

Optimierung von innovativen Zustellkonzepten auf der letzten Meile

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)

von der Fakultät Maschinenbau
der Technischen Universität Dortmund
Dissertation

vorgelegt von
Stefan Schaudt
aus Köln

Tag der mündlichen Prüfung: 05.12.2025

1. Gutachter: Prof. Dr.-Ing. Uwe Clausen
2. Gutachter: Prof. Dr. Christoph Buchheim

Dortmund, 2025

Danksagung

Diese Arbeit entstand überwiegend während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Transportlogistik (ITL) der Technischen Universität Dortmund.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Uwe Clausen für die Betreuung meiner Dissertation sowie für die stets vertrauensvolle und konstruktive Zusammenarbeit während meiner Zeit am Institut. Durch seine Unterstützung konnte ich an zahlreichen Konferenzen teilnehmen, wertvolles Feedback erhalten und entscheidende Impulse für meine Arbeit gewinnen. Zudem ermöglichte er mir einen längeren Auslandsaufenthalt, während dessen ich bedeutende Beiträge zu dieser Dissertation leisten konnte.

Ebenso danke ich Herrn Prof. Dr. Christoph Buchheim vom Lehrstuhl für Diskrete Optimierung der TU Dortmund für die Übernahme des Korreferats sowie für die inspirierende Zusammenarbeit im Rahmen des Graduiertenkollegs 1855.

Meinen Kolleginnen und Kollegen sowie studentischen Hilfskräften, die mich bei der Forschung und in Projekten begleitet und unterstützt haben, danke ich von Herzen. Besonderer Dank gilt Dr.-Ing. Moritz Pötting für die enge Zusammenarbeit zur Simulation innovativer Zustellkonzepte im urbanen Raum, Dr. Nicklas Klein für unsere gemeinsame Forschung zur Optimierung von Zustellrobotern auf der letzten Meile, Herrn Prof. Tolga Bektaş für die produktive Zeit in Liverpool und die gemeinsame Arbeit zur Geschwindigkeitsoptimierung von Paketdrohnen sowie Dr. Felix Hommelsheim und Felix Kreuz für ihre stete Hilfsbereitschaft und den wertvollen Austausch am Institut. Für die gemeinsame Zeit und die vielen bereichernden, motivierenden und oftmals wegweisenden Gespräche bin ich aufrichtig dankbar.

Meinem Bruder PD Dr. Oliver Schaudt danke ich für seine wertvollen Hinweise und Denkanstöße. Seine fachliche Unterstützung, seine tatkräftige Hilfe und seine aufbauenden Worte haben mir geholfen, motiviert zu bleiben und stets den Blick für das Wesentliche zu bewahren.

Mein tief empfundener Dank gilt auch meiner Ehefrau Isabell Schaudt. Sie hat mir in dieser herausfordernden Zeit mit Geduld und Verständnis den Rücken freigehalten. Ihre Bereitschaft, in vielen Lebensbereichen Mehrarbeit zu übernehmen, hat mir überhaupt erst ermöglicht, mich mit der notwendigen Konzentration dieser Dissertation zu widmen. Diese Arbeit ist auch ihr Verdienst.

Allen, die mich auf meinem Weg zur Promotion unterstützt, motiviert oder einfach an mich geglaubt haben, gilt mein aufrichtiger Dank.

Stefan Schaudt
Brühl, 2025

Zusammenfassung

Die Forschung an innovativen Zustellkonzepten für die letzte Meile der Logistik hat in den vergangenen Jahren erhebliche Fortschritte gemacht. Autonome Zustellkonzepte mit Zustellrobotern und Paketdrohnen werden weltweit erprobt oder bereits eingesetzt. Durch ihre Autonomie ermöglichen sie Zustellungen innerhalb vorgegebener Zeitfenster. Allerdings ist ihre Transportkapazität begrenzt, sodass häufig Pendeltouren zwischen Mikrodepots und Empfängern erforderlich sind. Die beiden Transportmittel unterscheiden sich grundlegend in ihrer Fortbewegung und Geschwindigkeit. Paketdrohnen fliegen mit hoher Geschwindigkeit auf direktem Weg durch die Luft, während Zustellroboter sich mit Schrittgeschwindigkeit auf Gehwegen bewegen. Bei Zustellrobotern ist die begrenzte Akkukapazität besonders relevant, da das Aufladen Zeit erfordert. Paketdrohnen hingegen erhalten bei ihrer Rückkehr ins Mikrodepot einen geladenen Akku. Aufgrund der hohen Geschwindigkeit steht bei Drohnen der Energieverbrauch im Fokus. Beide Transportmittel werden daher im Rahmen dieser Dissertation separat betrachtet.

Der Fokus dieser Arbeit liegt auf der Optimierung der letzten Meile unter Verwendung von Zustellrobotern und Paketdrohnen. Der Zustellprozess beginnt mit dem Transport der Sendungen von einem Depot in das Zustellgebiet, wo sie in Mikrodepots zwischengelagert werden. Anschließend erfolgt die Zustellung innerhalb definierter Zeitfenster mithilfe von Zustellrobotern oder Paketdrohnen. Nach jeder Zustellung kehren die Transportmittel zu einem Mikrodepot zurück, um eine neue Sendung aufzunehmen.

Für Zustellroboter wird ein Algorithmus zur Auftragsverteilung entwickelt, der die Anzahl zugestellter Sendungen maximiert. Dabei steht eine Flotte von Robotern zur Verfügung. Die Zustellung muss innerhalb der vorgegebenen Zeitfenster der Empfänger erfolgen und der Akku der Roboter muss an den Mikrodepots wieder aufgeladen werden. Zur Lösung kommt ein Branch-and-Price-Algorithmus zum Einsatz. Dazu wird zuerst das Teilproblem mit einem einzelnen Roboter und einer beliebigen, festen Empfängerreihenfolge betrachtet. Hierfür gilt es zu entscheiden, welche Mikrodepots zwischen den Empfängern angefahren werden, wie lange die Ladevorgänge dauern und ob überhaupt eine zulässige Lösung existiert. Dazu wird ein polynomieller Algorithmus entwickelt, dessen Ergebnisse in den Branch-and-Price-Algorithmus integriert werden. Laufzeitexperimente zeigen, dass Instanzen mit bis zu 60 Empfängern gelöst werden können.

Für Paketdrohnen ist die Problemstellung vergleichbar, wobei der Fokus auf der Minimierung des Gesamtenergieverbrauchs liegt. Auch hier wird zunächst eine feste Empfängerreihenfolge für eine einzelne Drohne betrachtet. Die Berechnung der optimalen Geschwindigkeiten erfolgt mithilfe eines aus der Literatur bekannten Algorithmus, der in polynomieller Zeit arbeitet. Im Rahmen dieser Arbeit wird dieser Algorithmus erweitert, um Mindest- und Höchstgeschwindigkeiten zu berücksichtigen. Darauf aufbauend wird ein weiterer Algorithmus entwickelt, der alle relevanten Belieferungszeitpunkte des letzten Empfängers einer gegebenen Kundenreihenfolge sowie die optimalen Geschwindigkeitsvektoren und Energieverbräuche berechnet. Beide Verfahren fließen in einen Branch-and-Price-Algorithmus ein, bei dem die berechneten Ankunftszeiten zur Formulierung einer Dominanzregel verwendet werden. Der entwickelte Algorithmus übertrifft bestehende Verfahren signifikant und ist im Durchschnitt über 17,5-mal schneller.

Abschließend wird eine Simulationsstudie vorgestellt, die die Paketzustellung mit Zustellrobotern und Paketdrohnen abbildet und den Mehrwert der Optimierungsansätze in verschiedenen Szenarien demonstriert.

Abstract

The research in the field of innovative delivery concepts for the last mile of logistics has made significant progress in recent years. Autonomous delivery solutions using delivery robots and parcel drones are being tested or have already been deployed worldwide. Thanks to their autonomy, these transport systems enable deliveries within predefined time windows. However, their transport capacity is limited, necessitating frequent shuttle trips between micro depots and recipients. The two transport systems differ fundamentally in terms of movement and speed. Parcel drones fly at high speeds on a direct path through the air, while delivery robots move at walking pace along sidewalks. For delivery robots, limited battery capacity is particularly critical, as recharging takes time. Parcel drones, on the other hand, return to the micro depot with a charged battery. Due to their high speed, energy consumption is a key factor. For this reason, both innovative delivery concepts are examined separately in the context of this dissertation.

The focus of this thesis is the optimization of last-mile delivery using delivery robots and parcel drones. The delivery process begins with the transport of parcels from a central depot to the delivery area, where they are temporarily stored in micro depots. Deliveries are then carried out within specified time windows using delivery robots or parcel drones. After each delivery, the vehicle returns to a micro depot to pick up the next parcel.

For delivery robots, an assignment algorithm is developed to maximize the number of deliveries. A fleet of robots is available for this purpose. Deliveries must take place within the specified time windows of the recipients, and the robots must recharge their batteries at the micro depots. A Branch-and-Price algorithm is employed to solve the problem. To this end, a subproblem is first considered: a single robot with an arbitrary, fixed customer sequence. The key decisions include which micro depots to visit between deliveries, how long to charge at each stop, and whether a feasible solution exists at all. A polynomial-time algorithm is developed for this subproblem, and its results are integrated into the overall Branch-and-Price framework. Runtime experiments show that this approach efficiently solves instances with up to 60 recipients.

The problem for parcel drones is comparable, although the focus here is on minimizing total energy consumption. Again, a fixed customer sequence for a single drone is considered initially. The optimal speeds are computed in polynomial time using a known algorithm, which is extended in this work to incorporate minimum and maximum speed limits. Building on this, an additional algorithm is developed to determine all relevant delivery times for the last recipient of a given customer sequence, as well as the corresponding optimal speed vectors and energy consumption values. Both procedures are integrated into a Branch-and-Price algorithm, in which the computed arrival times are used to formulate a dominance rule. The developed algorithm significantly outperforms existing methods and is, on average, more than 17.5 times faster.

Finally, a simulation study is presented that models parcel delivery using delivery robots and parcel drones, demonstrating the added value of the proposed optimization approaches across various scenarios.

Inhaltsverzeichnis

Abbildungsverzeichnis	xi
Tabellenverzeichnis	xiii
Abkürzungsverzeichnis	xv
1 Einleitung	1
1.1 Ziele der Arbeit	3
1.2 Aufbau der Arbeit	4
2 Innovative Zustellkonzepte für die letzte Meile der Logistik	5
2.1 Letzte Meile der Logistik	5
2.2 Zustellroboter	7
2.2.1 Zustellroboter für den Straßenverkehr	8
2.2.2 Zustellroboter für den Gehweg	9
2.2.3 Zustellkonzept	10
2.2.4 Vorteile und Nachteile	11
2.3 Paketdrohnen	12
2.3.1 Bauarten	13
2.3.2 Einsatzbereiche	14
2.3.3 Vorteile und Nachteile	16
3 Grundlagen der mathematischen Optimierung	17
3.1 Mathematische Optimierungsmodelle	17
3.1.1 Lineare Optimierung	18
3.1.2 Ganzzahlige Optimierung	19
3.2 Branch-and-Price-Verfahren	21
3.3 Graphentheorie	24
3.4 Dynamische Programmierung	25
4 Optimierung von Zustellnetzwerken mit Zustellrobotern	27
4.1 Einordnung in die Literatur	28
4.2 Notationen	31
4.3 Problembeschreibung	32
4.4 Gemischt-ganzzahliges lineares Programm	35
4.5 Dynamisches Programm für feste Sequenz	37
4.5.1 Zeit-Batterie-Kurven	38
4.5.2 Dynamisches Programm	41
4.6 Branch-and-Price-Algorithmus	48
4.6.1 Ganzzahliges lineares Programm und Pricing-Problem	48
4.6.2 Labeling-Algorithmus	50
4.6.3 Branching-Regeln	52

4.6.4	Beschleunigungsverfahren	54
5	Geschwindigkeitsoptimierung von Paketdrohnen	57
5.1	Energieverbrauch eines Multikopters	58
5.2	Einordnung in die Literatur	60
5.3	Notationen	63
5.4	Problembeschreibung	64
5.4.1	Optimierung eines Zustellnetzwerks mit Paketdrohnen	64
5.4.2	Vehicle Routing Problem With Time Windows and Speed Optimization .	67
5.4.3	Ganzzahliges lineares Programm	69
5.5	Geschwindigkeitsoptimierung für Sequenzen	69
5.5.1	Optimale Geschwindigkeiten für Sequenzen	70
5.5.2	Berechnung aller Belieferungszeitpunkte des letzten Kunden	77
5.6	Branch-and-Price-Algorithmus	86
5.6.1	Duales Programm und Pricing-Problem	86
5.6.2	Labeling-Algorithmus	87
5.6.3	Branching-Regel	90
5.6.4	Beschleunigungsverfahren	90
6	Analyse und Diskussion der Optimierungsergebnisse	93
6.1	Ergebnisse für Zustellroboter	93
6.2	Ergebnisse für die Geschwindigkeitsoptimierung	100
6.2.1	Seetransport	100
6.2.2	Straßengüterverkehr	104
7	Simulationsstudie	109
7.1	Simulationsmodell	109
7.1.1	Modellierung des Agenten Zustellroboter	113
7.1.2	Problemvarianten für Zustellroboter	115
7.1.3	Modellierung des Agenten Drohne	118
7.1.4	Problemvarianten für Paketdrohnen	121
7.1.5	Verifikation des Simulationsmodells	122
7.2	Experimente und Ergebnisse	123
7.2.1	Ergebnisse für Zustellroboter	125
7.2.2	Ergebnisse für Paketdrohnen	137
8	Zusammenfassung und Ausblick	143
A	Anhang	155
A	Korrektheitsbeweis von Algorithmus 1	155
B	Rückwärtsgerichtetes dynamisches Programm	168
C	Korrektheitsbeweise der Algorithmen 5 und 6	171
D	Resultate für das Pollution Routing Problem	184

Abbildungsverzeichnis

1.1	Vorgehensmodell für Optimierungsprobleme in der Logistik (Clausen und Geiger, 2013, S. 330)	4
2.1	Struktur eines Hub-and-Spoke-Netzwerks (Wellbrock et al., 2022, S.6)	6
2.2	Struktur einer zweistufigen letzten Meile	7
2.3	Anwendungsfelder für Zustellroboter	8
2.4	Zustellroboter für die Straße und den Gehweg	9
2.5	Paketdrohnen unterschiedlicher Bauarten	12
3.1	Schematischer Aufbau des Branch-and-Price-Verfahrens	23
4.1	Visualisierung des Graphen und einer Lösung	33
4.2	Graph mit zwei Kunden und zwei Depots	34
4.3	Akkustand über die Zeit für zwei unterschiedliche Zustellpfade	35
4.4	Beispiel einer gebrochenen Geraden mit drei Abschnitten	40
4.5	Vergleich zwischen temporären Paaren und der resultierenden gebrochenen Geraden	45
4.6	Zwei Beispiele für den vorwärtsgerichteten Vergleich von gebrochenen Geraden	55
5.1	Zusammensetzung der Leistung in Abhängigkeit der Geschwindigkeit (Zeng et al., 2019)	59
5.2	Einfluss des Sendungsgewichts auf den Energieverbrauch (Dukkanci et al., 2021)	60
5.3	Strikt konvexe Funktion (durchgezogen) transformiert in eine monoton steigende konvexe Funktion (gestrichelt und durchgezogen)	62
5.4	Gerichteter Graph mit einem Depot (Quadrat) und vier Kunden (Kreise)	65
5.5	Binärer Baum zur Visualisierung der Teilprobleme	73
5.6	Visualisierung der Ergebnisse des Geschwindigkeitsalgorithmus	74
5.7	Verlauf der Kosten in Abhängigkeit des Belieferungszeitpunkts	77
5.8	Visualisierung der Ergebnisse des Algorithmus 6	81
5.9	Entwicklung der Geschwindigkeit mit dem naiven Ansatz und Algorithmus 6	81
5.10	Aufteilung einer Sequenz s in Teilsequenzen $\tilde{s}(\mathbf{v})$ und $\bar{s}_1(\mathbf{v}), \dots, \bar{s}_p(\mathbf{v})$	83
5.11	Beispiel für drei Sequenzen, die zusammen mit ihrem Geschwindigkeitsvektor ein Match bilden und zwei Sequenzen mit Geschwindigkeitsvektoren ohne Match	84
5.12	Zwischen den Kurven besteht keine Dominanz	91
6.1	Lösung mit 30 Kunden, sechs Depots und zwei Zustellrobotern	95
6.2	Ergebnisse für Instanzen mit 34 bis 50 Kunden und drei Robotern	97
6.3	Ergebnisse für Instanzen mit 60 Kunden und einem bis sechs Robotern	98
6.4	Ergebnisse für Instanzen mit 60 Kunden, vier Robotern und Zeitfensterlängen zwischen 10 und 60 Minuten	99
6.5	Kostenfunktion des Pollution-Routing-Problems in Abhängigkeit von der Geschwindigkeit	106
6.6	Ergebnisse für das PRP auf den Instanzen der Teilmengen UK-A, UK-B und UK-C107	

7.1	Zustandsdiagramm des Agenten Zustellroboter	114
7.2	Zustandsdiagramm des Agenten Drohne	120
7.3	Flugmuster einer Paketdrohne (Kirschstein, 2020)	120
7.4	Zeitlicher Ablauf eines Simulationstages	123
7.5	Zustellgebiet im Stadtteil Altstadt Süd mit 2 Mikrodepots	125
7.6	Servicegrade für vier Zustellroboter mit und ohne fester Zuweisung zu den Mikrodepots	126
7.7	Zustellgebiet in den Stadtteilen Altstadt Nord und Süd mit 5 Mikrodepots	127
7.8	Gegenüberstellung des Servicegrads mit fünf Zustellrobotern und 30 bis 54 Kunden mit und ohne fester Zuweisung zu den Mikrodepots	128
7.9	Zustellgebiet im Stadtteil Bayenthal mit einem Mikrodepot	129
7.10	Servicegrade für verschiedene Kundenanzahlen und Ladestrategien	129
7.11	Gegenüberstellung der Ladezeiten für verschiedene Ladestrategien	131
7.12	Verlauf des Ladestandes eines Zustellroboters zwischen 9 und 15 Uhr	132
7.13	Verteilung auf fünf Mikrodepots für die Altstadt Nord und Altstadt Süd	133
7.14	Verteilung auf drei Mikrodepots für die Altstadt Nord und Altstadt Süd	134
7.15	Verteilung der Distanz je Belieferung	135
7.16	Verteilung der Gesamtdistanz für 5 Roboter über 15 Tage	135
7.17	Verteilung des Energieverbrauchs je Belieferung	136
7.18	Verteilung des Gesamtenergieverbrauchs für fünf Roboter über 15 Tage	136
7.19	Depot und Zustellgebiet in der Altstadt Nord und Süd der Stadt Köln	138
7.20	Verteilung der optimalen Geschwindigkeiten für zwei Drohnen	139
7.21	Verteilung diskreter Geschwindigkeiten für zwei Drohnen bei unterschiedlicher Anzahl an Diskretisierungsschritten	140
7.22	Zustellgebiet in der Altstadt Nord und Altstadt Süd mit einem Mikrodepot	142
C.1	Ankunftszeitpunkte mit Geschwindigkeitsvektor $\bar{\mathbf{v}}$	178
C.2	Visualisierung von Fall (2.1)	180
C.3	Visualisierung von Fall (2.2)	180
C.4	Visualisierung von Fall (2.3.1)	181
C.5	Visualisierung von Fall (2.3.2)	181

Tabellenverzeichnis

2.1	Spezifikation des Modells R2 von Nuro (Ferguson, 2020)	8
2.2	Spezifikation des Zustellroboters von Starship Technologies (Schweizerische Post AG, 2017)	10
2.3	Spezifikation der Paketdrohne W178 von Wingcopter (2021)	13
2.4	Spezifikation der Paketdrohne MK27-2 von Amazon (Federal Aviation Administration, 2022)	13
4.1	Notationen für Kapitel 4	31
4.2	Fahrzeiten zwischen den Depots und zwischen Depots und Kunden	44
4.3	Zulässigkeits-, Erweiterungs- und Vereinigungsregeln für den Labeling-Algorithmus	53
5.1	Notationen für den Energieverbrauch einer Drohne (Dukkanci et al., 2021)	59
5.2	Notationen für Kapitel 5	63
5.3	Übersicht zu Beispiel 3	73
5.4	Übersicht zu Beispiel 4	80
5.5	Zulässigkeits- und Erweiterungsregel für den Labeling-Algorithmus	89
6.1	Vergleich zwischen MIP und Branch-and-Price-Algorithmus für 15 Kunden und einem Roboter	96
6.2	Vergleich zwischen MIP und Branch-and-Price-Algorithmus für 30 Kunden und zwei Robotern	97
6.3	Resultate des Branch-and-Price-Algorithmus im Vergleich zu Fukasawa et al. (2018) auf den deep-sea shipping-Instanzen und short-sea shipping-Instanzen	103
6.4	Parameter des Pollution-Routing Problems (Demir et al., 2012)	105
6.5	Vergleich zwischen dem Branch-and-Price-Algorithmus und dem Algorithmus von Fukasawa et al. (2017, 2018) auf den UK-A, UK-B und UK-C Instanzen	108
7.1	Annahmen für das Simulationsmodell	111
7.2	Beispielhafte Auftragsliste für einen Zustellroboter	113
7.3	Beispielhafte Auftragsliste für eine Paketdrohne	119
7.4	Energieverbrauch mit kontinuierlichen und diskreten Geschwindigkeiten	140
D.1	Ergebnisse für 10, 15, 20 und 25 Kunden auf den UK-A Instanzen	185
D.2	Vergleich zwischen dem BCP-Algorithmus von (Fukasawa et al., 2017, 2018) und dem Branch-and-Price-Algorithmus auf den UK-A Instanzen	186
D.3	Ergebnisse für 10, 15, 20 und 25 Kunden auf den UK-B Instanzen	187
D.4	Ergebnisse für 50 und 75 Kunden auf den UK-B Instanzen	188
D.5	Vergleich zwischen dem BCP-Algorithmus von (Fukasawa et al., 2017, 2018) und dem Branch-and-Price-Algorithmus auf den UK-B Instanzen	189
D.6	Ergebnisse für 10, 15, 20 und 25 Kunden auf den UK-C Instanzen	190
D.7	Ergebnisse für 50 und 75 Kunden auf den UK-C Instanzen	191

D.8 Vergleich zwischen dem BCP-Algorithmus von (Fukasawa et al., 2017, 2018) und dem Branch-and-Price-Algorithmus auf den UK-C Instanzen	192
--	-----

Abkürzungsverzeichnis

Akku Akkumulator

B2B Business-to-Business

B2C Business-to-Consumer

BCP Branch-Cut-and-Price

DP duales Programm

EMP eingeschränktes Master Problem

EV RPTW Electric Vehicle Routing Problem with Time Windows

FRVCP Fixed Route Vehicle Charging Problem

GIS Geoinformationssysteme

IP ganzzahliges lineares Programm

KEP Kurier-, Express- und Paketdienste

MIP gemischt-ganzzahliges lineares Programm

MP Master Problem

NLP nichtlineares Programm

RPIT Routing- und Planungsprobleme von Industrie- und Trampschiffen

PRP Pollution-Routing Problem

VRP Vehicle Routing Problem

VRPTWSO Vehicle Routing Problem with Time Windows and Speed Optimization

Kapitel 1

Einleitung

Der weltweite Online-Handel verzeichnet seit seiner Entstehung ein kontinuierliches Wachstum, das sich in den letzten Jahren durch technologische Innovationen und verändertes Konsumverhalten nochmals erheblich beschleunigt hat. Im Jahr 2023 wurde in Deutschland ein Umsatz von 85,4 Milliarden € erzielt. Damit hat sich der Umsatz seit 2015 mehr als verdoppelt (Handelsverband Deutschland (HDE), 2024). Dieses Wachstum wirkt sich auch auf den Kurier-, Express- und Paketdienstmarkt (KEP-Markt) aus. Im Jahr 2023 wurden in Deutschland 4,18 Milliarden Sendungen zugestellt, was eine Steigerung von 40 % innerhalb der letzten 8 Jahre bedeutet. Für 2028 wird eine Menge von etwa 4,7 Milliarden Sendungen erwartet (BIEK, 2024).

Die steigenden Sendungsmengen und das Ziel, nachhaltiger zu agieren, stellen die KEP-Branche vor große Herausforderungen. Dabei steht die letzte Meile der Logistik besonders im Fokus. Einerseits steigt die Anzahl der zuzustellenden Sendungen, andererseits wird die Erreichbarkeit der Empfänger durch zunehmende Erwerbstätigkeit und die wachsende Zahl von Single-Haushalten erschwert (Clausen und Geiger, 2013, S. 287). Daneben führt der wachsende Lieferverkehr zu mehr Lärm- und Schadstoffemissionen, verschlechtert die Verkehrssituation und verursacht Staus im urbanen Raum (Hölter und Ninnemann, 2020, S. 31).

Neben der steigenden Anzahl an Sendungen hat auch die ökologische Verantwortung der Branche deutlich zugenommen. KEP-Unternehmen haben durch ihre Fahrzeugflotten eine große Sichtbarkeit im Straßenraum und stehen daher im Fokus der Öffentlichkeit (BIEK, 2024). In Deutschland betrugen die Treibhausgasemissionen des Verkehrssektors im Jahr 2022 148,6 Millionen Tonnen CO_2 -Äquivalente (Bundesministerium für Digitales und Verkehr, 2024, S. 315). Dabei beschreiben CO_2 -Äquivalente die Maßeinheit, die verschiedene Treibhausgase in einem Wert zusammenfassen, basierend auf ihrem jeweiligen Beitrag zum Treibhauseffekt im Vergleich zu Kohlendioxid (Green Vision Solution, 2023). Um die Emissionen zu reduzieren, hat die Bundesregierung das Bundes-Klimaschutzgesetz erlassen. Demnach sollen die Emissionen bis 2030 auf 84 Millionen Tonnen CO_2 -Äquivalente sinken und der Verkehrssektor soll bis 2045 treibhausgasneutral werden (Umweltbundesamt, 2023). In Reaktion auf diese Vorgaben haben KEP-Dienstleister begonnen, ihre Flotten auf Elektrofahrzeuge umzustellen. Im Jahr 2022 lag der Anteil an elektrischen

Nutzfahrzeugen bis 3,5 t bei 16 %. Zum Vergleich: Der Anteil elektrischer Kfz in Deutschland insgesamt betrug lediglich 2 %. Damit können Emissionen gesenkt werden. Allerdings löst die Umrüstung auf eine nachhaltigere Fahrzeugflotte nicht die Verkehrsbelastung im urbanen Raum.

Zur Reduzierung der Verkehrsbelastung und der Emissionen werden neben modernen Fahrzeugflotten auch alternative Zustellkonzepte entwickelt. Ein Beispiel ist der Einsatz von Lastenrädern. Dabei werden die Pakete gebündelt in ein Zustellgebiet transportiert und dort mit Lastenrädern zugestellt. Dadurch lassen sich die Emissionen einer Paketsendung um 10 % und die Emissionen der letzten Meile um über 50 % senken. Im Jahr 2022 wurden in Deutschland 450 Lastenräder für die Paketzustellung eingesetzt (BIEK, 2024). Daneben gibt es weitere innovative Konzepte, die auf autonomen Transportmitteln basieren.

Der Fokus dieser Arbeit liegt auf der Optimierung der Paketzustellung mit innovativen Zustellkonzepten, insbesondere Zustellrobotern und Paketdrohnen. Beide Transportmittel sind elektrisch angetrieben und bewegen sich autonom. Ein weiteres gemeinsames Merkmal ist, dass sie nicht auf die Straße als Verkehrsträger angewiesen sind. Zustellroboter fahren auf Gehwegen oder in Fußgängerzonen und Paketdrohnen nutzen die Luft als Verkehrsträger. Dadurch kann der Verkehr auf den Straßen entlastet werden. Beide Transportmittel haben jedoch eine eingeschränkte Transportkapazität. Die meisten Modelle können nur eine Sendung oder eine Einheit transportiert werden. Dies führt zu Pendeltouren zwischen einer Lagerstätte und den Empfängern. Aufgrund der geringen Kapazität und der Autonomie eignen sich diese Transportmittel besonders für zeitkritische Zustellungen oder Zustellungen in einem bestimmten Zeitfenster. So kann ein hoher Servicegrad für die Kunden gewährleistet werden.

Zur Veranschaulichung des Konzepts wird ein beispielhafter Zustellprozess mit Zustellrobotern vorgestellt, wie er von Starship Technologies in Kooperation mit der Hermes Germany GmbH in einem Pilotversuch in Hamburg umgesetzt wurde (Bertram I., 2016). Im Prozess werden im ersten Schritt die Sendungen mit einem Transporter von einem Depot zum Zustellgebiet transportiert und dort in Mikrodepots zwischengelagert. Von dort aus starten die Zustellroboter die Belieferungen der Empfänger. Jeder Empfänger konnte am Vortag ein Zeitfenster für die Belieferung aussuchen. Nach jeder Zustellung fährt der Roboter zurück zu einem Mikrodepot, wo er entweder aufgeladen oder für die nächste Zustellung vorbereitet wird. Für Paketdrohnen lässt sich der gleiche Zustellprozess anwenden, wobei es auch die Möglichkeit gibt, dass die Drohnen direkt vom Depot starten. Somit würde der gebündelte Transport zu den Mikrodepots entfallen.

Der Hauptunterschied zwischen den beiden Transportmitteln liegt in der Geschwindigkeit und dem Energieverbrauch. Paketdrohnen erreichen hohe Geschwindigkeiten, was zu einem signifikant höheren Energieverbrauch führt im Vergleich zu den Zustellrobotern. Aus diesem Grund wird für die vorliegende Arbeit angenommen, dass der Akku der Drohnen mit jeder Beladung gewechselt werden muss. Der Fokus liegt auf der Auswahl der optimalen Fluggeschwindigkeiten. Für die Zustellroboter werden Ladevorgänge modelliert und in die Optimierung einbezogen. Obwohl diese Art von Problemstellungen durch die Pendeltouren simpel erscheint, ist das zugrundeliegende Planungsproblem sehr eng mit dem Tourenplanungsproblem verwandt, wodurch die Berechnung einer optimalen Lösung aufwändig ist.

