dies hat allerdings schon bei dem kleinen Datensatz die Laufzeit verdoppelt. Für den vollen Datensatz erhöht sich die Laufzeit (von 46 Minuten auf 196) um ganze zweieinhalb Stunden. Diese Laufzeit betrachten wir als zu hoch, um sie in den anderen Experimenten zuzulassen. Die im Durchschnitt gelaufene Distanz hat sich allerdings von $2,00$ km auf nur noch $1,20$ km verringert. Das bedeutet, mit mehr Rechenleistung oder Rechenzeit können wir die Genauigkeit der Ergebnisse für die zu laufende Distanz und Zeit deutlich verbessern. Statt 8178 km laufen die Passagiere nun insgesamt nur noch 5115 km, was nur $62,5\%$ sind.

Zum Schluss findet sich in Tabelle 4.11 noch die Ergebnisse eines Experiments, bei dem wir die Exklusionsradien für die Agenten verändert haben. Das bedeutet dass wir statt allen Agenten, die zwischen $2,0$ und $25,0$ km vom jeweiligen Universitätstsandort entfernt wohnen, nun Agenten innerhalb anderer Radien betrachten. Dafür enthält die erste Spalte die Werte für die Radien und die zweite die sich dadurch ergebende Anzahl

an Agenten. In der dritten Spalte tragen wir die Anzahl der Fahrer (v) ein, in die beiden darauf folgenden die Anzahl an einzelnen Fahrern auf dem Hin- (v_e_h) und Rückweg (v_e_z). Die Prozentsätze beschreiben hier jeweils den Anteil an der gesamten Menge der Agenten. Zuletzt geben wir noch die Durchschnittliche Auslastung der Fahrzeuge an (s_d).

Tab. 4.1: Spaltenbezeichnungen

Bezeichnung	Bedeutung
dat	zugrunde liegender Datensatz
a	Anzahl an Agenten
p	Anzahl an Passagieren
v	Anzahl an Fahrern/Fahrzeugen
v_e_t	Gesamtanzahl an Fahrten, bei denen sich nur der Fahrer im Auto befindet
v_e_h	Gesamtanzahl an Fahrten zum Zielort, bei denen sich nur der Fahrer im Auto befindet
v_e_z	Gesamtanzahl an Fahrten vom Zielort, bei denen sich nur der Fahrer im Auto befindet
s_d	durchschnittlich von Passagieren und Fahrer zusammen belegte Plätze
CO_2_t	Gesamtausstoß von CO_2 durch alle Fahrzeuge (in t)
t_	Zeit (in min)
t_v_t	Gesamtzeit, die sich alle Autos aufsummiert auf der Straße befinden (in min)
t_a_t	Gesamtzeit, die alle Agenten summiert benötigen, um ihr Ziel zu erreichen (in min)
t_a_f_t	Gesamtzeit, die alle Agenten summiert in Fahrzeugen verbringen (in min)
t_a_f_d	Zeit, die Agenten durchschnittlich im Fahrzeug verbringen (in min)
+t_a_d	Zeit, die Agenten durchschnittlich länger im Fahrzeug verbringen als beim EverybodyDrives-Anzatz
d_	Distanz (in km)
d_v_t	Distanz, die alle Autos aufsummiert zurücklegen (in km)
d_a_t	Distanz, die alle Agenten aufsummiert zurücklegen (in km)
d_f_t	Distanz, die alle Agenten aufsummiert zu Fuß zurücklegen (in km)
d_v_d_v	durchschnittliche Distanz, die ein Fahrzeug/Fahrer zurücklegt (in km)

4.3 Ergebnisse der Analyse

Hier möchten wir die Ergebnisse der verschiedenen Experimente auswerten und einordnen.

Tab. 4.2: Ergebnisse - Verteilung 1 (6524 Agenten) - Teil 1

Algo	dat	v	t_a_f_t	t_v_t	t_a_t	t_a_f_d	+t_a_d
Ev-Dr	alle	6524	130833	130833	130833	10,03 (100%)	100%
RS-St	1_1	2171	143916	71372	246022	11,03 (110,0%)	211%
RS-St	1_2	2137	143833	70292	246792	11,02 (109,9%)	211%
RS-St	1_3	2170	144212	70979	246105	11,05 (110,1%)	210%
RS-St	1_4	2160	144094	70871	246953	11,02 (109,9%)	212%
RS-St	1_5	2137	143833	70594	246531	11,02 (109,9%)	210%
RS-St	1_6	2147	143801	70623	247121	11,02 (109,9%)	210%
RS-St	1_7	2122	144108	69798	247391	11,04 (110,1%)	212%
RS-St	1_8	2153	143765	70782	246698	11,02 (109,9%)	211%
RS-St	1_9	2140	143883	69882	246414	11,03 (110,0%)	211%
RS-St	1_10	2162	143697	70549	246417	11,01 (109,8%)	211%
RS-St avg	1_x	2150	143914	70648	246644	11,03 (110,0%)	210,9%

Für die Ergebnisse in Tabelle 4.2, Tabelle 4.3, Tabelle 4.4 und Tabelle 4.5 fällt auf, dass die Ergebnisse mit verschiedenen Datensätzen sehr ähnlich scheinen. Konkret haben wir den Unterschied der einzelnen Werte zum Durchschnitt untersucht und festgestellt, dass dieser tatsächlich sehr klein ist. Diese Berechnung haben wir für die Anzahl an Fahrern, sowie die gesamt von allen Agenten im Auto verbrachte und gesamt benötigte Zeit und die über die Fahrzeuge summierte Gesamtzeit durchgeführt. Allgemein ist kein Unterschied größer als $1,5\%$ des Durchschnitts. Das gilt sowohl für die zeitlichen, als auch die örtlichen Verteilungen, für die wir den Durchschnitt jeweils separat berechnet haben. Für die zeitlichen Verteilungen haben wir den Durchschnitt, der in Tabelle 4.2 zu finden ist, genutzt und für die örtliche den aus Tabelle 4.4. Die Anzahl an Fahrern unterscheidet sich bei der zeitlichen Verteilung (Datensätze 1_1 bis 1_10) um maximal $1,3\%$ vom Durchschnitt, bei den Datensätzen 1_1 bis 5_1 sogar nur um $0,8\%$. Für die von Agenten im Auto verbrachte Gesamtzeit liegt die größte Abweichung bei $0,21\%$ des Durchschnitts für die zeitlichen Verteilungen und $0,45\%$ für die lokalen. Bei der von

Tab. 4.3: Ergebnisse - Verteilung 1 - Teil 2

dat	Laufzeit	d_a_t	d_v_t	d_f_t	d_f_d	v_e_h	v_e_z	CO_2_t	s_d
alle	0,3 min	116821	116821	0	0	6524	6524	15,13	1
1_1	46 min	128764	58335	8178	1,88	315	381	7,48	3,005
1_2	51 min	128812	57343	8251	1,88	297	347	7,35	3,053
1_3	51 min	128928	57972	8156	1,87	353	387	7,43	3,006
1_4	54 min	128842	57881	8240	1,89	316	388	7,42	3,020
1_5	53 min	128685	57810	8226	1,88	302	348	7,41	3,053
1_6	48 min	128839	57703	8272	1,89	301	354	7,40	3,039
1_7	52 min	128839	56826	8275	1,88	324	381	7,29	3,074
1_8	49 min	128821	57922	8243	1,89	302	393	7,42	3,030
1_9	52 min	128767	56966	8213	1,87	319	371	7,30	3,049
1_10	50 min	128701	57510	8234	1,89	322	387	7,38	3,018
1_x	50,6 min	128800	57627	8229	1,88	315	374	7,39	3,035

Agenten benötigten Gesamtzeit ist der Unterschied noch kleiner. Hier liegt er bei $0,3\%$ für die zeitlichen und $0,5\%$ für die örtlichen Verteilungen. Auch die über alle Fahrzeuge summierte Zeit zeigt nur kleine Abweichungen, die sich für die lokalen und zeitlichen Verteilungen kaum unterscheiden ($1,2\%$ zeitlich und $1,33\%$ lokal). Das bedeutet, dass die Ergebnisse durch Unterschiede in der Gesamtheit eines Datensatzes, besonders wenn die Datensätze auf die gleiche Weise generiert sind, keinen großen Einfluss auf das Ergebnis hat. Darum ist davon auszugehen, dass Ergebnisse, die mit einem Datensatz von 6500 zufälligen tatsächlichen an der Universität Würzburg studierenden Agenten entstehen könnten, ebenfalls ähnlich wären.

In Abbildung 4.1 sehen wir die Verteilung der Distanzen, die Agenten auf dem Rückweg zu Fuß zurücklegen. Die Distanzen für den Hinweg sind sehr ähnlich. Sie betragen für die Fahrer natürlich null. Für Passagiere gilt, dass es wenige gibt, deren Distanz am unteren Ende des Spektrums liegt. Wir sehen, dass zwei Drittel der Agenten mindestens 900 m einfach laufen müssen. Selbst wenn man die Distanz nur von $1,2$ km auf $1,0$ km reduzieren würde, so sieht es so aus, als wären schon über die Hälfte der hier entstandenen Matches nicht mehr möglich. Allerdings kommt das vermutlich wieder dadurch zustande, dass wir einfach die erste passende Strecke zwischen dem Heimatort eines Agenten und der Strecke eines anderen nutzen und die tatsächlich kürzesten Strecken deutlich kleiner sind.

