Hochschule Niederrhein
University of Applied Sciences

SWK E^2 Working Paper 3/2023

Modellierung von Ridepooling-Angeboten mit unterschiedlichen Service Level Agreements

Luca Ciecierski, Marc Gennat

Juni 2023

SWK E^2
Institut für Energietechnik und Energiemanagement
Hochschule Niederrhein

SWK E^2 Institut für Energietechnik und Energiemanagement

Hochschule Niederrhein
Reinarzstraße 49
47805 Krefeld

Tel.: +49 (0) 2151-822 6693
E-Mail: energiezentrum@hs-niederrhein.de
Internet: https://www.hs-niederrhein.de/swk-e2/publikationen/

Luca Ciecierski, Prof. Dr.-Ing. Marc Gennat
E-Mail: lucaciecie@hotmail.de
marc.gennat@hs-niederrhein.de

ISSN: 2751-0050

Kurzzusammenfassung

Im Rahmen der Verkehrswende werden neue Möglichkeiten der Mobilität erforscht. Ein On-Demand Ridepooling-Service bietet Kunden eine effiziente Erweiterung dieser Mobilität. In diesem Working Paper wird ein Algorithmus implementiert, um Fahranfragen zu bündeln. Es wird geprüft, wie viele Kosten in Form von Zeit und Fahrwegen, durch die Kombinierung einzelner Fahranfragen, gespart werden können. Zusätzlich soll ermittelt werden, wie verschiedene Service Level Agreements aussehen können, um potentiellen Kunden ein breites Spektrum an individuellen Möglichkeiten zu bieten. Dabei wird der untersuchte Raum auf den Bezirk Fischeln der Stadt Krefeld begrenzt. Mit dem vorgestellten Algorithmus wird eine Grundlage zum Theorie-Praxis-Transfer in den realen Straßenverkehr gegeben.

Abstract

As transport changes, new mobility options are being explored. An on-demand ride-pooling service offers customers an efficient extension of this mobility. This paper implements an algorithm that aggregates travel requests. The cost, in terms of time and trips, is examined to determine how many individual trips can be combined. Furthermore, it is investigated how different service level agreements can be designed in order to offer a wide range of individual options to potential customers. The study area is limited to the district of Fischeln in the city of Krefeld. The presented algorithm will serve as a basis for a theory-practice transfer to real road traffic.

Keywords: Ridepooling auf Abruf, Service Level Angebote, Ridepooling-Problem, Mobilität

Inhaltsverzeichnis

1. Einleitung

Das Thema Mobilität ist in der deutschen Gesellschaft auf allen Ebenen präsent. Motorisierte Fahrzeuge dominieren in diesem Zusammenhang den Personentransport. Mit 78 Prozent hält der motorisierte Individualverkehr (MIV) den größten Anteil der privaten Verkehrsleistung [1]. Der MIV umfasst alle motorisierten Straßenverkehrsteilnehmer von Privatpersonen, wie Personenkraftwagen (PKW), Motorräder und Motorroller. Zum MIV zählen sowohl Privatautos als auch Autos von Taxiunternehmen oder Mietwagen. Privatpersonen nutzen am liebsten das Auto im Rahmen des MIV.

Für den Personennahverkehr werden ebenfalls Taxis genutzt. Um Taxis in das Prinzip des Ridepooling einzubinden und in 2.2 beschriebenen Vorteile auszuweiten, gilt es, ein geeignetes Modell zu finden. Der Ridepooling-Dienst muss in der Lage sein, mehr als eine Anfrage gleichzeitig zu berücksichtigen und dabei wirtschaftlich agieren zu können. Forschung, Modelle und Anbieter existieren bereits. Spezifisch in Krefeld gibt es das SWCAR. Dieser Bus On-Demand-Service der SWK Mobil GmbH soll den ÖPNV ergänzen. Die SWK Mobil GmbH ist eine Tochterfirma der SWK Stadtwerke Krefeld AG. Dieses Modell, betrieben von ioki GmbH, ist ausschließlich in den Abendstunden verfügbar [2]. Daten aus diesem Ridepooling-Modell sind öffentlich nicht einsehbar, weshalb hier komplett unabhängig davon gearbeitet wird.

Im Vordergrund des Ridepooling-Modells dieses Working Papers steht die Minimierung der Verzögerungen, die durch das Zusammenlegen von Routen entsteht. Darüber hinaus sollen belastbare Aussagen zur

Ankunftszeit gegeben werden können. Die Zufriedenheit des Kunden muss im Vordergrund stehen, damit der Dienst tatsächlich genutzt wird. Hierzu soll die Berechnung der Abfahrt- und Ankunftszeit der Kundschaft mitgeteilt werden. Zusätzlich soll eine transparente Darstellung der Routenplanung, mögliche Verzögerungen und Angaben der gesparten Kosten wiedergegeben werden. Mit unterschiedlichen Service Level Agreements (SLA), besteht die Möglichkeit, ein breites Spektrum an Bedürfnissen zu bedienen.

2. Grundlagen

2.1. Multiple Traveling Salesman Problem

Ein wichtiger Teil der TSP-Familie ist das Multiple Traveling Salesman Problem (MTSP). In dieser Abwandlung existieren *m* Handlungsreisende, welche in einem Depot starten. Es gilt eine sinnvolle Route für jeden Reisenden zu finden, sodass alle Städte besucht werden. Weitere Abwandlungen sind beispielsweise die Anzahl der Reisenden, die Anzahl der Depots, wie viele Reisende tatsächlich eingesetzt werden, ob es eine zeitliche Frist gibt oder auch individuelle Anforderungen wie minimale und maximale Reisezeiten oder besuchte Städte [3].

Das Vehicle Routing Problem (VRP), welches von Logistik-Unternehmen angewandt wird, besitzt dabei die zusätzliche Anforderung einer begrenzten Kapazität. Damit gehört es zu der Gruppe der MTSP. Übertragen auf Ridepooling-Algorithmen kann die Kapazität als die Anzahl von Passagieren verstanden werden, welche in einem Fahrzeug Platz finden [4].

Taxiflottenplanung ist der direkte Gegensatz zum traditionellen Herbeirufen von Taxis, welches ineffizient ist. Die solidarische Planung soll verhindern, dass Taxis eine eigene Agenda ausführen, und möglicherweise in bereits von Taxis dicht besiedelten Bezirken nach Passagieren suchen [5].

2.2. Ridepooling Problem

Das RPP hat die Aufgabe, die Routen von Fahranfragen zusammenzulegen beziehungsweise zu kombinieren. Dabei wird zwischen statischen und dynamischen Systemen unterschieden. Bei statischen Systemen werden die Fahranfragen gesammelt, kombiniert und daraufhin wird ein Taxi geschickt, welches diese Route bedienen kann. In dem vorliegenden Working Paper wird sich auf den statischen Teil der Routenkombinierung konzentriert.

Im Vergleich dazu sammelt ein dynamisches System ebenfalls Fahranfragen, jedoch in einem sehr kurzen Zeitraum und prüft, ob ein Fahrzeug, welches bereits eine Route bedient, noch eben jene Fahranfrage zusätzlich bedienen kann. Nach Furuhata et al. [6] gibt es noch keinen Standard in der praktischen Ausführung, allerdings werden verstärkt dynamische Methoden untersucht [7]. Auf die unterschiedlichen Herangehensweisen dieser beiden Unterteilungen wird später in 2.2 näher eingegangen.

In der umfassenden Studie von Yu et al. [8] werden die positiven Auswirkungen des Ridepoolings in Peking untersucht. Es wird beschrieben, dass die Einsparung von Kraftstoff einen erheblichen Einfluss auf Energie und CO_2-Emissionen haben. Im besten Szenario würden 67 Prozent Energie und 57 Prozent CO_2-Ausstoß gespart werden. Damit dieser Fall eintrifft, muss die Elektromobilität ausgebaut werden und somit

die Durchführung aller Ridepooling Anfragen mittels Elektroautos stattfinden. Darüber hinaus sollen diese Elektroautos durch erneuerbare Energien betrieben werden. Außerdem wird in diesem Szenario davon ausgegangen, dass die Elektroautos durchschnittlich 70 Prozent weniger Ressourcen benötigen.

In Deutschland wird Ridepooling bereits erforscht und angeboten. Im Rahmen einer Potenzialanalyse des Verkehrsverbund Rhein-Ruhr (VRR) und Kompetenzcenters Digitalisierung (KCD) [9] wurde festgestellt, dass On-Demand Ridepooling-Angebote eine sinnvolle Ergänzung zum ÖPNV darstellen. Anbieter sind in Deutschland unter anderem: **MOIA** in Hamburg und Hannover, **ioki** in Krefeld und Hamburg, **Via** in Bielefeld und Mainz sowie **CleverShuttle** in Dresden und Braunschweig.

Eine Befragung durch Bitkom [10] zeigt, dass die Zufriedenheit von Ridepooling-Nutzenden grundsätzlich höher ist, als die von üblichen ÖPNV-Kunden. Im Jahr 2021 haben neun Prozent der Befragten häufig das Ridepooling-Angebot wahrgenommen. Im Gegensatz dazu nutzten durchschnittlich etwa 25 Prozent die Dienste des ÖPNV häufig.