1.1 Ziele der Arbeit

Das primäre Ziel dieser Arbeit ist die Entwicklung, Implementierung und Simulation von Optimierungsverfahren für innovative Zustellkonzepte auf der letzten Meile der Logistik. Im Fokus stehen hierbei Zustellroboter und Paketdrohnen. Beide Konzepte zeichnen sich durch ihre begrenzte Transportkapazität aus, da oft nur eine Sendung gleichzeitig transportiert werden kann. Diese Arbeit befasst sich insbesondere mit der Optimierung der Zustellung auf der letzten Meile, wobei für Zustellroboter der Fokus auf der Planung von Ladevorgängen an Mikrodepots und für Paketdrohnen auf der Wahl der optimalen Fluggeschwindigkeit und der Minimierung des Energieverbrauchs liegt.

Ziel dieser Arbeit ist es, Algorithmen zu entwickeln, die die operative Planung eines Mikrodepot-Netzwerks mit Zustellrobotern oder Paketdrohnen ermöglichen. Wesentliche Beiträge zum Erreichen dieses Ziels sind:

- **Flexibles Zustellnetzwerk:** Ziel ist der Aufbau eines anpassungsfähigen Netzwerks, das es den Zustellrobotern ermöglicht, nach jeder Lieferung ein anderes Mikrodepot anzusteuern. Dies erhöht sowohl die betriebliche Effizienz als auch die Auslastung der verfügbaren Transportmittel.
- **Berücksichtigung von Lademengen, -orten und -zeitpunkten:** Angesichts des elektrischen Antriebs der Zustellroboter müssen die Ladevorgänge an den Mikrodepots so geplant werden, dass der Ladestand des Akkus in den vorgegebenen Grenzen bleibt.
- **Optimierung der Fluggeschwindigkeit der Paketdrohnen:** Der Energieverbrauch in Abhängigkeit von der Geschwindigkeit ist eine streng konvexe Funktion. Daher ist es entscheidend, die Geschwindigkeit so zu wählen, dass der Energieverbrauch minimiert wird.
- **Effiziente Bewertung von festen Zustellreihenfolgen:** Für das Branch-and-Price-Verfahren ist es von zentraler Bedeutung, das Entscheidungsproblem, ob eine gegebene Reihenfolge von Empfängern zulässig ist, in polynomieller Zeit zu berechnen. Zudem müssen die entstehenden Kosten berechnet werden.
 - Im Fall der Zustellroboter müssen zwischen den Empfängern Mikrodepots mit entsprechenden Ladezeiten eingeplant werden, um zu entscheiden, ob eine gegebene Reihenfolge von Empfängern zulässig ist.
 - Im Fall der Paketdrohnen gilt es eine zulässige Lösung für eine Reihenfolge der Empfänger zu finden, die gleichzeitig den Energieverbrauch minimiert. Dazu müssen die Fluggeschwindigkeiten entsprechend gewählt werden.

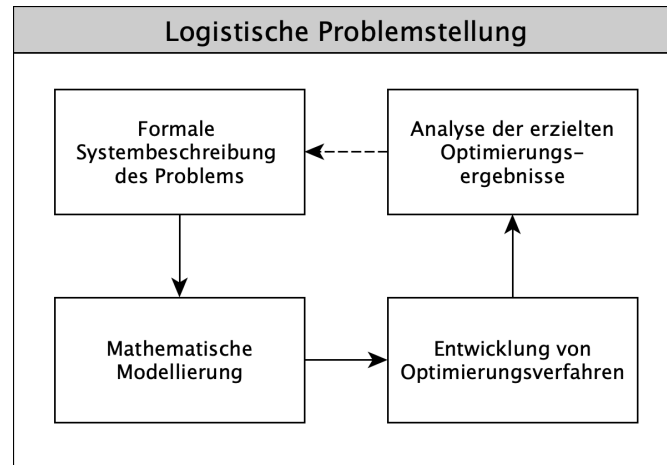


Abbildung 1.1: Vorgehensmodell für Optimierungsprobleme in der Logistik (Clausen und Geiger, 2013, S. 330)

1.2 Aufbau der Arbeit

Der Aufbau der Arbeit orientiert sich an dem Vorgehensmodell für Optimierungsprojekte in der Logistik, wie es von Geiger und Clausen (Clausen und Geiger, 2013, S. 329ff.) beschrieben wird. Die einzelnen Schritte sind in Abbildung 1.1 dargestellt. Zunächst wird die Problemstellung formal beschrieben und im Anschluss folgt die mathematische Modellierung. Dann werden geeignete Algorithmen zur Lösung entwickelt. Zum Schluss werden Ergebnisse generiert und analysiert.

Kapitel 2 beschreibt die logistischen Grundlagen, die zum Verständnis der Arbeit erforderlich sind. Dazu zählen die Definition der *letzten Meile* sowie die Darstellung innovativer Zustellkonzepte. Kapitel 3 vermittelt die mathematischen Grundlagen. Der Schwerpunkt liegt auf der Vorstellung des Branch-and-Price-Verfahrens, das in den folgenden Kapiteln Anwendung findet.

Die nächsten beiden Kapitel bilden den Kern der Arbeit. Kapitel 4 fokussiert sich auf die Optimierung eines Zustellnetzwerks mit Zustellrobotern. Es wird ein exakter Algorithmus vorgestellt, der das Problem der Zuordnung der Pakete auf die Mikrodepots und Roboter löst. Zudem werden für die Roboter Reihenfolgen der Zustellung und Ladezeiten an den Mikrodepots berechnet. Im Kapitel 5 liegt der Fokus auf der Optimierung eines Zustellnetzwerks mit Paketdrohnen. Auch für dieses Problem wird ein exakter Algorithmus vorgestellt, der die Auftragsverteilung auf die Paketdrohnen löst. Dabei wird der Energieverbrauch über alle Paketdrohnen minimiert.

Das Kapitel 6 stellt für die beiden vorangegangenen Kapitel Optimierungsergebnisse vor. Dabei werden die Korrektheit und Performance der Algorithmen analysiert. Kapitel 7 stellt eine Simulationsstudie für die innovativen Zustellkonzepte vor. Die Simulation untersucht die Paketzustellung mit Zustellrobotern und Paketdrohnen im urbanen Raum. Der Fokus liegt auf der Untersuchung des Nutzens der Algorithmen und der Anwendbarkeit innovativer Zustellkonzepte in der letzten Meile. Die Arbeit wird im Kapitel 8 mit einem Fazit abgeschlossen.

Zur besseren Lesbarkeit wird in der vorliegenden Arbeit die männliche Form verwendet.

Kapitel 2

Innovative Zustellkonzepte für die letzte Meile der Logistik

In diesem Kapitel werden die für das Verständnis der Arbeit notwendigen logistischen Grundbegriffe eingeführt. Zusätzlich werden zwei innovative Zustellkonzepte für den Transport von Waren auf der *letzten Meile* vorgestellt. Dazu zählen Zustellroboter und Paketdrohnen.

2.1 Letzte Meile der Logistik

Der Begriff der letzten Meile lässt sich anhand eines logistischen Transportnetzes verdeutlichen (siehe Abbildung 2.1). Die Abbildung visualisiert ein sogenanntes Hub-and-Spoke-Netzwerk, wie es typischerweise von KEP-Dienstleistern genutzt wird. In einem Hub-and-Spoke-Netzwerk wird generell zwischen Vor-, Haupt- und Nachlauf unterschieden. Im Vorlauf werden die Sendungen gesammelt, im Hauptlauf gebündelt transportiert und im Nachlauf an die Empfänger verteilt (Thaller et al., 2017, S.448). Der Nachlauf wird auch als *letzte Meile* bezeichnet.

Die letzte Meile bezeichnet allgemein den finalen Abschnitt einer Distributionskette bis zum Empfänger. Für Paketsendungen wird dieser Abschnitt von KEP-Dienstleistern durchgeführt. Die Betrachtung der letzten Meile ist von besonderer Bedeutung, da etwa 50 % der Gesamtkosten des Transports in diesem Abschnitt anfallen. Ein Grund für die hohen Kosten sind unter anderem Fehlzustellungen sowie der erhebliche personelle Aufwand für die Paketzustellung (Clausen und Geiger, 2013, S. 287).

Im Folgenden wird der Zustellprozess für Paketsendungen auf der letzten Meile mit einem Zustellfahrzeug beschrieben. Die letzte Meile startet in einem Depot, wie in Abbildung 2.1 dargestellt. Dort erfolgt die Endsortierung der Pakete in sogenannte Empfangsgebiete. Diese Gebiete sind anhand der Einwohnerzahl, Haushaltsdichte sowie der zu bewältigenden Strecke aufgeteilt. Im Depot wird je Gebiet ein Zustellfahrzeug beladen (Clausen und Geiger, 2013, S.287). Anschließend werden die Pakete in die Empfangsgebiete transportiert und an die Empfänger zugestellt.

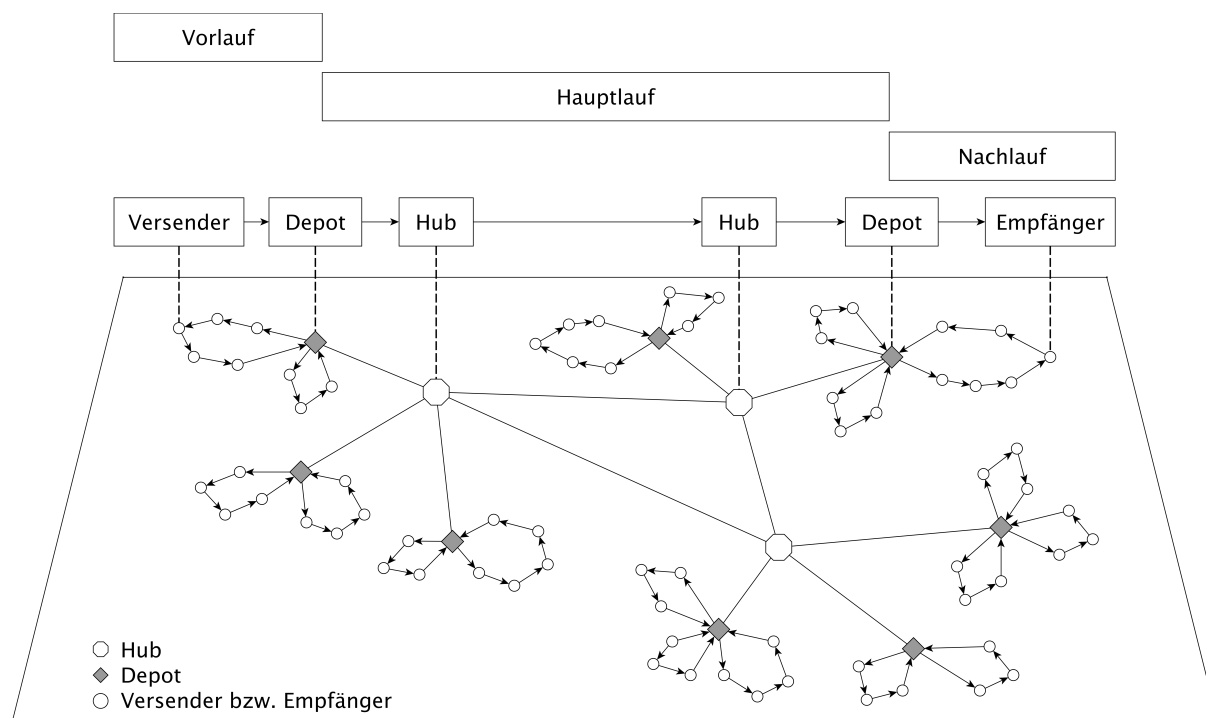


Abbildung 2.1: Struktur eines Hub-and-Spoke-Netzwerks (Wellbrock et al., 2022, S.6)

Falls es zu einer Fehlzustellung kommt, da der Empfänger nicht angetroffen werden konnte, wird entweder ein erneuter Zustellversuch am folgenden Tag unternommen oder die Pakete werden in einem Paketshop oder einer Paketstation hinterlegt. Dabei gilt, dass die Zwischenlagerstätte in der Nähe des Kunden liegen sollte (Clausen et al., 2016, S.44). Das Paket kann dort innerhalb der Öffnungszeiten abgeholt werden. Paketshops können Geschäfte des KEP-Dienstleisters oder private Einrichtungen, wie etwa Kioske oder Tankstellen, sein. Paketstationen sind automatisierte, autarke Stationen, die von den Dienstleistern betrieben werden. Je nach Kundenwunsch ist auch eine direkte Belieferung des Paketshops oder der Paketstation möglich, die im Durchschnitt 25 bis 30 % günstiger ist als eine Übergabe an der Haustür (Wellbrock et al., 2022, S.9). Damit ist die Vorstellung des Zustellprozesses auf der letzten Meile mit einem Zustellfahrzeug abgeschlossen. Neben der Zustellung mit einem Transporter kann die Zustellung auf der letzten Meile auch mit innovativen Konzepten durchgeführt werden. Eine Übersicht über solche Konzepte geben Boysen et al. (2020). Im Folgenden wird eine zweistufige letzte Meile für den Einsatz innovativer Zustellkonzepte beschrieben.

Zweistufige letzte Meile der Logistik

Für die Auslieferung von Paketsendungen mit Zustellrobotern oder Paketdrohnen wird die letzte Meile in zwei Abschnitte unterteilt (Bakach et al., 2021). Im ersten Abschnitt werden die Sendungen vom Depot aus in das Zustellgebiet transportiert und dort auf Mikrodepots verteilt. Als Mikrodepots dienen Paketshops oder Paketstationen. Im zweiten Schritt werden die Sendungen mithilfe innovativer Transportmittel, wie etwa Zustellrobotern oder Paketdrohnen, zu den Empfängern transportiert (Clausen et al., 2016, S.44). Dieser zweite Schritt wird auch als

allerletzte Meile bezeichnet. Eine schematische Darstellung der zweistufigen letzten Meile ist in Abbildung 2.2 dargestellt. Neben der zweistufigen letzten Meile sind auch Mischformen möglich. Beispielsweise können Zustellfahrzeuge, die den ersten Abschnitt der letzten Meile übernehmen, auch direkte Zustellungen durchführen (Poeting et al., 2019b). Hierbei werden sowohl Mikrodepots als auch Empfänger angefahren. Nach der Beschreibung der zweistufigen letzten Meile wird nun ein innovatives Zustellkonzept erläutert, das auf dem Einsatz von Zustellrobotern basiert.

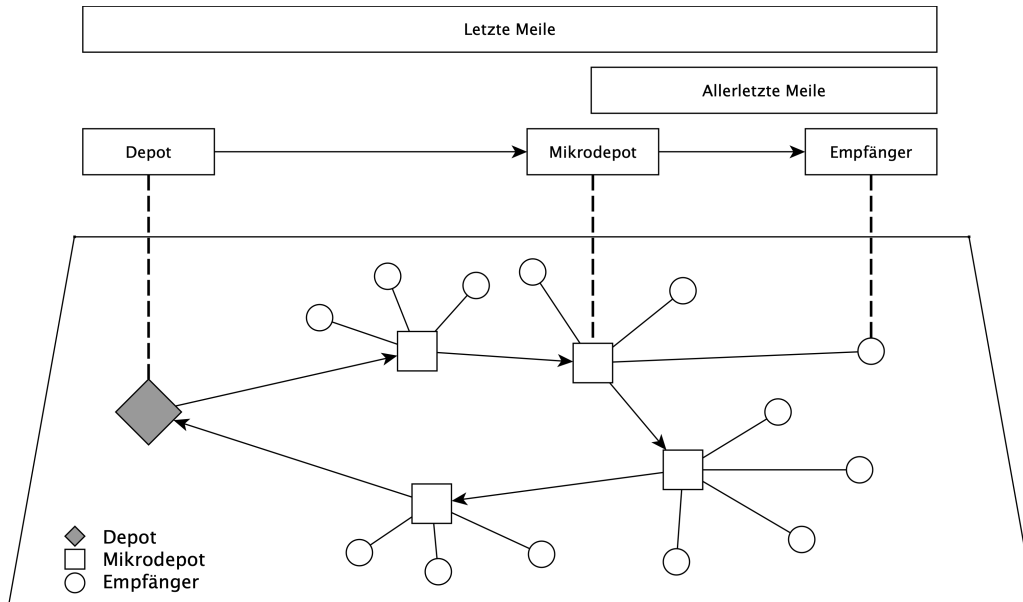


Abbildung 2.2: Struktur einer zweistufigen letzten Meile

2.2 Zustellroboter

Ein Konzept für die Paketzustellung auf der allerletzten Meile beruht auf dem Einsatz autonomer, batteriebetriebener Zustellroboter. Die Roboter transportieren Waren über Straßen oder Gehwege. Der Fokus dieser Arbeit liegt auf Robotern, die auf Gehwegen fahren und eine Transportkapazität von einer Sendung haben, wie es der Regelfall ist. Durch den Transport auf den Gehwegen sind die Größe und damit verbunden die Transportkapazität eingeschränkt. Die Autonomie und der begrenzte Laderaum der Roboter ermöglichen eine zeitlich genaue Zustellung an die Empfänger. Dabei eignen sich besonders zeitkritische Waren oder solche, die eine zeitgenaue Lieferung erfordern. Beispiele hierfür sind Medikamente, Mahlzeiten, Blumen, Einkäufe oder bestimmte Paketsendungen (siehe Abbildung 2.3). Ein führender Hersteller gibt an, dass Lebensmittel am häufigsten mit Zustellrobotern transportiert werden (StarShip Technologies (2022); Bogdanski (2017)). Zustellroboter lassen sich nach ihrem Einsatzort kategorisieren (Leerkamp et al., 2021). Neben dem Transport auf Gehwegen gibt es auch Modelle, die auf der Straße fahren. Diese Modelle werden in dieser Arbeit nicht näher betrachtet, aber zur Vollständigkeit mit vorgestellt. Innerhalb jeder Kategorie ähneln sich die Roboter hinsichtlich ihrer Größe, Geschwindigkeit und Transportkapazität. Im Folgenden werden zunächst Zustellroboter für den Straßenverkehr und anschließend solche für den Einsatz auf Gehwegen vorgestellt.

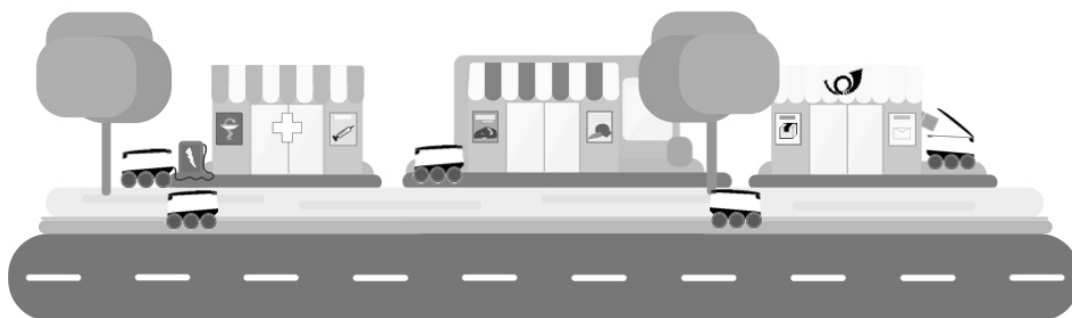


Abbildung 2.3: Anwendungsfelder für Zustellroboter

2.2.1 Zustellroboter für den Straßenverkehr

Zustellroboter lassen sich grundsätzlich in zwei Kategorien einteilen: konventionelle Fahrzeuge, die für die autonome Paketzustellung umgerüstet wurden, und speziell für die autonome Zustellung entwickelte Fahrzeuge. Zu den konventionellen Fahrzeugen zählen häufig Mittelklassewagen oder Kastenwagen, von denen nur ein Teil batteriebetrieben ist. Der Vorteil dieser Herangehensweise besteht darin, dass bereits vorhandene Fahrzeuge genutzt und mit zusätzlicher Hardware für den autonomen Einsatz ausgestattet werden können. Ein Beispiel hierfür ist das Unternehmen Gatik, das einen Ford Transit Connect für die autonome Zustellung umgerüstet hat.

Zur zweiten Kategorie gehören Roboter, die speziell für den autonomen Transport und die Paketzustellung entwickelt wurden. Ein wesentlicher Unterschied zu den konventionellen Fahrzeugen ist, dass kein Fahrersitz vorhanden ist. Ein Beispiel für einen solchen Roboter ist das Modell R2 von Nuro (siehe Abbildung 2.4(a)). Der Roboter ist mit der erforderlichen Sensorik für autonomes Fahren ausgestattet und besitzt die Größe eines Kleinwagens. Er verfügt über zwei Innenfächer, die über Flügeltüren zugänglich sind (siehe Abbildung 2.4(a)). Die Innenfächer können durch vorgefertigte Einsätze je nach Anwendungszweck unterteilt werden. Der Roboter erreicht eine Höchstgeschwindigkeit von 40 km/h und wiegt 1,15 Tonnen. Eine Übersicht der technischen Daten des R2 findet sich in Tabelle 2.1. Beispiele für weitere Hersteller dieser Transportmittel sind Neolix, Udelv, Toyota und JD.com (Cregger et al., 2020, S. 61ff.). Damit ist die Vorstellung des Zustellroboters für die Straße abgeschlossen.

Maße	L 2,74 m x B 1,10 m x H 1,86 m
Gewicht ohne Zuladung	1150 kg
Maximale Zuladung	190 kg
Ladevolumen	633 l
Maximale Geschwindigkeit	40 km/h
Batteriekapazität	31 kWh
Ladegeschwindigkeit	6,6 kWh/h

Tabelle 2.1: Spezifikation des Modells R2 von Nuro (Ferguson, 2020)

2.2.2 Zustellroboter für den Gehweg

Zustellroboter für den Einsatz auf Gehwegen haben eine kompakte Bauweise und bewegen sich mit Schrittgeschwindigkeit fort. Im September 2020 gab es weltweit 38 verschiedene Modelle von Zustellrobotern für den Einsatz auf dem Gehweg, hergestellt von 32 Unternehmen (Cregger et al., 2020). Einer der am weitesten verbreiteten Zustellroboter wurde von dem Hersteller Starship Technologies entwickelt. Starship Technologies, ein estnisches Unternehmen, wurde 2014 gegründet. Die technischen Daten des Roboters finden sich in Tabelle 2.2 und eine Abbildung des Modells in Abbildung 2.4(b). Der Roboter verfügt über einen Laderaum mit einem Volumen von 47 Litern. Die maximale Geschwindigkeit ist auf 6 km/h begrenzt und die Reichweite des Roboters beträgt 6 km. Ein Vergleich der Außenabmessungen zeigt, dass der Roboter von Starship Technologies im Vergleich zum für die Straße konzipierten Nuro R2 deutlich kompakter ist und das Ladevolumen etwa 13-mal kleiner ist.



(a) Nuro R2 (Ferguson, 2020)



(b) Starship Technologies

Abbildung 2.4: Zustellroboter für die Straße und den Gehweg

Zustellroboter, die auf dem Gehweg fahren, werden für den Einsatz im urbanen oder suburbanen Raum konzipiert. Dazu werden Größe, Geschwindigkeit und Gewicht der Roboter entsprechend an die Einsatzbedingungen angepasst. Allerdings liegen für diesen Einsatzort zwei konträre Zielvorgaben vor. Zum einen sollten die Distanzen zwischen Mikrodepots und Empfängern minimiert werden, da die Roboter nur in Schrittgeschwindigkeit fahren und eine begrenzte Reichweite haben. Daher eignen sich besonders urbane Räume mit einem eng verzweigten Netz an Mikrodepots und damit verbundenen kurzen Distanzen zu den Empfängern. Zum anderen sollte der Begegnungsverkehr mit Passanten möglichst gering bleiben, da die Roboter auf diese Rücksicht nehmen und unerwartetes Verhalten vorhersehen müssen. Dies reduziert die Geschwindigkeit der Roboter und erhöht damit die Anfahrtszeiten. Daher wären Räume mit geringer Bevölkerungsdichte von Vorteil. Starship Technologies gibt an, dass pro Minute höchstens drei Personen passieren sollten (Tänavsuu, 2017).

Maße	L 68 cm x B 57 cm x H 54 cm/125 cm (mit Fahne)
Gewicht ohne Zuladung	23 kg
Maximale Zuladung	10 kg
Ladevolumen	47 l
Maximale Geschwindigkeit	6 km/h
Batterietyp	Lithium-Polymer-Akku
Batteriekapazität	8000 mAh
Ladedauer	45 min
Maximale Reichweite	6 km

Tabelle 2.2: Spezifikation des Zustellroboters von Starship Technologies (Schweizerische Post AG, 2017)

2.2.3 Zustellkonzept

Dieser Abschnitt stellt den Ablauf der Paketzustellung mit Zustellrobotern vor. Dabei wird angenommen, dass die Roboter die allerletzte Meile eines zweistufigen Zustellprozesses absolvieren (vgl. Unterkapitel 2.1). Der beschriebene Ablauf ähnelt den Pilotversuchen von Starship Technologies in Hamburg (Brandt et al., 2018), London (BBC, 2017) oder Milton-Keynes (Großbritannien) (Osborne C., 2018). In allen drei Fällen wurden Zustellungen oder Abholungen innerhalb festgelegter Zeitfenster mit Zustellrobotern des Unternehmens durchgeführt.

Es wird angenommen, dass eine Person online einen Gebrauchsgegenstand bestellt und die Zustellung mit einem Roboter durchgeführt werden soll. Einen Tag vor der Zustellung kann der Empfänger ein verfügbares Zeitfenster für die Belieferung wählen. Dieses legt fest, in welchem Zeitraum der Roboter am nächsten Tag eintreffen soll. Am Zustelltag bringt ein Fahrzeug die konsolidierten Sendungen von einem Depot zu einem nahegelegenen Mikrodepot. Vor Beginn des Zeitfensters wird der Roboter im Mikrodepot von einem Mitarbeiter mit der Sendung beladen und startet seine Fahrt zum Empfänger. Während der Fahrt überwacht ein Operator den Roboter aus der Ferne. Der Operator hat Zugriff auf die Kameras des Roboters und kann über Lautsprecher mit Passanten kommunizieren. Außerdem ist ein manueller Eingriff über eine Fernsteuerung des Roboters durch den Operator möglich. In der Zukunft wird dieser Operator gegebenenfalls nicht mehr nötig sein und die Roboter fahren vollständig autonom. Der Empfänger wird über den Beginn der Fahrt des Roboters mit einer Anwendung auf seinem Smartphone informiert und erhält den voraussichtlichen Ankunftszeitpunkt. Während der gesamten Fahrt kann der Empfänger den Standort des Roboters abrufen. Nach dem Eintreffen des Roboters bestätigt der Empfänger die Ankunft und kann mittels Knopfdruck in der Anwendung den Roboter öffnen. Nach Entnahme der Sendung fährt der Roboter zurück zu einem Mikrodepot. Dort wird entweder der Akku geladen oder die nächste Zustellung vorbereitet. Erwähnenswert ist auch die Möglichkeit eines sogenannten *On-Demand-Services*. Hierbei wählen die Kunden vorab kein Zeitfenster, sondern der Wunsch einer zeitnahen Lieferung der Sendung wird durch einen Knopfdruck in der Anwendung bestätigt. Dieses Konzept bietet dem Kunden höhere Flexibilität. Allerdings müssen für einen reibungslosen Ablauf mehr Ressourcen in Form einer größeren Flotte vorgehalten werden, da die Kundenwünsche schwer zu planen sind. Abschließend werden Vor- und Nachteile des Konzepts gegenübergestellt und ein Fazit gezogen.

2.2.4 Vorteile und Nachteile

Die Autonomie der Zustellroboter bietet mehrere Vorteile. Personalkosten sowie die Gesamtkosten der Zustellung können reduziert werden (Heid et al., 2018, S. 8). Zudem können die Roboter rund um die Uhr arbeiten. Dadurch sind auch Zustellungen außerhalb üblicher Dienstzeiten möglich (Bogdanski, 2017, S. 81). Da die Roboter aufgrund der geringen Transportkapazität in Pendeltouren fahren, ist eine Zustellung im Zeitfenster oder on-demand möglich. Damit können die Bedürfnisse der Kunden besser berücksichtigt und die Kundenzufriedenheit gesteigert werden.

Ein weiterer Vorteil ist die geringere Belastung für die Umwelt. Zustellroboter sind elektrisch betrieben, wodurch sie den CO_2 -Ausstoß im Vergleich zu herkömmlichen Lieferfahrzeugen senken (Bogdanski, 2017, S. 81). Durch die Verlagerung der Lieferungen auf Gehwege wird der Verkehr auf den Straßen entlastet, was zu weniger Staus und Emissionen führt (Hossain, 2022, S. 6). Allerdings können Zustellroboter in dicht besiedelten Gebieten Gehwege überlasten, was Konflikte mit Fußgängern und Frustration verursachen kann (Gehrke et al., 2023). Bereits 2017 wurde in San Francisco der Einsatz von Zustellrobotern stark reglementiert. Dies erfolgte aufgrund von Beschwerden der Anwohner (Wong, 2017). Nachteilig ist auch, dass der Einsatz der Roboter regionalen Regularien und Gesetzen unterliegt, die sich, Stand 2025, noch stark unterscheiden. In den USA wird der Einsatz autonomer Roboter auf Ebene der einzelnen Bundesstaaten geregelt, wodurch unterschiedliche gesetzliche Vorgaben je nach Staat bestehen (Cregger et al., 2020, S. 31). In Deutschland ist eine Sondergenehmigung für den Einsatz erforderlich. Zudem musste bei den bisherigen Feldversuchen in Deutschland eine Person den Roboter auf dem Gehweg begleiten und ihn vor Ort überwachen (Brandt et al., 2018). Dies ersetzt allerdings nicht den Operator, der aus der Ferne die Roboter überwacht und die Steuerung übernehmen kann.