Es scheint zunächst unpassend, dass beim vollen Datensatz von über 6500 Agenten trotzdem noch etwa die Hälfte der von Fahrzeugen zurückgelegter Gesamtstrecke, ausgestoßenem CO_2 und anfallenden Kosten des EverybodyDrives-Ansatzes zustande kommt, obwohl nur etwa ein Drittel der Agenten Fahrer sind. Allerdings werden durch den genutzten Algorithmus eher diejenigen Agenten zu Fahrern, die eine längere Strecke zurücklegen, da diese tendenziell weniger mögliche Mitfahrgelegenheiten haben. Ein Agent

Tab. 4.4: Ergebnisse - andere Verteilungen - Teil 1

Algo	dat	a	v	t_a_f_t	t_v_t	t_a_t	t_a_f_d	+t_a_d
Ev-Dr	2_1	6561	6561	131438	131438	131438	10,02 (100%)	100%
RS-St	2_1	6561	2178	144613	71091	247797	11,02 (110,0%)	211%
Ev-Dr	3_1	6544	6544	131540	131540	131540	10,05 (100%)	100%
RS-St	3_1	6544	2171	144768	70952	247256	11,06 (110,0%)	211%
Ev-Dr	4_1	6545	6545	131129	131129	131129	10,02 (100%)	100%
RS-St	4_1	6545	2142	144512	70601	248132	11,04 (110,1%)	212%
Ev-Dr	5_1	6528	6528	130161	130161	130161	9,97 (100%)	100%
RS-St	5_1	6528	2144	143641	69827	247145	11,00 (110,3%)	212%
RS-St avg	x_1	6540	2161	144290	70789	247270	11,03	211,4%

legt im allgemeinen Durchschnitt etwa 20 km zurück, aber der Durchschnitt bei den Fahrern liegt bei etwa 27 km. Damit liegt die durchschnittliche Distanz der Mitfahrer bei etwa 16 km. Insgesamt bedeutet das, dass wir mit unserem Ansatz unter den genutzten Annahmen etwa die Hälfte der von Fahrzeugen zurückgelegten Strecke und damit auch des ausgestoßenen CO_2 sowie der Gesamtkosten einsparen können.

Im ersten Experiment, dessen Ergebnisse in Tabelle 4.6 zu finden sind, haben wir untersucht wie sich eine Reduzierung der Anzahl an Agenten über die Dichte des Datensatzes auf das Ergebnis auswirkt. Hier ist zu sehen, dass der Anteil an Fahrern und einzelnen Fahrern mit einer geringeren Anzahl an Agenten deutlich steigt. Wie Abbildung 4.3 zeigt, ist die Steigung bei weniger Agenten größer und flacht dann ab. Das bedeutet, dass ein Hinzufügen von zum Beispiel 100 zufälligen Agenten das Ergebnis prozentual mehr beeinflussen würde, wenn der Datensatz eher klein (zum Beispiel 1000 Agenten) ist, als wenn er schon größer ist (zum Beispiel 4000 Agenten). Außerdem haben wir in Abbildung 4.4 die Verteilung der von Passagieren zum Laufen benötigte Zeit betrachtet. Dass der Median im Allgemeinen bei allen Datensätzen bei einem Wert von über 13 liegt, ist erneut darauf zurückzuführen, dass wir immer die erste passende Laufstrecke akzeptieren. Allerdings sehen wir hier trotzdem, dass der Median für einen kleineren Datensatz etwas höher liegt. Das kann daran liegen, dass kürzere Laufstrecken nur dadurch zustande kommen können, dass die Wohnorte von Agenten nahe beieinander liegen und somit die Distanz zum ersten betrachteten Punkt schon deutlich unter

Tab. 4.5: Ergebnisse - andere Verteilungen - Teil 2

dat	Laufzeit	d_a_t	d_v_t	d_f_t	d_f_d	v_e_h	v_e_z	CO_2_t	s_d
2_1	0,3 min	117065	117065	0	0	6561	6561	15,07	1
2_1	46 min	127498	57680	8266	1,89	361	416	7,28	3,012
3_1	0,3 min	117270	117270	0	0	6544	6544	15,14	1
3_1	51 min	128242	57670	8216	1,87	352	407	7,29	3,014
4_1	0,3 min	117108	117108	0	0	6545	6545	15,14	1
4_1	53 min	128181	57667	8303	1,89	347	380	7,29	3,056
5_1	0,3 min	115928	115928	0	0	6528	6528	15,01	1
5_1	52 min	127747	56528	8298	1,89	314	371	7,23	3,045
x_1	50 min	128086	57576	8252	1,88	338	391	7,31	3,026

Tab. 4.6: Ergebnisse - Niedrigere Dichte

a	v	v_e_h	v_e_z	d_v_t	d_v_d_v	s_d
6524	2171 (33,3%)	315 (4,8%)	381 (5,8%)	58335	30,31	3,005
5880	2010 (34,2%)	320 (5,4%)	380 (6,4%)	54214	26,97	2,925
5278	1858 (35,2%)	309 (5,9%)	375 (7,1%)	50185	27,01	2,841
4631	1684 (36,4%)	320 (6,9%)	392 (8,5%)	45290	26,89	2,750
3977	1515 (38,1%)	332 (8,3%)	384 (9,7%)	40549	26,77	2,625
3322	1328 (40,0%)	341 (10,3%)	391 (11,8%)	35291	26,57	2,502
2656	1118 (42,1%)	321 (12,1%)	364 (13,7%)	30042	26,87	2,376
1949	897 (46,0%)	308 (15,8%)	344 (17,7%)	23502	26,20	2,173
1288	663 (51,5%)	281 (21,2%)	296 (23,0%)	16665	25,14	1,943
631	378 (59,9%)	196 (31,1%)	201 (31,9%)	8827	23,35	1,669

der maximal akzeptierten Laufstrecke liegt. Anhand von Abbildung 4.2 können wir diese Situationen vergleichen. Wenn für einen Agenten B die Punkte der Reihe nach bei Punkt 1 beginnend abgearbeitet werden, so ist schon beim zweiten Punkt das Kriterium der maximalen Distanz erfüllt, wodurch er als Ort ausgewählt wird, an dem Agent B den Agenten A mitnimmt. Allerdings ist die Distanz zwischen Agent A und Punkt 3 kleiner und uns entgeht somit die bessere Lösung. In unserem Algorithmus wird für alle Strecken, die auf diese Weise am Wohnort eines anderen Agenten vorbeiführen, der erste passende Punkt gewählt, der damit fast immer nahe am erlaubten Maximum liegt. Wenn Agent B nun aber an Punkt 3 wohnt und wir die Punkte damit von 3 beginnend aufsteigend betrachten, so haben wir mit dem ersten betrachteten Punkt schon einen gefunden, dessen Distanz unter der maximal akzeptierten liegt. Nur dadurch können in der aktuellen Implementierung deutlich geringere zu Fuß zurückgelegte Distanzen entstehen. Dieser Fall, dass Agenten nahe zusammen wohnen, scheint bei Datensätzen mit

Tab. 4.7: Ergebnisse - Verschiedene Distanzen zu Fuß

Distanz von Start	Distanz zu Ziel	v	v_e_h	v_e_z	t_a_f_t	t_a_t	d_v_t
0	0	3298	3298	3298	68926	68926	60521
0	1200	3298	3298	3298	68926	68926	60521
100	0	2756	2260	2280	70387	71214	54761
100	1200	2661	2082	2106	70044	74455	53704
600	0	1704	695	726	72874	91158	40680
600	600	1606	576	621	72669	94326	39214
1200	0	1327	350	376	74511	121068	35168
1200	1200	1126	188	241	73109	138081	31708
1200	2400	925	32	55	65399	192596	27620
2400	0	1015	95	141	75290	182608	29583
2400	1200	886	44	76	72623	200730	26832
2400	2400	765	26	29	65857	252864	23457
3600	0	886	56	76	73863	244093	26744

geringerer Dichte seltener vorzukommen.

Außerdem zeigen die Ergebnisse, dass ein Wert von mehr als 90% gematchter Anfragen schon bei um die 3500 Agenten erreicht werden kann. Eine Matchingrate von 95% erreichen wir in unseren Experimenten allerdings nie. Das zeigt auch, dass die letzten 40% der Agenten im Vergleich eine deutlich kleinere Verbesserung von nur noch etwa 4% mit sich bringen.

Das zweite Experiment (Tabelle 4.7) untersucht den Einfluss verschiedener maximaler Laufdistanzen, sowohl vom Ausgangsort eines Agenten zur Route eines Anderen als auch von der Route zum Zielort, auf die Anzahl an Fahrern und einzelnen Fahrern. In Abbildung 4.5 ist dies visuell dargestellt. Dabei zeigen die Achsen die maximalen Lauf-

Tab. 4.8: Ergebnisse - Vergleich mit Carpooling

	Ev-Dr	Cluster Carp.	RS-St	Ev-Dr	Cluster Carp.	RS-St
a	6524	6524	6524	3322	3322	3322
d_v_t	116821	59506	**58335**	62380	35280	**34894**
t_v_t	130833	76482	**71372**	67644	43892	**41403**
s_d	0	**3,147**	3,005	1	**2,827**	2,575
v_e_t	13048	913	**696**	6644	**652**	672
t_a_d	**10,02**	12,91	18,86	**10,18**	13,24	18,57
Laufzeit	17 s	51 min	46 min	17 s	31 min	11 min

Tab. 4.9: Ergebnisse - Verschiedene Fahrzeuggrößen - 4_1 - 6545 Agenten

Plätze	v	v_e_h	v_e_z	v_e_t	d_v_t	s_d
5	2171 (33,2%)	347 (5,30%)	380 (5,81%)	727 (5,55%)	58335	3,056
4	2332 (35,6%)	364 (5,56%)	400 (6,11%)	764 (5,84%)	59612	2,807
6	2041 (31,2%)	355 (5,42%)	370 (5,65%)	725 (5,54%)	56737	3,207