Statische und dynamische Modelle

Grundsätzlich muss zur Lösung eines RPP die Route der Fahranfrage bekannt sein. Ein Passagier muss angeben wo die Fahrt gestartet und beendet wird. Bei der Überprüfung der Routen muss, wie auch in der traditionellen Routenführung, ein *shortest path* Algorithmus angewendet werden. Dieser berechnet den kürzesten Weg zwischen zwei Knoten in einem Graph. Hierfür sind Algorithmen mit $O(n^2)$ Zeitkomplexität üblich, die auf dem effizienten Dijkstra-Algorithmus basieren [11]. Dieser ist in der Lage bei positiven Kantengewichten, den Weg mit minimalem Gewicht zu wählen [12].

Furuhata et al. [6] beschreiben ebenfalls wie genau das Zusammenlegen von Routen unterteilt werden kann. Eine Route besteht aus der Verbindung von Start- und Zielort. Zwei Routen R_1 und R_2 werden schematisch verglichen, dabei gibt es vier Möglichkeiten zur Unterteilung der Muster von diesen Routen in Tabelle 1.

Tabelle 1: Routenmuster beim Ridepooling [6]

Identische Fahrten	beschreiben Routen, auf welchen alle Anfangs- und Zielorte identisch sind	$R_1 = R_2$
Teilmengen einer Fahrt	liegen vor, wenn eine Route in einer anderen enthalten ist	$R_1 \subset R_2$
Partielle Fahrten	bedeuten, dass Ridepooling nicht die einzige Form des Transports für einen oder mehrere Passagiere ist. Beispielsweise nutzt R_2 ein öffentliches Verkehrsmittel anschließend an das Ridepooling	$R_1 \cap R_2$
Verzögerte Fahrten durch Umwege	sind jene, in denen weder Anfangs- noch Zielorte identisch sind. Ein Fahrer muss von den jeweils optimalen Routen für R_1 oder R_2 abweichen, damit eine insgesamt effizientere Lösung zum zeitlichen Nachteil der Passagiere gefunden wird. Diese Verzögerung kann ebenfalls mit einer partiellen Fahrt verbunden sein, sodass auch hier das Ridepooling nicht der einzige Bestandteil der Fahrt ist	$R_1 \neq R_2$

Ob diese Routen statisch oder dynamisch bearbeitet werden sollen, hängt von der Zielsetzung ab. Eine statische Implementierung ist robust anzuwenden. Fahrer wissen vorher, welche Route sie nehmen müssen, und werden nicht spontan auf einen Umweg geschickt. Eine dynamische Implementierung spart z.B. mehr Fahrten ein. In dem Modell Santi et al. [11], welches eine statische Implementierung vorstellt, wurden etwa 50 Prozent der Fahrten geteilt. Routen im dynamischen Modell nach Alonso-Mora et al. [4] konnten

hingegen zu etwa 90 Prozent geteilt werden. Hierbei wurde ein Weg gefunden die vorhandene Flotte von 13.000 Fahrzeugen auf 3.000 zu reduzieren.

Beide dieser Modelle arbeiten mit einem *Shareability Graph*. In diesem Graph sind die kombinierbaren Fahrten gekennzeichnet und die zu bedienenden Fahrten werden daraus ausgewählt. Der Shareability Graph wird in Kapitel 3.3 näher erläutert, da diese Arbeit ein solches Modell umsetzt. Die dynamische Implementierung dieses Graphen ist in der Lage ein VRP mit Heuristiken an eine optimale Lösung zu führen. Die beschriebenen Heuristiken sollen es ermöglichen, auch Kleinbusse oder Taxis mit einer Kapazität größer als zwei für das Ridepooling einsetzen zu können. Welche Modelle die Anbieter in Deutschland anwenden ist meistens öffentlich nicht zugänglich. Ein Vergleich mit diesen Anbietern ist deshalb nicht möglich. Lediglich das Unternehmen via [13] gibt an, ein dynamisches Modell zu nutzen.

3. Methodik

3.1. Methodik des Ridepooling-Algorithmus

Aus der bisherigen Literatur wurde als Grundlage das statische Modell von Santi et al. [11] genutzt. In der vorliegenden Arbeit wird es als *Shareability-Modell* bezeichnet. Aufgrund der vorhandenen Datenvorlage, das Straßennetz von Krefeld, konnte dieses Modell transparent für die Erarbeitung des Ridepooling-Modells für Fischeln angewendet und entsprechend angepasst werden. Um ein Shareability-Modell umsetzen zu können, ist ein ausgearbeiteter Graph $G = (V, E)$ nötig, welcher das Straßennetz darstellt. Dieser Graph ist die Basis, auf der das weitere Modell aufgebaut wurde, um realistische Daten zu berechnen. Diese wurden vor Beginn der vorliegenden Arbeit von Lukas Spengler, Mitarbeiter des E^2 Instituts, kompiliert und zur Verfügung gestellt. Die Daten haben ihren Ursprung in OpenStreetMap [14]. Das genutzte Straßennetz von Krefeld baut auf einem *directed graph* auf. Ein Digraph hat richtungsbestimmende Kanten, welche wichtig sind, um Einbahnstraßen simulieren zu können.

Die Aufstellung und Berechnungen des vorliegenden Modells wurden mit dem Programm Matlab umgesetzt. In Matlab bestehen Graphen aus zwei verknüpften Tabellen. Diese repräsentieren einerseits Knoten und andererseits Kanten. Der Digraph besteht aus etwa V = 17.000 Knoten und E = 40.000 Kanten. Zu jedem Knoten liegen jeweils die Koordinaten in Form von Latitude und Longitude vor.

Diese Daten können anschaulich auf eine Karte projiziert werden. Abbildung 1 zeigt alle verfügbaren Kanten in Schwarz auf der Karte von Krefeld.

3.2. Filterung der Daten

Im ersten Schritt wurden die Daten gefiltert. Um dem Bezirk Fischeln zu entsprechen, wurden die Knoten außerhalb der Koordinaten lat. $= [51.2894; 51.3136]$, lon. $= [6.5714; 6.6038]$ verworfen.

Weil die durchschnittliche Geschwindigkeit normalerweise nicht die maximal erlaubte Geschwindigkeit ist und die durchschnittlichen Geschwindigkeiten noch nicht für alle Straßen bekannt ist, wurde eine generelle Geschwindigkeit von 20 $\frac{\text{km}}{\text{h}}$ angenommen. Dieser Wert ist im Raum Fischeln plausibel. Aus der Distanz d in Metern und der generellen Geschwidigkeit wird eine Zeit tt in Sekunden errechnet.

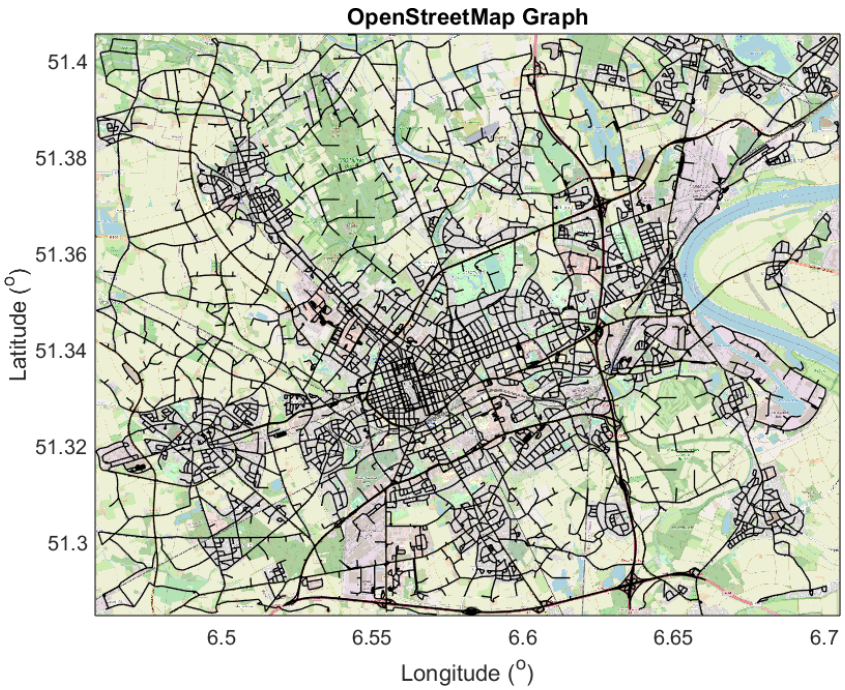

Abbildung 1: Karte Krefeld mit allen eingezeichneten Kanten

Zu beachten ist, dass bestimmte Straßentypen auch gefiltert werden müssen. Beispielsweise dürfen Taxis nicht auf dedizierten Fußgängerwegen fahren.

3.3. Erstellung der Fahrten und des Shareability-Graph

Mit dem gefilterten Digraph können nun Fahrten mit ihren entsprechenden Kosten definiert werden. Diese Fahrten werden als Synonym *Trips* benannt. Der Schwerpunkt des vorliegenden Modells liegt auf der gebündelten Bearbeitung eingehender Anfragen. In einem implementierten realen Modell müssen n-Trips gesammelt werden, beispielsweise in einem Zeitraum von fünf Minuten. Dieses Modell simuliert diese gebündelten Anfragen. Ein Trip wird mit mehreren Eigenschaften beschrieben. Und hier ist $T_i = \{o_i, d_i, st_i, pt_i, at_i, path_i, cost_i\}$ mit $i \in \{1, 2, ...n\}$ beschrieben worden. Tabelle 2 erläutert diese Attribute.