Die Nutzlast der Roboter ist relativ gering, weshalb sie für größere oder schwerere Lieferungen ungeeignet sind. Allerdings sind die Transportmittel für kleinere Lieferungen, die den Einsatz großer Fahrzeuge nicht rechtfertigen, ideal geeignet, wie beispielsweise Mahlzeiten (Bogdanski, 2017; Weinberg et al., 2023).

Zusammenfassend lässt sich schließen, dass für den Einsatz von Zustellrobotern der Transportweg zum Ziel möglichst kurz gehalten werden sollte, da diese nur mit Schrittgeschwindigkeit fahren. Zudem sollten die Gehwege für den Einsatz geeignet sein, sowohl in der Dimensionierung als auch in der Auslastung. Für den Transport eignen sich besonders Güter, bei deren Beförderung der Einsatz eines Fahrzeugs wirtschaftlich nicht sinnvoll ist. Dazu zählen insbesondere Essensbestellungen, Einkäufe und Medikamente. Im Vergleich zu anderen Zustellmethoden lassen sich Lieferungen on-demand mit Zustellrobotern deutlich leichter realisieren. Insgesamt bieten die Roboter einige Vorteile und ermöglichen durch ihre Autonomie einen kostengünstigen Betrieb. Dadurch können sie eine spezifische Marktnische besetzen und eine effiziente Alternative zu herkömmlichen Liefermethoden bieten.

2.3 Paketdrohnen

Drohnen sind unbemannte Transportmittel, die ohne Besatzung fliegen. Sie können jedoch vom Boden aus gesteuert werden. In dieser Arbeit werden ausschließlich Drohnen betrachtet, die für den zivilen Einsatz genutzt werden. Drohnen werden in der Regel elektrisch betrieben und sind entweder mit Flügeln oder einem oder mehreren Rotoren ausgestattet, die den benötigten Auftrieb erzeugen.

Im Folgenden werden Drohnen, die für den autonomen Einsatz konzipiert sind, betrachtet. Allerdings ist für deren Einsatz in den USA und Europa ein lizenzierter Pilot erforderlich, der die Drohne aus der Ferne steuern kann. Für Flüge mit einem erhöhten Risiko, wie beispielsweise bei der Auslieferung von Paketsendungen, wird eine spezielle Lizenz benötigt. Die vollständige Ausarbeitung dieser Regularien steht allerdings noch aus (Europäische Agentur für Flugsicherheit, 2022a). In den USA ist es beispielsweise möglich, dass ein Pilot mehrere Drohnen gleichzeitig überwacht (Federal Aviation Administration, 2022). Im Folgenden werden diese Drohnen auch als Paketdrohnen bezeichnet.



(a) Wingcopter W178 (DHL, 2018)



(b) Amazon MK27-2 (Amazon, 2022)

Abbildung 2.5: Paketdrohnen unterschiedlicher Bauarten

Paketdrohnen lassen sich in unterschiedliche Kategorien einordnen. Eine mögliche Unterscheidung basiert darauf, wie der Auftrieb erzeugt wird. Entweder wird dieser durch Flügel oder durch Rotorblätter erzeugt (Barmpounakis et al., 2016). Drohnen mit Flügeln lassen sich weiter unterteilen in Modelle mit starren oder beweglichen Flügeln. Auch Drohnen mit Rotoren lassen sich weiter unterteilen, etwa in Helikopter-Modelle und Multikopter mit mehreren Rotoren (DHL Customer Solutions & Innovation, 2022, S. 6f.). Zudem sind Kombinationen der Auftriebsarten möglich. In Abbildung 2.5 ist ein Modell mit starren Flügeln, aber beweglichen Rotoren und ein Multikopter mit sechs Rotoren abgebildet. Eine weitere Unterscheidungsmöglichkeit ist das Gewicht der Drohnen. Die Europäische Agentur für Flugsicherheit nimmt beispielsweise eine solche Kategorisierung vor. Hierbei erfolgt die Einteilung in vier Gewichtsklassen, von unter 250 Gramm bis 25 kg (Europäische Agentur für Flugsicherheit, 2022c). Allerdings fallen Paketdrohnen unter die Kategorie *Certified Category — Civil Drones*, da durch die Paketzustellung ein höheres Risiko vorliegt (Europäische Agentur für Flugsicherheit, 2022b). Im Folgenden werden zwei Paketdrohnen exemplarisch vorgestellt.

2.3.1 Bauarten

Das deutsche Unternehmen Wingcopter produziert eine Drohne mit starren Flügeln und kippbaren Rotoren. Die Drohne mit der Bezeichnung W178 hat eine Spannweite von 178 cm und gehört zur Kategorie der *electric vertical take-off and landing* (eVTOL)-Drohnen. Diese Kategorie umfasst elektrisch betriebene Drohnen, die senkrecht starten und landen können. Die Drohne ist mit vier kippbaren Rotoren ausgestattet und erreicht eine Höchstgeschwindigkeit von 151 km/h. Die maximale Reichweite beträgt 90 km und die maximale Zuladung liegt bei 5,5 kg. Bei maximaler Beladung verringert sich die Reichweite jedoch auf 35 km. Eine Übersicht der technischen Daten befindet sich in Tabelle 2.3. Abbildung 2.5(a) zeigt die Drohne.

Maße	L 1,46 m x B 1,78 m x H 0,48 m
Leergewicht	8,4 kg
Batteriegewicht	1,8 kg / Batterie (max. 4 Batterien)
Batterietyp	Lithium-Polymer-Akku
Batteriespannung & -kapazität	22,2 V & 16Ah
Maximales Abfluggewicht	18 kg
Maximale Zuladung	5,5 kg
Ladevolumen	16 l
min./ Ø /max. Geschwindigkeit	50 km/h / 86,4 km/h / 151 km/h
Reichweite	35 km (5,5 kg Zuladung) bis 90 km (leer)

Tabelle 2.3: Spezifikation der Paketdrohne W178 von Wingcopter (2021)

Ein Multikopter mit der Bezeichnung MK27-2 wird von Amazon hergestellt und zur Belieferung von Kunden eingesetzt. Auch diese Drohne zählt zur Kategorie der eVTOL-Drohnen. Mit sechs Rotoren erreicht die Drohne eine Geschwindigkeit von bis zu 80 km/h. Die Drohne hat eine maximale Reichweite von 12 km und eine Zuladung von 2,3 kg. Eine Übersicht der technischen Daten befindet sich in Tabelle 2.4, Abbildung 2.5(b) zeigt den Multikopter.

Maße	L 1,98 m x B 1,65 m x H 1,17 m
Gewicht mit Batterien	36 kg
Maximale Zuladung	2,3 kg
Geschwindigkeit	80 km/h
Reichweite	12 km

Tabelle 2.4: Spezifikation der Paketdrohne MK27-2 von Amazon (Federal Aviation Administration, 2022)

Generell zeichnen sich Drohnen mit Flügeln und kippbaren Rotoren durch eine große Reichweite, Langlebigkeit, geringeren Platzbedarf für Start und Landung sowie hohe Manövrierbarkeit aus. Nachteile dieser Bauweise sind die komplexe Technologie der Kippvorrichtungen und die vergleichsweise hohen Produktionskosten. Multikopter hingegen sind kostengünstig und einfach zu starten. Zusätzlich haben sie meist ein geringes Eigengewicht. Allerdings führt dies zu einem stärkeren Einfluss von Wetterfaktoren und einem niedrigeren maximalen Zuladungsgewicht (DHL Customer Solutions & Innovation, 2022). Die Einsatzbereiche der verschiedenen Bauarten werden im Folgenden diskutiert. Hierzu werden Anwendungsfelder sowie ein Logistikkonzept für die Paketauslieferung mit Drohnen vorgestellt.

2.3.2 Einsatzbereiche

Durch verschiedene Feldversuche und Untersuchungen haben sich diverse Einsatzbereiche für Drohnen ergeben. Generell eignen sich Drohnen für Einsatzorte, die schwer mit anderen Transportmitteln erreichbar sind, oder für Aufgaben, die mit anderen Transportmitteln große Kosten verursachen. Ein Beispiel hierfür ist der Einsatz in Offshore-Windparks, also bei Windkraftanlagen auf dem Meer. Wartungsarbeiten werden hier häufig mit bemannten Helikoptern durchgeführt. Durch den Einsatz von Drohnen lässt sich die Effizienz steigern und Kosten sparen. Außerdem können die Drohnen leichter gestartet werden und durch den Einsatz von unbemannten Transportmitteln werden keine Personen gefährdet.

Im Bereich der Agrar- und Forstwirtschaft können Drohnen Echtzeitdaten von Pflanzen erfassen und Luftaufnahmen erstellen. Mit den Daten kann beispielsweise der Einsatz von Düngemitteln, Herbiziden und Pestiziden besser geplant und durchgeführt werden. Darüber hinaus lassen sich Tiere beobachten und lokalisieren. Damit lassen sich Bewegungen von Herden erheben oder Bestände von Tierarten überprüfen. Ein weiteres bedeutendes Einsatzfeld von Drohnen liegt im Transport von Waren oder der Paketzustellung. Hier profitiert man insbesondere von der Geschwindigkeit der Drohnen, die z. B. den Transport von lebenswichtigen Gütern wie Medikamenten oder Blutkonserven in abgelegene Gebiete erleichtern. Aber auch der Transport von Paketsendungen zu Privathaushalten ist möglich. Dies wurde bereits von Unternehmen wie Amazon und DHL getestet (Amazon, 2022; DHL, 2018).

Weitere mögliche Einsatzgebiete sind (DHL Customer Solutions & Innovation, 2022):

- **Energie/Infrastruktur:** Überwachung großer Industrieanlagen, Stromleitungen und Pipelines.
- **Standort- und Layoutplanung:** Erstellung von Luftbildaufnahmen und Analyse von Baustellen.
- **Umweltschutz:** Schutz bedrohter Tierarten und Aufdeckung illegaler Aktivitäten wie Wilderei oder Brandrodung.
- **Notfallhilfe und Polizei:** Datenerfassung in Katastrophengebieten und risikoreiche Einsätze in der Strafverfolgung.
- **Film und Fotografie:** Erstellung von Luftbildaufnahmen und Produktion von Filmen.
- **Überwachung von Infrastruktur:** Überwachung von Logistikanlagen wie Lagerhäusern und Ölfeldern.
- **Intralogistik:** Unterstützung des Transports innerhalb großer Produktionsstätten und Lagerhäuser.

Damit ist die Vorstellung der Einsatzbereiche abgeschlossen. Im Weiteren wird der Einsatz von Drohnen in der Logistik genauer betrachtet und der Prozess der Paketzustellung mit Drohnen vorgestellt.

Eine umfassende Übersicht aktueller Drohnenmodelle für logistische Anwendungen und Feldversuche liefert Moshref-Javadi und Winkenbach (2021). Demnach gab es 27 Unternehmen im Jahr 2021, die Drohnen für logistische Zwecke eingesetzt oder getestet haben. Diese Zwecke sind der Transport für den Einzelhandel, Postdienste und Postzustellung, Essenslieferungen und Flüge mit einem medizinischen Hintergrund.

Im Folgenden wird der Einsatz von Paketdrohnen zur Auslieferung von Sendungen genauer betrachtet. Verschiedene Konzepte wurden entwickelt, um Paketdrohnen in Zustellprozesse zu integrieren. Grundsätzlich erfolgt die Beladung der Paketdrohne am Startpunkt durch einen Mitarbeiter und bei Bedarf wird der Akku gewechselt. Daraufhin steigt die Paketdrohne auf ihre vorgesehene Reiseflughöhe und setzt den Flug in Richtung Zielort fort. Bei der Abgabe der Sendung muss sichergestellt sein, dass keine Personen durch die Paketdrohne gefährdet werden und die Paketdrohne nicht beschädigt wird. Hierfür wurden unterschiedliche Konzepte entwickelt (Otto et al., 2018, S.5). Die Drohne vom Unternehmen Wingcopter landet und die Sendung wird manuell vom Empfänger entnommen. Diese Methode setzt voraus, dass ausreichend Platz für die Landung am Zielort vorhanden ist und der Empfänger mit der Handhabung der Drohne vertraut ist. Alternativ kann die Drohne die Sendung an einem Seil ablassen. Eine vollständig kontaktlose Übergabe wurde von Amazon entwickelt. Auch für diese Methode wird ein vordefinierter Abgabepunkt benötigt. Die Drohne MK27-2 verfügt über ein Fach, aus dem das Paket fällt. Die Fallhöhe beträgt dabei ca. 3,6 m (CNBC Television, 2022). Dies limitiert jedoch die Anzahl an Produkten, die mit der Drohne ausgeliefert werden können. Beide Übergabemethoden erfordern die Definition eines geeigneten Ablage- oder Landeplatzes (BBC, 2022). Nach erfolgreicher Abgabe der Sendung kehrt die Drohne zum Startort zurück.

Für den Einsatz auf der letzten Meile ist ein Konzept ähnlich zu den Zustellrobotern möglich. Dabei wird die letzte Meile aufgeteilt und die Drohnen starten von einem Mikrodepot, das in der Nähe der Empfänger liegt. Daneben ist es auch möglich, nicht nur die allerletzte Meile von Drohnen durchführen zu lassen, sondern die komplette letzte Meile. In diesem Fall starten die Drohnen von einem Depot, welches weiter von den Empfängern entfernt liegt. Ein weiteres Konzept besteht darin, Drohnen von einem Zustellfahrzeug aus starten zu lassen (Congress, 2017). Hierfür ist das Zustellfahrzeug mit einer Startvorrichtung auf dem Dach ausgestattet. Das Fahrzeug kann entweder geparkt bleiben, während die Drohne die Zustellungen übernimmt, oder parallel zur Drohne weitere Sendungen zustellen. Eine Übersicht über die entwickelten Varianten des Problems wird von Moshref-Javadi und Winkenbach (2021) gegeben. Alle Varianten haben gemein, dass das Zustellfahrzeug als Lager für die Pakete und Austauschpunkt für die Akkus dient.

2.3.3 Vorteile und Nachteile

Der Einsatz elektrischer Drohnen kann die Umweltbelastung im Vergleich zu traditionellen Liefermethoden reduzieren (Figliozi, 2017). Da Drohnen autonom betrieben werden, besteht das Potential, dass Unternehmen die Personalkosten reduzieren können. Allerdings erfordern aktuelle Regularien in der Regel noch eine Fernsteuerung durch einen Piloten. Ein wesentlicher Vorteil von Drohnen ergibt sich aus ihrem Einsatz im Luftraum. Dadurch sind Drohnen auf keine Straßeninfrastruktur angewiesen. Insbesondere in Entwicklungsländern werden Drohnen bereits für die Versorgung ländlicher Gebiete genutzt, etwa für medizinische Lieferungen und Notfalldienste in schwer zugänglichen Regionen (Moshref-Javadi und Winkenbach, 2021, S.1). Generell eignen sich Drohnen für den Transport zu entlegenen oder schwer erreichbaren Orten. Außerdem eignen sie sich für Einsatzbereiche, die risikobehaftet oder sehr kostspielig sind. Hierzu zählen beispielsweise Aufnahmen von Waldbränden oder der Einsatz bei Protesten (DHL Customer Solutions & Innovation, 2022). Durch die hohe Geschwindigkeit und den direkten Flug zum Ziel lassen sich kurze Transportzeiten realisieren und die Reaktionsfähigkeit der Logistiksysteme erhöhen (Thiels et al., 2015).

Zu den Nachteilen der Drohnen zählt die Lautstärke sowie das Risiko technischer Ausfälle, bei denen herabfallende Teile Menschen gefährden können (Clausen et al., 2016, S. 40). Daneben besteht auch die Gefahr, dass Personen die Kontrolle über Drohnen übernehmen könnten und diese für illegale Aktivitäten einsetzen oder bewusst zum Absturz bringen (Shakhatreh et al., 2019, S. 48593). Weiterhin gibt es Bedenken hinsichtlich des Datenschutzes und der Privatsphäre der Bevölkerung (Lee et al., 2022).

Die Paketdrohnen verfügen über eine begrenzte Transportkapazität, die üblicherweise 3 kg nicht überschreitet (Agatz et al., 2018). In den meisten Feldversuchen transportieren Drohnen lediglich ein einzelnes Paket, sodass keine Bündelungseffekte erzielt werden können. Auch die Reichweite ist ein limitierender Faktor und hängt stark von Bauart, Flughöhe, Beladungsgewicht und Wetterbedingungen ab (Otto et al., 2018).

Zusammenfassend lässt sich folgern, dass für die Paketdrohnen eine Vielzahl an Einsatzmöglichkeiten existiert und sie für schwer erreichbare Orte eine kostengünstige Alternative zur konventionellen Zustellung darstellen. Die Vorteile der Drohnen können besonders ausgespielt werden, wenn der Transport oder Prozess in der Luft durchgeführt werden muss, da die Alternativen hierfür Flugzeuge oder Hubschrauber sind. Beim Transport bieten die Paketdrohnen den Vorteil, dass sie einfach gestartet werden können und durch die hohe Geschwindigkeit eine kurze Transportzeit erreichen. Ein großes Hindernis für eine flächendeckende Einführung sind jedoch noch fehlende gesetzliche Regularien, die die Sicherheit und Akzeptanz in der Öffentlichkeit gewährleisten. Damit ist die Vorstellung der innovativen Zustellkonzepte abgeschlossen. Im nächsten Unterkapitel werden die Grundlagen für die mathematische Optimierung vorgestellt.

Kapitel 3

Grundlagen der mathematischen Optimierung

In dieser Arbeit werden Optimierungsmodelle und deren Lösungsverfahren vorgestellt. Die Grundlagen der mathematischen Optimierung, die für das Verständnis und die Durchdringung der Modelle nötig sind, werden in diesem Kapitel zusammengefasst erläutert. Neben den Grundlagen der mathematischen Optimierung werden auch die Grundlagen der Graphentheorie und der dynamischen Programmierung erläutert. Diese Grundlagen sind für das Verständnis des Branch-and-Price-Algorithmus erforderlich. Für eine vertiefte Einführung wird auf folgende Lehrbücher verwiesen (Korte und Vygen, 2018; Diestel, 2016; Tittmann, 2021).

Das Kapitel ist wie folgt gegliedert. In Unterkapitel 3.1 werden die Grundlagen mathematischer Optimierungsmodelle, unterteilt in lineare und ganzzahlige Modelle, vorgestellt. Für die ganzzahlige Optimierung wird das Branch-and-Bound-Verfahren beschrieben. Im Abschnitt 3.2 werden die Grundlagen des Branch-and-Price-Verfahrens vorgestellt. Anschließend wird in Unterkapitel 3.3 eine Einführung in die Graphentheorie gegeben. Das Kapitel wird mit der Vorstellung der dynamischen Programmierung in Abschnitt 3.4 abgeschlossen.

3.1 Mathematische Optimierungsmodelle

In den Kapiteln 4 und 5 werden Problemstellungen für die innovativen Zustellkonzepte beschrieben, die mittels mathematischer Optimierung gelöst werden sollen. Die Problemstellungen werden dabei durch Optimierungsmodelle beschrieben und mit geeigneten Verfahren gelöst. Ein Optimierungsmodell setzt sich aus Entscheidungsvariablen, einer Zielfunktion und Nebenbedingungen zusammen. Durch die Zuweisung von Werten an die Entscheidungsvariablen können verschiedene Lösungen für die Problemstellung erzeugt werden. Beispiele für Entscheidungsvariablen sind Entscheidungen darüber, ob ein Fahrzeug eine bestimmte Reihenfolge von Kunden besuchen soll oder wie viele Fahrzeuge für die Lösung eines Problems benötigt werden. Zur Bewertung der Belegung der Entscheidungsvariablen wird eine Zielfunktion festgelegt. Diese weist jeder Wertekombination

einen Zielfunktionswert zu, wodurch die Bewertung und der Vergleich der gefundenen Lösungen ermöglicht werden. Nebenbedingungen sind Funktionen, die die Entscheidungsvariablen durch Gleichungen oder Ungleichungen einschränken und somit Restriktionen der Problemstellung in dem Optimierungsmodell widerspiegeln.

Grundsätzlich können Optimierungsmodelle in unterschiedliche Klassen unterteilt werden. Dabei wird häufig zwischen linearen und nichtlinearen sowie kontinuierlichen und ganzzahligen Modellen unterschieden. Lineare Modelle zeichnen sich dadurch aus, dass sowohl die Zielfunktion als auch die Nebenbedingungen lineare Funktionen der Entscheidungsvariablen sind. Ein nichtlineares Optimierungsmodell besteht aus Funktionen, die in den Entscheidungsvariablen nicht linear sind. Eine nichtlineare Nebenbedingung ist beispielsweise die Begrenzung des frühesten Ankunftszeitpunkts bei einem Kunden einer Tour, wenn die Fahrzeit von einer variablen Geschwindigkeit abhängt. Der Wertebereich der Entscheidungsvariablen bestimmt, ob ein kontinuierliches oder ganzzahliges Modell vorliegt. Wenn die Variablen auf den reellen Zahlen $\mathbb{R}$ definiert sind, spricht man von einem kontinuierlichen Modell. Wenn die Variablen hingegen aus den ganzen Zahlen $\mathbb{Z}$ stammen, handelt es sich um ein ganzzahliges Modell. Im nächsten Abschnitt werden die Grundlagen linearer Optimierungsmodelle vorgestellt.

3.1.1 Lineare Optimierung

In diesem Abschnitt wird die lineare Optimierung basierend auf dem Werk von (Korte und Vygen, 2018) beschrieben. Dazu wird im ersten Schritt die Definition eines allgemeinen linearen Optimierungsmodells gegeben.

Definition 1. *Ein lineares Optimierungsproblem ist gegeben durch eine Instanz und eine Aufgabe zu der Instanz.*

Instanz: Matrix $A \in \mathbb{R}^{m \times n}$ und Spaltenvektoren $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$.

Aufgabe: Bestimme, welche der folgenden drei Möglichkeiten auf das lineare Optimierungsproblem zutrifft. Für ein lineares Optimierungsproblem muss gelten, dass es entweder

- (i) eine optimale Lösung besitzt, das heißt einen Spaltenvektor $\mathbf{x} \in \mathbb{R}^n$, der $A\mathbf{x} \leq \mathbf{b}$ erfüllt und bei dem $\mathbf{c}^T \mathbf{x}$ maximiert wird,
- (ii) unzulässig ist, das heißt, die Menge $\{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b}\}$ ist leer,
- (iii) unbeschränkt ist, das heißt für alle $\alpha \in \mathbb{R}$ gibt es ein $\mathbf{x} \in \mathbb{R}^n$ mit $A\mathbf{x} \leq \mathbf{b}$ und $\mathbf{c}^T \mathbf{x} > \alpha$.

Eine Instanz dieses Problems wird als *lineares Programm* (LP) bezeichnet. Ein LP wird in dieser Arbeit in der folgenden Form dargestellt:

$$\begin{array}{ll} \text{maximiere} & \mathbf{c}^T \mathbf{x} \end{array} \quad (3.1)$$

$$\begin{array}{ll} \text{unter der Nebenbedingung} & A\mathbf{x} \leq \mathbf{b}. \end{array} \quad (3.2)$$

Die Multiplikation der Vektoren in (3.1) wird als *Zielfunktion* bezeichnet, während die Bedingungen in (3.2) als *Nebenbedingung* bezeichnet werden. Die Einträge des Vektors $\mathbf{x}$ werden als *Variablen* bezeichnet. Falls zusätzlich $\mathbf{x} \geq 0$ gefordert ist, wird $\mathbf{x}$ als *nichtnegativ* bezeichnet. Das LP (3.1)-(3.2) wird auch als *primales LP* bezeichnet. Neben dem primalen LP existiert auch ein *duales LP*, das wie folgt definiert ist.

Definition 2. Sei (3.1)-(3.2) ein primales LP. Das zugehörige duale LP wird durch das folgende lineare Programm definiert:

$$\text{minimiere} \quad \mathbf{y}^T \mathbf{b} \quad (3.3)$$

$$\text{u. d. N.} \quad \mathbf{y}A = \mathbf{c}, \quad (3.4)$$

$$\mathbf{y} \geq 0. \quad (3.5)$$

Die Zielfunktion des dualen LP ist durch (3.3) gegeben. Es ist zu beachten, dass die Zielfunktion, im Gegensatz zum primalen LP, minimiert werden muss. Die Nebenbedingungen sind durch die Gleichungen (3.4) definiert, während die Nichtnegativität der Variablen durch (3.5) gewährleistet wird. Nach Einführung des primalen LP und des zugehörigen dualen LP kann nun der Dualitätssatz formuliert werden.

Satz 1. (Gale et al., 1951) Falls ein primales LP und das zugehörige duale LP optimale Lösungen $\mathbf{x}^*$ bzw. $\mathbf{y}^*$ besitzen, gilt $\mathbf{c}^T \mathbf{x}^* = \mathbf{y}^{*T} \mathbf{b}$.

Der Beweis dieses Satzes findet sich in (Korte und Vygen, 2018, S.70). Eine weitere Beziehung zwischen dem primalen und dualen LP wird durch den folgenden Satz beschrieben. Auch der Beweis dieses Satzes ist in der Veröffentlichung von Korte und Vygen zu finden.

Satz 2. Ist ein LP unbeschränkt, so ist das zugehörige duale LP unzulässig. Besitzt ein LP eine optimale Lösung, so hat auch das entsprechende duale LP eine optimale Lösung.

Ein LP wird als *unbeschränkt* bezeichnet, wenn die Nebenbedingungen den optimalen Wert der Zielfunktion nicht einschränken (siehe Definition 1). Ein LP wird als *unzulässig* bezeichnet, wenn es keine Lösung gibt, die alle Nebenbedingungen gleichzeitig erfüllen kann (siehe Definition 1). Damit ist die Einführung in die lineare Optimierung abgeschlossen.

3.1.2 Ganzzahlige Optimierung

Die Grundlagen der linearen Optimierung lassen sich durch die Forderung der Ganzzahligkeit der Variablen erweitern. Die Beschreibung der ganzzahligen Optimierung orientiert sich an der Veröffentlichung von Korte und Vygen (2018).

Definition 3. Ein ganzzahliges Optimierungsproblem ist gegeben durch eine Instanz und eine Aufgabe zu der Instanz.

Instanz: Matrix $A \in \mathbb{Z}^{m \times n}$ und Spaltenvektoren $\mathbf{b} \in \mathbb{Z}^m, \mathbf{c} \in \mathbb{Z}^n$.

Aufgabe: Entscheide, welche der drei Möglichkeiten auf das ganzzahlige Optimierungsproblem zutrifft. Für ein ganzzahliges Optimierungsproblem gilt, dass es entweder

- (i) *eine optimale Lösung enthält, das heißt ein Spaltenvektor $\mathbf{x} \in \mathbb{Z}^n$ existiert, der $A\mathbf{x} \leq \mathbf{b}$ erfüllt und $\mathbf{c}^T \mathbf{x}$ maximiert,*
- (ii) *unzulässig ist, das heißt die Menge $\{\mathbf{x} \in \mathbb{Z}^n : A\mathbf{x} \leq \mathbf{b}\}$ ist leer,*
- (iii) *unbeschränkt ist, das heißt für alle $\alpha \in \mathbb{Z}$ gibt es ein $\mathbf{x} \in \mathbb{Z}^n$ mit $A\mathbf{x} \leq \mathbf{b}$ und $\mathbf{c}^T \mathbf{x} > \alpha$.*

Dabei wird eine Instanz eines ganzzahligen Optimierungsproblems als *ganzzahliges lineares Programm* (IP) bezeichnet. Ein IP mit Variablen, die keine Ganzzahligkeitsforderung haben, wird als *gemischt-ganzzahliges lineares Programm* (MIP) bezeichnet. Im nächsten Abschnitt wird ein Verfahren zum Lösen von IPs und MIPs vorgestellt.

Ganzzahlige Optimierungsprobleme können mithilfe des *Branch-and-Bound-Verfahrens* gelöst werden. Im Folgenden werden die zwei zentralen Schritte des Branch-and-Bound-Verfahrens vorgestellt. Die Beschreibung basiert auf Domschke et al. (2015). Für eine weiterführende Erläuterung wird diese Veröffentlichung empfohlen. Die zentralen Schritte lauten wie folgt:

- **Branching:** Ein zu lösendes Problem P_0 wird so in k Teilprobleme $P_1, \dots, P_k$ *verzweigt* (to branch) oder zerlegt, dass

$$X(P_0) = \bigcup_{i=1}^k X(P_i) \text{ und möglichst } X(P_i) \cap X(P_j) = \emptyset \text{ für alle } i \neq j \text{ gilt.}$$

Dabei bezeichnet $X(P_i)$ die Menge der zulässigen Lösungen von Teilproblem P_i . Die Probleme $P_1, \dots, P_k$ können weiter verzweigt werden.

- **Bounding:** Die Begrenzung des Verzweigungsprozesses wird als *bounding* bezeichnet. Dabei wird anhand von Schranken überprüft, ob ein Teilproblem weiter verzweigt werden muss. Es wird generell zwischen einer *unteren Schranke* und einer *oberen Schranke* unterschieden. Die untere Schranke stellt den besten Zielfunktionswert einer zulässigen Lösung dar. Der Wert der Schranke kann im Verlauf des Verfahrens steigen und beträgt zu Beginn $-\infty$. Die obere Schranke stellt eine Begrenzung des Zielfunktionswertes eines *relaxierten* Teilproblems dar. Ein relaxiertes Teilproblem ist ein Teilproblem, das durch das Weglassen von Nebenbedingungen vereinfacht wurde. Beispielsweise wird bei der Relaxation die Ganzzahligkeitsforderung einer Lösung aufgehoben. Ein Teilproblem wird nicht weiter betrachtet, wenn die obere Schranke kleiner als die untere Schranke ist oder wenn die Lösung des relaxierten Problems zugleich eine Lösung des ursprünglichen Teilproblems darstellt.

Damit ist die Vorstellung des Branch-and-Bound-Verfahrens abgeschlossen und es folgt ein Verfahren, welches eben dieses erweitert.