Tab. 4.10: Ergebnisse - Verschiedene Fahrzeuggrößen - 1_7 - 6524 Agenten

Plätze	v	v_e_h	v_e_z	v_e_t	d_v_t	s_d
5	2122 (32,5%)	324 (5,0%)	381 (5,8%)	705(5,4%)	56826	3,074
4	2301 (35,3%)	348 (5,3%)	386 (5,9%)	734 (5,6%)	58708	2,835
6	2014 (30.9%)	321(4,9%)	362(5,5%)	683(5,2%)	55770	3,239

strecken und die Größe der Kreise stellt die Anzahl an Fahrern dar. Dabei fällt auf, dass sich beide Werte auf die Anzahl an Fahrern auswirken, was sich im Diagramm dadurch zeigt, dass die Kreise nach oben rechts kleiner werden. Das bedeutet, das Ergebnis mit den größten Werten für die Distanzen führt zur kleinsten Anzahl an Fahrern. Außerdem können wir hier auch erkennen, dass eine Änderung der erlaubten maximalen Distanz zwischen der Strecke und dem Zielort das Ergebnis nicht beeinflusst, wenn die erlaubte Distanz zwischen dem Startort und der Strecke null ist. Dann werden in allen Fällen alle Agenten zu Fahrern, da keine Matches zustande kommen können. Das ist auch in Abbildung 4.6 zu sehen. Dort haben wir die Distanzen addiert. In dieser Darstellung ist besonders gut zu sehen, dass eine zu niedrige Distanz für den Weg von zu Hause zur Strecke dazu führt, dass keine Matches zustande kommen, unabhängig davon, wie hoch die Distanz zwischen Strecke und Universität gewählt wird. Ansonsten ist zu sehen, dass eine höhere gesamte Distanz die Anzahl an benötigten Fahrern allgemein verringert. Zudem können wir in Tabelle 4.7 und Abbildung 4.6 erkennen, dass bei gleicher Summe die Aufteilungen, bei denen ein größerer Anteil der Gesamtstrecke für die Distanz vom Wohnort eines Passagiers zur Route des Fahrers genutzt wird, bessere Ergebnisse liefern. Daraus lässt sich schließen, dass dieser Teil der Laufstrecke relevanter ist und nicht verringert werden sollte, um im Gegenzug eine weitere Strecke von der Route eines Fahrers zum einem Passagier zugeordneten Universitätsstandort zu erlauben. Bei höheren

Tab. 4.11: Ergebnisse - Verschiedene Distanzen

Distanz	a	v	v_e_h	v_e_z	s_d
2 - 25	6524	2171 (33,28%)	315 (4,83%)	381 (5,84%)	3,005
5 - 25	2697	1449 (53,73%)	641 (23,77%)	696 (25,81%)	1,861
5 - 40	4098	2279 (55,61%)	1042 (25,43%)	1118 (27,28%)	1,798
2 - 40	7925	2914 (36,77%)	503 (6,35%)	574 (7,24%)	2,720

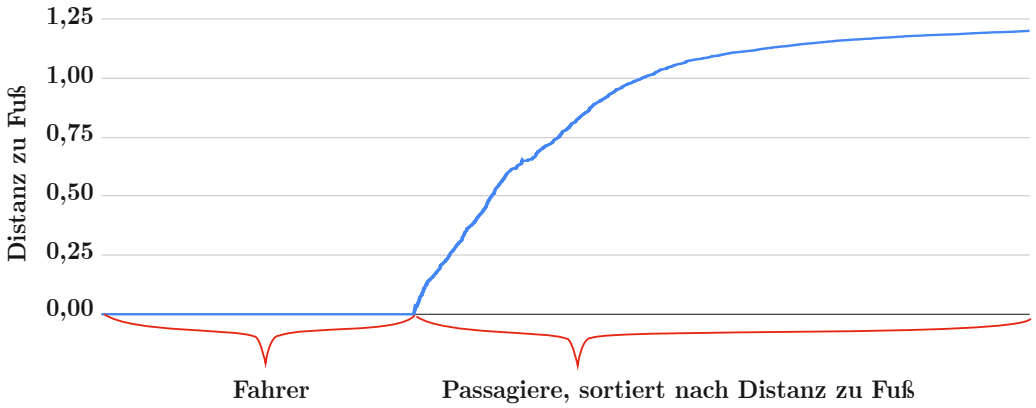


Abb. 4.1: Laufdistanzen der Agenten

Distanzen für beide Werte ist der Unterschied zwischen den Verteilungen jedoch kleiner. So ist das Ergebnis für die Verteilungen $2400 - 1200$, $1200 - 2400$ und $3600 - 0$ sehr ähnlich. Aber auch hier liefert die Verteilung $3600 - 0$ das beste Ergebnis. Das Ergebnis für die Verteilung $2400 - 1200$ ist nur gleich gut. Noch größere Distanzen zu testen wäre vielleicht theoretisch interessant, aber für die Praxis eher nicht relevant.

In Tabelle 4.8 finden wir die Ergebnisse des dritten Experiments. Dabei haben wir unseren Algorithmus mit einem Carpooling-Algorithmus[Bar24] verglichen, der Umwege erlaubt, um weniger Fahrzeuge zu benötigen. Es ist vermutlich durch diese Umwege erklärbar, dass die über die Fahrzeuge summierte Zeit und Distanz bei unserem Algorithmus etwas niedriger sind. Hierbei führt bei der Verwendung des vollen Datensatzes 1_7 unser Algorithmus auch zu weniger einzelnen Fahrern, obwohl wir keine Umwege erlauben. Das kann aber dadurch erklärt werden, dass wir direkt versuchen, die Passagiere so gut wie möglich auf die Fahrer aufzuteilen, indem wir bei der Zuweisung eines Passagiers immer das am wenigsten ausgelastet Fahrzeug bevorzugen. Die Anzahl an Fahrern allgemein ist für unseren Algorithmus etwas höher, was zu einer etwas niedrigeren durchschnittlichen Auslastung der Fahrzeuge führt. Das ist im Algorithmus von Jonas Barth im Gegensatz zur Gesamtanzahl der Fahrer nicht relevant. Die durchschnittliche Reisezeit eines Agenten liegt für unseren Algorithmus deutlich höher, da die Zeit zu Fuß, wie vermutet, einen großen Einfluss darauf hat. Dir Fahrzeit alleine liegt im Durchschnitt nur bei etwa elf Minuten.

Experiment vier, dessen Ergebnisse in Tabelle 4.9 und Tabelle 4.10 zu finden sind, betrachtet, wie sich die Anzahl an Fahrern und einzelnen Fahrern verändert, wenn wir die maximale Sitzplatzkapazität der Fahrzeuge verändern. Die Unterschiede in der Anzahl der einzelnen Fahrer sind hier sehr klein, die in der Anzahl der Fahrer im Allgemeinen etwas größer, aber trotzdem im Bereich von wenigen Prozent. Das zeigt, dass nicht viele Fahrer dadurch selbst alleine fahren müssen, dass die möglichen Mitfahrgelegenheiten schon voll sind. Insgesamt sind die Unterschiede bei der Reduzierung der Kapazität minimal größer, als wenn wir sie erhöhen, was den Erwartungen entspricht. Eine niedrigere

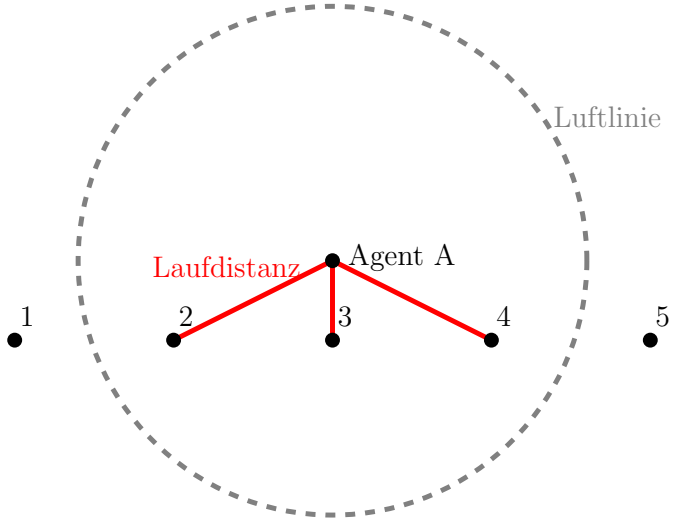


Abb. 4.2: Beispielhafte Situation

Kapazität führt aber doch zu mehr Fahrer, die dann aber auch meist trotzdem Mitfahrer zugewiesen bekommen. Das zeigt aber auch, dass das Ergebnis durch größere Fahrzeuge nicht deutlich besser wird. Da Fahrzeuge mit mehr Kapazität oft auch in den Dimensionen größer sind und mehr Treibstoff verbrauchen, ist es unter Umständen nicht die bessere Wahl für eine Verbesserung der Anzahl der einzelnen Fahrer von unter einem halben Prozent größere Fahrzeuge zu bevorzugen.

Für Experiment 5 haben wir betrachtet, was passiert, wenn wir statt den ersten gefundenen Pfad zu akzeptieren, weiter nach dem kürzesten Weg suchen. Dabei zeigt sich, dass wir deutlich andere Ergebnisse erhalten, die näher an der realen Situation sind. Da sich die Matches nicht ändern, werten wir hier nur die zum Laufen benötigte Zeit aus, die sich zusammen mit der Distanz tatsächlich ändert. In Abbildung 4.7 sehen wir, dass die Verteilung deutlich ausgeglichener ist, da im Standardfall die erste Strecke unter $1,2$ km akzeptiert wird, wodurch sie im Regelfall nicht weit unter $1,2$ km liegt und noch deutlich kürzer gewählt werden kann. So liegt nun zum Beispiel der Median bei $7,2$ Minuten liegt anstatt bei $13,2$ und hat sich somit fast halbiert. Im Maximum kommt es aber trotzdem noch vor, dass eine Strecke gerade noch so innerhalb der maximalen Distanz verläuft. Die insgesamt gelaufene Distanz hat sich so um $62,5\%$ verringert. Das heißt wir können auch für die anderen vollen Datensätze annehmen, dass wie die gelaufene Distanz um etwa 60% überschätzt haben.