Tabelle 2: Eigenschaften eines Trip

o_i	Origin	Anfangsort
d_i	Destination	Zielort
st_i	Start time	Anfangszeitpunkt bzw. wann der Kunde am Anfangsort ist
pt_i	Pickup time	Abholzeit bzw. wann der Kunde ins Taxi steigt
at_i	Arrival time	Ankunftszeit bzw. wann der Kunde das Taxi verlässt
$path_i$		Der kürzeste Weg
$cost_i$		Kosten des Weges, hier in Resezeit

Zur Simulation wird ein zufälliger Anfangs- und Zielort bestimmt. Dies passiert mit der *randi*() Matlab-Funktion, welche eine pseudozufällige gleichverteilte Zahl generiert. In diesem Fall wurden für die beiden Orte jeweils eine Zahl, welche innerhalb der Knotenpunktanzahl liegt, bestimmt. Die Start- und Abholzeit wird hier als identisch angenommen. Die Kosten berechnen sich aus der Summierung tt aller Kanten die zum Zielort führen. In Matlab funktioniert dies mit der *shortestpath* [12] Funktion. Diese berechnet *cost* und *path*. Die Ankunftszeit erschließt sich folglich aus der Summe von Startzeit und Kosten. Es ist zu beachten,

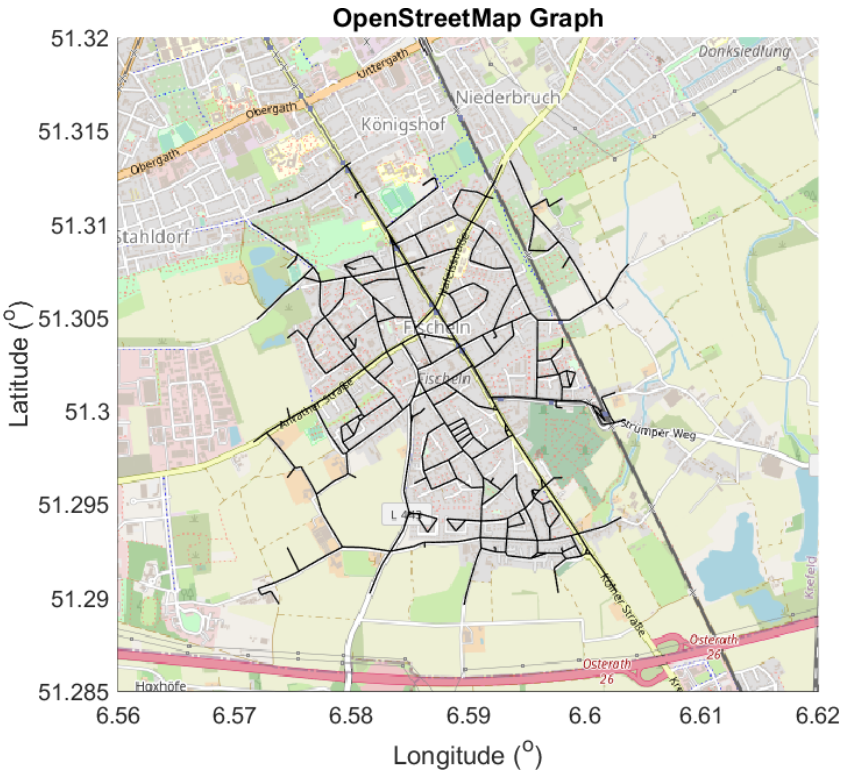

Abbildung 2: Karte Fischeln

dass bestimmte Trips grundlegend nicht rentabel sind. Aufgrund dessen werden diese Ausreißer ebenfalls gefiltert. Es wurde angenommen, dass keine Fahrten unternommen werden, welche weniger als eine Minute Fahrtzeit in Anspruch nehmen. Darüber hinaus werden keine Fahrten generiert, welche außerhalb des Servicebereiches liegen.

Nachdem die n Fahrten bekannt sind, wird der Shareability-Graph erstellt. Dieser wird auch *Kombinierbarkeitsgraph* genannt. Der Shareability-Graph fällt unter die Kategorie *undirected graph*. Das bedeutet, dass er im Gegensatz zum Digraph richtungsunabhängige Kanten besitzt. In diesem Graph ist jede Fahrt ein Knoten. Alle Knoten werden miteinander auf Kombinierbarkeit geprüft. Falls zwei Knoten kombinierbar sind, werden diese mit einer Kante verbunden und eine Kombinierung liegt vor. Im Folgenden werden zwei Knoten beziehungsweise Trips beispielhaft überprüft. T_i und T_j mit $i \in \{1,2,...n\}$, $j \in \{1,2,...n\}$, $j \neq i$.

Das Anfangskriterium für die Prüfung ist, ob diese überhaupt im selben Zeitraum stattfinden. mit $st_j \leq at_i + \Delta$ wird geprüft ob T_i losfährt, nachdem T_j mit der maximalen Verzögerung Δ schon am entsprechenden Zielort sein kann. Damit ist eine Kombinierung der beiden Fahrten ausgeschlossen. Das Gleiche gilt für den umgekehrten Fall $st_i \leq at_j + \Delta$. Die Berechnungen der Zeiten sind also notwendige, aber nicht hinreichende Bedingungen.

Daraufhin können die hinreichenden Bedingungen für die Kombinierbarkeit überprüft werden mit

$$st_i \leq pt_i \leq st_i + \Delta,$$
$$st_j \leq pt_j \leq st_j + \Delta,$$
$$dt_i \leq at_i + \Delta \text{ und}$$
$$dt_j \leq at_j + \Delta,$$

wobei dt $(Delivery\ time)$ die Übergabezeit bzw. wann der Kunde nach der Kombinierung am Zielort ist. Mit diesen Konditionen wird sichergestellt, dass die kombinierten Fahrten die maximale Verzögerung nicht

überschreiten. Da das vorliegende Working Paper sich darauf beschränkt zwei Fahrten (k=2) zu kombinieren, gibt es vier Permutationen von möglichen Routen, die ein Taxi ansteuern kann mit

$$o_i \to o_j \to d_i \to d_j,$$
$$o_i \to o_j \to d_j \to d_i,$$
$$o_j \to o_i \to d_i \to d_j \text{ und}$$
$$o_j \to o_i \to d_j \to d_i.$$

Die Bedingungen werden nun auf Route 1 übertragen. In dem erarbeiteten Modell wird angenommen, dass die Abholzeit pt_i bei Anfangsort o_i ohne Verspätung geschieht. Wenn eine durchschnittliche Verspätung bekannt wäre, könnte diese statisch, mit st_i aufgerechnet werden. Die Abholzeit pt_j bei o_j berechnet sich aus $pt_j = pt_i + tt(o_i, o_j)$, tt ist dabei die $travel\ time$ zwischen den beiden Punkten o_i und o_j. Dieses Vorgehen lässt sich auch auf die Übergabezeiten dt übertragen: $dt_i = pt_j + tt(o_j, d_i)$ und $dt_j = dt_i + tt(d_i, d_j)$. Für die Überprüfung der Fahrtzeit wird der Dijkstra Algorithmus[12] genutzt. Konkret existieren die Bedingungen

$$st_i \leq pt_i \leq st_i + \Delta \tag{1}$$
$$st_j \leq pt_i + tt(o_i, o_j) \leq st_j + \Delta \tag{2}$$
$$pt_i + tt(o_i, o_j) + tt(o_j, o_i) \leq at_i + \Delta \tag{3}$$
$$pt_i + tt(o_i, o_j) + tt(o_j, d_i) + tt(d_i, d_j) \leq at_j + \Delta \tag{4}$$

für Route 1. Diese lassen sich auf die Routen 2 bis 4 übertragen. Wenn das Ziel die Minimierung der Fahrten ist, kann die Überprüfung abgebrochen werden, nachdem die Bedingungen durch eine Route erfüllt worden sind. Falls am Ende die günstigsten Kombinierungen gewählt werden sollen, müssen alle Routen überprüft werden und abschließend verglichen, ob die Kosten der kombinierten Fahrt $cost_{i,j} < cost_i + cost_j$ erfüllen. Dadurch wird sichergestellt, dass die Kosten der gesamten Fahrt nicht größer sind als die der einzelnen Fahrten.

Sobald die Kombinierung geprüft wurde, kann eine Kante zwischen den beiden Knoten gesetzt werden. Im Falle der Kostenminimierung wird das Gewicht der Kante mit $weight_{i,j} = cost_i + cost_j - cost_{i,j}$ bemessen. Dies entspricht der Kostenersparnis der beiden Fahrten. Dementsprechend werden die nächsten beiden Fahrten verglichen. Die Zeit, die zum Ein- und Aussteigen des Autos verwendet wird, wurde in diesem Modell nicht berücksichtigt, da diese unabhängig von Ridepooling ist und die Fahrten darüber hinaus nicht in Echtzeit simuliert werden. Der maximale Zeitaufwand der Funktion beträgt im Worst-Case-Szenario $O\left(n^2\right)$. Abbildung 3 visualisiert ein Shareability-Graph mit 15 verglichenen Knoten. Die Knoten sind mit blauen Zahlen gekennzeichnet und das Gewicht der Kanten wird in Schwarz dargestellt. Die maximal zulässige Verzögerung liegt bei zehn Minuten und der benutzte Algorithmus hat für die $\binom{n}{k} = \binom{15}{2} = 105$ Kombinationen gerundet 0,16 Sekunden gerechnet.