3.2 Branch-and-Price-Verfahren

Das *Branch-and-Price-Verfahren* spielt eine zentrale Rolle in dieser Arbeit, da es zur Lösung der in den Kapiteln 4 und 5 beschriebenen Problemstellungen verwendet wird. Deshalb wird das Verfahren ausführlich vorgestellt. Es handelt sich dabei um eine Erweiterung des Branch-and-Bound-Verfahrens. Der wesentliche Unterschied zwischen den beiden Verfahren besteht darin, dass im Branch-and-Price-Verfahren das Optimierungsproblem nur mit einer Teilmenge der Variablen initialisiert wird. Im Verlauf des Verfahrens werden bei Bedarf weitere Variablen hinzugefügt, um die Optimalität der Lösung sicherzustellen. Die Beschreibung des Verfahrens folgt der Einführung von Feillet (2010) und wird durch das IP (3.6)-(3.9) veranschaulicht. Dieses IP dient als Beispiel, um das Verständnis des Verfahrens zu erleichtern.

Das IP (3.6)-(3.9) stellt beispielhaft eine Formulierung des Vehicle Routing Problems (VRP) dar. Ein VRP umfasst eine Menge von n Kunden, m Fahrzeugen sowie ein Depot. Das Ziel besteht darin, die Kunden in Touren vom Depot aus zu beliefern, sodass die Kosten minimiert werden. Das Depot ist sowohl Quelle als auch Senke jeder Tour. Eine mögliche Kostenfunktion ist die Summe der gefahrenen Distanzen.

Das Optimierungsproblem lässt sich wie folgt definieren. Sei Ω die Menge der möglichen Touren und $x_s, s \in \Omega$, eine binäre Variable, die entscheidet, ob eine Tour gefahren wird. Die Kosten der Tour $s, s \in \Omega$, seien c_s . Der Koeffizient e_{is} ist gleich eins, falls der Kunde i , für $i = 1, \dots, n$, in Tour s vertreten ist. Damit lässt sich das VRP wie folgt darstellen:

$$\text{minimiere} \quad \sum_{s \in \Omega} c_s x_s \quad (3.6)$$

$$\text{u.d.N.} \quad \sum_{s \in \Omega} e_{is} x_s \geq 1 \quad \text{für } i = 1, \dots, n, \quad (3.7)$$

$$\sum_{s \in \Omega} x_s \leq m, \quad (3.8)$$

$$x_s \in \{0, 1\} \quad \forall s \in \Omega. \quad (3.9)$$

Die Zielfunktion (3.6) minimiert die Gesamtkosten der Entscheidungsvariablen. Die Nebenbedingungen (3.7) stellen sicher, dass jeder Kunde in mindestens einer Tour enthalten ist. Die Nebenbedingung (3.8) garantiert, dass maximal m Touren gefahren werden. Mit der letzten Bedingung (3.9) wird die Binarität der Variablen gefordert.

Ein direkter Ansatz zur Lösung des IP (3.6)-(3.9) ist mit dem Branch-and-Bound-Verfahren möglich. Das IP wäre mit diesem Verfahren jedoch kaum lösbar, da die Menge Ω exponentiell mit der Anzahl der Kunden wächst. Eine mögliche Lösung bietet das Branch-and-Price-Verfahren, da hierbei nur eine Teilmenge der Variablen berücksichtigt wird. Sei hierfür $\Omega' \subseteq \Omega$ die Teilmenge der Variablen. Zur Relaxation des IP (3.6)-(3.9) wird die Bedingung (3.9) wie folgt ersetzt

$$x_s \geq 0, \quad s \in \Omega.$$

Das relaxierte Programm wird als Masterproblem (MP) bezeichnet. Sei $MP(\Omega)$ das Masterproblem auf Basis von Menge Ω und sei $MP(\Omega')$ das Problem auf der Teilmenge Ω' der Variablen. Das Problem $MP(\Omega')$ wird auch als eingeschränktes Masterproblem (EMP) bezeichnet. Das $MP(\Omega)$ entspricht dem relaxierten, zu lösenden Problem P_0 des Branch-and-Bound-Verfahrens. Da das Branch-and-Price-Verfahren das EMP betrachtet, wird im Folgenden erläutert, wie durch die Hinzunahme von Variablen zum EMP eine optimale Lösung des $MP(\Omega)$ gefunden werden kann. Dafür ist die Einführung des dualen Problems $DP(\Omega')$ zum $MP(\Omega')$ erforderlich.

Seien y_i , für $i = 1, \dots, n$, die dualen Variablen, die den Nebenbedingungen (3.7) zugeordnet sind. Die duale Variable $\tilde{y}$ ist der Nebenbedingung (3.8) zugeordnet.

$$\text{maximiere} \quad \sum_{i=1}^n y_i - m\tilde{y} \quad (3.10)$$

$$\text{u.d.N.} \quad \sum_{i=1}^n e_{is}y_i - \tilde{y} \leq c_s \quad \forall s \in \Omega', \quad (3.11)$$

$$y_i, \tilde{y} \geq 0 \quad i = 1, \dots, n. \quad (3.12)$$

Die Zielfunktion (3.10) maximiert die Summe der dualen Variablen, abzüglich des Produkts der dualen Variable $\tilde{y}$ und der Anzahl der Fahrzeuge m . Die Nebenbedingungen (3.11) verlangen, dass für jede Tour $s \in \Omega$ die Summe der dualen Variablen y_i , die den in s enthaltenen Kunden zugeordnet sind, abzüglich des Wertes von $\tilde{y}$, kleiner oder gleich den Kosten der Tour c_s ist. Die Nichtnegativität von y_i , für $i = 1, \dots, n$, und $\tilde{y}$ wird durch die Nebenbedingungen (3.12) sichergestellt. Mit dem dualen Programm lässt sich folgender Satz formulieren.

Satz 3. *Sei Ω' eine Teilmenge von Ω und $\mathbf{y}^*$ eine optimale Lösung des $DP(\Omega')$. Ist $\mathbf{y}^*$ eine zulässige Lösung des $DP(\Omega)$, so ist sie auch eine optimale Lösung für das $DP(\Omega)$.*

Beweis. Der zulässige Bereich von $DP(\Omega)$ ist eine Teilmenge des zulässigen Bereichs von $DP(\Omega')$, und die Zielfunktion ist in beiden Modellen identisch. $\square$

Der Satz spielt eine zentrale Rolle im Branch-and-Price-Verfahren. Da somit gilt, dass eine optimale Lösung des $MP(\Omega')$ auch eine optimale Lösung des $MP(\Omega)$ ist, sofern die optimale Lösung $\mathbf{y}^*$ des $DP(\Omega')$ zulässig für das $DP(\Omega)$ ist. Der Algorithmus überprüft die Zulässigkeit für jede optimale Lösung des $DP(\Omega')$. Falls mindestens eine Nebenbedingung des $DP(\Omega)$ durch $\mathbf{y}^*$ verletzt wird, muss die entsprechende Variable zu Ω' hinzugefügt und das Problem erneut gelöst werden.

Es ist zu beachten, dass in einer Iteration auch mehrere Variablen zu Ω' hinzugefügt werden können. Zudem muss sich der Zielfunktionswert des EMP durch die Hinzunahme einer Variablen nicht zwingend verbessern. Trotzdem terminiert das Verfahren nach einer endlichen Anzahl an Schritten, sofern Ω endlich ist. Die Überprüfung der Zulässigkeit des $DP(\Omega)$ erfolgt im sogenannten *Pricing-Problem*. Im Folgenden wird das Pricing-Problem genauer erläutert.

Sei $\mathbf{y}^* = (y_1^*, y_2^*, \dots, y_n^*, \tilde{y}^*)$ eine optimale Lösung des $DP(\Omega')$. Im Pricing-Problem wird anschlie-

ßend geprüft, ob die entsprechenden Nebenbedingungen (3.11) für jede Tour $s \in \Omega$ erfüllt sind. Hierfür werden die reduzierten Kosten jeder Tour $s \in \Omega$ berechnet. Diese sind definiert als

$$\bar{c}_s = c_s - \sum_{i=1}^n e_{is} y_i^* + \tilde{y}^*.$$

Wenn die reduzierten Kosten kleiner als 0 sind, ist die Nebenbedingung verletzt, und die der Tour s entsprechende Variable wird zu Ω' hinzugefügt. Die Generierung und das Hinzufügen neuer Variablen wird als *Column Generation* (Spaltengenerierung) bezeichnet. Eine optimale Lösung des Pricing-Problems ist somit gegeben durch $\bar{c}^* = \min_{s \in \Omega} \bar{c}_s$. Diese Schritte werden wiederholt, bis keine Variable mit negativen reduzierten Kosten mehr gefunden wird. In diesem Fall sind die Zielfunktionswerte der optimalen Lösung des $MP(\Omega')$ und des $MP(\Omega)$ gleich.

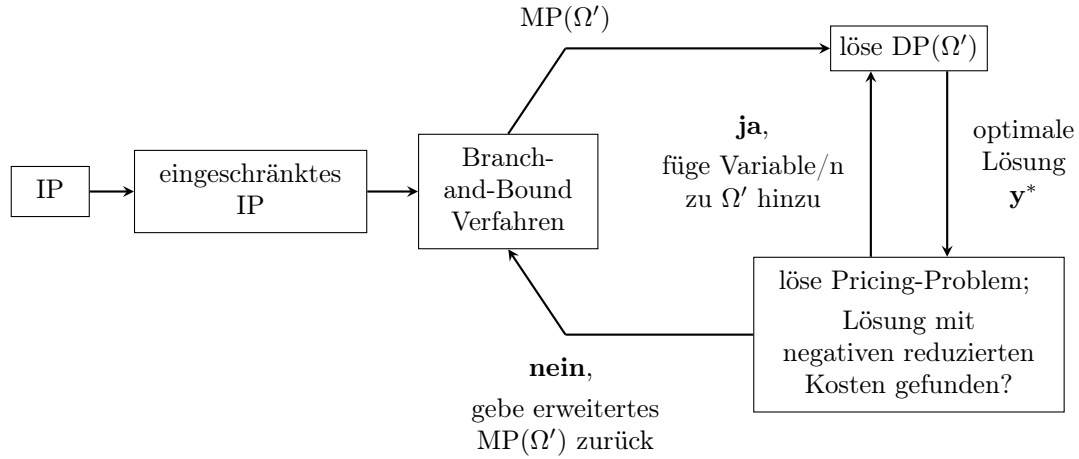


Abbildung 3.1: Schematischer Aufbau des Branch-and-Price-Verfahrens

Eine Übersicht des Branch-and-Price-Verfahrens ist in Abbildung 3.1 dargestellt. Die Eingabe besteht aus einem ganzzahligen linearen Programm. Im ersten Schritt wird die Menge der Variablen eingeschränkt, wodurch das reduzierte IP definiert wird. Das reduzierte IP wird mithilfe des Branch-and-Bound-Verfahrens gelöst. Es ist zu beachten, dass für jeden Knoten mithilfe des Column-Generation-Verfahrens geprüft wird, ob weitere Variablen hinzugefügt werden müssen. Ist dies der Fall, wird das erweiterte Teilproblem wiederholt gelöst, bis keine zusätzlichen Variablen mehr gefunden werden. Damit wird sichergestellt, dass die Teilprobleme optimal gelöst werden und somit auch eine optimale Lösung für das ursprüngliche Problem gefunden wird.

Da die Berechnung der Kosten aller Variablen zu aufwendig wäre, wird das Pricing-Problem auf ein *Kürzeste-Wege-Problem mit Beschränkung der Ressourcen* reduziert. Zur Einführung des Verfahrens werden die grundlegenden Begriffe der Graphentheorie erläutert.

3.3 Graphentheorie

Die Einführung von Graphen ist für diese Arbeit zentral, da die Planungsprobleme auf Graphen modelliert werden. Die Darstellung der Graphentheorie orientiert sich an den Grundlagenwerken von Tittmann (2021) und Diestel (2016). Ein *ungerichteter Graph* $G = (V, E)$ besteht aus einer Menge von Knoten V und einer Menge von Kanten E , wobei jede Kante zwei Knoten aus V verbindet. Ein *gerichteter Graph* $G = (V, A)$ besteht aus einer Knotenmenge V und einer Kantenmenge A , wobei jeder Kante (i, j) ein geordnetes Paar von Knoten (u_i, u_j) aus V zugeordnet ist. Ein Graph, der nur eine Kante zwischen je zwei Knoten hat und keine Kanten mit gleichen Endknoten enthält, wird als *schlicht* bezeichnet. In dieser Arbeit werden ausschließlich schlichte Graphen betrachtet.

Eine *Kantenfolge* ist eine Abfolge von Knoten und Kanten $u_1, e_1, u_2, e_2, u_3, \dots, u_{k-1}, e_{k-1}, u_k$ aus G . Dabei gilt, dass eine Kante e_i , für $i = 1, \dots, k-1$, die Knoten u_i, u_{i+1} verbindet. Ein *Weg* W ist eine Kantenfolge, in der kein Knoten mehrfach auftritt. Zur vereinfachten Notation kann ein Weg für schlichte Graphen auch nur durch die Knoten beschrieben werden. Damit gilt für einen Weg $W = (u_1, u_2, \dots, u_k)$, dass $u_i \in V$, für $i = 1, 2, \dots, k$. Ein Graph $G = (V, E)$ wird als *zusammenhängend* bezeichnet, falls zwischen je zwei Knoten aus V ein Weg existiert. Ein Graph $G = (V, E)$ wird als *gewichtet* bezeichnet, falls jeder Kante von G ein Gewicht $c : E \rightarrow \mathbb{R}$ zugeordnet ist. Die Kosten eines Weges W werden als c_W bezeichnet und berechnen sich als Summe der Gewichte der Kanten zwischen den Knoten.

Kürzeste Wege

Ein zentrales Problem der Graphentheorie ist die Suche nach dem kürzesten Weg zwischen zwei Knoten in einem ungerichteten oder gerichteten Graphen.

Definition 4. Die Suche nach einem kürzesten Weg in einem Graphen zwischen einer Quelle und einer Senke kann durch eine Instanz und eine Aufgabe definiert werden.

Instanz: Ein gewichteter Graph $G = (V, E)$ (gerichtet oder ungerichtet), Gewichte $c : E \rightarrow \mathbb{R}$ sowie zwei Knoten $u_s, u_t \in V$.

Aufgabe: Finde einen kürzesten Weg W von u_s nach u_t , also einen Weg mit minimalen Kosten c_W , oder entscheide, dass u_t von u_s aus nicht erreichbar ist.

Wenn die Kostenfunktion ausschließlich positive Gewichte den Kanten zuordnet, kann der kürzeste Weg mit dem Dijkstra-Algorithmus (siehe (Korte und Vygen, 2018, S.169)) berechnet werden. Der Algorithmus liefert entweder den kürzesten Weg und dessen Kosten oder zeigt an, dass keine zulässige Lösung existiert.

Das Kürzeste-Wege-Problem kann durch Beschränkungen der Knoten bzw. Kanten und der Einführung von Ressourcen erweitert werden. Das daraus resultierende Problem wird als *Kürzeste-Wege-Problem mit Beschränkung der Ressourcen* bezeichnet. Wie bereits erwähnt, kann das Pricing-Problem für die Optimierungsprobleme dieser Arbeit auf ein Kürzeste-Wege-Problem

mit Beschränkung der Ressourcen reduziert werden. Die mathematische Formulierung des Problems orientiert sich an (Feillet et al., 2004).

Sei $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ ein gewichteter Graph, wobei $\mathcal{V}$ die Knotenmenge und $\mathcal{A}$ die Kantenmenge beschreibt. Die Knoten u_s und u_t stellen die Quelle und Senke des Weges dar. Eine Kante $(i, j) \in \mathcal{A}$ besitzt Kosten in Höhe c_{ij} . Sei d_{ij}^l der Verbrauch der Ressource l bei der Benutzung der Kante $(i, j) \in \mathcal{A}$. Es wird angenommen, dass alle Werte d_{ij}^l die Dreiecksungleichung erfüllen. Zu den Beispielen für Ressourcen zählen Zeit, Kapazität oder der Ladestand eines Akkus. Jeder Knoten $u_i \in \mathcal{V}$ besitzt zwei nicht-negative Werte a_i^l, b_i^l für jede Ressource l . Ein Besuch des Knotens u_i ist für jede Ressource nur im Intervall $[a_i^l, b_i^l]$ möglich. Ein Beispiel für diese Intervalle sind Zeitfenster. Das Ziel ist es, einen kreisfreien Weg von u_s zu u_t zu finden, der alle Knotenbeschränkungen erfüllt und die geringsten Gesamtkosten aufweist. Dieses Problem kann als MIP formuliert werden (Feillet et al., 2004, S.217). Eine weitere Lösungsmöglichkeit stellt der sogenannte *Labeling-Algorithmus* dar, der auf dynamischer Programmierung basiert. Da Labeling-Algorithmen stark auf spezifische Problemstellungen zugeschnitten sind, erfolgt keine allgemeine Einführung und es wird auf die Kapitel 4 und 5 verwiesen.

3.4 Dynamische Programmierung

Die dynamische Programmierung befasst sich mit der Lösung von Entscheidungsproblemen, bei denen mehrere voneinander abhängige Entscheidungen getroffen werden müssen. Ziel ist es, für das Gesamtproblem ein Optimum zu finden. Die folgende Beschreibung der dynamischen Programmierung orientiert sich an den Ausführungen von Domschke et al. (2015). Eine Besonderheit der dynamischen Programmierung liegt in der sequentiellen Betrachtung mehrerer *Stufen*. Eine Stufe wird durch die Menge ihrer möglichen Zustände sowie durch die auf dieser Stufe zu treffenden Entscheidungen charakterisiert. Nach Domschke et al. (2015) lassen sich die Modelle der dynamischen Programmierung nach folgenden Punkten kategorisieren:

- (i) Zeitabstände der Stufen,
- (ii) Informationsgrad,
- (iii) Einwertigkeit oder Mehrwertigkeit der Zustands- und Entscheidungsvariablen,
- (iv) Endlichkeit oder Unendlichkeit der möglichen Zustände und Entscheidungen.

Zu (i): Es wird zwischen kontinuierlichen und diskreten Modellen unterschieden. Ein diskretes Modell umfasst Entscheidungen und Zustandsänderungen zu diskreten Zeitpunkten oder in diskreten Stufen. In einem kontinuierlichen Modell erfolgen Zustandsänderungen durch fortlaufende Entscheidungen. In dieser Arbeit werden ausschließlich diskrete Modelle betrachtet.

Zu (ii): Es wird zwischen deterministischen und stochastischen Modellen unterschieden. Dabei wird bei den deterministischen Modellen angenommen, dass äußere Einflüsse auf das System

stets nur einen festen Wert annehmen. Bei stochastischen Modellen werden die äußeren Einflüsse als Zufallsvariablen modelliert, die verschiedene Werte mit bekannten Wahrscheinlichkeiten annehmen können. In dieser Arbeit werden ausschließlich deterministische Modelle betrachtet.

Zu (iii): Zustände oder Entscheidungen können auch als Vektoren dargestellt werden, wenn sie aus mehreren Komponenten bestehen.

Zu (iv): Die Anzahl der möglichen Zustände oder Entscheidungen kann endlich oder unendlich sein. In dieser Arbeit werden ausschließlich endliche Zustände und Entscheidungen betrachtet.

Im Folgenden werden die Lösungsprinzipien der dynamischen Programmierung erläutert. Sei dazu z_k eine Zustandsvariable zur Wiedergabe eines Zustands, in dem sich das Problem oder System am Ende der Stufe/Periode k befindet. Sei Z_k die Menge aller Zustände, in denen sich das Problem oder System am Ende der Stufe/Periode k befinden kann. Damit lässt sich der Begriff der *Politik* definieren.

Definition 5. Eine Folge $(x_h, x_{h+1}, \dots, x_k)$ von Entscheidungen, die ein Problem von einem Zustand $z_{h-1} \in Z_{h-1}$ in einen Zustand $z_k \in Z_k$ überführt, wird als *Politik* bezeichnet. Entsprechend wird eine Folge $(x_h^*, x_{h+1}^*, \dots, x_k^*)$ von Entscheidungen, die ein Problem von einem Zustand $z_{h-1} \in Z_{h-1}$ in einen Zustand $z_k \in Z_k$ unter Einbeziehung der Zielfunktion überführt, als *optimale Politik* bezeichnet.

Auf Basis der Definition kann das *Bellman'sche Optimalitätsprinzip* eingeführt werden. Der Satz geht auf den Begründer der dynamischen Programmierung Bellman (1966) zurück.

Satz 4. Sei $(x_1^*, \dots, x_{k-1}^*, x_k^*, \dots, x_n^*)$ eine optimale Politik, die ein Problem von einem Anfangszustand in einen Endzustand überführt. Sei zusätzlich z_{k-1}^* der Zustand, den das System auf Stufe $k-1$ annimmt. Dann gilt:

- $(x_k^*, \dots, x_n^*)$ ist eine optimale (Teil-) Politik, die das Problem von einem Zustand z_{k-1}^* auf Stufe $k-1$ in einen Endzustand überführt.
- $(x_1^*, \dots, x_{k-1}^*)$ ist eine optimale (Teil-) Politik, die das Problem von einem Anfangszustand in einen Zustand z_{k-1}^* auf Stufe $k-1$ überführt.

Diesen Satz macht sich die dynamische Programmierung bei der Bestimmung einer optimalen Politik zunutze. Damit ist die Einführung in die mathematischen Grundlagen abgeschlossen. Im folgenden Kapitel wird die Optimierung eines Zustellnetzwerks unter Einsatz von Zustellrobotern betrachtet und ein Optimierungsalgorithmus für eine konkrete Problemstellung vorgestellt.

Kapitel 4

Optimierung von Zustellnetzwerken mit Zustellrobotern

In diesem Kapitel wird ein Optimierungsalgorithmus für ein Paketnetzwerk mit autonomen Zustellrobotern vorgestellt. Die im Folgenden betrachtete Problemstellung ähnelt der in Kapitel 2.2.3 beschriebenen Problemstellung. Es wird eine zweistufige letzte Meile mit autonomen Zustellrobotern untersucht. Dabei transportieren die Roboter Paketsendungen von Mikrodepots zu den Empfängern, die zuvor ein Zustellzeitfenster ausgewählt haben. Das Ziel besteht darin, einen Plan zu erstellen, der die Reihenfolge bestimmt, in der die Zustellroboter die Mikrodepots und Empfänger anfahren und dabei die Anzahl der Zustellungen maximiert. Es ist zu beachten, dass die Zustellroboter eine maximale Transportkapazität von einem Paket haben. Dies bedeutet, dass nach einer Zustellung ein Mikrodepot angefahren werden muss, um eine weitere Sendung zu laden. Das Mikrodepot kann nach jeder Zustellung gewechselt werden. Ein Roboter kann zwischen zwei Kunden mehrere Mikrodepots anfahren. Ein solches Vorgehen kann vorteilhaft sein, wie im späteren Verlauf anhand von Beispiel 1 gezeigt wird. Am Mikrodepot wird außerdem der Akku der Roboter wieder aufgeladen. Daher sind entsprechende Ladezeiten einzuplanen. Für die Zielfunktion wird jedem Kunden ein Profit zugewiesen, der eingesammelt wird, wenn der Kunde von einem Roboter beliefert wird. Das Ziel besteht darin, die Summe der gesammelten Profite zu maximieren. Allerdings ist es zur Zielerreichung nicht erforderlich, jeden Kunden zu beliefern. Aus logistischer Sicht können entgangene Profite als Opportunitätskosten interpretiert werden, die entstehen, wenn ein Kunde nicht durch das eigene System, sondern durch einen Kurier oder ein anderes Unternehmen versorgt werden muss.

Für eine realistische Modellierung ist es entscheidend, den Akku des Roboters und die Ladevorgänge zu berücksichtigen, da die Kapazität des Akkus oft restriktiv ist und Ladevorgänge Zeit in Anspruch nehmen. Beispielsweise hat der Zustellroboter von Starship Technologies eine Reichweite von sechs km und benötigt zum Laden 45 Minuten (Schweizerische Post AG, 2017). Der Ladezustand muss stets im zulässigen Bereich liegen und die Ladezeiten an den Mikrodepots müssen eingeplant werden. Bei Drohnen entfällt diese Betrachtung, da die meisten Modelle ein Konzept nutzen, bei dem die Akkus am Mikrodepot getauscht und durch geladene Akkus ersetzt

werden (siehe Wingcopter (2021); Shankland (2018)). Darüber hinaus gibt es bereits Stationen, die den Akkuwechsel bei Drohnen automatisieren (siehe Matternet (2020)). Im weiteren Verlauf des Kapitels werden die Mikrodepots der Einfachheit halber als Depots bezeichnet.

Dieses Kapitel ist wie folgt strukturiert: Zunächst erfolgt eine Einordnung in die verwandte Literatur. Anschließend wird eine Übersicht der in diesem Kapitel verwendeten Notationen präsentiert. Unterkapitel 4.3 beschreibt die Problemstellung formal und stellt ein einführendes Beispiel vor, das die Schwierigkeit der richtigen Depotwahl zwischen zwei Kunden aufzeigt. Anschließend wird die Problemstellung als MIP formuliert (siehe Unterkapitel 4.4). Das MIP dient als Vergleichsmodell für den in diesem Kapitel entwickelten Branch-and-Price-Algorithmus. Unterkapitel 4.5 betrachtet eine feste Reihenfolge der Kunden und untersucht das folgende Entscheidungsproblem: Für eine gegebene Menge von Depots, einen Zustellroboter und eine Kundenreihenfolge muss entschieden werden, ob zwischen je zwei Kunden ein Pfad von Depots mit entsprechenden Ladezeiten existiert, sodass jeder Kunde in seinem Zeitfenster beliefert werden kann. Für dieses Entscheidungsproblem wird ein dynamisches Programm vorgestellt, das es in Polynomialzeit basierend auf der Anzahl der Kunden und Depots exakt löst. Auf Basis dieses Programms wird in Unterkapitel 4.6 ein Branch-and-Price-Algorithmus präsentiert. Die Performance des Algorithmus wird anhand von Benchmark-Instanzen getestet und mit dem MIP-Modell in Kapitel 6 verglichen.

4.1 Einordnung in die Literatur

Im Folgenden wird die relevante Literatur vorgestellt. Da das Problem dieses Kapitels an zwei Themenfelder angrenzt, werden beide erläutert: zum einen die Optimierung von Zustellkonzepten mit Zustellrobotern und zum anderen Varianten des VRP, in denen die Ladevorgänge modelliert werden. Ostermeier et al. (2022) bieten eine Übersicht über Optimierungsprobleme mit Zustellrobotern. Die Forschung zu Zustellrobotern hat zu einer Vielzahl unterschiedlicher Problemvarianten geführt, die im Folgenden erläutert werden.

Zustellfahrzeug mit Zustellrobotern

Ein Konzept kombiniert ein Zustellfahrzeug, das neben Paketen auch Roboter transportiert und diese an bestimmten Orten zur Belieferung von Kunden startet. Boysen et al. (2018) beschreiben ein Problem mit einem Zustellfahrzeug, welches an Depots Roboter und Pakete aufnimmt und diese an vordefinierten Orten zu den Kunden losfahren lässt. Zur Lösung dieses Problems werden ein MIP-Modell und eine Heuristik entwickelt. Ziel der Berechnung ist es, die Anzahl verspäteter Zustellungen zu minimieren. Ostermeier et al. (2022) betrachten ein ähnliches Problem, wobei die Zielfunktion sowohl die Verspätung als auch die zurückgelegte Distanz reduziert. Darüber hinaus wird das Problem von Heimfarth et al. (2022) noch erweitert, sodass das Zustellfahrzeug neben dem Transport von Robotern und Paketen auch Kunden beliefern kann. Zur Lösung wird eine generalisierte variable Nachbarschaftssuche verwendet. Chen et al. (2021b,a) betrachten ein Problem, bei dem die Roboter aus dem Zustellfahrzeug direkt zu den Kunden starten. Dabei muss das Fahrzeug zur Weiterfahrt auf die Rückkehr der Roboter warten. Ziel ist es, die Gesamtzeit

aus Roboterfahrten und Fahrten des Zustellfahrzeugs zu minimieren. Heuristiken werden zur Problemlösung eingesetzt.

Zweistufige letzte Meile mit Zustellrobotern

Ein weiteres Konzept ähnelt dem in Unterkapitel 2.2.3 beschriebenen. Alfandari et al. (2022) untersuchen für eine zweistufige letzte Meile verschiedene Zielfunktionen zur Messung von Verspätungen. Zur Lösung werden ein MIP und ein exaktes Verfahren auf Basis der Benders-Dekomposition eingesetzt. Eine Fallstudie zur zweistufigen letzten Meile wird von Poeting et al. (2019a) vorgestellt. Die Autoren kombinieren eine Simulation und Optimierungsalgorithmen. Ziel ist es, einen bestimmten Servicegrad anhand der Anzahl der Verspätungen zu erreichen.

Location Routing Problem mit Zustellrobotern

Sonneberg et al. (2019) untersuchen neben der Zustellung mit Robotern auch die Entscheidung, ob ein potenzielles Depot eröffnet werden soll. Zusätzlich wird angenommen, dass die Roboter mehrere Fächer haben und damit mehrere Kunden in einer Tour beliefern können. Ziel ist es, die Betriebskosten, die durch die Eröffnung der Depots und den Betrieb der Roboter entstehen, zu minimieren.

Zustellroboter mit Betrachtung der Ladevorgänge

Yu et al. (2022) betrachten ebenfalls eine Variante des *Zustellfahrzeug mit Zustellroboter* Problems. Dabei können die Akkus der Roboter im Fahrzeug geladen werden. Die Belieferungen können sowohl vom Zustellfahrzeug als auch von den Robotern durchgeführt werden. Zur Lösung wird ein MIP-Modell und eine *Adaptive-Large-Neighborhood-Search*-Heuristik vorgestellt. Bakach et al. (2021) untersuchen eine Variante des *Location Routing Problems mit Zustellrobotern*. Dabei muss entschieden werden, welche Depots eröffnet werden sollen und von welchem Depot die Roboter ihre Auslieferungen durchführen sollen. Es wird angenommen, dass die Roboter nach jeder Belieferung ihren Akku vollständig aufladen müssen. Ziel ist es, die Gesamtkosten, bestehend aus Depotöffnungskosten und der Fahrzeit der Roboter, zu minimieren. Zur Lösung werden mehrere aufeinander aufbauende MIP-Modelle verwendet.