Im letzten Experiment waren die veränderten Variablen die Exklusionsradien von Agenten. Dabei fällt auf, dass deutlich weniger Matches zustande kommen, wenn wir die minimale Distanz zum Universitätstsandort erhöhen. Dieser Effekt kommt zum Teil dadurch zustande, dass sich die Anzahl der Agenten insgesamt verringert. Allerdings ist er nicht vollständig darüber erklärbar. Für die Distanz von $5,0$ bis $25,0$ km ist die Anzahl der Agenten derjenigen ähnlich, die wir durch Reduzierung der Dichte des glei-

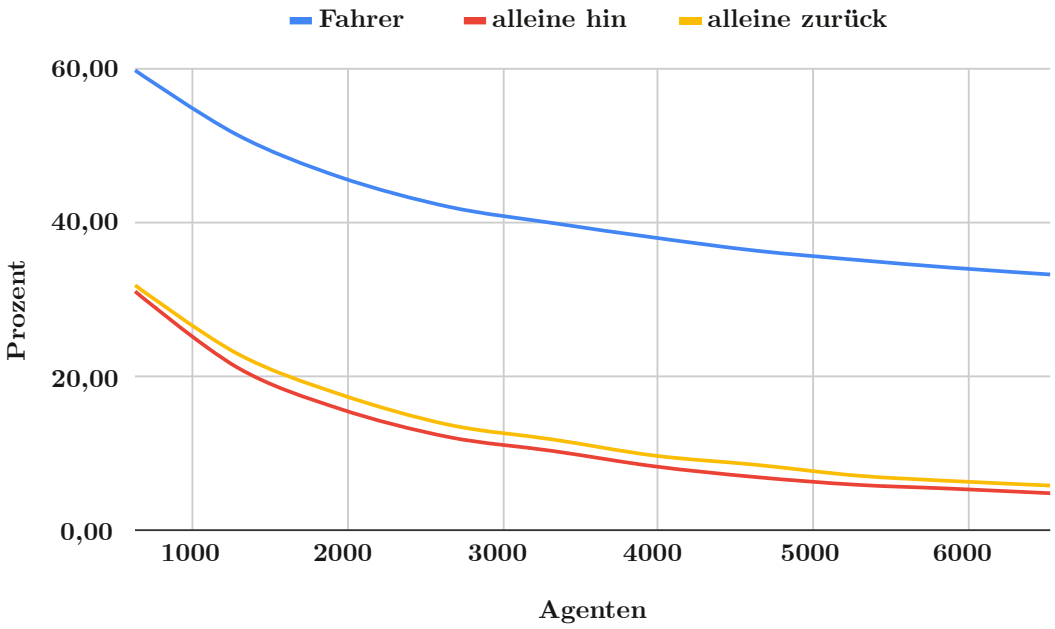


Abb. 4.3: Anteil der (einzelnen) Fahrer bei verschieden großen Datensätzen

chen zugrunde liegenden Datensatzes (1_1) auf 40% erhalten. Allerdings erhöht die Verringerung der Dichte die benötigen Fahrer auf 42, 1% der Agenten und die Anzahl der einzelnen Fahrer auf etwa 13% der Agenten, wohingegen die veränderte minimale Distanz zu deutlich andern Anteilen führt. Hier erhöht sich die Anzahl der Fahrer auf 53, 7% und die der einzelnen Fahrer auf etwa 25%. Diese deutlich höheren Anteile zeigen, dass die innerhalb von Würzburg zustande gekommenen Matches für einen deutlichen Anteil der sonst alleine fahrenden Agenten noch Mitfahrer bereitstellen. So bleiben durch das Ausschließen dieses Falles fast doppelt so viele Fahrer alleine. Die Version mit der Distanz zwischen 5, 0 bis 40, 0 km können wir rein nach der Anzahl der Agenten etwa mit 60% von 1_1 vergleichen. Dabei erhalten wir Werte von 38, 1% Fahrern und etwa 9% einzelnen Fahrern. Im Gegensatz dazu benötigen wir bei der Betrachtung der zwischen 5, 0 und 40, 0 km entfernten Agenten 55, 6% der Agenten als Fahrer und es fahren sogar etwa 26% alleine. Das zeigt, dass auch die Hinzunahme der zwischen 25, 0 und 40, 0 km entfernten Agenten das Ergebnis prozentual nicht verbessert. So ist auch die Auslastung der Fahrzeuge mit nur 1, 80 Agenten deutlich geringer als wenn wir die Agenten zwischen 2, 0 und 25, 0 km betrachten und sogar niedriger als für die zwischen 5, 0 und 25, 0 km. Das kann dadurch zustande kommen, dass zwischen weiter entfernt wohnende Agenten eher größere Distanzen liegen und damit ein Großteil der Matches mit näher am Universitätsstandort wohnenden Agenten geschehen muss. Wenn wir die Ergebnisse für die Agenten von 2, 0 bis 40, 0 km betrachten fällt besonders auf, dass wir prozentual trotzdem mehr Fahrer benötigen und mehr Fahrer alleine bleiben als bei nur den Agenten, die zwischen 2, 0 bis 25, 0 km weit entfernt wohnen (36, 8% und etwa 7% im Gegensatz zu 33, 2% und etwa 5%), obwohl die gesamte Anzahl der Agenten größer

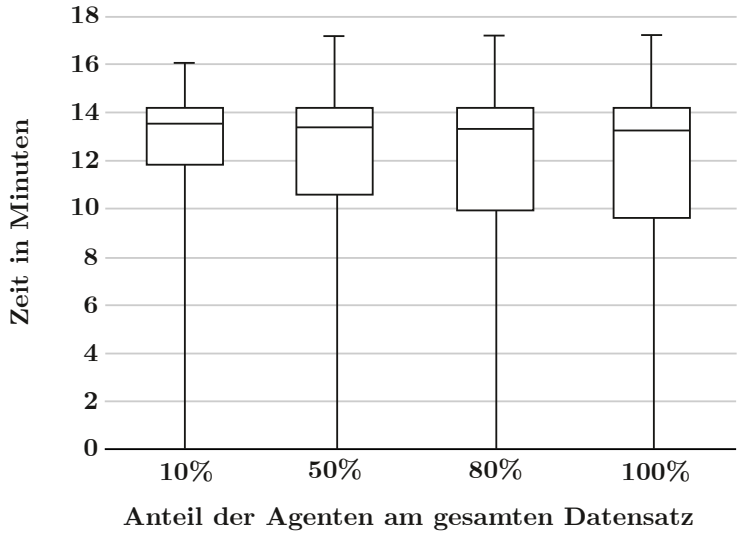


Abb. 4.4: von Passagieren zum Laufen benötigte Zeit für verschieden große Datensätze

ist. Das ist vermutlich wieder dadurch zu erklären, dass die Agenten über eine deutlich größere Fläche verteilt sind und damit die Chance kleiner ist, außerhalb von Würzburg schon ein Match zu finden. Genauer liegen die Ausganspunkte der 6524 Agenten, die zwischen $2,0$ und $25,0$ km entfernt wohnen auf einer Fläche von etwa 2000 m^2, wohingegen die 1401 Agenten, die zwischen $25,0$ und $40,0$ km entfernt wohnen, über eine Fläche von etwa 3000 m^2 verteilt sind. Da wir alleine dadurch schon mehr Fahrer benötigen, ist davon auszugehen, dass das die Anzahl der einzelnen Fahrer sich ebenfalls erhöht.

4.4 Laufzeit

Der Großteil der Laufzeit, das heißt bei einem vollen Datensatz mit einer Laufzeit vom 50 Minuten etwa 45 Minuten, kommt durch die Berechnung der tatsächlichen Laufdistanz mit GraphHopper für das Finden von möglichen Matches (Algorithmus 1) zustande, da sich viele Routen finden, die innerhalb der geforderten Luftlinie verlaufen, wodurch viele tatsächliche Distanzen berechnet werden müssen. Daher ist es nötig, die Anzahl der Streckenberechnungen so weit wie möglich zu verringern. Dafür haben wir andere Kriterien gesucht, anhand derer Agenten als mögliche Matches ausgeschlossen werden. Diese Kriterien werden im Algorithmus getestet, bevor die Distanz betrachtet wird. Um die tatsächlichen Streckenberechnungen zu reduzieren, betrachten wir für Paare von Agenten nun zuerst die Zeitintervalle und berechnen die Distanz nur noch für Paare, bei denen die zeitlichen Voraussetzungen gegeben sind, sowie für die in Abbildung 3.1 dargestellte Einschränkung der Distanz gegeben ist. Es wäre möglich, noch weitere Filter zu nutzen, aber die Laufzeit von ungefähr 50 Minuten auf dem vollen genutzten Datensatz haben wir als ausreichend akzeptiert. Wenn zuerst die Luftlinie geprüft wird, daraufhin die tatsächliche Distanz und erst danach die Zeit, dann beträgt die Laufzeit mit rund 6500

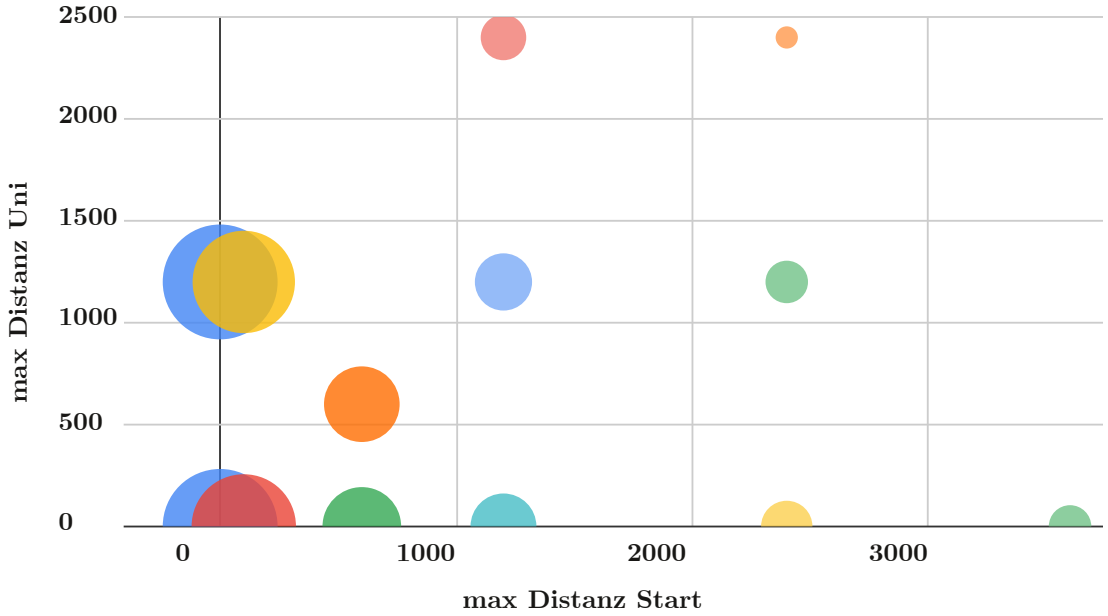


Abb. 4.5: Anzahl an benötigten Fahrern, dargestellt durch die Größe der Blasen, bei verschiedenen maximal erlaubten Laufdistanzen

Agenten fast 13 Stunden, wovon wieder nur wenige Minuten für den Rest der Berechnungen benötigt werden. Ähnlich erhöht auch jede Änderung, die die Anzahl an möglichen Matches erhöht, für die die Distanz berechnet werden muss, die Laufzeit. Da zeigt sich auch an den Ergebnissen des Experiments, bei dem wir, statt die erste passende Laufstrecke zu akzeptieren, weiter nach der kürzesten Strecke gesucht haben. Dadurch, dass die Strecke für alle anderen Punkte im Luftlinie-Radius ebenfalls berechnet werden muss, erhöht sich die Laufzeit von 46 Minuten auf ganze 196.