3.4. Matching

Sobald die kombinierbaren Fahrten berechnet wurden, müssen diese koordiniert werden. Ziel ist es, Doppelbelegungen zu vermeiden. Dazu eignet sich das aus der Graphentheorie bekannte *Matching*. Das Matching beschreibt eine Heuristik, um in richtungsunabhängigen Graphen Kanten auszuwählen, welche sich keine Knoten teilen. Ein Matching mit maximaler Kardinalität beschreibt, dass so viele Kanten wie möglich gewählt werden. Ein Matching mit maximalem Gewicht beschreibt, dass die auszuwählenden Kanten nach ihrem Gewicht maximiert werden. Eine mögliche Heuristik zur Implementierung von unterschiedlichen

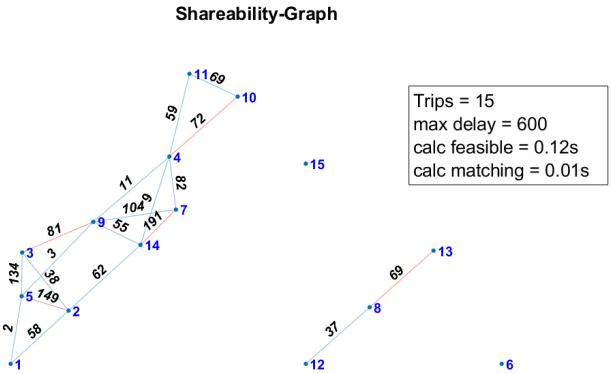

Abbildung 3: Beispiel eines Shareability-Graph

Matchings liefert Galil [15]. Diese Implementation baut auf den Blossom-Algorithmus von Jack Edmonds aus 1965 auf.

Die endgültigen Fahrten werden in einem *Tripset* oder einer Fahrtentabelle festgehalten. Zu Beginn wird die Anzahl von einzelnen Fahrten dargestellt. Sobald zwei Fahrten ein Matching bilden, werden diese Fahrten aus der Tabelle gelöscht und als eine verbundene Fahrt hinzugefügt. In Abbildung 3 wurde ein *weighted maximum matching in general graphs* angewendet und die relevanten Kanten rot eingezeichnet. Dieses Matching entspricht einer maximalen Kostenreduzierung. Die Zeitkomplexität beträgt beim Matching mit maximaler Kardinalität bei spärlichen Graphen $O\left(n \cdot \sqrt{n}\right)$ und beim Matching mit maximalen Gewicht $O\left(n^2 \log n\right)$. Spärlich bedeutet, dass von einem Knoten nur eine begrenzte Anzahl von Kanten ausgehen.

Diese Funktionen wurden nicht im Rahmen dieses Working Paper programmiert. Für das *maximum cardinality matching* wurde der Code von Peshkin [16] verwendet und für das *weighted maximum matching* konnte die Funktion von Saunders [17] genutzt werden.

3.5. Einstellbare Parameter und dessen Auswirkung

Raumvergrößerung

Das Modell besitzt viele Variablen, die verändert und angepasst werden können. Das implementierte Modell ist zum Beispiel in der Lage den Raum vergrößern. Eine Ausweitung von Fischeln auf ganz Krefeld ist folglich ohne viel Anpassung möglich, da die Straßendaten bereits vorhanden sind. Diese Erweiterung geht mit zusätzlichen Herausforderungen einher. Beispielsweise erhöht sich die Anzahl der Knoten und Kanten, dadurch sind längere Fahrten möglich und die Laufzeit wird verlängert.

Gewichtung

Die Gewichtung der Kanten kann ebenfalls angepasst werden. Die vorhandenen Distanzen wurden eingangs durch eine durchschnittliche Geschwindigkeit in eine Zeit umgewandelt. Es ist möglich, die tatsächlich benötigte Zeit pro Kante einzutragen, um so eine noch genauere Simulation zu erzeugen. Darüber hinaus ist eine abstraktere Gewichtung möglich, wenn beispielsweise verbrauchte Emissionen oder realistische Daten der aktuellen Verkehrslage einbezogen werden.

Fahrten und Verzögerung

Fahrten und Verzögerung stehen in einem kausalen Zusammenhang. Wenn die Anzahl der Fahrten erhöht wird, steigt generell die Laufzeit, weil die Zeitkomplexität $O(n)$ sich auf diese Anzahl bezieht. Darüber hinaus steigt durch mehr Fahrten auch die Möglichkeit, dass zwei Fahrten kombiniert werden können. Mit einer Verringerung der Verzögerungen geht ein Abfall der Kombinierbarkeit einher. Dieser verringert ebenfalls die Laufzeit des Algorithmus. Wenn wenige Fahrten in das System kommen, sollte also eine höhere Verzögerung gewählt werden. Wenn es viele Fahrten gibt, kann die Verzögerung niedriger gewählt werden. Dies wird in Kapitel 4 aufgezeigt und in einen konkreten Kontext gesetzt.

Kombinierungen

Eine Vergrößerung der Trip-Kombinierungen ist nur mit einer erheblichen Veränderung des Algorithmus möglich. In dem vorliegenden Modell wurde dies nicht angewendet. Die Erhöhung bedeutet eine gesteigerte Zeitkomplexität. Bei k Kombinierungen erhöht sich die Zeitkomplexität auf $O(n^k)$. Der daraus resultierende Shareability-Graph ist dementsprechend ein *Hyper-Graph*, aufgrund der Verbindung zwischen k Trips. Eine Priorisierung von Matchings in Hyper-Graphen wurde im Modell von Santi et al. [11] implementiert. Allerdings geschah dies für $k = 3$, indem ein *Greedy-Algorithmus* die größten Gewichte der *Hyper-Links* in jedem Schritt auswählt. Dieser Algorithmus rechnet mit einer Zeitkomplexität von $O(n \log n)$. Für den Fall von maximaler Kardinalität, ist das Gewicht der Hyper-Links die Anzahl der Fahrten welche verbunden werden. Im Fall des maximalen Gewichts ist es die Summe der Gewichte. Das bedeutet also, dass nach dem Greedy-Algorithmus auf jeden Fall ein gewichtetes Matching ausgeführt werden muss. Das Matching auf Hyper-Networks macht den Algorithmus zu einem *NP-hard* Problem. Die Anwendung der Heurisitk, also des Greedy Algorithmus, reduziert das Problem zu *NP-complete*.

4. Auswertung des Modells

4.1. Analyse der Ergebnisse

Erste Beispiele wurden bereits in Kapitel 3 zur Darstellung der angewendeten Methodik gegeben und erläutert. In diesem Beispiel wurden 15 Fahrten simuliert, um die Erklärungen der Methodik visuell zu unterstreichen und die theoretischen Möglichkeiten des Shareability-Graph aufzuzeigen. Ein Durchlauf mit 15 Fahrten ist allerdings nicht aussagekräftig für realistische Verhältnisse. In der Praxis muss ein Zeitraum oder eine maximale Anzahl an Fahrten gewählt werden, damit eine Berechnung in Echtzeitgeschwindigkeit gewährleistet werden kann.

In diesem Working Paper wurden beide vorgestellten Arten des Matchings verwendet. Die Ergebnisse haben gezeigt, dass das *maximum cardinality matching* die Anzahl der Fahrten nicht weiter reduziert als das *weighted matching*. Daraus lässt sich schließen, dass durch den kürzeren Algorithmus kein signifikanter Vorteil gegeben ist. Aufgrund dessen sind die vorgestellten Ergebnisse ausschließlich mit dem *weighted matching* generiert.

Im Jahr 2011 gab es durchschnittlich 450.000 Taxifahrten in Manhattan pro Tag, was in etwa 310 Trips pro Minute entspricht [11]. Manhattan ist der am dichtesten bevölkerte Bezirk von New York City, weshalb

das Matching dort größere Leistung erbringen muss, als das vorliegende theoretische Modell in Fischeln. Aufgrund dessen sollte das Ziel sein, eine ähnliche Anzahl an Trips bedienen zu können.

Für die Erstellung der Ergebnisse wurde jede Kombination aus den zufällig generierten Fahrten und den jeweiligen Verzögerungen 100 Mal ausgeführt und berechnet. Für die Anzahl von Fahrten T_n wurde $n \in \{10;20;50;75;100;150;200;300\}$ gewählt und die maximalen Verzögerungen Δ_d mit $d \in \{10;30;45;60;80;100;200;300;600\}$. Es wurden $k = 2$ Fahrten kombiniert. Die Ergebnisse der Durchläufe wurden in eine Liste übertragen, die zwölf Variablen enthält, welche für die Berechnung und Bewertung der Ergebnisse relevant sind.

1 Anzahl der Fahrten T_n,
2 maximale Verzögerung Δ_d,
3 Nummer des Durchlaufes,
4 benötigte Zeit für die Kombinierung,
5 benötigte Zeit für das Matching,
6 benötigte Zeit für den Durchlauf,
7 Summe der Zeit für einzelne Trips im Durchlauf mit $cost_{max} = \sum_i^n st\,(T_i)$,
8 Summe der gesparten Zeit mit $weight_{max} = \sum_i^n weight_{i,j}$,
9 absolut gesparte Zeit mit $cost_{min} = cost_{max} - weight_{max}$,
10 relative gesparte Zeit in Prozent mit $t_{saved} = \left(\frac{weight_{max}}{cost_{max}}\right)$,
11 Anzahl der kombinierten Trips $Matchings$ und
12 relativ gesparte Trips $T_{saved} = \left(\frac{Matchings}{T_n}\right)$.

Daraus ergibt sich ein Datensatz von 7200 Zeilen und zwölf Spalten. In den besten Fällen konnten 100 Prozent der gestellten Fahranfragen zusammengelegt werden. Die größte erreichte Zeitersparnis lag bei 35,68 Prozent. Für die weitere Betrachtung wurden die Ergebnisse auf Vollständigkeit und Korrektheit gefiltert, sodass kein Datensatz Not a Number (NaN) enthält. Falls keine Kombinierung in diesem Datensatz möglich war, schreibt Matlab NaN als Ergebnis in die Tabelle.