Electric Vehicle Routing Problem with Time Windows (EVRPTW)

Ein weiteres Forschungsgebiet, in dem Ladevorgänge betrachtet werden, ist das *Electric Vehicle Routing Problem with Time Windows* (EVRPTW) (siehe Schneider et al. (2014)). Das EVRPTW ähnelt dem VRP, wobei zusätzlich Zeitfenster für die Kunden berücksichtigt werden und die Fahrzeuge elektrisch sind. Dies erfordert die Berücksichtigung von Ladestationen und die Modellierung der Ladevorgänge. Eine mögliche Zielfunktion ist die Minimierung der zurückgelegten Distanz. Das hier beschriebene Problem kann als kapazitätsbeschränktes EVRPTW betrachtet werden, dessen Transportkapazität auf eine Einheit beschränkt ist. Aufgrund dieser Restriktion müssen die Fahrzeuge zwischen zwei Kunden jeweils ein Depot oder eine Ladestation anfahren.

Eine Übersicht über das allgemeine EVRP ohne Zeitfenster wird von Erdelić und Carić (2019) gegeben. Die erste Publikation zum EVRPTW stammt von Schneider et al. (2014). Die Autoren formulieren das Problem als MIP und stellen eine Kombination aus variabler Nachbarschaftssuche und Tabu-Heuristik vor. Hierbei wird allerdings angenommen, dass die Fahrzeuge bei Ankunft

an einer Ladestation die Akkus vollständig laden müssen. Diese Restriktion wird von Keskin und Çatay (2016) aufgehoben und es wird neben einem MIP auch eine *Adaptive Large Neighborhood Search* Heuristik vorgestellt. Desaulniers et al. (2016) untersuchen verschiedene Ladevarianten des EVRPTW. Neben einer Ladestrategie, die nur vollständige Ladevorgänge zulässt, werden auch partielle Ladevorgänge an den Depots berücksichtigt. Es wird ein exakter *Branch-and-Price-and-Cut*-Algorithmus vorgestellt. Ähnlich wie im folgenden Abschnitt werden feste Reihenfolgen betrachtet und Ladezeiten berechnet. Der Hauptunterschied liegt in den Knoten, welche in den Reihenfolgen enthalten sind. Während die Autoren sowohl Kunden als auch Ladestationen in den Reihenfolgen berücksichtigen, werden in dieser Arbeit ausschließlich Kunden betrachtet und die Ladestationen implizit durch ein dynamisches Programm modelliert.

Die Zielfunktion dieses Kapitels basiert auf einem Profit, der die Zahlungsbereitschaft der Kunden symbolisiert. Dieser Profit wird erzielt, wenn der Kunde von einem Zustellroboter beliefert wird. Diese Art der Zielfunktion für das VRP ist in der Literatur als *Team Orienteering Problem (TOP)* bekannt. Tae und Kim (2015) beschreiben ein TOP mit Zeitfenstern und stellen einen Branch-and-Price-Algorithmus zur Lösung des Problems vor. Ein kapazitiertes TOP mit Zeitfenstern wird von Park et al. (2017) betrachtet. Die Autoren stellen auch einen Branch-and-Price-Algorithmus für die Lösung des Problems vor. Beide Publikationen berücksichtigen jedoch weder Akkus noch Ladevorgänge.

4.2 Notationen

Ein Überblick über alle in diesem Kapitel verwendeten Notationen ist in Tabelle 4.1 zu finden.

Allgemeine Notationen	
C	Menge der Kunden
D	Menge der Depots
m	Anzahl der verfügbaren Roboter
p_i	Zahlungsbereitschaft von Kunde $u_i \in C$
$[a_i, b_i]$	Zeitfenster von Kunde $u_i \in C$
T	Zeithorizont
τ_i	Servicezeit bei Kunde $u_i \in C$
d_{ij}	Fahrzeit zwischen Knoten $u_i \in V$ und Knoten $u_j \in V$
d_i	Fahrzeit von Kunde $u_i \in C$ zu seinem nächsten Depot $u_{k(i)}$
B	maximale Fahrzeit des Akkus
β_i	Ladestand des Akkus an Knoten $u_i \in V$
λ_k	Ladezeit an Depot $u_k \in D$
α	Laderate gemessen in Fahrzeit je geladener Zeiteinheit
t_i	Ankunftszeitpunkt des Kunden oder Depots $u_i \in V$
s	Sequenz von Kunden mit $s = (u_1, u_2, \dots, u_h)$
MIP (Unterkapitel 4.4)	
$\tilde{S}_{ij}$	Menge der Depot-Pfade zwischen Knoten $u_i, u_j \in V$
d_{ij}^k	Fahrzeit von $u_i \in C$ nach $u_j \in C$ auf Depot-Pfad $k \in \tilde{S}_{ij}$
$\underline{d}_{ij}^k$	Fahrzeit von $u_i \in C$ zum ersten Depot auf dem Depot-Pfad $k \in \tilde{S}_{ij}$ zu $u_j \in C$
$\bar{d}_{ij}^k$	Fahrzeit vom letzten Depot auf dem Depot-Pfad $k \in \tilde{S}_{ij}$ zu $u_j \in C$
Dynamisches Programm (Unterkapitel 4.5)	
(t, β)	Zeit-Batterie-Paar
$(\underline{t}, \underline{\beta}), (\bar{t}, \bar{\beta})$	unteres und oberes Zeit-Batterie-Paar
$(\underline{t}_k, \underline{\beta}_k), (\bar{t}_k, \bar{\beta}_k)$	unteres und oberes temporäres Zeit-Batterie-Paar
D'	reduzierte Menge der Depots
σ	Anzahl Abschnitte auf gebrochener Geraden
ρ	Index eines Abschnitts auf gebrochener Gerade
Branch-and-Price-Algorithmus (Unterkapitel 4.6)	
e_{is}	binärer Koeffizient, der angibt, ob Kunde u_i in Sequenz s enthalten ist
Ω	Menge der zulässigen Sequenzen
Ω'	reduzierte Menge der zulässigen Sequenzen mit $\Omega' \subseteq \Omega$
$\bar{c}(s, \mathbf{y}^*)$	reduzierte Kosten von Sequenz s
$\mathcal{V}$	Menge der Knoten in Hilfsgraph $\mathcal{G}$
FW, BW	Menge der vorwärts- und rückwärtsgerichteten Kunden
W	Weg mit $W = (w_0, u_1, u_2, \dots, u_h, w_{n+1})$ oder Teilweg
w_0, w_{n+1}	Quelle und Senke der Wege
L	Label
$\mathcal{L}, \mathcal{K}$	Menge der Label

Tabelle 4.1: Notationen für Kapitel 4

4.3 Problembeschreibung

In diesem Unterkapitel erfolgt eine formale Beschreibung der Optimierung eines Zustellnetzwerks mit autonomen Robotern. Gegeben ist ein ungerichteter Graph $G = (V, E)$, wobei $V = C \cup D$ die Knotenmenge darstellt und $E = \{\{i, k\} : u_i \in V, u_k \in D\}$ die Menge der Kanten. Die Menge C beschreibt die Kunden und D die Menge der Depots. Es ist zu beachten, dass zwei Kunden nicht direkt miteinander verbunden sind und der Zustellroboter zwischen je zwei Kunden mindestens ein Depot anfahren muss. Der Grund hierfür ist die Kapazität des Laderaums des Roboters, die maximal eine Sendung fasst. Daher sind die Kunden ausschließlich mit den Depots verbunden. Es wird angenommen, dass jeder Kunde $u_i \in C$ ein Zeitfenster $[a_i, b_i]$ hat, wobei stets $a_i \leq b_i$ gilt. Die Belieferung kann nur innerhalb des Zeitfensters erfolgen. Bei erfolgreicher Zustellung wird ein Preis von p_i vom Kunden gezahlt. Die Entnahme der Sendung am Kunden wird als Servicezeit bezeichnet. Diese ist für jeden Kunden $u_i \in C$ individuell und benötigt τ_i Einheiten.

Die Fahrzeit von u_i nach u_k , $\{i, k\} \in E$, beträgt d_{ik} Zeiteinheiten. Im Folgenden wird angenommen, dass die Fahrzeiten symmetrisch sind, sodass $d_{ik} = d_{ki}$ für jede Kante $\{i, k\} \in E$ gilt. Das Depot mit der kürzesten Fahrzeit zu Kunde $u_i \in C$ wird mit $u_{k(i)}$ bezeichnet und die Fahrzeit von u_i zu $u_{k(i)}$ wird mit d_i gekennzeichnet.

Zur Belieferung der Kunden stehen m Zustellroboter zur Verfügung. Wie bereits erwähnt, haben diese eine Transportkapazität von einer Sendung und steuern nach jeder Belieferung ein Depot an. Jedoch können zwischen je zwei Kunden auch mehrere Depots angefahren werden. Es wird angenommen, dass die Zuordnung der Sendungen zu den Depots zu Beginn noch nicht erfolgt ist und somit Teil der Optimierung ist, ebenso wie die Zuordnung der Zustellroboter zu den Depots. Alle Roboter fahren mit der gleichen Geschwindigkeit, welche als fest angesehen wird. Jeder Roboter ist elektrisch angetrieben und besitzt daher einen Akku. Dieser kann an jedem Depot nachgeladen werden. Es wird angenommen, dass die Zunahme des Ladestandes während des Ladevorgangs und die Abnahme während der Fahrt jeweils linear in Abhängigkeit von der Zeit sind. Marra et al. (2012) zeigen, dass diese Annahme für Lithium-Ionen-Akkumulatoren bei einem Ladestand von 20% bis 90% gerechtfertigt ist. Außerdem gilt, dass das komplette Entladen und vollständige Laden des Akkus die Lebenszeit verringert. Daher wird im Folgenden angenommen, dass der Ladestand nicht unter 20% sinken kann und Ladevorgänge nur bis 80% der Kapazität durchgeführt werden. Es wird angenommen, dass der Energieverbrauch des Roboters während der Be- oder Entladung mit Sendungen sowie eventuellen Wartezeiten am Depot vernachlässigbar ist (Bakach et al., 2021). Der Ladestand des Akkus gibt die verbleibende Fahrzeit an. Sei B die maximale Fahrzeit. Im Folgenden wird zur Vereinfachung der Ladestand im Intervall $[0, B]$ abgebildet, wobei dieses Intervall innerhalb der Grenzen zwischen 20% und 80% liegt. Dies bedeutet, dass der tatsächliche Ladestand des Akkus noch 20% beträgt, wenn dieser im Modell leer ist. Nach einer vollen Ladung des Akkus im Modell entspricht der Ladestand tatsächlich 80%. Die Ladegeschwindigkeit beträgt α . Dies bedeutet, dass sich mit jeder Zeiteinheit, die geladen wird, die Fahrzeit um α erhöht. Die Ladezeit am Depot $u_k \in D$ wird mit λ_k bezeichnet. Der Ladestand des Roboters beim Eintreffen an Knoten $u_i \in V$ wird mit β_i bezeichnet.

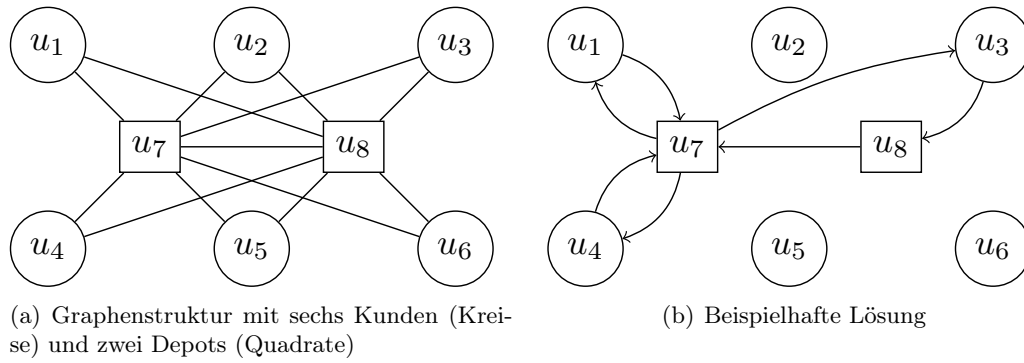


Abbildung 4.1: Visualisierung des Graphen und einer Lösung

Ziel der Optimierung ist die Maximierung der gesammelten Profite unter den folgenden Nebenbedingungen:

- Jeder Kunde wird höchstens einmal beliefert.
- Kunden werden in ihren Zeitfenstern beliefert.
- Der Ladestand des Akkus bewegt sich immer im Bereich $[0, B]$.

Es ist anzumerken, dass nicht alle Kunden beliefert werden müssen. Für den Graphen G gelten folgende Einschränkungen:

- Der Subgraph, bestehend aus den Depotknoten, ist bezüglich der maximalen Fahrzeit zusammenhängend. Das heißt, dass es zwischen je zwei Depots $u_i, u_k \in D$ einen Weg gibt, wobei jede Kante auf dem Weg die maximale Fahrzeit B nicht überschreitet.
- Jeder Kunde kann von mindestens einem Depot erreicht werden. Das bedeutet, dass für jeden Kunden $u_i \in C$ gilt $d_i \leq B/2$.
- Die Roboter starten von einem beliebigen Depot mit einem Ladestand, der B entspricht.

Ein Beispiel für einen Graphen mit sechs Kunden (Kreise) und zwei Depots (Quadrate) ist in Abbildung 4.1(a) dargestellt. Eine beispielhafte Lösung ist zusätzlich in Abbildung 4.1(b) eingezeichnet. Die resultierende Reihenfolge lautet $(u_7, u_1, u_7, u_3, u_8, u_7, u_4, u_7)$.

Die Einführung schließt mit einem Beispiel, das zeigt, dass die Wahl eines oder mehrerer Depots zwischen zwei Kunden nicht trivial ist. Ohne Akkukapazitätsgrenze B wäre das Depot mit minimaler Fahrzeit stets optimal. Wird B jedoch berücksichtigt, kann ein anderes Depot vorteilhafter sein, wie Beispiel 1 verdeutlicht.

Beispiel 1. Dieses Beispiel betrachtet die Depotauswahl zwischen den Kunden u_i und u_j . Aktuell befindet sich der Roboter bei Kunde u_i . Zur Auswahl stehen die Depots u_k und u_l . Die benötigten Fahrzeiten sind in Abbildung 4.2 neben den Kanten dargestellt. In diesem Beispiel ist u_k das Depot, das die Fahrzeit zwischen u_i und u_j minimiert. Die Strecke von u_i nach u_j über u_k

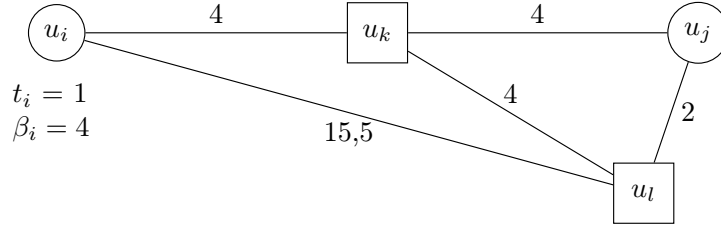


Abbildung 4.2: Graph mit zwei Kunden und zwei Depots

beträgt acht Einheiten. Es wird angenommen, dass die maximale Fahrzeit zehn Einheiten beträgt ($B = 10$).

Das Zeitfenster von Kunde u_j lautet $[19, 24]$. Es wird angenommen, dass Kunde u_i zum Zeitpunkt eins beliefert wird und die Servicezeit beträgt eine Einheit. Der aktuelle Ladestand des Fahrzeugs liegt bei vier Einheiten. Damit kann der Roboter zum Zeitpunkt zwei nur zu u_k fahren, da das Depot u_l 15,5 Einheiten entfernt liegt und somit nicht erreichbar ist.

Zum Zeitpunkt sechs erreicht der Roboter das Depot u_k mit einem leeren Akku. Die Ladegeschwindigkeit für dieses Beispiel beträgt zwei Einheiten, was bedeutet, dass $\alpha = 2$ ist. Am Depot gibt es zwei Möglichkeiten zur Weiterfahrt: 1) Der Roboter kann zu Kunde u_j fahren oder 2) zuerst zu Depot u_l und dann zu u_j . Im Folgenden werden beide Möglichkeiten genauer betrachtet.

- Falls der Roboter direkt zu u_j fährt, benötigt er mindestens zwei Zeiteinheiten zum Laden, um die erforderlichen vier Einheiten zu erhalten. Allerdings startet das Zeitfenster von u_j erst zum Zeitpunkt 19. Daher kann der Roboter in fünf Zeiteinheiten seinen Akku vollständig laden und wartet dann noch vier Zeiteinheiten. Anschließend startet der Roboter zum Zeitpunkt 15 vom Depot und erreicht den Kunden u_j zum frühestmöglichen Zeitpunkt mit einem Ladestand von sechs Einheiten. Der Verlauf des Ladestandes über die Zeit ist in Abbildung 4.3 durch eine gepunktete Linie dargestellt.
- Im zweiten Fall fährt der Roboter zum Depot u_l . Dafür lädt er zwei Zeiteinheiten und erreicht das Depot zum Zeitpunkt 12 mit einem leeren Akku. Am Depot angekommen, kann der Roboter fünf Zeiteinheiten laden und dann mit vollem Akku zu u_j starten. Dadurch erreicht der Roboter u_j auch zum frühestmöglichen Zeitpunkt, der bei 19 liegt, mit einem Ladestand von acht Einheiten. Der Verlauf des Ladestandes für diese Option ist als gestrichelte Linie in Abbildung 4.3 aufgezeigt.

Es bleibt festzuhalten, dass der Roboter auf beiden Pfaden den Kunden u_j zum frühesten Zeitpunkt beliefert. Allerdings ist der Ladestand beim Kunden u_j im Fall des zweiten Pfads höher als bei einer direkten Fahrt. Dieses Beispiel zeigt, dass nicht zwangsläufig das Depot optimal ist, welches die Fahrzeit zwischen zwei Kunden minimiert.

Die Problemstellung vereint zwei gegenläufige Ziele: einen möglichst frühen Belieferungszeitpunkt und einen möglichst hohen Ladestand. Das in Unterkapitel 4.5 vorgestellte dynamische Programm löst diesen Zielkonflikt.

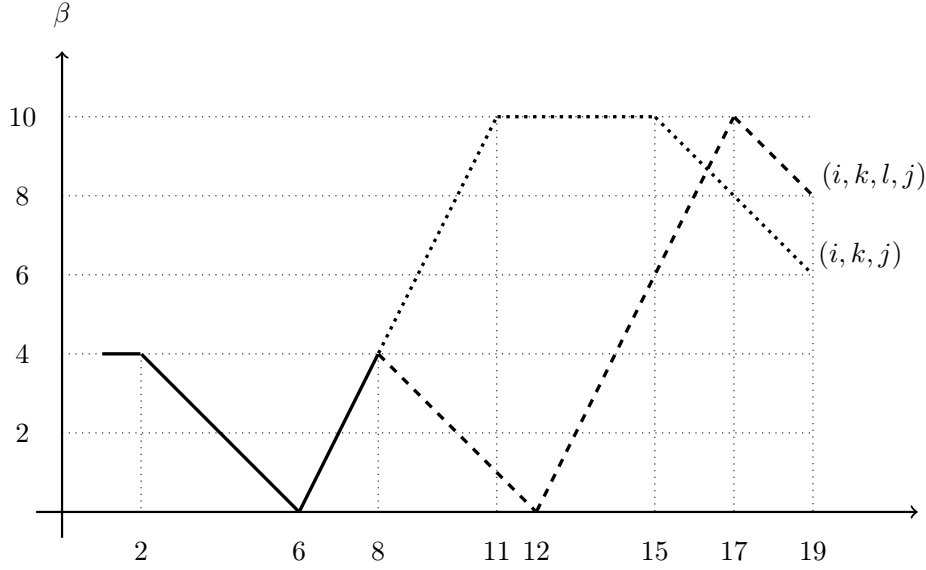


Abbildung 4.3: Akkustand über die Zeit für zwei unterschiedliche Zustellpfade

4.4 Gemischt-ganzzahliges lineares Programm

In diesem Abschnitt wird das Optimierungsproblem aus Unterkapitel 4.3 als MIP formuliert. Das Programm ist ähnlich zur Formulierung von Roberti und Wen (2016). Allerdings betrachten die Autoren das sogenannte *Electric Traveling Salesperson Problem* (E-TSP) mit Zeitfenstern, wobei Ladezeiten und -mengen als ganzzahlig angenommen werden. Beim E-TSP gilt es, eine kürzeste Tour mit einem elektrischen Fahrzeug durch eine Menge von Knoten zu finden. Dabei wird die Kapazität des Akkus berücksichtigt und das Fahrzeug an Ladesäulen wieder aufgeladen.

Sei 0 ein fiktiver Knoten, der Start- und Endpunkt einer jeden Tour ist. Es gilt, dass 0 keinen Profit hat und die Fahrzeit von 0 zu allen Depots null ist. Sei $\tilde{S}_{ij}$ die Menge aller Depot-Pfade zwischen den Knoten u_i und u_j , wobei $u_i, u_j \in C \cup \{0\}$ gelten. Die Fahrzeit von u_i nach u_j über Depot-Pfad $\tilde{s}_{ij}^k$ ist als d_{ij}^k definiert. Die Fahrzeit von u_i zum ersten Depot auf dem Pfad $\tilde{s}_{ij}^k$ beträgt $\underline{d}_{ij}^k$ und die Zeit vom letzten Depot zu u_j beträgt $\bar{d}_{ij}^k$. Es ist zu beachten, dass die Anzahl der Depot-Pfade zwischen je zwei Kunden exponentiell in der Menge der Depots ist. Um die Anzahl der betrachteten Depot-Pfade zu verringern, wird eine Dominanzregel eingeführt. Seien u_i, u_j zwei Kunden und $\tilde{s}_{ij}^1, \tilde{s}_{ij}^2$ zwei Depot-Pfade zwischen diesen Kunden. Es gilt, dass $\tilde{s}_{ij}^1$ von $\tilde{s}_{ij}^2$ dominiert wird, falls

- $d_{ij}^1 \geq d_{ij}^2$,
- $\underline{d}_{ij}^1 \geq \underline{d}_{ij}^2$,
- $\bar{d}_{ij}^1 \geq \bar{d}_{ij}^2$.

Es werden binäre Variablen x_{ij}^k mit $u_i, u_j \in C \cup \{0\}$ und $\tilde{s}_{ij}^k \in \tilde{S}_{ij}$ definiert, die 1 sind, falls der Depot-Pfad $\tilde{s}_{ij}^k$ von u_i nach u_j genutzt wird, und andernfalls 0 betragen. Sei t_i der Belieferungs-

zeitpunkt des Kunden $u_i \in C$ und β_i der Ladestand am Kunden. Außerdem sei λ_i die Ladezeit des Roboters vor der Belieferung von Kunde $u_i \in C$. Das MIP kann wie folgt formuliert werden:

$$\text{maximiere} \quad \sum_{i,j \in C \cup \{0\}} \sum_{\tilde{s}_{ij}^k \in \tilde{S}_{ij}} p_i x_{ij}^k \quad (4.1)$$

$$\text{u.d.N.} \quad \sum_{i \in C \cup \{0\}} \sum_{\tilde{s}_{ij}^k \in \tilde{S}_{ij}} x_{ij}^k \leq 1 \quad \forall j \in C, \quad (4.2)$$

$$\sum_{j \in C \cup \{0\}} \left(\sum_{\tilde{s}_{ij}^k \in \tilde{S}_{ij}} x_{ij}^k - \sum_{\tilde{s}_{ji}^k \in \tilde{S}_{ji}} x_{ji}^k \right) = 0 \quad \forall i \in C, \quad (4.3)$$

$$\sum_{i \in C} \sum_{\tilde{s}_{0i}^k \in \tilde{S}_{0i}} x_{0i}^k \leq m, \quad (4.4)$$

$$t_i + \tau_i + \lambda_j + \sum_{\tilde{s}_{ij}^k \in \tilde{S}_{ij}} d_{ij}^k x_{ij}^k \leq t_j + \left(1 - \sum_{\tilde{s}_{ij}^k \in \tilde{S}_{ij}} x_{ij}^k\right) M \quad \forall i, j \in C, \quad (4.5)$$

$$\beta_i - \sum_{\tilde{s}_{ij}^k \in \tilde{S}_{ij}} d_{ij}^k x_{ij}^k + \alpha \lambda_j \geq \beta_j - \left(1 - \sum_{\tilde{s}_{ij}^k \in \tilde{S}_{ij}} x_{ij}^k\right) M \quad \forall i, j \in C, \quad (4.6)$$

$$\sum_{j \in C \cup \{0\}} \sum_{\tilde{s}_{ij}^k \in \tilde{S}_{ij}} d_{ij}^k x_{ij}^k \leq \beta_i \quad \forall i \in C, \quad (4.7)$$

$$\beta_i \leq B - \sum_{j \in C \cup \{0\}} \sum_{\tilde{s}_{ji}^k \in \tilde{S}_{ji}} \bar{d}_{ji}^k x_{ji}^k \quad \forall i \in C, \quad (4.8)$$

$$d_i \leq t_i \quad \forall i \in C, \quad (4.9)$$

$$a_i \leq t_i \leq b_i \quad \forall i \in C, \quad (4.10)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i, j \in C \cup \{0\}, \tilde{s}_{ij}^k \in \tilde{S}_{ij}. \quad (4.11)$$

Die Zielfunktion (4.1) maximiert die Summe der Profite. Die Nebenbedingungen (4.2) stellen sicher, dass jeder Kunde maximal einmal angefahren wird. Die Flusserhaltung in den Kundenknoten wird mit den Nebenbedingungen (4.3) gewährleistet. Dies bedeutet, dass in einer Lösung jeder Kundenknoten ebenso häufig angefahren wie verlassen wird. Durch die Nebenbedingung (4.4) wird garantiert, dass maximal m Roboter eingesetzt werden. Die korrekten Belieferungszeitpunkte der Kunden werden durch die Nebenbedingungen (4.5) festgelegt, wobei M eine große Zahl ist. Die Korrektheit des Ladestandes zwischen zwei Kunden wird mit den Nebenbedingungen (4.6) gewährleistet. Der minimale und maximale Ladestand an einem Kunden wird durch die Nebenbedingungen (4.7) und (4.8) eingehalten. Eine untere Begrenzung an den frühesten Ankunftszeitpunkt an einem Kunden ist durch Nebenbedingung (4.9) gegeben. Die Definition der Zeitfenster und die Binarität der Entscheidungsvariablen ist durch (4.10) und (4.11) gegeben. Damit ist die Vorstellung des MIP abgeschlossen. Im folgenden Unterkapitel werden Depot-Pfade berechnet, wobei die Reihenfolge der Kunden als gegeben angenommen wird.

4.5 Dynamisches Programm für feste Sequenz

In diesem Abschnitt wird eine feste Reihenfolge der Kunden betrachtet. Dazu wird die Definition einer *Sequenz* eingeführt.

Definition 1. Eine Sequenz $s = (u_1, u_2, \dots, u_h)$ beschreibt eine Reihenfolge der Kunden, mit $u_1, u_2, \dots, u_h \in C$.

Für diese Sequenz wird ein dynamisches Programm vorgestellt, das das folgende Entscheidungsproblem lösen kann.

Entscheidungsproblem 1. Gegeben sind eine Menge von Depots, ein Zustellroboter und eine Sequenz. Gibt es zwischen je zwei Kunden einen zulässigen Depot-Pfad mit entsprechenden Ladezeiten, sodass jeder Kunde in seinem Zeitfenster beliefert wird und der Ladestand im zulässigen Bereich bleibt?

Wie auch die Problemstellung 4.3 beinhaltet das Entscheidungsproblem 1 zwei konträre Ziele: die Minimierung der Zeit zur Einhaltung der Zeitfenster und die Maximierung des Ladestandes zur Erhöhung der Reichweite. Um diesen Konflikt zu lösen, wird ein dynamisches Programm vorgestellt, das das Entscheidungsproblem 1 in Polynomialzeit löst. Das dynamische Programm wird anschließend in den Branch-and-Price-Algorithmus eingebunden und ermöglicht die Lösung des Pricing-Problems (siehe Unterkapitel 4.6.2). Dieses Entscheidungsproblem kann auch als Variante des *Fixed Route Vehicle Charging Problem* (FRVCP) aufgefasst werden. Das Ziel des FRVCP ist es, für eine gegebene Reihenfolge von Kunden Ladestationen und Ladezeiten einzuplanen, sodass der Ladestand des Akkus nicht unter null sinkt. Eine Übersicht über Publikationen für das FRVCP wird von Kullman et al. (2021) gegeben. Roberti und Wen (2016) betrachten eine Variante mit einem Fahrzeug, Zeitfenstern und partiellen Ladevorgängen. Die Autoren stellen einen exakten Labeling-Algorithmus vor, der auf Depot-Pfaden zwischen je zwei Kunden basiert. Allerdings gilt, dass die Ladezeit und die Lademenge ganzzahlig sind und für jede geladene Menge ein neues Label erschaffen wird. Hiermann et al. (2019) betrachten eine Variante des FRVCP mit Zeitfenstern und partiellen Ladevorgängen, bei der maximal eine Ladestation zwischen je zwei Kunden angefahren werden darf. Wie in der zuvor erwähnten Publikation wird ein exakter Algorithmus vorgestellt, der auf Depot-Pfaden zwischen je zwei Kunden basiert. Eine Variante mit einer nichtlinearen Ladefunktion wird von Froger et al. (2019) vorgestellt. Die Autoren betrachten eine stückweise lineare, konkave Ladefunktion und präsentieren einen exakten Algorithmus, der auf Labeln basiert. Diese Art der Modellierung ermöglicht es, dass zwischen zwei Kunden mehrere Ladestationen angefahren werden können. Dieser Algorithmus wurde von Kullman et al. (2021) implementiert. Nach der Literaturvorstellung für das FRVCP werden einführende Definitionen präsentiert, die für das dynamische Programm und den anschließenden Beweis benötigt werden.