4.5 Mögliche Ungenauigkeiten

Die meisten dieser Ungenauigkeiten sind schon in Abschnitt 2.4 – Annahmen und Eingeständnisse beschrieben. Allerdings möchten wir hier eine Einschätzung geben, inwiefern sie das Ergebnis beeinflussen, beziehungsweise, ob durch die Beseitigung der Ungenauigkeiten ein besseres oder ein schlechteres Ergebnis zu erwarten wäre.

In der Betrachtung, ob der Wohnort eines Agenten A oder ein Universitätsstandort nahe genug an der Strecke eines Agenten B liegt, wird nicht berücksichtigt, ob es an dem Punkt auf der Strecke Parkmöglichkeiten gibt. Die aktuelle Verteilung dieser Orte ist in Abbildung 4.8 zu sehen. Die Abbildung 4.9 zeigt einen Ausschnitt, in dem zu erkennen ist, dass einige Punkte für die Mitnahme von Passagieren auf der Autobahn (A3) liegen. Obwohl hier auch einige ungünstige Orte zustande kommen, scheint das nur

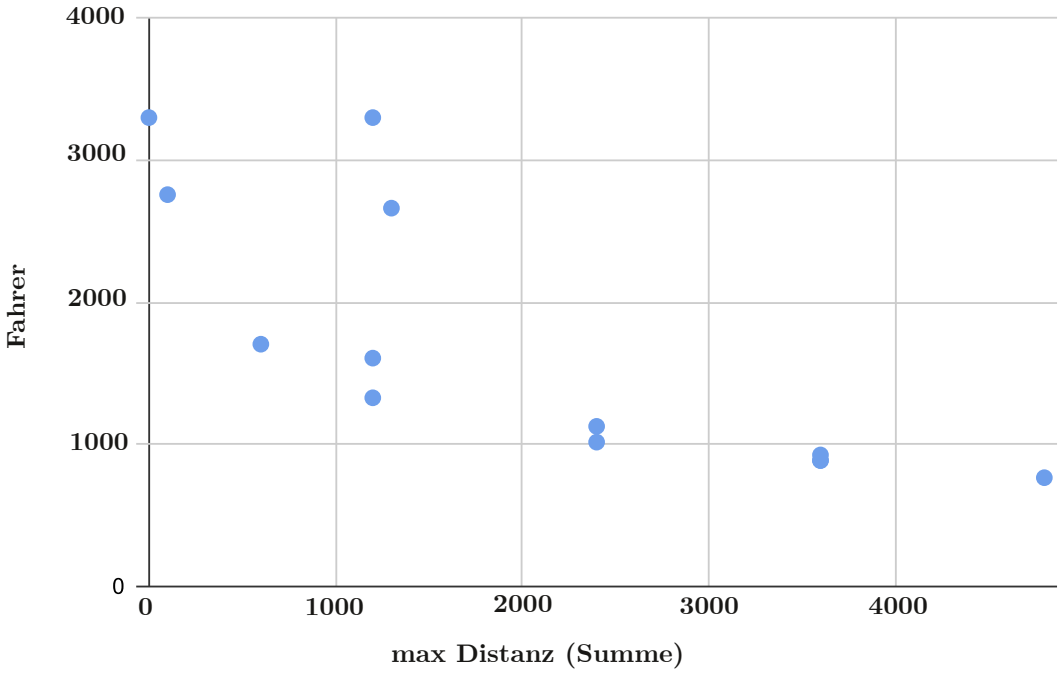


Abb. 4.6: Anzahl an Fahrern bei verschiedenen maximalen Distanzen zu Fuß

ein kleiner Teil der gesamten Menge an berechneten Punkten zu sein. Allerdings wäre es sehr aufwendig, alle Haltemöglichkeiten zu finden, unter anderem, da für das Ein- oder Aussteigen nicht nur Parkplätze infrage kommen, sondern auch zum Beispiel manche Straßenränder. Würde man das berücksichtigen, gäbe es weniger mögliche Matches. Entweder, es gibt eine Haltemöglichkeit, oder das Match kommt nicht zustande. Aktuell wird dieses Problem nicht gelöst, da wir annehmen, dass der Fahrer in diesem Fall den Passagier abholen kann, ohne zu viel Zeit zu verlieren, da die Distanz nicht deutlich über $2,4$ km gesamt betragen sollte, was im Regelfall nur wenige Minuten in Anspruch nimmt. In der Realität wären für die Strecke insgesamt sogar nur etwa $1,2$ km zu erwarten, was der tatsächliche Durchschnitt ist. Hier ist außerdem zu erwähnen, dass durch unseren Algorithmus die gesamt zu laufende Strecke deutlich überschätzt wird.

Eventuell ist auch nicht jeder Agent bereit oder in der Lage, die hier angenommenen 1200 m zu Fuß zurückzulegen. Andere Agenten wiederum könnten auch bereit sein, weiter zu laufen. Etwa, weil sie statt zu Fuß mit dem Fahrrad zur Strecke kommen würden. Wenn sich die akzeptierte Strecke insgesamt verringert, ist von weniger Matches auszugehen. Umgekehrt können dann mehr mögliche Matches zustande kommen, wenn die Agenten bereit sind, sich weiter zur Route eines anderen Agenten zu bewegen. Diese Annahmen basieren auf dem Experiment, dessen Ergebnisse in Tabelle 4.7 zu finden sind, bei dem im Allgemeinen für höhere akzeptierte Distanzen die Anzahl an einzelnen Fahrern niedriger war. Wir gehen aber hier in den Experimenten immer von einer festen Distanz für alle Agenten aus, was in der Realität nicht angenommen werden kann.

Außerdem arbeiten wir im Standardfall mit Agenten die $2,0$ und $25,0$ km vom jewei-

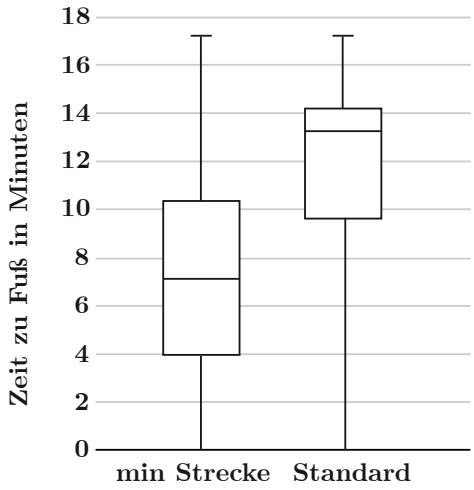


Abb. 4.7: Zeiten zu Fuß vom Startort zur Strecke

ligen Universitätsstandort entfernt wohnen. Dabei ist zu beachten, dass viele der nah wohnenden Agenten vermutlich eher andere Verkehrsmittel nutzen würden.

Dadurch, dass bei verschiedenen Zielorten nur die Fahrzeit bis zum nächsten Punkt betrachtet wird, müsste sich ein Mitfahrer selbst darum kümmern, trotzdem rechtzeitig anzukommen. Daher lassen wir für den Standardfall nur übereinstimmende Standorte für Matches zu und betrachten die Ergebnisse in Tabelle 4.7 als Indikator dafür, wie sich die Anzahl an von den lokalen Gegebenheiten passenden Matches erhöht, wenn die zeitlich möglichen Matches gleich bleiben. Dadurch können wir den Einfluss der Distanz betrachten, ohne gleichzeitig eine der Zeit geschuldete Veränderung vorliegen zu haben. Dann können wir diesen ohne den Störfaktor auswerten.

Wenn Mitfahrer dadurch nur mehr Zeit einberechnen müssen, dann wirkt sich das nicht negativ auf die Anzahl der möglichen zeitlichen Matches aus (da es durch die Art der Implementierung darauf im Allgemeinen keinen Einfluss hat), sondern nur auf die Gesamtzeit der Agenten, sowie die theoretische Zufriedenheit der Passagiere. Durch eine höhere mögliche Distanz können trotzdem mehr potenzielle Matches zustande kommen, da weniger potenzelle Fahrer durch nicht übereinstimmende Universitätsstandorte herausgefiltert werden. Allerdings wäre es bei zufälliger Zuweisung für die Mitfahrer selbst nicht möglich, diese Distanz, beziehungsweise die benötigte Zeit, um diese zurückzulegen, einzuplanen, da einem Passagier vorher nicht bekannt ist, ob er eventuell bei einer Distanz von maximal 1200 m zwischen dropoff-Punkt und Zielort bis zu 15 Minuten zusätzlich laufen muss. Da dann jeder Studierende vorsichtshalber diese 15 Minuten einplanen müsste, aber dann eventuell deutlich eher ankommt als gewünscht, halten wir den möglichen Einfluss auf die Zufriedenheit der Mitfahrer für sehr groß. Daher müsste man diese Zeit für das Matching mit einberechnen. Aber selbst wenn die Zeit zu Fuß beachtet wird, kann die Größe der Universitätsstandorte dazu führen, dass der Mitfahrer zu früh oder zu spät ankommt, da der dropoff-Punkt unter Umständen näher an dem