Die Information, dass es unter bestimmten Umständen zu keinem Ergebnis gekommen ist, wird ebenfalls ausgewertet. Mithilfe dieser Information können weitere Entscheidungen getroffen werden. Beispielsweise kam es zu keinem Ergebnis, wenn die Parameter T_n oder Δ_d zu klein gewählt worden sind. Um die Daten zu bündeln und zu komprimieren, mit dem Ziel einer verbesserten Übersichtlichkeit, wurde jedes Paar von Fahrten und Verzögerung gruppiert. Es existieren also die unterschiedlichen Gruppen $T_{10}\Delta_{10}$, $T_{20}\Delta_{10}$, $T_{10}\Delta_{30}$ und weitere, welche die Tabelle auf 72 Zeilen beziehungsweise Gruppen konzentriert. Aus den Durchläufen dieser Gruppen wurde das arithmetische Mittel berechnet, damit diese ohne Umwege untereinander verglichen werden können.

Tabelle 3 visualisiert die Anzahl der erfolgreichen Durchläufe entsprechend den vorgenommenen Gruppierungen. Diese Gruppen wurden auf den Achsen aufgetragen. Die Anzahl der Fahrten wird jeweils auf der X-Achse dargestellt. Die maximale Verzögerung der Fahrten in Sekunden ist auf der Y-Achse zu finden. Die Z-Achse beinhaltet die Gruppengröße. Ab etwa 150 Fahrten hat jede Verzögerung zu Ergebnissen geführt. Andersherum hat ab 300 Sekunden Verzögerung jede Anzahl an Fahrten Ergebnisse erzeugt. Bei zehn Fahrten mit zehn Sekunden Verzögerung gab es nur einen Durchlauf, welcher ein Ergebnis generiert hat. Es wird deutlich, dass auch klein gewählte Parameter zu quantitativem Erfolg führen, jedoch entsprechend angepasst werden müssen, um robuste Ergebnisse liefern zu können.

Tabelle 3: Anzahl der Durchläufe mit mindestens einer Kombination

Maximale Verzögerung Δ [s] \ Anzahl der Fahrten	10	20	50	75	100	150	200	300
600	100	100	100	100	100	100	100	100
300	100	100	100	100	100	100	100	100
200	90	100	100	100	100	100	100	100
100	58	95	100	100	100	100	100	100
80	46	94	100	100	100	100	100	100
60	21	65	100	100	100	100	100	100
45	17	52	99	100	100	100	100	100
30	10	24	87	100	100	100	100	100
10	1	13	37	66	86	97	100	100

In Tabelle 4 wird die prozentual gesparte Zeit der Gruppen dargestellt. Die inhaltliche Bedeutung der X- und Y-Achse wurde gleich gewählt, um den direkten Bezug zu vereinfachen. Auch hier sind bessere Ergebnisse in der rechten oberen Ecke angesiedelt. Im Folgenden wird der Index der referenzierten Punkte mit $(X = n, Y = \Delta)$ angegeben. Der niedrigste Wert beträgt 1,5 Prozent bei $(100, 10)$ und der höchste 34,4 Prozent bei $(300, 600)$. Aufgrund der niedrigen Gruppengröße in der linken unteren Ecke bei Werten kleiner als $(20, 100)$ haben diese Werte keine große Aussagekraft.

Tabelle 4: Durchschnittlich gesparte Zeit in Prozent

Maximale Verzögerung Δ [s] \ Anzahl der Fahrten	10	20	50	75	100	150	200	300
600	10,1	16,5	23,1	26,1	28,1	30,6	32,4	34,4
300	10,4	15,6	23,1	26,1	27,8	30,4	32,3	34,4
200	8,0	13,1	21,7	24,6	27,3	30,2	31,8	34,3
100	6,0	6,6	13,4	17,3	20,0	24,0	27,1	31,0
80	5,2	5,0	10,2	13,0	15,9	20,4	23,4	27,9
60	6,2	4,4	6,8	9,3	11,5	15,3	17,8	22,3
45	6,5	4,1	4,5	5,9	7,9	10,7	13,1	16,6
30	7,3	3,8	2,8	3,7	4,6	6,3	8,3	10,9
10	3,9	3,9	2,4	1,7	1,5	2,0	2,5	3,7

Darüber hinaus ist zu erkennen, dass der Einfluss der Verzögerung bei $(X \leq 20, Y)$ nur geringfügig ist, denn die Werte wachsen nur langsam. Bei einer Verzögerung von $(X, 10)$ scheint außerdem die Effizienz nicht erstrebenswert, denn die Werte wachsen im Vergleich zu einer höher gewählten Anzahl langsamer. Es ist allerdings möglich, dass bei einer sehr viel höheren Anzahl an Fahrten dieser Wert auch befriedigende Ergebnisse liefert. Mit dem aktuellen Stand des Algorithmus ist ein ausführliches Testen auf dem verwendeten System allerdings nicht möglich, da die Laufzeit den Rahmen der Testphase übertrifft. Darauf wird später näher eingegangen. Wenn eine so geringe Verzögerung auf die Realität übertragen wird, müssten zwei Fahrten beinahe identische Start- und Endpunkte besitzen, um tatsächlich ein relevantes Ergebnis zu erzeugen. Ab dem Punkt (75,200) beträgt die durchschnittlich gesparte Zeit etwa 25 Prozent. Dieser Punkt stellt also eine erstrebenswerte Kosten-Nutzen-Relation dar.

Die in Tabelle 5 dargestellte Standardabweichung für die gesparte Zeit verdeutlicht die Robustheit der Ergebnisse. Das Diagramm visualisiert die Anzahl der Fahrten auf der X-Achse und die Standardabweichung in Prozent auf der Y-Achse. Es wird ersichtlich, dass die Ergebnisse mit steigenden Fahrten und maximaler Verzögerung ebenfalls an Robustheit gewinnen, also eine konstante Quote aufweisen. Bei $n = 300,\ \Delta \geq 200$

ist der 3σ Bereich, in welchem 99,7 Prozent der Daten liegen, bei 1,5 Prozent. Das bedeutet, dass der abgebildete Mittelwert eine exzellente Einschätzung über den tatsächlich auf den realen Straßen eintreffenden Wert gibt. Eine höhere erlaubte Verzögerung bietet generell robustere Mittelwerte an. Die niedrigeren Standardabweichungen bei $\Delta \leq 30$ können durch die generell niedrige Kombinierbarkeit der Fahrten erklärt werden. Der Großteil der kombinierbaren Fahrten liegt an der Grenze zur maximalen Verzögerung und dadurch können die Werte weniger streuen.

Tabelle 5: Standardabweichung σ für gesparte Zeit

Maximale Verzögerung Δ [s] \ Anzahl der Fahrten	10	20	50	75	100	150	200	300
600	4,6	3,6	1,8	1,4	1,2	0,9	0,7	0,5
300	5,1	3,7	1,7	1,5	1,1	0,9	0,8	0,5
200	5,2	3,9	2,3	1,6	1,2	0,9	0,7	0,5
100	4,2	3,8	2,9	2,5	1,8	1,6	1,2	1,0
80	3,4	3,4	3,1	2,4	2,1	1,7	1,7	1,1
60	3,6	2,4	2,5	2,5	2,4	1,8	1,6	1,5
45	4,0	3,1	2,2	2,0	2,1	1,7	1,7	1,4
30	2,5	2,5	1,9	1,6	1,7	1,6	1,4	1,4
10	0,0	1,9	1,7	1,0	1,0	1,1	1,0	0,9

Tabelle 6 visualisiert die prozentual gesparten Trips. Obwohl der Algorithmus sich darauf fokussiert, kein Matching mit maximaler Kardinalität zu generieren, ist zu erkennen, dass die gesparten Fahrten schnell an die 100 Prozent konvergieren. Schon ab dem Punkt (50,200) liegen die prozentual gesparten Fahrten bei 80 Prozent. Also schon bei diesen sehr niedrig gewählten Parametern kann der resultierende Wert mit einer Verringerung der Fahrzeugflotte verglichen werden. Die Auswirkungen dieser Ergebnisse werden im folgenden Kapitel interpretiert und in Bezug gesetzt.

Tabelle 6: Durchschnittlich gesparte Fahrten in Prozent

Maximale Verzögerung Δ [s] \ Anzahl der Fahrten	10	20	50	75	100	150	200	300
600	55,4	72,4	85,0	88,6	90,9	92,9	94,2	96,0
300	53,2	67,5	84,0	88,1	90,5	93,0	94,4	96,0
200	39,6	57,4	79,2	84,7	88,7	92,1	93,7	95,9
100	24,1	25,9	46,7	57,2	64,9	75,2	81,3	89,0
80	23,0	18,2	35,1	42,8	50,6	62,5	69,6	79,8
60	21,9	16,9	22,0	29,6	36,0	46,0	52,2	63,4
45	21,2	12,7	13,9	18,4	24,2	31,3	37,6	46,6
30	20,0	11,7	8,2	11,2	13,6	17,8	23,4	30,2
10	20,0	10,0	5,6	4,2	4,3	5,7	7,0	10,1

4.2. Grenzen des Modells

Tabelle 7 stellt die durchschnittlich benötigte Zeit zur Errechnung dar. Das Modell wurde nicht auf Laufzeit optimiert, was erkennbar durch die rasche Steigung bei einer erhöhten Anzahl an Fahrten ist. Zu diesem Entwicklungstand des Modells ist diese Optimierung noch nicht notwendig, um die Ziele und Ergebnisse begründen zu können. Die Laufzeit kann gesenkt werden, wenn die Kombinierung der Fahrten optimiert wird, zum Beispiel durch Parallelisierung.