4.5.1 Zeit-Batterie-Kurven

Gemäß der obigen Beschreibung soll ein Pfad in G gefunden werden, sodass alle Kunden einer Sequenz $s = (u_1, \dots, u_h)$ in ihren Zeitfenstern beliefert werden und die Grenzen des Akkus eingehalten werden. Ein solcher Pfad muss an einem Depot starten und enden.

Das dynamische Programm baut den gesuchten Pfad iterativ auf. Dabei wird in jeder Iteration das Präfix einer Sequenz $s' = (u_1, \dots, u_i)$ über alle besten Depot-Pfade zum nächsten Kunden u_{i+1} erweitert. Nach der Erweiterung zum letzten Kunden u_h wird die Lösung des Entscheidungsproblems zurückgegeben.

Definition 6. Sei s' eine Sequenz. Eine Lösung π zu s' ist ein Pfad in G zusammen mit Ladezeiten an den Depots, welcher an einem Depot beginnt und die Kunden in der gegebenen Reihenfolge besucht. Eine Lösung π ist zulässig, wenn

- jeder Kunde in s' in seinem Zeitfenster beliefert wird und
- sich der Ladestand zu jedem Zeitpunkt zwischen $[0, B]$ befindet.

Wenn π im Kunden u_j endet, dann muss der Ladestand mindestens d_j betragen. Damit ist sichergestellt, dass der Zustellroboter noch das nächstgelegene Depot erreichen kann. Existiert keine zulässige Lösung für eine Sequenz s' , wird diese Sequenz als unzulässig bezeichnet.

Es ist anzumerken, dass eine Lösung auch an einem Depot nach dem letzten Kunden aus s' enden kann. Außerdem ist es nach der Definition möglich, ein Depot zwischen zwei Kunden mehrfach anzufahren. Dies gilt nicht für Kunden, da dies bereits in der Problembeschreibung ausgeschlossen ist. Es ist zu beachten, dass diese Annahme den Lösungsraum nicht beschränkt, sofern die zugrunde liegende Metrik die Dreiecksungleichung erfüllt.

Auf Grundlage der Definition einer zulässigen Lösung lässt sich das sogenannte *Zeit-Batterie-Paar* definieren.

Definition 7. Sei s' eine Sequenz und sei π eine zulässige Lösung für s' . Angenommen, π endet am Knoten u_i zum Zeitpunkt t mit Ladestand β . Das Paar (t, β) wird als Zeit-Batterie-Paar für u_i in Bezug auf s' bezeichnet. Des Weiteren wird die Lösung π als Zertifikat für (t, β) bezeichnet.

Zur Vereinfachung wird im Folgenden nur das Paar (t, β) als *Zeit-Batterie-Paar* bezeichnet, wenn s' und/oder u_i aus dem Kontext klar sind. Zum Vergleich von zwei Zeit-Batterie-Paaren an einem Kunden oder einem Depot werden Dominanzregeln aufgestellt. Diese Regeln beschreiben pareto-optimale Zeit-Batterie-Paare.

Definition 8. Für eine gegebene Sequenz, die bei Kunden $u_i \in C$ endet, seien (t_i, β_i) und $(\tilde{t}_i, \tilde{\beta}_i)$ zwei Zeit-Batterie-Paare für u_i . Es gilt, dass (t_i, β_i) von $(\tilde{t}_i, \tilde{\beta}_i)$ dominiert wird, wenn

$$t_i \geq \tilde{t}_i \text{ und} \tag{4.12}$$

$$\beta_i \leq \tilde{\beta}_i. \tag{4.13}$$

Dies bedeutet, dass ein Zeit-Batterie-Paar ein anderes Paar dominiert, wenn es nicht später und nicht mit einem niedrigeren Ladestand den Kunden beliefert. Auch für die Depots kann eine Dominanzregel definiert werden. Diese unterscheidet sich jedoch von der Regel für die Kunden, da an den Depots geladen werden kann.

Definition 9. Sei $u_k \in D$ ein Depot und seien (t_k, β_k) und $(\tilde{t}_k, \tilde{\beta}_k)$ zwei Zeit-Batterie-Paare für u_k , die auf der gleichen Sequenz basieren. Es gilt, dass (t_k, β_k) von $(\tilde{t}_k, \tilde{\beta}_k)$ dominiert wird, falls

$$t_k \geq \tilde{t}_k \text{ und} \quad (4.14)$$

$$\beta_k \leq \tilde{\beta}_k + \alpha(t_k - \tilde{t}_k). \quad (4.15)$$

Dies bedeutet, dass der Ankunftszeitpunkt des dominierenden Paares nicht später als der des dominierten Paares am Depot sein darf. Zusätzlich darf der Ladestand im Vergleich zum dominierten Paar nicht niedriger sein, wenn die Zeitdifferenz, die durch ein früheres Eintreffen am Depot entsteht, zum Laden genutzt wird.

Wie zuvor erläutert, verfolgt das Kapitel das Ziel, das Entscheidungsproblem zu lösen, ob für eine Sequenz s eine zulässige Lösung existiert. Zur Beantwortung des Problems werden iterativ an jedem Kunden alle nicht dominierten Zeit-Batterie-Paare berechnet. Jedes dieser Paare muss eine zulässige Lösung als Zertifikat besitzen. Die Berechnung aller dieser Paare für die Kunden der Sequenz ist ein Schlüsselkonzept für die Lösung des Entscheidungsproblems.

Definition 10. Sei s eine Sequenz, die bei Kunde $u_i \in s$ endet und eine zulässige Lösung besitzt. Die Vereinigung aller nicht dominierten Zeit-Batterie-Paare (Selbstdominanz ausgeschlossen) für Kunde u_i wird als Zeit-Batterie-Kurve bezeichnet.

Im weiteren Verlauf des Kapitels wird gezeigt, dass die Zeit-Batterie-Kurven in polynomieller Zeit berechnet werden können. Außerdem besitzen sie eine einfache Struktur. Die Kurven sind stückweise Funktionen, die aus linearen Segmenten mit Steigung α und horizontalen Lücken bestehen.

Definition 11. Für eine Sequenz, die bei Kunden u_i endet, seien $(\underline{t}^1, \underline{\beta}^1), (\underline{t}^2, \underline{\beta}^2), \dots, (\underline{t}^\sigma, \underline{\beta}^\sigma)$ und $(\bar{t}^1, \bar{\beta}^1), (\bar{t}^2, \bar{\beta}^2), \dots, (\bar{t}^\sigma, \bar{\beta}^\sigma)$, $\sigma \geq 1$, Zeit-Batterie-Paare für u_i mit

$$\underline{t}^\rho \leq \bar{t}^\rho \text{ und } \underline{\beta}^\rho \leq \bar{\beta}^\rho \text{ für alle } \rho = 1, \dots, \sigma, \quad (4.16)$$

$$\bar{t}^\rho < \underline{t}^{\rho+1} \text{ und } \bar{\beta}^\rho = \underline{\beta}^{\rho+1} \text{ für alle } \rho = 1, \dots, \sigma - 1. \quad (4.17)$$

Eine gebrochene Gerade ist eine abschnittsweise definierte Funktion $\beta : [\underline{t}^1, \bar{t}^1] \cup \bigcup_{\rho=2}^\sigma [\underline{t}^\rho, \bar{t}^\rho] \rightarrow [\underline{\beta}^1, \bar{\beta}^\sigma]$ mit:

$$\beta(t) = \begin{cases} \underline{\beta}^1 + \alpha(t - \underline{t}^1) & \text{für alle } t \in [\underline{t}^1, \bar{t}^1], \\ \underline{\beta}^\rho + \alpha(t - \underline{t}^\rho) & \text{für alle } t \in [\underline{t}^\rho, \bar{t}^\rho] \text{ und } \rho \in \{2, \dots, \sigma\}, \end{cases}$$

sodass jedes Paar $(t, \beta(t))$ der Funktion β ein Zeit-Batterie-Paar für u_i repräsentiert.

Trotz der kompliziert anmutenden Definition einer gebrochenen Geraden lassen sich diese Geraden leicht visuell darstellen. Abbildung 4.4 zeigt eine gebrochene Gerade mit drei Abschnitten, die jeweils nach rechts verschoben sind.

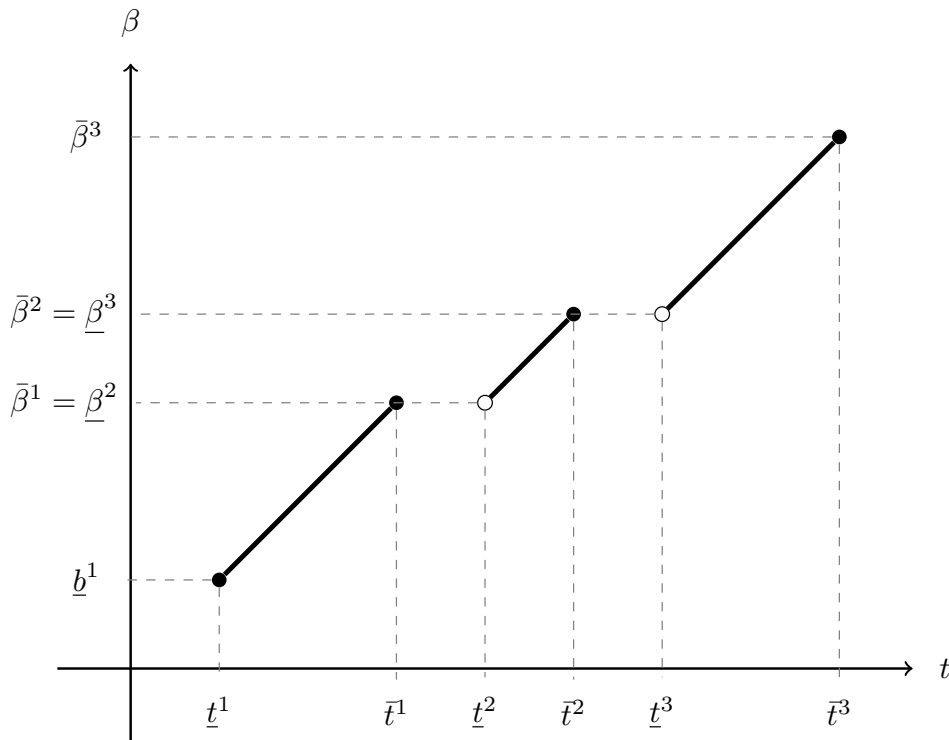


Abbildung 4.4: Beispiel einer gebrochenen Geraden mit drei Abschnitten

In Lemma 7 wird gezeigt, dass jede Zeit-Batterie-Kurve eine gebrochene Gerade ist. Es ist anzumerken, dass die Gegenrichtung nicht notwendigerweise zutrifft.

Der untere Endpunkt $(\underline{t}^\rho, \underline{\beta}^\rho)$ eines Segments ρ einer gebrochenen Geraden wird als *unteres gebrochenes Paar* bezeichnet. Entsprechend wird ein oberer Endpunkt $(\bar{t}^\rho, \bar{\beta}^\rho)$ als *oberes gebrochenes Paar* bezeichnet. Die folgende Beobachtung ergibt sich direkt aus der Definition 11.

Beobachtung 1. *Die Steigung der Interpolation zweier Punkte auf einer gebrochenen Geraden ist durch α begrenzt.*

Diese Beobachtung gilt, da eine gebrochene Gerade die Kombination ihrer Abschnitte ist, die eine Steigung von α haben. Für zwei aufeinanderfolgende Abschnitte gilt außerdem, dass der Ladestand des oberen gebrochenen Paares des ersten Abschnitts gleich dem Ladestand des unteren gebrochenen Paares des zweiten Abschnitts ist, jedoch die Zeit des unteren Paares echt größer ist als die des oberen Paares. Nach der Einführung des Konzepts der Zeit-Batterie-Kurve und der gebrochenen Geraden wird ein dynamisches Programm zur Lösung des Entscheidungsproblems vorgestellt.

4.5.2 Dynamisches Programm

Das hier vorgestellte dynamische Programm kann das Entscheidungsproblem für eine gegebene Sequenz in Polynomialzeit lösen. Zusätzlich zur Lösung des Entscheidungsproblems kann das Programm leicht erweitert werden, um eine zulässige Lösung zurückzugeben, falls eine solche existiert. Das dynamische Programm wird im folgenden Unterkapitel im Branch-and-Price-Algorithmus verwendet. Zur Vereinfachung wird angenommen, dass der Roboter die Sequenz mit einem vollständig geladenen Akku von dem Depot aus startet, das dem ersten Kunden der Sequenz am nächsten liegt.

Bevor das dynamische Programm im Detail beschrieben wird, wird die zugrunde liegende Idee erläutert. Gegeben sind zwei aufeinanderfolgende Kunden aus einer Sequenz s . Der Algorithmus berechnet eine Zeit-Batterie-Kurve am zweiten Kunden basierend auf der Zeit-Batterie-Kurve am ersten Kunden. Dabei werden sowohl der direkte Weg zu den Depots als auch die Möglichkeit, mehrere Depots anzufahren, berücksichtigt. Dafür wird vom ersten Kunden aus das früheste Zeit-Batterie-Paar an jedem Depot berechnet. Auf der Basis dieser Paare wird die Zeit-Batterie-Kurve am zweiten Kunden berechnet. Der Pseudocode des dynamischen Programms und seiner Subroutine sind in Algorithmus 1 bzw. Algorithmus 2 dargestellt.

Die Eingabe des Algorithmus 1 ist eine Sequenz $s = (u_1, u_2, \dots, u_h)$ und die Menge der Depots D . Für beide Algorithmen wird angenommen, dass alle weiteren Parameter bereits global definiert sind. Die Rückgabe des Algorithmus ist die Entscheidung, ob eine zulässige Lösung für s existiert. In der Initialisierung wird das früheste Zeit-Batterie-Paar am ersten Kunden aus s berechnet (siehe Zeile 2). Dieses Paar stellt gleichzeitig die erste Zeit-Batterie-Kurve am ersten Kunden dar. Als Nächstes wird Algorithmus 2 für jeweils zwei aufeinanderfolgende Kunden aufgerufen, beginnend mit u_1 und u_2 (siehe Zeilen 6 bis 11).

Die Eingabe von Algorithmus 2 sind zwei Kunden $u_i, u_j \in s$, die Menge der Depots D und eine gebrochene Gerade für u_i . Im ersten Teil des Algorithmus werden die frühesten Zeit-Batterie-Paare an allen Depots berechnet, die direkt von u_i erreicht werden können (siehe Zeilen 2 bis 8). Dafür verwendet der Algorithmus das erste Zeit-Batterie-Paar von der gebrochenen Geraden am Kunden, sodass der Ladestand ausreichend ist, um das jeweilige Depot zu erreichen. Ein Depot gilt als erreichbar, wenn der höchste Ladestand auf der gebrochenen Geraden für u_i ausreicht, um das Depot zu erreichen.

Dann versucht der Algorithmus, die Ankunftszeitpunkte an den Depots zu minimieren (siehe Zeilen 10 bis 21). Ähnlich zum Dijkstra-Algorithmus wird die Menge der Depots, für die der früheste Ankunftszeitpunkt bekannt ist, iterativ erweitert. Es wird ein nicht betrachtetes Depot mit kleinstem Ankunftszeitpunkt gewählt. Für das Depot werden alle erreichbaren Depots betrachtet, und es wird versucht, den frühesten bekannten Ankunftszeitpunkt zu minimieren. Falls Ladevorgänge nötig sind, wird die Ladezeit minimal gehalten. Somit wird garantiert, dass die Ankunftszeitpunkte kleinstmöglich sind. Nach dieser Berechnung besitzt jedes Depot ein Zeit-Batterie-Paar, da der Subgraph der Depots bezogen auf die Akkukapazität zusammenhängend ist.

Algorithmus 1 Lösung des Entscheidungsproblems für Sequenz s

Eingabe: Sequenz $s = (u_1, \dots, u_h)$, Depots D
Ausgabe: s ist zulässig oder unzulässig

```

1:  $\# (\underline{t}_1^1, \underline{\beta}_1^1), (\bar{t}_1^1, \bar{\beta}_1^1)$  definieren die gebrochene Gerade für Kunde  $u_1$ 
2:  $(\underline{t}_1^1, \underline{\beta}_1^1), (\bar{t}_1^1, \bar{\beta}_1^1) \leftarrow (\max(a_1, d_1), B - d_1)$ 
3: falls  $\underline{t}_1^1 > b_1$  oder  $\underline{\beta}_1^1 < \frac{B}{2}$  dann
4:   Rückgabe unzulässig
5: ende falls
6: für alle  $i = 2$  bis  $h$  mache
7:   rufe Algorithmus 2 auf und erweitere von  $u_{i-1}$  zu  $u_i$ 
8:   falls Erweiterung unzulässig dann
9:     Rückgabe unzulässig
10:  ende falls
11: ende für alle
12: Rückgabe zulässig

```

Im nächsten Teil berechnet der Algorithmus die Menge D' . Diese Menge umfasst alle Depots u_k , für die ein Zeit-Batterie-Paar für u_j über die Kante $\{k, j\}$ berechnet werden kann (siehe Zeile 22). Falls D' leer ist, bricht der Algorithmus ab und gibt zurück, dass die Erweiterung nicht möglich ist (siehe Zeile 23). Für jedes Depot aus D' werden zwei Zeit-Batterie-Paare für u_j berechnet, siehe Zeilen 27 bis 30. Zum einen das Zeit-Batterie-Paar mit dem kleinsten Ankunftszeitpunkt für u_j , welches als *unteres temporäres Paar* bezeichnet wird und zum anderen das früheste Zeit-Batterie-Paar mit dem höchsten erreichbaren Ladestand, welches als *oberes temporäres Paar* bezeichnet wird. Diese temporären Paare werden zum Aufbau der gebrochenen Geraden verwendet. Dazu werden die Depots aus D' nach folgenden Kriterien sortiert (siehe Zeile 31):

- Aufsteigend nach dem frühestmöglichen Ankunftszeitpunkt für u_j ,
- Absteigend nach dem Ladestand zum frühestmöglichen Zeitpunkt,
- Absteigend nach dem höchstmöglichen Ladestand für u_j .

Die Sortierung erfolgt zuerst nach dem ersten Kriterium, bei Gleichheit nach dem zweiten Kriterium und bei weiterer Gleichheit nach dem dritten Kriterium. Sollte auch nach dem dritten Kriterium immer noch Gleichheit herrschen, dann sind die Abschnitte gleich und nur ein Depot muss betrachtet werden. In diesem Fall kann die Sortierung auf Basis der Indizes der Depots erfolgen.

Im letzten Teil des Algorithmus (siehe Zeilen 33 bis 43) wird die gebrochene Gerade für u_j berechnet und zurückgegeben. Der erste Abschnitt der gebrochenen Geraden ergibt sich aus dem ersten Depot der Sortierung und der linearen Interpolation der dazugehörigen unteren und oberen temporären Paare. Anschließend werden die Depots in der sortierten Reihenfolge betrachtet. Ein neuer Abschnitt wird hinzugefügt, falls der Ladestand des oberen temporären Paares den höchsten Ladestand der aktuellen gebrochenen Geraden überschreitet. Beim Hinzufügen eines Abschnitts muss das untere gebrochene Paar so angepasst werden, dass sein Ladestand dem höchsten Ladestand des vorangegangenen Abschnitts entspricht. Nach der Betrachtung jedes

Depots in der Sortierung wird die gebrochene Gerade für u_j zurückgegeben. Um einen besseren Einblick in den Algorithmus zu geben, wird im Folgenden ein Beispiel vorgestellt.

Algorithmus 2 Erweiterung von Kunde u_i zu Kunde u_j

Eingabe: Kunden $u_i, u_j \in s$, Depots D , gebrochene Gerade $(\underline{t}_i^\rho, \underline{\beta}_i^\rho), (\bar{t}_i^\rho, \bar{\beta}_i^\rho)$ mit $\rho = 1, \dots, \sigma$

- 1: $t_k \leftarrow \infty, \beta_k \leftarrow 0$ für alle $u_k \in D$
- 2: # Berechne Zeit-Batterie-Paare an den Depots direkt von Kunde u_i
- 3: **für alle** $u_k \in D$ mit $d_{ik} \leq \bar{\beta}_i^\sigma$ **mache**
- 4: $\tilde{\rho} \leftarrow \min\{\rho : \bar{\beta}_i^\rho \geq d_{ik}, \rho \in \{1, \dots, \sigma\}\}$
- 5: $\lambda \leftarrow \frac{1}{\alpha} \min(0, d_{ik} - \underline{\beta}_i^{\tilde{\rho}})$
- 6: $t_k \leftarrow \underline{t}_i^{\tilde{\rho}} + \lambda + \tau_i + d_{ik}$
- 7: $\beta_k \leftarrow \underline{\beta}_i^{\tilde{\rho}} + \alpha\lambda - d_{ik}$
- 8: **ende für alle**
- 9: # Depot-zu-Depot-Evaluierung
- 10: $\tilde{D} \leftarrow \emptyset$
- 11: **während** $\tilde{D} \neq D$ **mache**
- 12: $k \leftarrow \arg \min\{t_l : u_l \in D \setminus \tilde{D}\}$
- 13: $\tilde{D} \leftarrow \tilde{D} \cup \{u_k\}$
- 14: **für alle** $u_l \in D \setminus \tilde{D}$ mit $d_{kl} \leq B$ **mache**
- 15: $\lambda \leftarrow \frac{1}{\alpha} \max(0, d_{kl} - \beta_k)$
- 16: **falls** $t_k + \lambda + d_{kl} < t_l$ **dann**
- 17: $t_l \leftarrow t_k + \lambda + d_{kl}$
- 18: $\beta_l \leftarrow \beta_k + \alpha\lambda - d_{kl}$
- 19: **ende falls**
- 20: **ende für alle**
- 21: **ende während**
- 22: $D' \leftarrow \{u_k \in D : t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj} \leq b_j, d_{kj} + d_j \leq B\}$
- 23: **falls** D' leer ist **dann**
- 24: **Rückgabe** Erweiterung unzulässig
- 25: **ende falls**
- 26: # Berechnung unterer und oberer temporärer Paare
- 27: $\underline{t}_k \leftarrow \max(a_j, t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj})$ für alle $u_k \in D'$
- 28: $\bar{t}_k \leftarrow \max(\underline{t}_k, \min(b_j, t_k + \frac{1}{\alpha}(B - \beta_k) + d_{kj}))$ für alle $u_k \in D'$
- 29: $\underline{\beta}_k \leftarrow \min(B - d_{kj}, \beta_k + \alpha(\underline{t}_k - t_k - d_{kj}) - d_{kj})$ für alle $u_k \in D'$
- 30: $\bar{\beta}_k \leftarrow \min(B - d_{kj}, \beta_k + \alpha(\bar{t}_k - t_k - d_{kj}) - d_{kj})$ für alle $u_k \in D'$
- 31: Sortiere in aufsteigender Reihenfolge nach $\underline{t}_k$, bei Gleichheit in absteigender Reihenfolge nach $\underline{\beta}_k$, bei Gleichheit in absteigender Reihenfolge nach $\bar{\beta}_k$
- 32: Ersetze Indizes der Depots in D' basierend auf Sortierung von 1 bis $|D'|$
- 33: # Aufbau der gebrochenen Geraden für Kunde u_j
- 34: $(\underline{t}_j^1, \underline{\beta}_j^1) \leftarrow (\underline{t}_1, \underline{\beta}_1), (\bar{t}_j^1, \bar{\beta}_j^1) \leftarrow (\bar{t}_1, \bar{\beta}_1)$
- 35: $\beta \leftarrow \bar{\beta}_1, \rho \leftarrow 2$
- 36: **für alle** $k = 2$ bis $|D'|$ **mache**
- 37: **falls** $\bar{\beta}_k > \beta$ **dann**
- 38: $(\underline{t}_j^\rho, \underline{\beta}_j^\rho) \leftarrow (\underline{t}_k + \frac{1}{\alpha}(\beta - \underline{\beta}_k), \beta)$
- 39: $(\bar{t}_j^\rho, \bar{\beta}_j^\rho) \leftarrow (\bar{t}_k, \bar{\beta}_k)$
- 40: $\beta \leftarrow \bar{\beta}_k$
- 41: $\rho \leftarrow \rho + 1$
- 42: **ende falls**
- 43: **ende für alle**
- 44: **Rückgabe** Gebrochene Gerade für Kunde u_j

Beispiel 2. Gegeben sind die Sequenz $s = (u_1, u_2, u_3)$, die Depotmenge $D = (\tilde{u}_1, \tilde{u}_2, \tilde{u}_3)$, eine Akkukapazität von $B = 10$ und eine Laderate von $\alpha = 1$. Die Fahrzeiten zwischen Kunden und Depots sind in Tabelle 4.2 dargestellt. Zeitfenster und Servicezeiten bleiben unberücksichtigt.

	$\tilde{u}_1$	$\tilde{u}_2$	$\tilde{u}_3$	u_1	u_2	u_3
$\tilde{u}_1$	0	5	3	3	7	3
$\tilde{u}_2$	5	0	3	4	4	4,5
$\tilde{u}_3$	3	3	0	6	3	5

Tabelle 4.2: Fahrzeiten zwischen den Depots und zwischen Depots und Kunden

Zu Beginn initialisiert Algorithmus 1 für den Kunden u_1 jeweils ein unteres und oberes gebrochene Paar. Beide Paare sind identisch und lauten $(3, 7)$. Sie bilden zugleich die erste gebrochene Gerade, dargestellt in Abbildung 4.5(b). In Abbildung 4.5(a) sind alle möglichen Ankunftszeitpunkte mit zugehörigem Ladestand abgebildet. Demnach kann u_1 ab Zeitpunkt 3 mit Ladestand 7 von Depot $\tilde{u}_1$ oder zu späteren Zeitpunkten auch von $\tilde{u}_2$ bzw. $\tilde{u}_3$ beliefert werden. Alle diese Optionen werden jedoch durch das Paar $(3, 7)$ dominiert (siehe Definition 8).

Auf Basis der ersten gebrochenen Geraden berechnet Algorithmus 2 eine gebrochene Gerade für Kunde u_2 . Zunächst werden die frühesten Ankunftszeitpunkte an allen von u_1 erreichbaren Depots bestimmt (Zeilen 2–8). Da alle Depots höchstens sechs Zeiteinheiten entfernt sind, wird je ein Zeit-Batterie-Paar berechnet: $(6, 4)$ für $\tilde{u}_1$, $(7, 3)$ für $\tilde{u}_2$ und $(9, 1)$ für $\tilde{u}_3$. In Zeilen 10–21 wird geprüft, ob ein früherer Zugang über andere Depots möglich ist. Dies ist hier nicht der Fall.

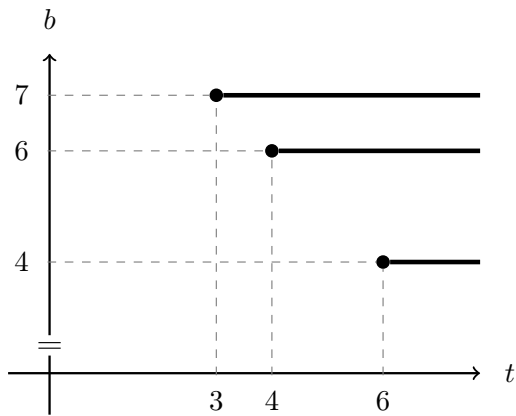
Zunächst wird die Menge D' bestimmt (Zeile 22). Sie enthält alle Depots, von denen aus ein Zeit-Batterie-Paar für u_2 berechnet werden kann, d.h. eine Ankunft im Zeitfenster mit ausreichendem Ladestand für die Weiterfahrt zum nächstgelegenen Depot ist möglich. Es sind alle Depots in D' .

Im nächsten Schritt werden die temporären Paare von jedem Depot aus D' für u_2 berechnet (siehe Zeilen 27 bis 30). Das untere temporäre Paar beschreibt die früheste Ankunft, das obere temporäre die früheste Ankunft mit maximalem Ladestand (jeweils unter Einhaltung des Zeitfensters und der Erreichbarkeit des nächsten Depots). Die resultierenden Paare lauten: $(19, 3)$ und $(19, 3)$ von $\tilde{u}_1$, $(15, 3)$ und $(18, 6)$ von $\tilde{u}_2$, $(17, 3)$ und $(21, 7)$ von $\tilde{u}_3$.

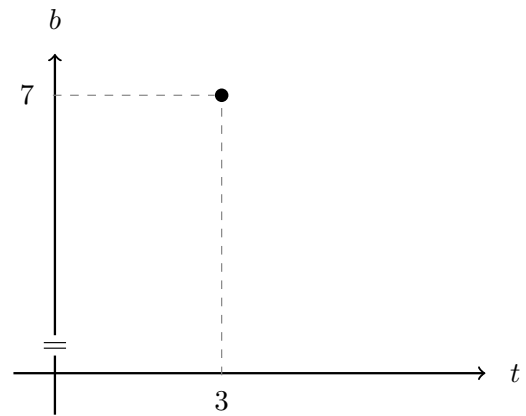
Allerdings kann durch partielles Laden an den Depots auch jedes Zeit-Batterie-Paar innerhalb der konvexen Hülle gewählt werden, das aus dem unteren und oberen temporären Paar besteht. Alle diese Paare sind in Abbildung 4.5(c) dargestellt.

Die Paare werden sortiert, um die gebrochene Gerade für u_2 zu berechnen (siehe Zeile 31), wobei sich die Reihenfolge $\tilde{u}_2, \tilde{u}_3, \tilde{u}_1$ ergibt. Die Berechnung der gebrochenen Gerade erfolgt in Zeile 33 bis 43 und das Ergebnis ist in Abbildung 4.5(d) dargestellt. Die Paare von $\tilde{u}_1$ entfallen, da sie von $\tilde{u}_2$ dominiert werden. Von $\tilde{u}_3$ werden nur Paare mit Ladestand über 6 berücksichtigt. Die resultierende gebrochene Gerade wird an Algorithmus 1 übergeben, der daraufhin die gebrochene Gerade für u_3 berechnet.