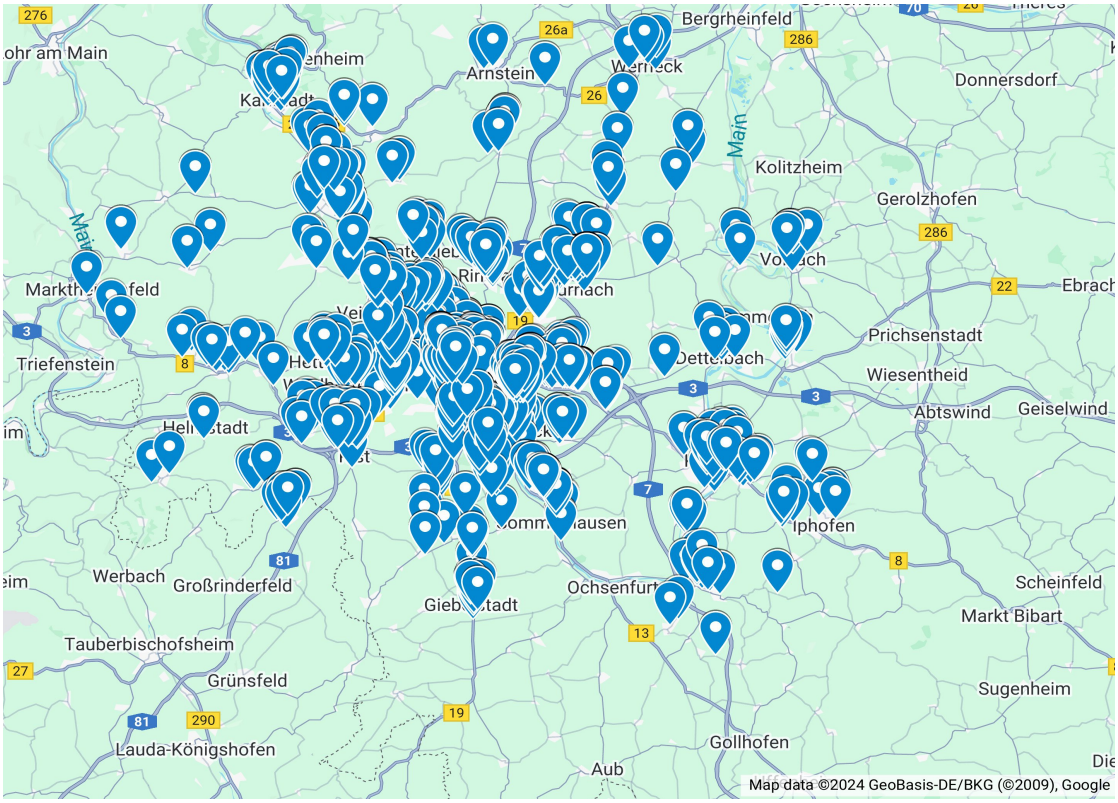


Abb. 4.8: Ein- und Aussteigepunkte, erstellt auf https://www.google.com/maps/d/

Ort liegt, zu dem der Passagier eigentlich gelangen möchte, als an der Koordinate, die für den entsprechenden Universitätsstandort festgelegt wurde. Aber da der Ankunftsort festgelegt ist, wäre es für Passagiere einfach, die Zeit einzuplanen, die sie vom festgelegten Ankunftsort zum tatsächlichen Zielort benötigen, wenn sie am Ankunfstort abgestzt werden.

Den Algorithmus so umzuschreiben, dass die Ankunft am für den Universitätsstandort festgelegten Ort im gewünschten Zeitfenster geschieht, kann, genauso wie dass die Haltezeit für andere aussteigende Agenten nicht beachtet wird, je nach Datensatz zu mehr oder zu weniger Matches führen. Für ein Match muss eine Überschneidung von mindestens einer Minute vorliegen. Dafür sind die präferierten Zeiten der Agenten ausschlaggebend. Beide Versionen können zufällig die Anzahl an Matches positiv oder negativ beeinflussen, aber das lässt sich für einen Datensatz nicht voraussagen, ohne alle Zeiten zu überprüfen.

Dass jedes Auto hier fünf Sitze hat, halten wir für eine passende Annahme. Es gibt zwar Fahrzeuge mit weniger oder mehr Sitzen, aber, dass im Durchschnitt vier Passagiere mitgenommen werden können, scheint eine gute Annahme zu sein, die die Ergebnisse im Durchschnitt nicht deutlich verändern sollte. Das bedeutet genauer, wir gehen davon aus, dass einige wenige Fahrzeuge mit mehr oder weniger als fünf Plätzen das Ergebnis nicht signifikant verändern würden, sondern, dass eine signifikante Anzahl an in der

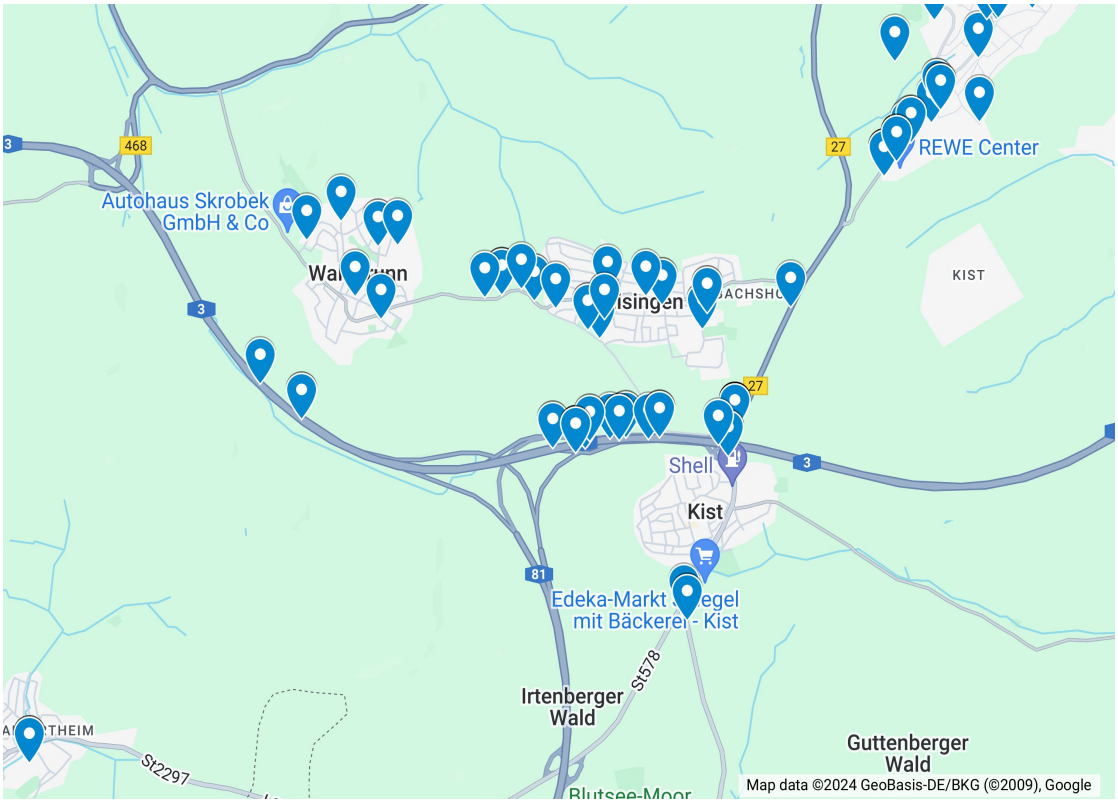


Abb. 4.9: Ein- und Aussteigepunkte, Ausschnitt, erstellt auf https://www.google.com/maps/d/

Sitzplatzanzahl deutlich abweichenden Fahrzeugen vorliegen müsste. Wir haben gezeigt, dass eine gesamt veränderte Sitzplatzanzahl das Ergebnis etwas verändert, aber hier gehen wir davon aus, dass der Durchschnitt der Sitzplatzanzahl auch dann gleich bleibt, wenn wir Abweichungen nach oben und unten für einzelne Fahrzeuge zulassen.

Wir nehmen an, dass Agenten nur dann die Fahrt zur Universität antreten würden, wenn auch eine Mitfahrgelegenheit für den Rückweg vorhanden ist. Grundsätzlich halten wir dies für eine bessere Annahme als die, dass jeder Agent die Mitfahrgelegenheit auch ohne gesicherte Rückfahrt annehmen würde. In der Realität wäre die beste Annahme vermutlich, dass manche Agenten auch eine Mitfahrgelegenheit nur für die Hin- oder die Rückfahrt akzeptieren würden, da sie alternative Optionen haben. Allerdings müsste man dafür zunächst Zahlen ermitteln, wie groß diese Anteile tatsächlich sind. Durch das Berücksichtigen dieser Möglichkeit könnte es sogar zu mehr Matches kommen, da die Anzahl an Einschränkungen für mögliche Matches reduziert wird.

Der hier genutzte Algorithmus bearbeitet nur eine statische Version des Ridesharing-Problems. Grundsätzlich halten wir dies für eine relativ gute Approximation für ein Ridesharing-Problem, bei dem Pendler betrachtet werden. Allerdings gibt es in der Realität auch immer kurzfristige Planänderungen. Je nachdem, ob weniger oder mehr Agenten Fahrten antreten möchten, kann das dann zu mehr, beziehungsweise weniger Matches

führen.

Auch dass die Strecke nur einmal für die Hinrichtung berechnet wird, kann positive oder negative Auswirkungen auf das Ergebnis haben. Je nachdem, wie sehr sich die Strecke für die Rückrichtung unterscheidet, kann es zu größeren Diskrepanzen in der Menge der möglichen Matches kommen. In welche Richtung dies das Ergebnis beeinflusst, lässt sich so nicht pauschal voraussagen.

In unseren Annahmen hat jeder Agent die Möglichkeit Fahrer oder Passagier zu sein. In der Realität besitzt aber nicht jeder ein Auto. Das würde die Rollenauswahl einschränken und dadurch vermutlich zu schlechteren Ergebnissen führen. Außerdem wird ebenso angenommen, dass es keine Präferenz für Rollen gibt, wie zum Beispiel Agenten, die zwar bereit sind, am System teilzunehmen, aber nur als Fahrer. Das würde wie alle Einschränkungen zu einem Ergebnis mit weniger Matches führen.