Tabelle 7: Durchschnittliche Laufzeit in Sekunden

Maximale Verzögerung Δ [s]	Anzahl der Fahrten							
	10	20	50	75	100	150	200	300
600	0,06	0,22	1,34	3,06	5,53	12,58	23,61	54,16
300	0,05	0,17	1,02	2,32	4,20	9,69	18,01	41,88
200	0,04	0,13	0,74	1,71	3,06	7,00	12,95	29,53
100	0,03	0,10	0,52	1,21	2,06	4,61	8,27	18,53
80	0,03	0,09	0,49	1,12	1,93	4,28	7,51	16,83
60	0,03	0,09	0,47	1,04	1,82	4,03	7,10	15,83
45	0,03	0,09	0,46	1,00	1,76	3,89	6,80	15,65
30	0,05	0,09	0,45	0,98	1,73	3,79	6,57	15,01
10	0,23	0,09	0,44	0,97	1,68	3,72	6,41	14,69

Die Grenze des derzeit programmierten Modells, liegt also bei etwa 300 Fahrten pro Durchlauf. Wenn das Modell länger als 60 Sekunden benötigt um eine Auskunft für die vorhandenen Fahrten zu erstellen, kann keine Garantie für eine rechtzeitige Vorhersage gegeben werden. Dies ist zum aktuellen Stand der Ausarbeitung kein Schwerpunkt und kann bei Überarbeitung zielführend angepasst werden.

Das implementierte Modell generiert ähnliche Ergebnisse wie in der Arbeit von Santi et al. [11]. Dort wurden etwa 50 bis 60 Prozent der Fahrten gespart. Obwohl in dem vorliegenden Modell hauptsächlich das gewichtete Matching benutzt wird, ist zu sehen, dass die kombinierten Trips bei großen Parametern gegen 100 Prozent Kombinierbarkeit annähert. Im Modell, welches von Alonso-Mora et al. [4] vorgestellt wurde, konvergiert die Anzahl der geteilten Fahrten ebenfalls an 100 Prozent. Die prozentual gesparte Zeit wird in dem dynamischen Modell nicht quantifiziert. Sowohl bei dem vorliegenden Modell, als auch in dem grundlegenden Modell liegt die gesparte Zeit bei etwa 30 Prozent.

4.3. Mögliche Service Level Agreements

Ein SLA oder auch *Dienstleistungsgütevereinbarung* beschreibt einen Vertrag mit unterschiedlichen Stufen zwischen zwei Parteien. Eine höhere Stufe bietet dabei mehr Funktionen oder Komfort, bei einem höheren Preis und andersherum bei einer niedrigen Stufe.

SLA können auf das Shareability-Modell angewendet werden. Eine Abstufung kann durch eine spezifische maximalen Verzögerung erreicht werden. Die günstige Variante würde einen hohen Verzug vorsehen, was dem System eine gute Kombinierbarkeit liefert. Dabei zahlt die höhere Stufe den Ausgleich für die niedrige Kombinierbarkeit. Umsetzbar wäre diese Abstufung, wenn anstelle eines statischen globalen Δ, den Fahrten diese Verzögerung zugewiesen wird. Angenommen T_i ist auf einer niedrigen SLA-Stufe und diese ist mit einer Verzögerung von $\Delta_i = 10$ min bemessen, und T_j auf einer hohen mit $\Delta_j = 2$ min, dann werden Gleichungen 1 bis 4 wie folgt geändert:

$$st_i \leq pt_i \leq st_i + \Delta_i \tag{5}$$

$$st_j \leq pt_i + tt(o_i, o_j) \leq st_j + \Delta_j \tag{6}$$

$$pt_i + tt(o_i, o_j) + tt(o_j, o_i) \leq at_i + \Delta_i \tag{7}$$

$$pt_i + tt(o_i, o_j) + tt(o_j, d_i) + tt(d_i, d_j) \leq at_j + \Delta_j \tag{8}$$

Eine Eingrenzung der Reichweite kann ebenfalls vorgenommen werden, sodass nur Fahrten mit einer bestimmten Dauer, beispielsweise maximal 30 Minuten, berücksichtigt werden. In einem fortgeschrittenen

Modell könnte außerdem die Art der Fahrt verändert werden. Dieses Modell sieht ein On-Demand Door to Door System vor. Die Einbeziehung des bestehenden ÖPNV könnte berücksichtigt werden, sodass die Routenführung überprüft, ob ein kurzer Weg zu einer Haltestelle führt und anschließend ans Ziel des Kunden. Weiterhin wäre es möglich, den Kunden eine Fahrt ablehnen zu lassen, sodass eine neue Fahrt vorgeschlagen wird. Weitere Unterschiede können auch abseits des Shareability-Modells vorgenommen werden. Beispiele hierfür sind die Kapazität des Taxis, bei welchen nur eine weitere Person darf mitfahren, die Beschaffenheit des Taxis, in welchen eine eigene Kabine mit Trennwand vorhanden ist oder auch ein Flatrate-Modell mit einer bestimmten Anzahl an freien Fahrten.

5. Diskussion und Fazit

Fazit

Das Ziel war die Entwicklung eines Algorithmus zur Lösung des RPP bezogen auf den Bezirk Fischeln der Stadt Krefeld. Der implementierte Algorithmus ist in der Lage Fahranfragen zu generieren, diese zu kombinieren und die entsprechenden Kombinationen mit der größten Kostenersparnis auszuwählen. Die T_n Fahranfragen besitzen die gewünschten Attribute, die das Berechnungsverfahren benötigt, um die beschriebenen Ergebnisse zu liefern. Diese Attribute umfassen unter anderem die o_i Anfangs- und d_i Endpunkte. Die Dauer der Fahrt wurde mit den Kosten $cost_i$ gleichgesetzt, um die Kostenersparnis zu ermitteln. Zum Schluss wurde die Route mit $path_i$ bestimmt, woraus insgesamt die Fahrt der Passagiere mittels des Algorithmus bearbeitet werden kann. Diese Attribute werden auch den kombinierten Fahrten zugeschrieben.

Die relevanten Ergebnisse werden ermittelt und können entsprechend visualisiert werden, wie beispielsweise die Zeitersparnis durch die Kombinierung der Fahrten und die damit einhergehenden gesparten einzelnen Fahrten. Diese Ergebnisse sind vergleichbar mit denen, welche in der Literatur beschrieben sind. Das RPP konnte also mittels des erstellten Algorithmus erfolgreich gelöst werden.

Diskussion

Grundsätzlich entsprechen die Ergebnisse den gesetzten Anforderungen. Auf einige Modifikationen kann trotzdem eingegangen werden. Der geschriebene Code ist nicht auf die Laufzeit hin optimiert. Die Funktion zur Kombinierung soll eine Zeitkomplexität von $O(n^2)$ aufweisen. Diese Funktion hat allerdings den größten Anteil der Laufzeit. Möglichkeiten zur Optimierung der Funktion wurden bereits in Sektion 4.2 kurz angeschnitten. Inwieweit diese Funktion optimierbar ist, muss noch untersucht und getestet werden.

In diesem Modell wurde das gewichtete Matching verwendet, damit die gesparte Zeit optimiert wird. Die Ergebnisse wurden anhand dieser Methodik generiert. Ein weiterer Test, ob das Maximum Cardinality Matching bedeutend kürzere Laufzeiten generiert, kann überprüft werden, sobald die Methode der Kombinierung optimiert ist. Darüber hinaus ist zu hinterfragen, ob die Vergrößerung des Raumes dabei ausschlaggebend ist. Diese Aspekte sollten hinsichtlich der bisherigen Literatur kritisch betrachtet werden, um den hier beschriebenen Code zu modifizieren.

Darüber hinaus können die erweiterten Anforderungen an die Simulation implementiert werden. Zum Beispiel können im Digraph die durchschnittlich benötigten Zeiten pro Straßeeingefügt werden. Auch aktuelle Staus könnten für die Routenführung berücksichtigt werden, um so den Wunsch der Zeitersparnis der Kunden gerecht zu werden. Zusätzliche Parameter sind für den Theorie-Praxis-Transfer unabdingbar.

Der Einsatz eines *Fleet Operators*, um eine tatsächliche Taxiflotte zu simulieren, ist für die Anwendung des Modells auf Krefeld notwendig. Im gleichen Zuge kann der Raum von Fischeln auf das Krefelder Stadtgebiet erweitert werden. Der verwendete Algorithmus kann die unterschiedlichen Kategorien von Fahrten verarbeiten, wie in Sektion 2.2 beschrieben. Beispielsweise identische Fahrten, Teilmengen einer Fahrt und verzögerte Fahrten durch Umwege. Partielle Fahrten, eine Teilfahrt durch Ridepooling die andere Verkehrsmittel einbezieht, sind für die praktische Umsetzung und die Zusammenarbeit mit den ÖPNV ebenfalls interessant. Grundsätzlich soll das Ridepooling eine Erweiterung des ÖPNV sein und eine Einbindung dessen würde zu einer gesteigerten Effizienz führen.