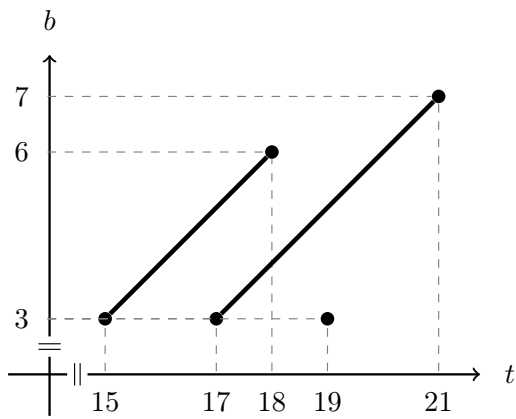
Die frühesten Zeit-Batterie-Paare an den Depots von u_2 sind $(28, 0)$ für $\tilde{u}_1$, $(20, 0)$ für $\tilde{u}_2$ und $(18, 0)$ für $\tilde{u}_3$. Im nächsten Schritt wird versucht, die Ankunftszeitpunkte durch die Betrachtung



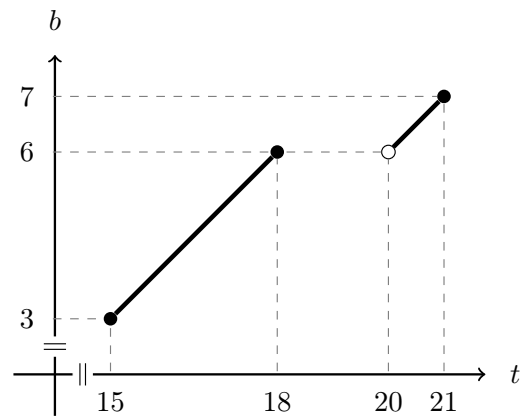
(a) Alle Kombinationen aus Ankunftszeit und Lade-stand für u_1



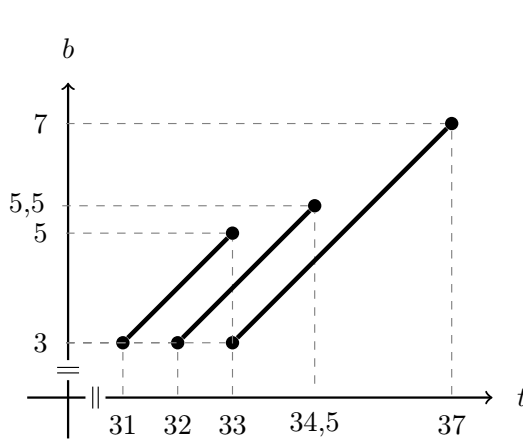
(b) Gebrochene Gerade für u_1



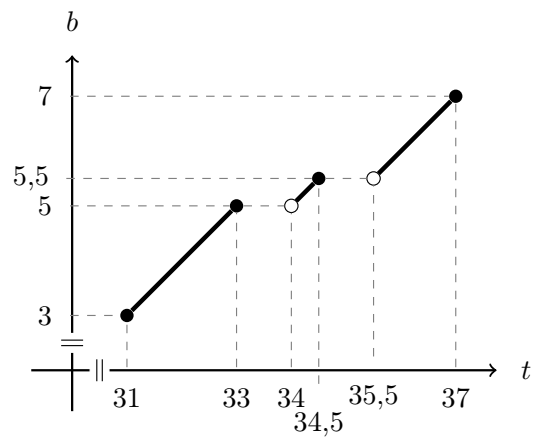
(c) Alle Kombinationen aus Ankunftszeit und Lade-stand für u_2



(d) Gebrochene Gerade für u_2



(e) Alle Kombinationen aus Ankunftszeit und Lade-stand für u_3



(f) Gebrochene Gerade für u_3

Abbildung 4.5: Vergleich zwischen temporären Paaren und der resultierenden gebrochenen Geraden

von Depot-Pfaden zu verringern. Der Ankunftszeitpunkt für $\tilde{u}_1$ kann über $\tilde{u}_3$ verringert werden und das neue Zeit-Batterie-Paar für $\tilde{u}_1$ lautet $(24, 0)$.

Es folgt die Berechnung der Menge D' , die alle Depots umfasst. Die unteren und oberen temporären Paare für u_3 lauten: $(33, 3)$ und $(37, 7)$ von $\tilde{u}_1$, $(32, 3)$ und $(34, 5)$ von $\tilde{u}_2$ sowie $(31, 3)$ und $(33, 5)$ von $\tilde{u}_3$. Diese sind in Abbildung 4.5(e) dargestellt.

Anschließend werden die Depots sortiert, was zur Reihenfolge $\tilde{u}_3, \tilde{u}_2, \tilde{u}_1$ führt. Daraufhin wird die gebrochene Gerade für u_3 berechnet. Diese ist in Abbildung 4.5(f) dargestellt. Algorithmus 1 erhält diese zurück und bestätigt eine zulässige Lösung. Damit ist das Beispiel zu Algorithmus 1 und 2 abgeschlossen.

Es bleibt zu zeigen, dass Algorithmus 1 korrekt ist. Alle Beweise und Hilfsaussagen sind im Anhang A zu finden. Wie bereits erwähnt, ist die Berechnung der Zeit-Batterie-Kurven und der gebrochenen Geraden ein Schlüsselkonzept für den Korrektheitsbeweis. Daher wird vorab gezeigt, dass Algorithmus 1 zusammen mit Algorithmus 2 für jeden Kunden der Sequenz eine gebrochene Gerade berechnet.

Lemma 1. *Sei s eine Sequenz, die Algorithmus 1 als zulässig bewertet. Dann gilt, dass Algorithmus 1 zusammen mit Algorithmus 2 an jedem Kunden aus s eine gebrochene Gerade, wie in Definition 11 formuliert, berechnet.*

Der Beweis wird mittels Induktion geführt. Um die Korrektheit des dynamischen Programms zu beweisen, wird zunächst gezeigt, dass es möglich ist, eine zulässige Lösung zu erzeugen, falls der Algorithmus eine Sequenz als zulässig bewertet.

Lemma 2. *Falls Algorithmus 1 eine Sequenz s als zulässig bewertet, dann existiert eine zulässige Lösung.*

Es ist zu beachten, dass eine zulässige Lösung, falls vorhanden, durch eine einfache Erweiterung von Algorithmus 1 sowie Algorithmus 2 zurückgegeben werden kann. Es bleibt zu zeigen, dass keine zulässige Lösung existiert, wenn der Algorithmus eine Sequenz als *unzulässig* bewertet.

Lemma 3. *Falls Algorithmus 1 eine Sequenz s als unzulässig bewertet, dann existiert keine zulässige Lösung für s .*

Zum Beweis dieser Aussage wird zunächst gezeigt, dass Algorithmus 1 zusammen mit Algorithmus 2 an jedem Kunden eine Zeit-Batterie-Kurve gemäß Definition 10 berechnet. Diese Behauptung wird mittels vollständiger Induktion bewiesen. Der Induktionsanfang wird wie folgt formuliert.

Proposition 1. *Gegeben ist eine Sequenz s für die eine zulässige Lösung existiert. Algorithmus 1 berechnet eine Zeit-Batterie-Kurve am ersten Kunden aus s .*

Die folgenden beiden Lemmata bilden den Induktionsschritt. Zuerst wird gezeigt, dass, falls eine Zeit-Batterie-Kurve an Algorithmus 2 übergeben wird, dann wird jedes Depot, welches in einer beliebigen Lösung auf den aktuell betrachteten Kunden folgt, dominiert.

Lemma 4. *Gegeben sind eine Sequenz s , zwei aufeinanderfolgende Kunden u_i, u_j aus s und eine zulässige Lösung π zu s . Sei u_k ein Depot, welches in π zwischen u_i und u_j besucht wird und sei (t_k, b_k) das Zeit-Batterie-Paar für u_k in π . Falls Algorithmus 1 oder 2 eine Zeit-Batterie-Kurve für u_i berechnet, gilt, dass Algorithmus 2 ein Zeit-Batterie-Paar für u_k berechnet, das (t_k, b_k) dominiert.*

Um den Induktionsschritt abzuschließen, muss gezeigt werden, dass eine Zeit-Batterie-Kurve am zweiten Kunden berechnet wird, wenn eine solche Kurve am ersten Kunden gegeben ist.

Lemma 5. *Gegeben sind eine Sequenz s und eine zulässige Lösung π zu s . Seien u_i und u_j zwei aufeinanderfolgende Kunden in s . Falls Algorithmus 1 oder 2 eine Zeit-Batterie-Kurve für u_i berechnet, dann gilt, dass Algorithmus 2 eine Zeit-Batterie-Kurve für u_j berechnet.*

Es bleibt zu zeigen, dass die Laufzeit des Algorithmus 1 zusammen mit Algorithmus 2 polynomiell in der Eingabe beschränkt ist.

Lemma 6. *Sei s eine Sequenz mit $|s|$ Kunden und sei $|D|$ die Anzahl der Depots. Die Laufzeit von Algorithmus 1 lässt sich asymptotisch abschätzen durch $O(|s| \cdot |D|^2)$.*

Die Korrektheit des dynamischen Programms wird im folgenden Theorem formuliert.

Theorem 1. *Algorithmus 1 löst das Entscheidungsproblem, ob zu einer gegebenen Sequenz eine zulässige Lösung existiert, in Polynomialzeit. Falls eine zulässige Lösung existiert, dann kann diese in Polynomialzeit berechnet werden.*

Beweis. Lemma 2 besagt, dass, falls Algorithmus 1 *zulässig* zurückgibt, eine zulässige Lösung erzeugt werden kann. Lemma 3 besagt, dass, falls Algorithmus 1 *unzulässig* zurückgibt, keine zulässige Lösung existieren kann. Die polynomielle Laufzeit wird mit Lemma 6 gezeigt. $\square$

Aus Theorem 1 folgt das folgende Lemma.

Lemma 7. *Jede Zeit-Batterie-Kurve ist eine gebrochene Gerade.*

Beweis. Sei s eine Sequenz, für die eine zulässige Lösung existiert. Lemma 1 zeigt, dass Algorithmus 1 zusammen mit Algorithmus 2 an jedem Kunden aus s eine gebrochene Gerade berechnet. Mittels Induktion wurde gezeigt, dass Algorithmus 1 und 2 an jedem Kunden aus s eine Zeit-Batterie-Kurve berechnen. Damit folgt, dass Zeit-Batterie-Kurven gebrochene Geraden sind. $\square$

Abgeschlossen wird dieses Unterkapitel mit dem Verweis auf einen Algorithmus, welcher die Sequenzen von hinten nach vorne, bezogen auf die Reihenfolge, betrachtet und der Idee ,die Transportkapazität zu erhöhen.

Neben der Betrachtung einer Sequenz vom ersten bis zum letzten Kunden ist es ebenfalls möglich, die Sequenz in umgekehrter Reihenfolge zu betrachten und mit dem letzten Kunden der Sequenz zu beginnen. Die rückwärtsgerichtete Evaluierung einer Sequenz wird in Anhang B beschrieben.

In diesem Kapitel wird angenommen, dass die Transportkapazität der Zustellroboter eine Sendung beträgt und nach jeder Zustellung eine Rückkehr zum Mikrodepot erforderlich ist. Eine interessante Frage ist, ob sich Algorithmus 1 und 2 so modifizieren lassen, dass auch größere Transportkapazitäten abgebildet werden können. Bei Aufhebung dieser Beschränkung wird eine direkte Fahrt von einem Kunden zum nächsten möglich. Diese Option muss bei der Konstruktion der gebrochenen Geraden berücksichtigt werden. Dazu müsste eine gebrochene Gerade vom aktuellen zum nächsten Kunden übertragen und um die entsprechende Fahrzeit sowie den Akkuverbrauch verschoben werden, wobei unzulässige Paare entfernt werden müssten. Die neuen Abschnitte, welche über die Depots hinzugefügt werden, müssten in die übertragene, gebrochene Gerade integriert werden. Dies würde einen exakten Algorithmus für eine Variante des FRVCP mit Zeitfenstern und linearen Ladefunktionen ermöglichen, der in Polynomialzeit arbeitet.

4.6 Branch-and-Price-Algorithmus

Dieses Unterkapitel stellt einen Branch-and-Price-Algorithmus zur Optimierung eines Zustellnetzwerks mit Robotern vor. Der grundsätzliche Aufbau dieses Verfahrens wurde bereits in Unterkapitel 3.2 vorgestellt. Mit diesem Verfahren wird das in Unterkapitel 4.3 beschriebene Problem gelöst. Dazu wird das Problem im Folgenden als IP formuliert. Anschließend wird das dazugehörige duale Programm eingeführt. Auf Basis des dualen Programms wird das Pricing-Problem definiert und ein Labeling-Algorithmus zur Lösung des Pricing-Problems präsentiert. Das Kapitel schließt mit der Vorstellung einer Branching-Strategie sowie verschiedener Beschleunigungsverfahren für den Branch-and-Price-Algorithmus. Die Ergebnisse des Algorithmus werden in Kapitel 6 vorgestellt.

4.6.1 Ganzzahliges lineares Programm und Pricing-Problem

Im Folgenden wird das Problem aus Unterkapitel 4.3 als IP formuliert. Ähnlich zu Unterkapitel 4.5 werden feste Sequenzen betrachtet, wobei die Depots und die entsprechenden Ladezeiten zwischen den Kunden nur implizit berücksichtigt werden. Sei dazu Ω die Menge aller zulässigen Sequenzen. Die Entscheidung über die Zulässigkeit einer Sequenz kann mit Algorithmus 1 getroffen werden. Sei $x_s, s \in \Omega$, eine binäre Variable, die angibt, ob eine zulässige Sequenz von einem der m Roboter durchgeführt wird. Damit lässt sich das Problem als Mengenpackungsproblem mit einer zusätzlichen Nebenbedingung beschreiben:

$$\begin{array}{ll} \text{maximiere} & \sum_{s \in \Omega} \sum_{u_i \in C} e_{is} p_i x_s \end{array} \quad (4.18)$$

$$\begin{array}{ll} \text{u.d.N.} & \sum_{s \in \Omega} e_{is} x_s \leq 1 \quad \forall u_i \in C, \end{array} \quad (4.19)$$

$$\sum_{s \in \Omega} x_s \leq m, \quad (4.20)$$

$$x_s \in \{0, 1\} \quad \forall s \in \Omega. \quad (4.21)$$

Die Zielfunktion (4.18) maximiert den gesammelten Profit über alle Roboter. Der Parameter e_{is} ist eins, falls Kunde $u_i \in C$ in Sequenz $s \in \Omega$ enthalten ist. Die Nebenbedingungen (4.19) stellen sicher, dass kein Kunde mehrfach beliefert wird. Durch Nebenbedingung (4.20) ist garantiert, dass die Anzahl der verfügbaren Roboter nicht überschritten wird. Die Binarität der Variablen ist mit Bedingungen (4.21) sichergestellt. Die Relaxierung von (4.18)-(4.21) ist das MP.

Zur Ausführung des Branch-and-Price-Algorithmus werden die Werte der dualen Variablen benötigt. Daher wird das duale Programm zusätzlich vorgestellt. Dazu wird das MP dualisiert. Es ist nicht notwendig, die binären Variablen nach oben durch 1 zu begrenzen, da dies bereits mit Nebenbedingung (4.19) gewährleistet wird. Falls eine Variable in keiner der Nebenbedingungen (4.19) enthalten ist, kann sie auch nicht in der Zielfunktion vorkommen. Das Problem besitzt für jeden Kunden $u_i \in C$ eine Variable y_i und für die Nebenbedingung (4.20) eine Variable, welche mit $\tilde{y}$ bezeichnet wird. Damit lässt sich das duale Problem als folgendes LP formulieren:

$$\text{minimiere} \quad \sum_{u_i \in C} y_i + m\tilde{y} \quad (4.22)$$

$$\text{u.d.N.} \quad \sum_{u_i \in C} e_{is}y_i + \tilde{y} \geq \sum_{u_i \in C} e_{is}p_i \quad \forall s \in \Omega, \quad (4.23)$$

$$y_i, \tilde{y} \geq 0 \quad \forall u_i \in C. \quad (4.24)$$

Die Zielfunktion (4.22) minimiert die Summe der Werte der dualen Variablen. Für jede Sequenz $s \in \Omega$ besitzt das duale Problem eine Nebenbedingung (4.23). Die Nichtnegativität der Variablen ist mit Bedingung (4.24) gewährleistet.

Da die Anzahl der Sequenzen möglicherweise exponentiell in der Anzahl der Kunden wächst, wird nur eine Teilmenge der Sequenzen betrachtet, das heißt $\Omega' \subseteq \Omega$. Es ist zu beachten, dass Ω' auch die leere Menge sein kann. Durch Reduzierung der Variablen auf die Menge Ω' wird das EMP gebildet. Allerdings erfordert die Beschränkung der Menge Ω die Lösung des Pricing-Problems. Dazu wird eine optimale Lösung des EMPs genommen und die Werte der dualen Variablen betrachtet. Auf Basis der Werte bewertet das Pricing-Problem die Entscheidungsvariablen für das EMP, welche noch nicht in Ω' enthalten sind, und überprüft, ob weitere Variablen für eine optimale Lösung des MP betrachtet werden müssen.

Eine Einführung in das allgemeine Pricing-Problem findet sich in Unterkapitel 3.2. Sei $\mathbf{y}^* = (y_1^*, \dots, y_{|C|}^*, \tilde{y}^*)$ eine optimale Lösung für das LP (4.22)-(4.24). Dann ergeben sich die reduzierten Kosten einer Variable x_s als

$$\bar{c}(s, \mathbf{y}^*) = \sum_{u_i \in C} e_{is}(p_i - y_i^*) - \tilde{y}^*. \quad (4.25)$$

Eine optimale Lösung ist definiert als $\bar{c}^*(\mathbf{y}^*) = \max_{s \in \Omega} \bar{c}(s, \mathbf{y}^*)$. Zur Berechnung dieser wird das Problem auf ein Kürzeste-Wege-Problem mit Beschränkung der Ressourcen reduziert (siehe Unterkapitel 3.3) und mit einem Labeling-Algorithmus gelöst.

4.6.2 Labeling-Algorithmus

Die Idee des Labeling-Algorithmus besteht darin, jede mögliche Sequenz basierend auf der optimalen dualen Lösung des EMPs zu betrachten und diese auf Zulässigkeit sowie reduzierte Kosten zu bewerten. Dabei werden gleichzeitig vorwärtsgerichtete Teilwege von einer Quelle und rückwärtsgerichtete Teilwege von einer Senke generiert. Dieses Verfahren wird als *bidirectional labeling search* bezeichnet (Righini und Salani, 2006).

Sei $\mathbf{y}^* = (y_1^*, \dots, y_{|C|}^*, \tilde{y}^*)$ eine optimale Lösung zu (4.22)-(4.24). Zur Beschreibung der Suche erfolgt die Einführung eines Hilfsgraphen $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. Die Knotenmenge ist definiert als $\mathcal{V} = FW \cup BW \cup \{w_0, w_{n+1}\}$. Hierbei ist $FW = \{u_i : u_i \in C, a_i \leq t^*\}$ die Menge der vorwärtsgerichteten Knoten und $BW = \{u_i : u_i \in C, t^* \leq b_i\}$ die Menge der rückwärtsgerichteten Knoten, wobei $FW \cup BW = C$ gilt. Seien w_0 und w_{n+1} die Quelle bzw. Senke der Wege. Der Parameter t^* wird so berechnet, dass die größere der beiden Mengen balanciert ist. Sei $\mathcal{A} = \{(i, j) : u_i, u_j \in \mathcal{V}\}$ die Kantenmenge von $\mathcal{G}$. Es ist zu beachten, dass in $\mathcal{G}$ die Depots nicht enthalten sind. Stattdessen berechnet Algorithmus 2 Depot-Pfade mit entsprechenden Ladezeiten zwischen den Kunden.

Jeder Kunde $u_i \in C$ hat Kosten in Höhe von $p_i - y_i^*$, verfügt über ein Zeitfenster $[a_i, b_i]$ und hat eine Servicezeit τ_i . Der Knoten w_0 hat Kosten in Höhe von $-\tilde{y}^*$. Das Zeitfenster von w_0 und w_{n+1} ist $[0, T]$, wobei der Zeithorizont T als nicht restriktiv angenommen wird. Die Fahrzeit eines Kunden $u_i \in C$ zur Quelle oder Senke, d_i , entspricht der Fahrzeit vom Kunden zum nächstgelegenen Depot. Die Fahrzeit zwischen zwei Kunden hängt vom gewählten Depot-Pfad ab. Alle weiteren Parameter können aus der Problemstellung 4.3 übertragen werden. Für die Berechnung eines kürzesten Weges wird zuerst eine Definition eines Weges gegeben.

Definition 12. *Ein Weg $W = (w_0, u_1, u_2, \dots, u_h, w_{n+1})$ beschreibt eine Reihenfolge von Kunden, wobei $u_1, u_2, \dots, u_h \in C$ gilt. Ein Weg startet an der Quelle w_0 und endet an der Senke w_{n+1} .*

Ein Weg wird als zulässig bezeichnet, wenn Algorithmus 1 die enthaltene Sequenz s als zulässig bewertet. Die Kosten eines Weges W ergeben sich aus der Summe der Kosten der enthaltenen Knoten und berechnen sich als

$$\bar{c}_W(\mathbf{y}^*) = \sum_{u_i \in W} (p_i - y_i^*) - \tilde{y}^*.$$

Es ist zu beachten, dass die Länge des kürzesten Weges in $\mathcal{G}$ dem Lösungswert des Pricing-Problems entspricht. Ein bidirektionaler Labeling-Algorithmus wird zur Lösung des Kürzeste-Wege-Problems mit Beschränkung der Ressourcen eingesetzt.

Der bidirektionale Algorithmus setzt sich aus einem vorwärts- und einem rückwärtsgerichteten Labeling-Algorithmus zusammen. Beide Algorithmen generieren Teilwege, die in einem Vereinigungsschritt zusammengeführt werden. Ziel der beidseitigen Suche ist es, die Anzahl der zu untersuchenden Wege zu minimieren. Der vorwärtsgerichtete Labeling-Algorithmus untersucht zulässige Teilwege W^{fw} . Diese vorwärtsgerichteten Teilwege beginnen in w_0 , besuchen eine Men-

ge von Kunden aus der Menge FW und enden in einem Knoten in $\mathcal{V}$. Der rückwärtsgerichtete Labeling-Algorithmus untersucht zulässige Teilwege W^{bw} in umgekehrter Reihenfolge. Diese starten an der Senke w_{n+1} und besuchen eine Menge von Kunden in BW .

Ein Label wird definiert als $L = (u_i, W, \bar{c}_W(\mathbf{y}^*), \underline{\mathbf{t}}, \bar{\mathbf{t}}, \underline{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}})$, wobei $u_i \in \mathcal{V}$ das aktuelle Ende des Teilwegs beschreibt, W den Teilweg, $\bar{c}_W(\mathbf{y}^*)$ die reduzierten Kosten des Teilwegs und die Vektoren $\underline{\mathbf{t}}, \bar{\mathbf{t}}, \underline{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}}$ die gebrochene Gerade für u_i für die enthaltene Sequenz in Teilweg W . Der vorwärtsgerichtete Algorithmus wird mit dem Label $L_0^{fw} = (w_0, (w_0), -\tilde{y}^*, (0), (0), (B), (B))$ und der rückwärtsgerichtete Algorithmus mit dem Label $L_0^{bw} = (w_{n+1}, (w_{n+1}), 0, (T), (T), (0), (0))$ initialisiert. Beide Algorithmen verfolgen das Ziel, in jeder Iteration alle noch nicht erweiterten Labels zu spezifischen Knoten zu erweitern. Dies bedeutet, dass der vorwärtsgerichtete Algorithmus Labels zu allen zulässigen Knoten in $\mathcal{V}$ erweitert und der rückwärtsgerichtete Algorithmus zu allen zulässigen Knoten in BW . Der vorwärtsgerichtete Algorithmus stoppt die Erweiterung eines Labels, falls der Endknoten des Teilwegs in $BW \setminus FW$ liegt oder der Senke w_{n+1} entspricht.

Der Pseudocode der vorwärtsgerichteten Suche ist in Algorithmus 3 dargestellt. Die Eingabe des Algorithmus ist Graph $\mathcal{G}$. Alle weiteren Parameter, wie die maximale Fahrzeit des Akkus oder der Ladefaktor, werden global definiert. Die Initialisierung des Labels L_0^{fw} erfolgt in Zeile 1. Das Label L_0^{fw} wird zur Menge der noch nicht erweiterten Labels $\mathcal{L}$ und zur Menge $\mathcal{K}$ hinzugefügt. Im Anschluss werden die Teilwege der Labels aus der Menge $\mathcal{L}$ zu Kunden erweitert, die die folgenden Kriterien erfüllen:

- Der Kunde ist nicht im Teilweg des Labels enthalten,
- Die Kante zwischen dem Endknoten des Teilwegs und dem nächsten Kunden ist in $\mathcal{A}$,
- Algorithmus 2 bewertet die Erweiterung zum nächsten Kunden als zulässig.

Wenn alle Kriterien erfüllt sind, wird ein neues Label erzeugt. Es wird in $\mathcal{K}$ gespeichert, wenn der aktuelle Knoten der Senke entspricht. Anzumerken ist, dass Algorithmus 2 für die Senke w_{n+1} nicht aufgerufen werden muss. Liegt der neue Endknoten des Weges in der Menge FW , wird das Label zu $\mathcal{L}$ hinzugefügt. Da der rückwärtsgerichtete Algorithmus ähnlich abläuft, wird auf die Ausführung des dazugehörigen Pseudocodes verzichtet.

Eine Übersicht aller erforderlichen Schritte des vorwärts- und rückwärtsgerichteten Algorithmus ist in Tabelle 4.3 dargestellt. Zur Vereinfachung wird die Menge X definiert, die alle Knoten eines Teilwegs umfasst. Dabei ist die Tabelle in die folgenden drei Zeilen unterteilt: In der Zeile *Zulässigkeit* wird entschieden, ob eine Erweiterung eines vorwärtsgerichteten Labels L^{fw} zu einem Knoten zulässig ist. Die gleiche Entscheidung wird für ein rückwärtsgerichtetes Label L^{bw} getroffen. Für beide Algorithmen gilt, dass der entsprechende Knoten noch nicht besucht sein darf, eine Kante existieren muss und Algorithmus 2 bzw. 8 die Erweiterung als zulässig bewertet. Hierbei bezeichnet BS die gebrochene Gerade, die durch Algorithmus 2 bzw. 8 berechnet wird. In der zweiten Zeile werden die Regeln für die Erweiterung definiert. Dabei wird für die vorwärtsgerichtete Routine ein Label L'^{fw} erzeugt und dessen entsprechende Komponenten defi-

Algorithmus 3 Vorwärtsgerichteter Algorithmus

Eingabe: Hilfsgraph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$
Ausgabe: Menge von Labeln $\mathcal{K}$

```

1:  $\mathcal{L}, \mathcal{K} \leftarrow (w_0, (w_0), -\tilde{y}^*, (0), (0), (B), (B))$ 
2: während  $\mathcal{L} \neq \emptyset$  mache
3:    $(u_i, W, \bar{c}_W(\mathbf{y}^*), \underline{\mathbf{t}}, \bar{\mathbf{t}}, \underline{\boldsymbol{\beta}}, \bar{\boldsymbol{\beta}}) \leftarrow$  entferne ein Label aus  $\mathcal{L}$ 
4:   für alle  $u_j \in \mathcal{V}$  mache
5:     falls  $u_j \notin X$  und  $(i, j) \in \mathcal{A}$  und Algorithmus 2 ist zulässig für Erweiterung von  $u_i$  nach
        $u_j$  basierend auf gebrochenen Geraden für  $u_i$  dann
6:       erzeuge neues Label  $L$ 
7:       falls  $u_i = w_{n+1}$  dann
8:          $\mathcal{K} \leftarrow L$ 
9:       ende falls
10:      falls  $u_i \in FW$  dann
11:         $\mathcal{L} \leftarrow L$ 
12:      ende falls
13:    ende falls
14:  ende für alle
15: ende während

```

niert. Gleiches gilt für den rückwärtsgerichteten Algorithmus. Die gebrochene Gerade wird dabei mit Algorithmus 2 bzw. Algorithmus 8 berechnet.

Nach der Terminierung der beiden Labeling-Algorithmen wird die Vereinigung der gefundenen Labels durchgeführt. Die Regeln der Vereinigung sind in der letzten Zeile der Tabelle definiert. Die Vereinigung eines vorwärts- und rückwärtsgerichteten Labels ist zulässig, wenn die folgenden drei Bedingungen erfüllt sind:

- Beide Label befinden sich am gleichen aktuellen Knoten,
- Der einzige Knoten, der in beiden Teilwegen enthalten ist, ist der aktuelle Knoten,
- Auf der gebrochenen Geraden des vorwärtsgerichteten Labels existiert ein Zeit-Batterie-Paar, das nicht später ist und einen Ladestand aufweist, der mindestens so groß ist wie bei einem Zeit-Batterie-Paar auf der gebrochenen Geraden des rückwärtsgerichteten Labels.

Damit sind die Labeling-Algorithmen beschrieben. Im nächsten Unterkapitel werden Branching-Regeln für den Branch-and-Price-Algorithmus vorgestellt und im Anschluss werden Verfahren betrachtet, welche die Laufzeit des Branch-and-Price-Algorithmus reduzieren sollen.