Auch andere potenzielle Präferenzen werden durch das bisher zufällige Matching ignoriert. In der Realität kann es vorkommen, dass eine Person mit einer anderen kein Auto teilen möchte. Ein naheliegendes Beispiel wäre hier ein Nichtraucher, der nicht in ein Auto steigen möchte, in dem geraucht wird oder wurde. Umgekehrt möchte er unter Umständen nur andere Nichtraucher im eigenen Auto. Besonders junge, weibliche Mitfahrer, oder Fahrer, können sich auch grundsätzlich unwohl dabei fühlen, mit fremden Menschen in einem Auto zu sitzen. Zum Beispiel bei BlaBlaCar[Blab] wird das mitigiert, indem alle Mitglieder überprüft werden, wobei Ausweisdokumente vorliegen müssen. In dieser Arbeit wird davon ausgegangen, dass alle Teilnehmer allen anderen vertrauen können und das auch tun. In der Realität muss das aber nicht zutreffen und da es um individuelle Persönlichkeiten geht, ist der Punkt der Präferenzen hier schwierig zu berücksichtigen und es setzt deutlich mehr Recherche voraus, Agenten mit realistischen Präferenzen zu generieren. Aber für jede zusätzliche Einschränkung ist ein Ergebnis mit weniger Matches zu erwarten.

Zusammenfassend lässt sich sagen, dass die Anzahl an Einschränkungen den großen Unterschied macht. Wenn Einschränkungen wegfallen, wie zum Beispiel, wenn Agenten auch nur Fahrten in eine Richtung akzeptieren würden, verbessert sich das Ergebnis potenziell. Genauso kann es sich deutlich verschlechtern, je mehr zusätzliche Einschränkungen, zum Beispiel für persönliche Präferenzen der Agenten, beachtet werden.

5 Ausblick

Zusammenfassend lässt sich sagen, dass viel Potenzial besteht, um mit Ridesharing-Programmen CO_2 einzusparen und weniger Autos auf den Straßen zu haben. Allerdings ist das Ergebnis sehr davon abhängig, wie viele Agenten sich im System befinden. Um das gewünschte Ergebnis von einer 90%igen Matchingrate zu erreichen, müssen in der simulierten Situation etwa 3500 Agenten am Ridesharing-Programm teilnehmen. Dabei ist die Verteilung dieser Agenten ähnlich wie die in den genutzten Datensätzen, nur die Dichte ist verringert. Das bedeutet, dass das Ergebnis nur eine realistische Mindestanzahl für ein reales Ridesharing-Programm liefern kann, wenn die Verteilung der tatsächlich teilnehmenden Agenten zufällig ist. Wenn allerdings zum Beispiel bevorzugt weiter entfernt (oder näher) wohnende Studierende bereit wäre, an einem solchen Angebot teilzunehmen, dann ist nicht davon auszugehen, dass diese Anzahl akkurat ist. Wie Experiment 6 gezeigt hat, finden sich weniger Matches und es müssen mehr Fahrer alleine fahren, wenn wir die minimale Distanz zum Universitätstsandort erhöhen. Vermutlich würden viele Studenten die nahe am Universitätsstandort wohnen, andere Verkehrsmittel nutzen.

Für die Zukunft besteht unter anderem die Möglichkeit, die in Abschnitt 2.4 erwähnten möglichen Ungenauigkeiten zu verringern. Dafür könnte man Agenten zufällige Präferenzen zuordnen und nur dann ein Match erzeugen, wenn die Präferenzen passen. So kann man den Agenten zusätzliche Attribute zuordnen, wie eine bevorzugte Rolle, Raucherstatus, Tierhaarallergien und so weiter. Diese können dann entweder als Muss- oder als Kann-Kriterien behandelt werden. Für Muss-Kriterien bietet sich ein einfacher Filter an, bei dem unpassende Agenten nicht als mögliche Mitfahrer betrachtet werden. Darunter würden zum Beispiel starke Allergien fallen, oder Agenten, die kein Auto besitzen. Für optionale Kriterien bietet sich unter Umständen eine Implementierung an, bei der für jedes nicht beachtet Kriterium Strafpunkte vergeben werden. Dann kann man zusätzlich zu den aktuellen Zielen auch versuchen, die Strafpunkte zu minimieren.

Auch die anderen Ungenauigkeiten können zum Teil beseitigt werden. So würde es zwar mehr Rechenaufwand darstellen, aber man kann zum Beispiel Routen für den Hin- und Rückweg separat berechnen und betrachten. Außerdem kann man die Fahrzeuge ebenfalls zufällig mit variablen Eigenschaften wie Kraftstoffverbrauch und Sitzplatzanzahl generieren und dann diejenigen Agenten als Fahrer bevorzugen, die umweltfreundlichere Fahrzeuge besitzen oder mehr Plätze anbieten können. Die höhere Sitzplatzanzahl zu bevorzugen ist aber nach den Ergebnissen in Tabelle 4.9 und Tabelle 4.10 eventuell sogar die schlechtere Wahl. Die Ergebnisse unter Bevorzugung von verschiedenen Fahrzeugeigenschaften von Fahrzeugen, die jeweils bestimmte Werte, für zum Beispiel

die Kapazität und den Treibstoffverbrauch besitzen, zu testen, wäre auch ein weiteres interessantes Experiment.

Statt anzunehmen, dass jeder auf der Strecke mitgenommen werden kann, wäre es möglich, alle Haltemöglichkeiten im Umkreis von Würzburg zu finden und zu speichern. Mit der aktuellen Implementierung ergeben sich auch teilweise ungünstige Haltemöglichkeiten, wie zum Beispiel auf größeren Straßen außerhalb von Ortschaften, wie Abbildung 4.9 zeigt. Alle Haltemöglichkeiten zu finden, ist ein großer Aufwand, der aber nur einmal durchgeführt werden muss. Die gefundenen Koordinaten können dann mit den Routen der Agenten verglichen werden. Daraufhin muss man die Wohnorte der Agenten nur noch mit den Koordinaten der Haltemöglichkeiten vergleichen und nicht mehr mit den gesamten Routen der anderen Agenten. Diesen Schritt kann man aber auch umgekehrt durchführen und zuerst mit den Wohnorten vergleichen. Welche Version man wählt, kommt auf den genutzten Algorithmus an. Wenn Umwege in Kauf genommen werden sollen, eignet sich der hier vorgeschlagene Algorithmus nicht gut. Es wäre möglich, Strecken aus einigen ähnlich guten Optionen so zu wählen, dass die mit den meisten nahen Wohnorten von anderen Agenten gewählt wird. Allerdings kann es dadurch passieren, dass die Agenten auf den weniger bewohnten Strecken keine Mitfahrgelegenheiten mehr finden, da alle Fahrer umgeleitet werden, und sie dann selbst zu Fahrern werden müssen. Eine Option wäre es, die Strecken aus den Optionen anhand von Wahrscheinlichkeiten zu wählen, die aus der Anzahl der Agenten auf der jeweiligen Route berechnet werden.

Direkt die nächsten Anhaltepunkte für jeden Agenten zu finden, würde auch das Problem der inakkurat berechneten Laufstrecken zum Teil lösen. In der aktuellen Implementierung nutzen wir statt der kürzesten Strecke die zum ersten passenden Punkt. Da die Matchings dadurch gleich bleiben wäre es hier auch möglich, die Berechnung der Distanzen zum Schluss vor der Ausgabe für alle entstandenen Matches erneut durchzuführen und dabei nach der kürzesten Distanz zu suchen. Aber wenn wir nur die Haltepunkte mit der Route vergleichen, die nahe genug am Wohnort eines Agenten liegen, könnten wir daraus dann direkt den mit der kürzesten Distanz zum Startpunkt wählen.

Es ist auch möglich, die Laufzeit des Algorithmus weiter zu verkürzen, indem die Anzahl der GraphHopper-Nutzungen weiter reduziert wird. So kann man zum Beispiel, ähnlich wie im Algorithmus von Helena Fehler[Feh24], die Vergleiche weiter lokal einschränken, indem man nur noch Vergleiche mit anderen Agenten zulässt, die in einer ähnlichen Richtung wohnen. Wenn man so zusätzlich Zeit gewinnen kann, oder grundsätzlich schon mehr Laufzeit akzeptiert oder mehr Ressourcen zur Verfügung stehen, ist es möglich and anderen Stellen mehr Zeit zu investieren, wie zum Beispiel für genauere Ergebnisse nach der kürzesten Laufstrecke zwischen dem Heimatort eines Agenten und der Route eines anderen zu suchen, statt irgendeine zu akzeptieren.

Außerdem stellt die nutzerorientierte Betrachtung eine Basis für die Erweiterung um eine variable Nachfrage dar. Dafür würde man die Simulation für einen längeren Zeitraum durchführen. Agenten, denen zu oft kein Match zugeordnet wurde, nehmen nicht weiter am System teil und stellen keine Anfragen mehr. Im Gegensatz dazu empfehlen Agenten,

die über einen längeren Zeitraum viele Matches erhalten haben, das System eventuell weiter, was für die Simulation bedeutet, dass neue Agenten hinzugefügt werden. Solch eine dynamische Nachfrage bildet die Realität besser ab, als eine statische Betrachtung.

Insgesamt gibt es im Bereich Ridesharing viele weitere Möglichkeiten, ein realistischeres Ergebnis zu erhalten. Aber da die Realität durch den Einfluss von Individuen oft zu kompliziert ist, um sie vollkommen zu simulieren, besteht nur die Möglichkeit, die Simulation immer weiter an die Realität anzupassen.

Literaturverzeichnis

[AB20] Vincent Armant und Kenneth N. Brown: Fast optimised ridesharing: Objectives, reformulations and driver flexibility. *Expert Systems with Applications*, 141:112914, 2020, https://doi.org/10.1016/j.eswa.2019.112914, ISSN 0957-4174. https://www.sciencedirect.com/science/article/pii/S0957417419306323.