Ob ein statisches oder dynamisches Modell implementiert wird, ist ebenfalls von Interesse. Eine Analyse von den Straßen in Krefeld wäre für eine Entscheidung für oder gegen diese Modellklassen notwendig. Welches Modell sich in Krefeld anbietet, ist nur mit Fahrzeugdaten einer Taxiflotte oder Ähnlichem adäquat darzustellen. Im Rahmen einer praktischen Implementierung, wäre es zielführend in Erfahrung zu bringen, welche Modelle andere Ridepoolinganbieter verwenden. Eine Konsortialstudie könnte deshalb ein erstrebenswerter Weg sein, das Ridepooling in Deutschland gesammelt zu erforschen und voranzutreiben.

Eine weitere Form der Interpretation dieser statischen Implementierung wäre die Betrachtung der einzelnen Trips, die in diesem Modell generiert werden, als eine Route, die so im MIV stattfindet. Wenn bekannt ist, wie viele Personen von welchen Startorten zu welchen Zielorten wollen, kann das Ausmaß der Ersparnisse durch Ridepooling näher eingeschätzt werden. Wenn also durch dieses Modell der komplette MIV in Krefeld simuliert werden könnte, ist eine erweiterte Aussage über die tatsächlichen Auswirkungen von Ridepooling in der Praxis möglich. Auf dieser Grundlage kann im Sinne von SLA eine engere Aussage von Anreizen gegeben werden.

Literatur

[1] Umweltbundesamt. „Mobilität privater Haushalte.“ (2022), Adresse: `https://www.umweltbundesamt.de/daten/private-haushalte-konsum/mobilitaet-privater-haushalte#verkehrsleistung-im-personentransport` (besucht am 21. 01. 2023).

[2] SWK Stadtwerke Krefeld AG. „Mein SWCAR.“ (2023), Adresse: `https://www.swk.de/privatkunden/mobilitaet/sharing-und-emobility/mein-swcar` (besucht am 22. 01. 2023).

[3] T. Bektas, „The multiple traveling salesman problem: an overview of formulations and solution procedures,“ *Omega*, Jg. 34, Nr. 3, S. 209–219, 2006, ISSN: 0305-0483. Adresse: `https://www.sciencedirect.com/science/article/pii/S0305048304001550`.

[4] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli und D. Rus, „On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment,“ *Proceedings of the National Academy of Sciences*, Jg. 114, Nr. 3, S. 462–467, 2017. Adresse: `https://www.pnas.org/doi/abs/10.1073/pnas.1611675114`.

[5] K. Yamamoto, K. Uesugi und T. Watanabe, „Adaptive Routing of Cruising Taxis by Mutual Exchange of Pathways,“ in *Knowledge-Based Intelligent Information and Engineering Systems*, I. Lovrek, R. J. Howlett und L. C. Jain, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 559–566, ISBN: 978-3-540-85565-1.

[6] M. Furuhata, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang und S. Koenig, „Ridesharing: The state-of-the-art and future directions,“ *Transportation Research Part B: Methodological*, Jg. 57, S. 28–46, 2013, ISSN: 0191-2615. Adresse: https://www.sciencedirect.com/science/article/pii/S0191261513001483.

[7] J. Wang, X. Wang, S. Yang, H. Yang, X. Zhang und Z. Gao, „Predicting the matching probability and the expected ride/shared distance for each dynamic ridepooling order: A mathematical modeling approach,“ *Transportation Research Part B: Methodological*, Jg. 154, S. 125–146, 2021, ISSN: 0191-2615. Adresse: https://www.sciencedirect.com/science/article/pii/S0191261521001880.

[8] B. Yu et al., „Environmental benefits from ridesharing: A case of Beijing,“ *Applied Energy*, Jg. 191, S. 141–152, 2017, ISSN: 0306-2619. Adresse: https://www.sciencedirect.com/science/article/pii/S0306261917300600.

[9] Verkehrsverbund Rhein-Ruhr (VRR) und Kompetenzcenters Digitalisierung (KCD). „Potenzialanalyse On-Demand-Ridepooling im Ruhrgebiet.“ (2022), Adresse: https://www.kcd-nrw.de/projekte/ergebnisse-potenzialanaylse-ridepooling-systeme.html (besucht am 08. 02. 2023).

[10] Bitkom. „Deutschlands Landbevölkerung will neue Mobilitätsdienste.“ (2021), Adresse: https://www.bitkom.org/Presse/Presseinformation/Deutschlands-Landbevoelkerung-will-neue-Mobilitaetsdienste#_ (besucht am 08. 02. 2023).

[11] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz und C. Ratti, „Quantifying the benefits of vehicle pooling with shareability networks,“ *Proceedings of the National Academy of Sciences*, Jg. 111, Nr. 37, S. 13 290–13 294, 2014. Adresse: https://www.pnas.org/doi/abs/10.1073/pnas.1403657111.

[12] MathWorks. „shortestpath.“ (2015), Adresse: https://de.mathworks.com/help/matlab/ref/graph.shortestpath.html (besucht am 29. 01. 2023).

[13] via. „Für einen modernen öffentlichen Nahverkehr in Städten weltweit.“ (2023), Adresse: https://ridewithvia.com/audience/cities (besucht am 08. 02. 2023).

[14] L. Spengler und M. Gennat, „Fahrzeitermittlung im städtischen Raum mittels Google API,“ in *Making Connected Mobility Work: Technische und betriebswirtschaftliche Aspekte*, H. Proff, Hrsg. Wiesbaden: Springer Fachmedien Wiesbaden, 2021, S. 839–853, ISBN: 978-3-658-32266-3. Adresse: https://doi.org/10.1007/978-3-658-32266-3_52.

[15] Z. Galil, „Efficient Algorithms for Finding Maximum Matching in Graphs,“ *ACM Comput. Surv.*, Jg. 18, Nr. 1, S. 23–38, März 1986, ISSN: 0360-0300. Adresse: https://doi.org/10.1145/6462.6502.

[16] L. Peshkin. „Maximum Cardinality matching.“ Version 1.2.0.0. (2023), Adresse: https://de.mathworks.com/matlabcentral/fileexchange/23159-maximum-cardinality-matching (besucht am 29. 01. 2023).

[17] D. Saunders. „Weighted maximum matching in general graphs.“ Version 1.1.0.0. (2013), Adresse: https://de.mathworks.com/matlabcentral/fileexchange/42827-weighted-maximum-matching-in-general-graphs (besucht am 29. 01. 2023).

A. Anhang 1

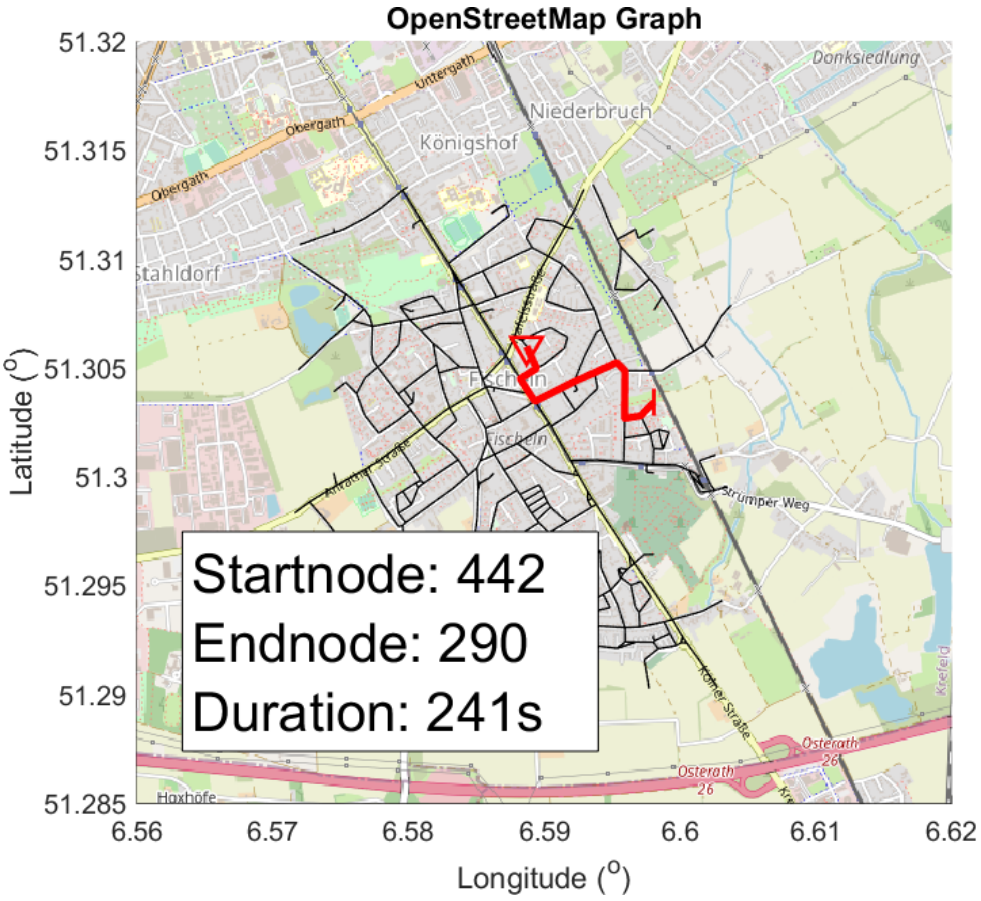

Abbildung 4: Beispiel für Fahrt 1

Tabelle 8: Ausschnitt der Kanten des Digraphen von Krefeld

	1 EndNodes		2 Weight	3 high-way	4 max-speed	5 max-speed type	6 un-known	7 name	8 ref
2734	962	121	12.86	prim.	50	unk.	yes	Obergath	B 57
2735	962	912	16.46	sec.	50	unk.	yes	Kölner Straße	unk.
2736	962	5482	78.04	prim.	70	unk.	yes	Obergath	B 57
2737	962	10335	13.51	sec.	50	unk.	yes	Kölner Straße	unk.
2738	963	121	96.70	sec.	50	unk.	yes	Kölner Straße	unk.
2739	963	2284	59.55	sec.	50	unk.	yes	Kölner Straße	unk.
2740	963	5480	50.44	sec.	50	unk.	no	Kölner Straße	unk.
2741	964	38	13.31	prim.	50	unk.	yes	Dießemer Bruch	unk.
2742	964	2167	103.7	sec.	50	unk.	yes	Oppumer Straße	L 382
2743	964	7803	12.07	prim.	50	unk.	yes	Dießemer Bruch	unk.