[ADA] ADAC: ADAC Pendlernetz App. https://www.adac.de/services/apps/pendlernetz/, besucht: 2024-07-16.

[AESW11] Niels A.H. Agatz, Alan L. Erera, Martin W.P. Savelsbergh und Xing Wang: Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological*, 45(9):1450–1464, 2011, https://doi.org/10.1016/j.trb.2011.05.017, ISSN 0191-2615. https://www.sciencedirect.com/science/article/pii/S0191261511000671, Select Papers from the 19th ISTTT.

[Bar24] Jonas Barth: Masterpraktikumsbericht. not yet finished as of the 21.07.2024, 2024.

[Blaa] GooglePlay - BlaBlaCar. https://play.google.com/store/apps/details?id=com.comuto, besucht: 2024-07-16.

[Blab] BlaBlaCar: BlaBlaCar. https://www.blablacar.de/carpool, besucht: 2024-07-16.

[Blac] BlaBlaCar: Über die Servicegebühr bei Mitfahrgelegenheiten. https://support.blablacar.com/hc/de/articles/360014533360-Über-die-Servicegebühr-bei-Mitfahrgelegenheiten, besucht: 2024-07-16.

[Blad] BlaBlaCar: Über Mitfahrten. https://support.blablacar.com/hc/de/articles/360015367779-Über-Mitfahrten, besucht: 2024-07-16.

[BMM04] Roberto Baldacci, Vittorio Maniezzo und Aristide Mingozzi: An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation. *Operations Research*, 52(3):337–497, 2004, 10.1287/opre.1030.0106. https://doi.org/10.1287/opre.1030.0106.

[CL07] Jean François Cordeau und Gilbert Laporte: The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007, 10.1007/s10479-007-0170-8.

[CS11] Nelson D Chan und Susan A Shaheen: Ridesharing in North Ameri-

ca: Past, Present, and Future. *Transport Reviews*, 32(1):93–112, 2011, 10.1080/01441647.2011.621557.

[dCG22] Vinicius Renan de Carvalho und Fatemeh Golpayegani: Satisfying user preferences in optimised ridesharing services:. *Applied Intelligence*, 52(10):11257–11272, 2022, 10.1007/s10489-021-02887-1. https://doi.org/10.1007/s10489-021-02887-1.

[FDO+13] Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc Etienne Brunet, Xiaoqing Wang und Sven Koenig: Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013, https://doi.org/10.1016/j.trb.2013.08.012, ISSN 0191-2615. https://www.sciencedirect.com/science/article/pii/S0191261513001483.

[Feh24] Helena Fehler: Modeling and Evaluation of Sustainable Mobility Concepts for the Campus Mobility. https://www.informatik.uni-wuerzburg.de/fileadmin/10030100/2024/Update_-_Modeling_and_Evaluation_of_Sustainable_Mobility_Concepts_for_the_Campus_Mobility_-_Helena_Fehler.pdf, 2024.

[fWA] Marktplatz GmbH Agentur für Web App: Pendlerportal. https://www.pendlerportal.de/help, besucht: 2024-07-16.

[gh] GraphHopper. https://github.com/graphhopper/graphhopper, besucht: 2024-07-21.

[GL23] Chung Piaw Teo Hai Wang Guodong Lyu, Wang Chi Cheung: Multiobjective Stochastic Optimization: A Case of Real-Time Matching in Ride-Sourcing Markets. *Manufacturing Service Operations Management*, 26(2):500–518, 2023, 10.1287/msom.2020.0247.

[JCMW18] Shan Jiang, Le Chen, Alan Mislove und Christo Wilson: On Ridesharing Competition and Accessibility: Evidence from Uber, Lyft, and Taxi. In: *Proceedings of the 2018 World Wide Web Conference*, WWW '18, Seite 863–872, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee, ISBN 9781450356398, 10.1145/3178876.3186134. https://doi.org/10.1145/3178876.3186134.

[Kel07] Kalon L. Kelley: Casual Carpooling—Enhanced. *Journal of Public Transportation*, 10(4):119–130, 2007, https://doi.org/10.5038/2375-0901.10.4.6, ISSN 1077-291X. https://www.sciencedirect.com/science/article/pii/S1077291X22003113.

[Laba] Oak Ridge National Laboratory: National Household Travel Survey 2022. https://nhts.ornl.gov, besucht: 2024-07-21.

[Labb] Oak Ridge National Laboratory: Transportation Energy Data Book 40. https://tedb.ornl.gov, besucht: 2024-07-21.

[LHA20] Chaya Levinger, Noam Hazon und Amos Azaria: Human satisfacti-

on as the ultimate goal in ridesharing. *Future Generation Computer Systems*, 112:176–184, 2020, https://doi.org/10.1016/j.future.2020.05.028, ISSN 0167-739X. https://www.sciencedirect.com/science/article/pii/S0167739X19302249.

[LQTF24] Chijia Liu, Alain Quilliot, Hélène Toussaint und Dominique Feillet: A filtering system to solve the large-scale shared autonomous vehicles Dial-a-Ride Problem. *Transportation Research Part C: Emerging Technologies*, 161:104551, 2024, https://doi.org/10.1016/j.trc.2024.104551, ISSN 0968-090X. https://www.sciencedirect.com/science/article/pii/S0968090X2400072X.

[LTY+20] Xiaobing Liu, Helena Titheridge, Xuedong Yan, Rui Wang, Weimin Tan, Deqi Chen und Jiechao Zhang: A passenger-to-driver matching model for commuter carpooling: Case study and sensitivity analysis. *Transportation Research Part C: Emerging Technologies*, 117:102702, 2020, https://doi.org/10.1016/j.trc.2020.102702, ISSN 0968-090X. https://www.sciencedirect.com/science/article/pii/S0968090X20306173.

[MJ17] Neda Masoud und R. Jayakrishnan: A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. *Transportation Research Part B: Methodological*, 106:218–236, 2017, https://doi.org/10.1016/j.trb.2017.10.006, ISSN 0191-2615. https://www.sciencedirect.com/science/article/pii/S0191261517301169.

[MYSJ19] Ruimi Ma, Lifei Yao, Lijun Song und Maozhu Jin: A novel algorithm for peer-to-peer ridesharing match problem. *Neural Computing and Applications*, 31(1):247–258, 2019, 10.1007/s00521-018-3733-5. https://doi.org/10.1007/s00521-018-3733-5.

[NR16] Mehdi Nourinejad und Matthew J. Roorda: Agent based model for dynamic ridesharing. *Transportation Research Part C: Emerging Technologies*, 64:117–132, 2016, https://doi.org/10.1016/j.trc.2015.07.016, ISSN 0968-090X. https://www.sciencedirect.com/science/article/pii/S0968090X15002661.

[PLH19] James J. Pan, Guoliang Li und Juntao Hu: Ridesharing: simulator, benchmark, and evaluation. *Proc. VLDB Endow.*, 12(10):1085–1098, jun 2019, 10.14778/3339490.3339493, ISSN 2150-8097. https://doi.org/10.14778/3339490.3339493.

[PXZ+15] Dominik Pelzer, Jiajian Xiao, Daniel Zehe, Michael H. Lees, Alois C. Knoll und Heiko Aydt: A Partition-Based Match Making Algorithm for Dynamic Ridesharing. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2587–2598, 2015, 10.1109/TITS.2015.2413453.

[rG] ride2Go GmbH: fahrgemeinschaft.de. https://fahrgemeinschaft.de, besucht: 2024-07-16.

[SASG16] Mitja Stiglic, Niels Agatz, Martin Savelsbergh und Mirko Gradisar: Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review*, 91:190–207, 2016, https://doi.org/10.1016/j.tre.2016.04.010, ISSN 1366-5545. https://www.sciencedirect.com/science/article/pii/S1366554515303033.

[SCZ20a] Yanshuo Sun, Zhi Long Chen und Lei Zhang: Nonprofit peer-to-peer ridesharing optimization. *Transportation Research Part E: Logistics and Transportation Review*, 142:102053, 2020, https://doi.org/10.1016/j.tre.2020.102053, ISSN 1366-5545. https://www.sciencedirect.com/science/article/pii/S1366554520307043.

[SCZ20b] Yanshuo Sun, Zhi Long Chen und Lei Zhang: Nonprofit peer-to-peer ridesharing optimization. *Transportation Research Part E: Logistics and Transportation Review*, 142:102053, 2020, https://doi.org/10.1016/j.tre.2020.102053, ISSN 1366-5545. https://www.sciencedirect.com/science/article/pii/S1366554520307043.

[SSW98] Susan Shaheen, Daniel Sperling und Conrad Wagner: Carsharing in Europe and North American: Past, Present, and Future. Institute of Transportation Studies, Research Reports, Working Papers, Proceedings qt4gx4m05b, Institute of Transportation Studies, UC Berkeley, Januar 1998. https://ideas.repec.org/p/cdl/itsrrp/qt4gx4m05b.html.

[Sta] Statistica: How the World Commutes. https://www.statista.com/chart/25129/gcs-how-the-world-commutes/, besucht: 2024-07-21.

[Tir20] Alejandro Tirachini: Ride-hailing, travel behaviour and sustainable mobility: an international review. *Transportation*, 47(4):2011–2047, 2020, 10.1007/s11116-019-10070-2, ISSN 1572-9435. https://doi.org/10.1007/s11116-019-10070-2.

[VT] Inc. Via Transportation: via. https://my.drivewithvia.com, besucht: 2024-07-16.

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Versicherung zur selbständigen Leistungserbringung

Titel der Bachelorarbeit:

Simulation eines neuen Matching-Algorithmus' für Studierenden-Ridesharing

Thema bereitgestellt von (Titel, Vorname, Nachname, Lehrstuhl):
Prof. Dr. Marie Schmidt, Lehrstuhl für Informatik I

Eingereicht durch (Vorname, Nachname, Matrikel):
Marie Holfelder, 2607493

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, entsprechend den Vorgaben der Prüfungsleiterin bzw. des Prüfungsleiters 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.

Iphofen, den 24.07.2024 Marie Holfelder
Ort, Datum, Unterschrift

Ref. 2.3 / Stand: 24.11.2023