unk. = unknown; prim. = primary; sec. = secondary

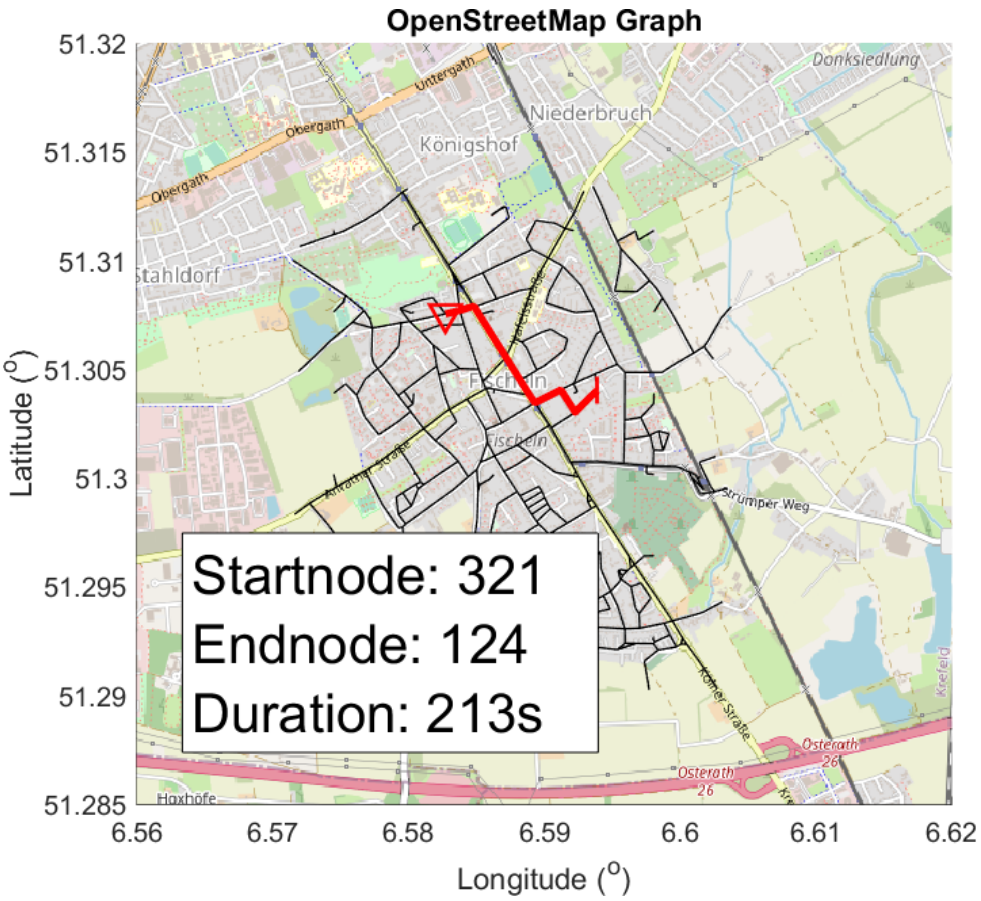

Abbildung 5: Beispiel für Fahrt 2

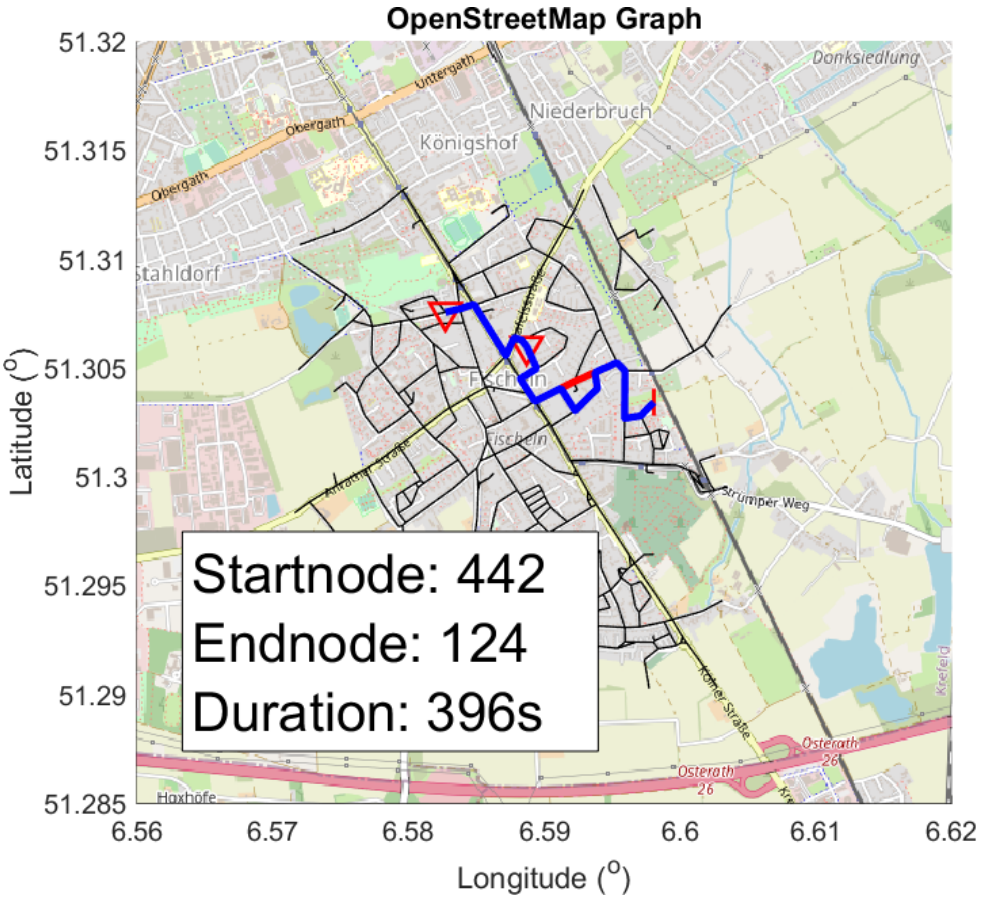

Abbildung 6: Beispiel der Kombination aus Fahrt 1 und 2

```
if (t2.st <= t1.at + delay) && (t1.st <= t2.at + delay)
  %% Route 1 %% Origin 1 -> Origin 2 -> Destination 1 -> Destination 2
  if (t1.st <= t1.pt) && (t1.pt <= t1.st + delay)
    [p,tt] = shortestpath(G,t1.o,t2.o);
    P{1} = p;
    traveltime(1) = tt;
    if (t2.st <= t1.pt + traveltime(1)) && (t1.pt + traveltime(1) <= t2.st + delay)
      [p,tt] = shortestpath(G,t2.o,t1.d);
      P{1} = [P{1} p];
      traveltime(1) = traveltime(1) + tt;
      if t1.pt + traveltime(1) <= t1.at + delay
        [p,tt] = shortestpath(G,t1.d,t2.d);
        P{1} = [P{1} p];
        traveltime(1) = traveltime(1) + tt;
        if t1.pt + traveltime(1) <= t2.at + delay
          Route(1) = true;
          Link = true;
        end
      end
    end
  end
end
```

Abbildung 7: Code zur Prüfung auf Kombinierbarkeit

```
  %% feasible Trips
  shareability = graph; % Erstelle shareability Network
  shareability = addnode(shareability,length(t)); % Node 1 = t(1), t sind alle Trips
  ticfeasible = tic; % Zeitmessung
  for i = 1:(length(t)-1)
    for j = (i+1):(length(t)) % combine all trips with each other
      [Link,c,p,cs,ps] = isfeasible(t(i),t(j),max_delay(val_delay),fischeln);
      if Link
        weight = cs-c; % time save
        shareability = addedge(shareability,i,j,weight);
        % Link in shareability Network
        ei = findedge(shareability,i,j);
        shareability.Edges.Cost(ei) = c; % combined cost
        shareability.Edges.Path{ei} = p; % combined path
        shareability.Edges.SingleTripCost(ei) = cs; % t1.cost + t2.cost
      end
    end
  end
  tocfeasible = toc(ticfeasible);
```

Abbildung 8: Code zur Erstellung eines Shareability-Graphen

Tabelle 9: Ausschnitt der Knoten des Digraphen von Krefeld

	1		2	3		4
	coor		type	UTMcoor		status
781	6.530	51.31	OSM	327900	5687000	OK
782	6.535	51.33	OSM	328300	5689000	OK
783	6.459	51.39	OSM	323200	5696000	OK
784	6.571	51.32	OSM	330700	5689000	OK
785	6.523	51.33	OSM	327500	5690000	OK