4.6.3 Branching-Regeln

Eine Möglichkeit zur Verringerung der Laufzeit besteht in der Betrachtung der Regeln des Branching im Branch-and-Price-Algorithmus. Im Folgenden werden zwei Branching-Regeln vorgestellt. Ein naiver Ansatz besteht darin, das Branching direkt auf den binären Variablen x_s des EMP durchzuführen. Dies würde dazu führen, dass eine auf eins gesetzte Variable eine sehr

	Vorw. ger. Label $L^{fw} =$ $(u_i, W^{fw}, \bar{c}_{W^{fw}}^{fw}(\mathbf{y}^*),$ $\underline{t}^{fw}, \bar{t}^{fw}, \underline{\beta}^{fw}, \bar{\beta}^{fw})$	Rückw. ger. Label $L^{bw} =$ $(u_i, W^{bw}, \bar{c}_{W^{bw}}^{bw}(\mathbf{y}^*),$ $\underline{t}^{bw}, \bar{t}^{bw}, \underline{\beta}^{bw}, \bar{\beta}^{bw})$
Zulässigkeit Erweiterung zu $u_j \in \mathcal{V} / u_k \in BW$ ist zulässig, falls	$u_i \in FW, u_j \notin X^{fw}$ $(i, j) \in \mathcal{A},$ Algo. 2 gibt BS zurück	$u_k \notin X^{bw}$ $(i, k) \in \mathcal{A},$ Algo. 8 gibt BS zurück
Erweiterung Neues Label L'^{fw} / L'^{bw} ist definiert als	aktueller Knoten = u_j $W'^{fw} = (w_0, \dots, u_i, u_j)$ $\bar{c}_{W'^{fw}}^{fw}(\mathbf{y}^*) = \bar{c}_{W^{fw}}^{fw}(\mathbf{y}^*) + p_j - y_j^*$ $(\underline{t}'^{fw}, \bar{t}'^{fw}, \underline{\beta}'^{fw}, \bar{\beta}'^{fw}) = \text{Algo. 2}$ $(u_i, u_j, D, \underline{t}^{fw}, \bar{t}^{fw}, \underline{\beta}^{fw}, \bar{\beta}^{fw})$	aktueller Knoten = u_k $W'^{bw} = (u_k, u_i, \dots, w_{n+1})$ $\bar{c}_{W'^{bw}}^{bw}(\mathbf{y}^*) = \bar{c}_{W^{bw}}^{bw}(\mathbf{y}^*) + p_k - y_k^*$ $(\underline{t}'^{bw}, \bar{t}'^{bw}, \underline{\beta}'^{bw}, \bar{\beta}'^{bw}) = \text{Algo. 8}$ $(u_i, u_k, D, \underline{t}^{bw}, \bar{t}^{bw}, \underline{\beta}^{bw}, \bar{\beta}^{bw})$
Vereinigung Label L^{fw} und L^{bw} werden vereinigt, falls	$X^{fw} \cap X^{bw} = \{u_i\},$ $\exists (t^{fw}, \beta^{fw}) \in BS^{fw}, (t^{bw}, \beta^{bw}) \in BS^{bw} :$ $t^{fw} \leq t^{bw}, \beta^{fw} \geq \beta^{bw}$	

Tabelle 4.3: Zulässigkeits-, Erweiterungs- und Vereinigungsregeln für den Labeling-Algorithmus

starke Einschränkung darstellt. Der Ausschluss einer bestimmten Sequenz hingegen stellt keine wesentliche Einschränkung dar und verkleinert den Suchraum nur unwesentlich. Folglich würde diese Regel zu einem starken Ungleichgewicht im Branch-and-Bound-Baum führen. Daher werden zwei alternative Branching-Regeln vorgestellt. Die erste Regel wird als *Kunden-Branching* bezeichnet (siehe Boussier et al. (2006); Azi et al. (2010)). Für eine nicht ganzzahlige Lösung wird geprüft, ob ein Kunde $u_i \in C$ nicht ganzzahlig oft beliefert wird. Ist dies der Fall, wird der Kunde gewählt, dessen Belieferungsanzahl am nächsten an 0,5 liegt. Anschließend werden zwei Branches erstellt. In einem Branch muss der Kunde beliefert werden, während er im anderen ausgeschlossen wird. Dafür wird im ersten Branch die Bedingung (4.19) für den Kunden gleich 1 gesetzt. Um einen Kunden auszuschließen, werden alle Variablen, welche den Kunden enthalten, aus Ω' entfernt und zusätzlich wird der Kunde aus $\mathcal{G}$ entfernt. Dies erfordert, dass der Graph $\mathcal{G}$ spezifisch für das Teilproblem des Branch-and-Bound-Baums angepasst und an die Kinderknoten weitergegeben wird.

Diese Regel allein garantiert jedoch nicht die Ganzzahligkeit. Daher wird eine zweite Regel eingeführt, welche die Kanten betrachtet. Diese Regel wird als *Kanten-Branching* bezeichnet (siehe Desrosiers et al. (1984)). Wenn alle Kunden ganzzahlig oft beliefert werden, wird überprüft, ob die Kanten auch ganzzahlig oft durchlaufen werden. Ist dies nicht der Fall, wird die Kante, die in einer Lösung am nächsten an 0,5 liegt, ausgewählt, und es werden zwei Teilprobleme erstellt. Im ersten Branch wird die Kante forciert, im anderen ausgeschlossen. Um eine Kante (i, j) zu forcieren, werden die Nebenbedingungen für u_i und u_j auf eins gesetzt, Variablen entfernt, die u_i oder u_j beliefern, aber (i, j) nicht nutzen, und alle ausgehenden bzw. eingehenden Kanten von u_i oder u_j außer (i, j) werden aus $\mathcal{G}$ entfernt. Um eine Kante auszuschließen, werden die entsprechenden Variablen entfernt und die Kante aus $\mathcal{G}$ gelöscht. Diese Regeln garantieren die Ganzzahligkeit der Lösung.

4.6.4 Beschleunigungsverfahren

Die Beschleunigungsverfahren lassen sich grundsätzlich in zwei Kategorien einteilen. Zum einen gibt es Verfahren, die die Optimalität der Lösung bewahren, und zum anderen solche, die eine heuristische Lösung liefern. Zunächst wird die erste Kategorie betrachtet. Dazu werden eine Dominanzregel, eine Reduktion des Graphen $\mathcal{G}$ und eine Variation der aufgenommenen Variablen zum EMP vorgestellt. Im Anschluss werden heuristische Verfahren erläutert.

Dominanz

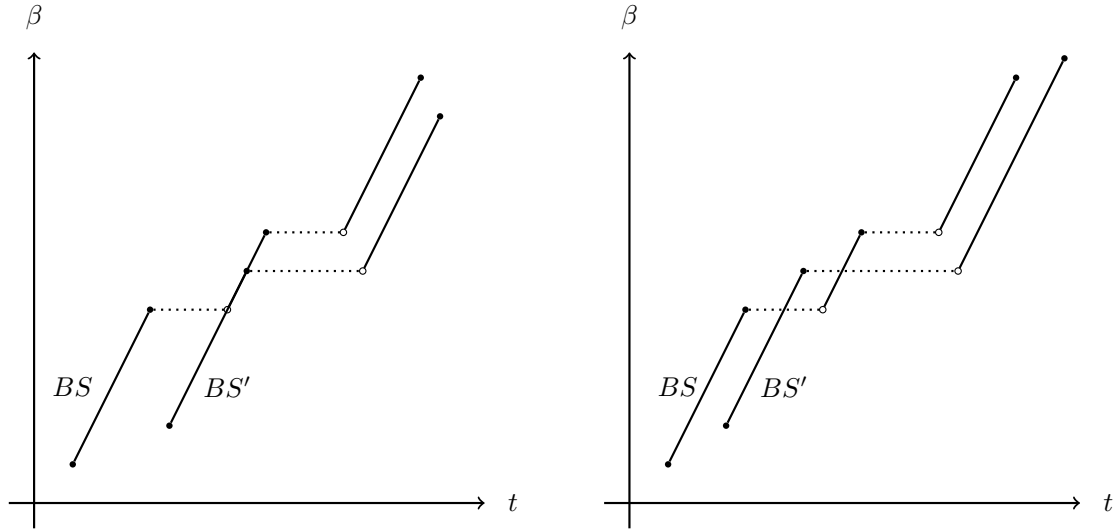
Die Grundidee der Dominanz besteht darin, zwei Labels zu vergleichen, deren Teilwege am gleichen Kunden enden, und zu entscheiden, ob eines dieser Labels das andere dominiert. Unabhängig von der Fortsetzung des Weges muss das dominierende Label in jedem Fall besser sein. Der Vergleich erfolgt auf Basis der reduzierten Kosten, der Zeit, des Ladestandes und der besuchten Kunden. Liegt Dominanz vor, muss das dominierte Label nicht weiter berücksichtigt werden. Dadurch kann die Menge der zu betrachtenden Labels reduziert werden. Die Prüfung kann bei der Aufnahme eines neuen Labels zur Menge der noch zu betrachtenden Labels durchgeführt werden (siehe Zeile 8 im Labeling-Algorithmus). Dabei kann das neue Label dominiert werden und wird dann nicht aufgenommen. Außerdem ist es möglich, dass Labels in der Menge der noch zu betrachtenden Labels dominiert werden und dementsprechend aus der Menge entfernt werden.

Zur Formulierung der Dominanzregel können der vorwärts- und rückwärtsgerichtete Labeling-Algorithmus weitestgehend zusammen betrachtet werden. Angenommen, L und L' sind zwei Labels, deren Teilwege am gleichen End- bzw. Startkunden $u_i \in C$ enden. Seien X, X' die entsprechenden Mengen der besuchten Knoten der Labels. Sei Y die Menge aller Knoten aus $X \setminus X'$, die von L' in ihrem Zeitfenster erreicht werden können. Es ist zu beachten, dass die Überprüfung der Erreichbarkeit vereinfacht durchgeführt wird und jeder Knoten aus $X \setminus X'$ getrennt betrachtet wird. Sei BS die gebrochene Gerade von Label L und BS' die entsprechende Gerade von Label L' .

Definition 13. Seien $L = (u_i, W, \bar{c}_W(\mathbf{y}^*), \underline{t}, \bar{t}, \underline{\beta}, \bar{\beta})$ und $L' = (u_i, W', \bar{c}_{W'}(\mathbf{y}^*), \underline{t}', \bar{t}', \underline{\beta}', \bar{\beta}')$ zwei Labels, die den gleichen aktuellen Kunden $u_i \in C$ besitzen. Label L dominiert Label L' , falls die folgenden Bedingungen erfüllt sind:

- (1) $\bar{c}_W(\mathbf{y}^*) \geq \bar{c}_{W'}(\mathbf{y}^*)$,
- (2.1) $\forall (t', \beta') \in BS' \exists (t, \beta) \in BS : t \leq t', \beta \geq \beta'$ (vorwärtsgerichteter Fall) bzw.
- (2.2) $\forall (t', \beta') \in BS' \exists (t, \beta) \in BS : t \geq t', \beta \leq \beta'$ (rückwärtsgerichteter Fall),
- (3) $Y = \emptyset$.

Bedingung (1) besagt, dass die reduzierten Kosten von L größer oder gleich den reduzierten Kosten von L' sein müssen. Bedingung (2) unterscheidet sich für den vorwärts- und rückwärtsgerichteten Fall. Im vorwärtsgerichteten Fall muss gelten, dass für jedes Zeit-Batterie-Paar auf



(a) Gebrochene Gerade von BS dominiert die gebrochene Gerade von BS'

(b) Zwei unvergleichbare gebrochene Geraden

Abbildung 4.6: Zwei Beispiele für den vorwärtsgerichteten Vergleich von gebrochenen Geraden

der gebrochenen Geraden von L' ein entsprechendes Paar auf der gebrochenen Geraden von L existiert, das nicht später ist und einen gleich großen oder größeren Ladestand aufweist. Im rückwärtsgerichteten Fall muss gelten, dass für jedes Zeit-Batterie-Paar von L' ein entsprechendes Paar von L existiert, das nicht früher ist und keinen höheren Ladestand aufweist. Bedingung (3) sagt aus, dass kein Kunde von Label L' erreichbar ist, der ausschließlich von L besucht wird. Sind diese Kriterien erfüllt, wird L' von L dominiert.

Zwei Beispiele für den vorwärtsgerichteten Vergleich zweier gebrochener Geraden sind in Abbildung 4.6 dargestellt. Dabei wird in der linken Abbildung 4.6(a) die gebrochene Gerade von L' von der gebrochenen Geraden von L dominiert, und in der rechten Abbildung 4.6(b) sind zwei unvergleichbare gebrochene Geraden abgebildet. Im Folgenden werden weitere Verfahren zur Verbesserung der Laufzeit vorgestellt.

Variation der Anzahl der aufgenommenen Variablen

Eine weitere Möglichkeit besteht darin, nach Abschluss des Labeling-Algorithmus mehrere Variablen zum EMP hinzuzufügen. Da der Algorithmus ohnehin alle nicht dominierten Variablen mit positiven reduzierten Kosten findet, können diese nach ihren Kosten sortiert und eine vordefinierte Anzahl aufgenommen werden. Es ist zu beachten, dass auch alle gefundenen Variablen mit positiven reduzierten Kosten aufgenommen werden könnten. Allerdings würde dabei die Anzahl der zu betrachtenden Variablen gegebenenfalls erheblich steigen, was die Lösung des EMPs erschweren würde. Das andere Extrem bestünde darin, nur eine einzelne Variable aufzunehmen. Dies könnte jedoch zu zusätzlichen Aufrufen des Pricing-Problems führen. Daher ist es wichtig, einen Wert zu wählen, der eine angemessene Balance zwischen den beiden entgegengesetzten Zielen gewährleistet.

Reduktion des Graphen $\mathcal{G}$

Ein weiteres Verfahren reduziert den Graphen $\mathcal{G}$ unter Wahrung der Optimalität. Bereits vor dem Branch-and-Price-Algorithmus kann die Kantenmenge unabhängig von den dualen Variablen verkleinert werden. Eine Kante $(i, j) \in \mathcal{A}$ wird entfernt, wenn u_j von u_i nicht ohne Zeitfensterverletzung erreichbar ist, also gilt:

$$a_i + \tau_i + \min_{k \in D} \{d_{ik} + d_{kj}\} + \frac{1}{\alpha} \max\{0, d_j + d_k + \min_{k \in D} \{d_{ik} + d_{kj}\} - B\} > b_j. \quad (4.26)$$

Zusätzlich können Kunden mit negativen reduzierten Kosten während des Pricing-Problems vorübergehend aus $\mathcal{V}$ entfernt werden.

Heuristik

Abschließend wird in Anlehnung an Desaulniers et al. (2008) eine Heuristik zur Beschleunigung des Pricing-Problems vorgestellt. Durch eine Begrenzung der ausgehenden Kanten jedes Kunden $u_i \in \mathcal{C}$ werden der Graph $\mathcal{G}$ und damit die Anzahl der zu generierenden Labels reduziert. Welche Kanten $(i, j) \in \mathcal{A}$ im Graphen verbleiben, entscheidet ein Bewertungsschema basierend auf (i) der Distanz, (ii) der Zeitfenster-Differenz und (iii) den reduzierten Kosten mit den Gewichten γ_1 bis γ_3 :

$$\gamma_1 \frac{d_{i,j}}{\max_{(i,k) \in \mathcal{A}} \{d_{i,k}\}} + \gamma_2 \frac{b_j - a_i}{\max_{(i,k) \in \mathcal{A}} \{b_k\} - a_i} + \gamma_3 \frac{p_j - y_j^*}{\min_{(i,k) \in \mathcal{A}} \{p_k - y_k^*\}}. \quad (4.27)$$

Außerdem wird für die Heuristik die Dominanzregel gelockert, sodass die Menge Y nicht zwingend leer sein muss. In diesem Fall wird gefordert, dass die reduzierten Kosten von L größer oder gleich den reduzierten Kosten von L' zuzüglich der maximal möglichen reduzierten Kosten der Menge Y sind. Hierfür werden die Kosten der Menge Y formal als $\bar{c}_Y(\mathbf{y}^*) = \sum_{u_j \in Y} \max\{0, p_j - y_j^*\}$ definiert. Bedingung (1) der Dominanzregel wird für die heuristische Suche demnach wie folgt modifiziert: $\bar{c}_W(\mathbf{y}^*) \geq \bar{c}_{W'}(\mathbf{y}^*) + \bar{c}_Y(\mathbf{y}^*)$. Es gilt zu beachten, dass diese Lockerung auch im Wurzelknoten des Branch-and-Price-Verfahrens zulässig ist. Der Grund hierfür ist, dass in diesem Knoten noch kein Branching durchgeführt wurde und somit noch keine Variablen oder Kanten durch den Suchbaum eingeschränkt sind.

Diese Kantenreduktion beschleunigt die Suche erheblich. Um die Optimalität der Lösung zu garantieren, ist jedoch eine erneute Suche im vollständigen Graphen erforderlich, falls keine Variablen mit positiven reduzierten Kosten gefunden werden. Die mit dem vorgestellten Branch-and-Price-Algorithmus erzielten Ergebnisse werden in Kapitel 6 präsentiert. Das nächste Kapitel betrachtet ebenfalls die Zustellung auf der letzten Meile, verlagert den Transport jedoch in die Luft. Beim Einsatz von Drohnen liegt der Fokus auf der Minimierung des Energieverbrauchs, welcher durch die Flugstrecke und -geschwindigkeit optimiert wird.

Kapitel 5

Geschwindigkeitsoptimierung von Paketdrohnen

In diesem Kapitel wird ein Optimierungsalgorithmus für ein Zustellnetzwerk mit Paketdrohnen vorgestellt. Dieses Kapitel basiert auf einer Arbeit mit dem Titel *Sequence and Speed Optimisation*, die im *INFORMS Journal on Computing* erschienen ist (Schaudt und Bektaş, 2026). Der Fokus liegt hierbei auf der Betrachtung der allerletzten Meile, also auf dem Abschnitt vom Mikrodepot zum Empfänger. Das Problem ähnelt der Fragestellung, die in Unterkapitel 2.2.3 beschrieben wird. In diesem Kapitel wird jedoch die Zustellgeschwindigkeit des Transportmittels als variabel angenommen. Ziel ist es, einen Plan zu erstellen, der den gesamten Energieverbrauch aller Drohnen minimiert. Konkret soll der Plan eine Zuordnung zwischen Sendungen und Drohnen festlegen und bestimmen, in welcher Reihenfolge und von welchem Mikrodepot die Sendungen geliefert werden. Wie die Zustellroboter verfügen auch die Drohnen über eine Transportkapazität von nur einer Sendung und können nach der Belieferung eines Empfängers das Mikrodepot wechseln. Den Empfängern sind ebenfalls Zeitfenster zugeordnet, in denen die Belieferung erfolgen soll. Ein Unterschied besteht darin, dass keine Ladevorgänge des Akkus an den Mikrodepots modelliert werden. Stattdessen wird angenommen, dass die Akkus an den Mikrodepots getauscht werden. Außerdem kann die Fluggeschwindigkeit der Drohnen zu jedem Zeitpunkt angepasst werden. Ziel der Problemstellung ist es, einen Plan zu erstellen, der sicherstellt, dass jeder Empfänger in seinem Zeitfenster beliefert wird und dabei der geschwindigkeitsabhängige Energieverbrauch der Drohnen minimiert wird.

Die Betrachtung des Energieverbrauchs von Paketdrohnen ist besonders relevant, da dieser nicht linear von der Fluggeschwindigkeit abhängt. Ursächlich dafür ist der Luftwiderstand, durch den der Energieverbrauch in quadratischer Abhängigkeit zur Geschwindigkeit steht. Für Zustellroboter ist die Betrachtung variabler Geschwindigkeiten nur bedingt sinnvoll, da sie mit Schrittgeschwindigkeit auf Bürgersteigen fahren und das Intervall zulässiger Geschwindigkeiten daher sehr gering ist. Folglich hätte die Optimierung der Geschwindigkeiten kaum einen Effekt. Im Folgenden wird der Begriff „Depot“ synonym für „Mikrodepot“ verwendet.

Dieses Kapitel ist wie folgt gegliedert. In Unterkapitel 5.1 wird ein Energieverbrauchsmodell vorgestellt, das den Energieverbrauch in Abhängigkeit von Gewicht und Geschwindigkeit der Paketdrohnen beschreibt. Im Anschluss erfolgt eine Einordnung in die verwandte Literatur in Unterkapitel 5.2, gefolgt von der Einführung der Notationen in Unterkapitel 5.3. In Unterkapitel 5.4 wird die Problemstellung formal beschrieben und eine ganzzahlige lineare Formulierung des Problems präsentiert. Die Unterkapitel 5.5 und 5.6 bilden den Kern dieses Kapitels. Zunächst wird die Problemstellung für eine einzelne Drohne und eine gegebene Reihenfolge der Empfänger betrachtet. Für die Reihenfolge wird ein exakter Algorithmus vorgestellt, der die Geschwindigkeiten so wählt, dass der Gesamtenergieverbrauch minimiert wird. Anschließend wird ein weiterer Algorithmus eingeführt, der für die gegebene Reihenfolge der Empfänger alle relevanten Belieferungszeitpunkte des letzten Empfängers berechnet. Zudem wird die Korrektheit der beiden Algorithmen bewiesen. Diese beiden Algorithmen fließen in einen Branch-and-Price-Algorithmus ein, der zur Optimierung eines Zustellnetzwerks mit Paketdrohnen genutzt werden kann. Die Ergebnisse des Branch-and-Price-Algorithmus werden in Kapitel 6 präsentiert.

5.1 Energieverbrauch eines Multikopters

In diesem Abschnitt wird ein Modell zur Berechnung des Energieverbrauchs einer Drohne in Abhängigkeit von der Geschwindigkeit im waagerechten Flug vorgestellt. Als Modell für eine Drohne wird ein Multikopter verwendet (siehe Unterkapitel 2.3.1). Die Herleitung einer Energieverbrauchsfunction basiert auf den Veröffentlichungen von Dukkanci et al. (2021) und Zeng et al. (2019). Dazu wird zunächst eine Funktion definiert, die die benötigte Leistung in Abhängigkeit von der Geschwindigkeit beschreibt. Zeng et al. (2019) definieren den Funktionsterm wie folgt

$$P(v) = \underbrace{\frac{\delta}{8}\rho s A \Omega^3 r^3 \left(1 + \frac{3v^2}{U_{tip}^2}\right)}_{\text{Widerstand des Rotorprofils}} + \underbrace{(1+k)\frac{W^{3/2}}{\sqrt{2\rho A}}\left(\sqrt{1 + \frac{v^4}{4v_0^4}} - \frac{v^2}{2v_0^2}\right)^{1/2}}_{\text{induzierter Widerstand}} + \underbrace{\frac{1}{2}d_0\rho s A v^3}_{\text{parasitärer Widerstand}}. \quad (5.1)$$

Die Leistung wird in Watt und die Geschwindigkeit in Metern pro Sekunde gemessen. In Tabelle 5.1 sind die verwendeten Notationen dargestellt. Diese beschränken sich auf dieses Unterkapitel.

In Abbildung 5.1 ist die Funktion (5.1) gestrichelt als Summe dargestellt. Diese weist ein lokales Maximum bei null und einen Tiefpunkt bei v_{me} auf. Dabei beschreibt die Geschwindigkeit null den Schwebestand der Drohne. Zu erkennen ist, dass die Funktion weder konvex noch konkav ist. Daher wird der Funktionsterm mittels einer Taylor-Approximation vereinfacht. Es wird dafür $P_0 = \frac{\delta}{8}\rho s A \Omega^3 r^3$ und $P_i = (1+k)\frac{W^{3/2}}{\sqrt{2\rho A}}$ gesetzt. Mit der ersten Taylor-Approximation ergibt sich:

$$\tilde{P}(v) = P_0\left(1 + \frac{3v^2}{U_{tip}^2}\right) + P_i \frac{v_0}{v} + \frac{1}{2}d_0\rho s A v^3.$$

Diese Funktion ist in Abbildung 5.1 als durchgezogene Kurve dargestellt. Es handelt sich dabei um eine strikt konvexe Funktion. Es ist zu beachten, dass v_{me} der Geschwindigkeit entspricht,

die die Flugzeit der Drohne maximiert, jedoch nicht die Reichweite. Dafür wird eine weitere Funktion eingeführt, die den Energieverbrauch der Drohne in Abhängigkeit der Geschwindigkeit je zurückgelegter Einheit beschreibt. Diese Funktion wird aus der approximierten Funktion $\tilde{P}(v)$ durch die Multiplikation mit der Flugzeit gewonnen und es folgt

$$E(v, d) \approx P_0 \left(\frac{d}{v} + \frac{3dv}{U_{tip}^2} \right) + P_i \frac{v_0 d}{v^2} + \frac{1}{2} d_0 \rho s A d v^2 \quad (5.2)$$

$$\approx \mu_1 \frac{d}{v} + \mu_2 d v + \mu_3 \frac{d}{v^2} + \mu_4 d v^2 \quad (5.3)$$

$$\approx 79,856 \frac{d}{v} + 0,0166 d v + 955,035 \frac{d}{v^2} + 0,009 d v^2. \quad (5.4)$$

Symbol	Beschreibung	Wert
δ	Profilwiderstandsbeiwert	0,012
ρ	Dichte Luft (kg/m^3)	1,225
s	Flächendichte des Rotors	0,05
A	Fläche des Rotors (m^2)	0,503
Ω	Winkelgeschwindigkeit des Rotorblatts ($radian/s$)	300
r	Radius des Rotors (m)	0,4
U_{tip}	Geschwindigkeit der Rotorspitze (m/s)	120
k	Inkrementeller Korrekturfaktor der induzierten Leistung	0,1
m_{air}	Drohnengewicht (kg)	2,04
m_{pack}	Sendungsgewicht (kg)	1
m_{batt}	Batteriegewicht (kg)	0,89
W	Gesamtgewicht ($Newton$)	38,53
v_0	Mittlere rotorinduzierte Geschwindigkeit im Schwebeflug (m/s)	4,03
d_0	Rumpfwiderstandsverhältnis	0,6

Tabelle 5.1: Notationen für den Energieverbrauch einer Drohne (Dukkanci et al., 2021)

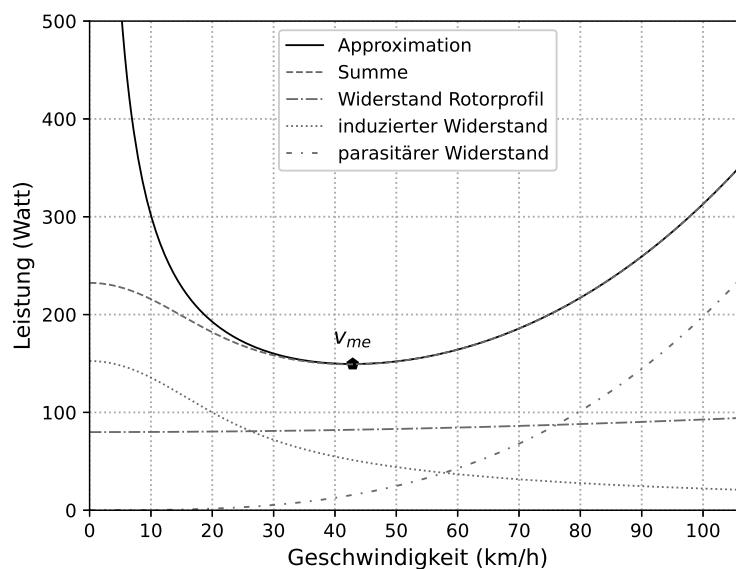


Abbildung 5.1: Zusammensetzung der Leistung in Abhängigkeit der Geschwindigkeit (Zeng et al., 2019)

In Abbildung 5.2 sind zwei Energieverbrauchsfunktionen abgebildet. Der Energieverbrauch wird in Wattstunden pro Kilometer gemessen. Die durchgezogene Kurve zeigt den Energieverbrauch bei einer Traglast von einem Kilogramm, während die gestrichelte Kurve den Energieverbrauch ohne Last darstellt. Die Geschwindigkeit, die die Reichweite maximiert, hängt vom Gesamtgewicht der Drohne ab. Sie beträgt bei einer zusätzlichen Last von einem Kilogramm 74,65 km/h und ohne zusätzliches Gewicht 70,13 km/h. Abbildung 5.2 verdeutlicht, dass das Sendungsgewicht den Energieverbrauch beeinflusst, wobei dieser Einfluss ab dem Minimum der Funktion relativ gering ist. Zur Vereinfachung wird im Folgenden angenommen, dass die Drohnen sowohl auf dem Hinflug zum Kunden als auch auf dem Rückflug gleich schwer sind.

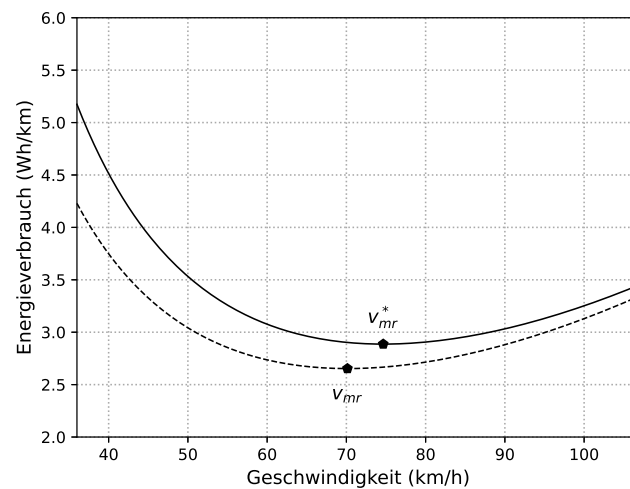


Abbildung 5.2: Einfluss des Sendungsgewichts auf den Energieverbrauch (Dukkanci et al., 2021)

5.2 Einordnung in die Literatur

Dieser Abschnitt präsentiert die relevante Literatur. Die Betrachtung beschränkt sich jedoch nicht nur auf die Geschwindigkeitsoptimierung, sondern umfasst auch Publikationen zur Optimierung logistischer Prozesse mit Paketdrohnen. Der Fokus liegt dabei auf Konzepten, die von einem stationären Depot ausgehen und den Energieverbrauch der Drohne modellieren. Eine Übersicht über Optimierungsprobleme mit Paketdrohnen wird in Vilorio et al. (2020) vorgestellt. Moshref-Javadi und Winkenbach (2021) bieten eine Übersicht über Publikationen, die den Einsatz von Paketdrohnen in der Logistik untersuchen.

Dorling et al. (2017) untersuchen ein Problem, das mit dem VRP verwandt ist. Drohnen starten dabei von einem Depot und beliefern Kunden. Jede Drohne kann mehrere Kunden in einer Tour bedienen. Zwei getrennte Zielfunktionen werden untersucht. Die erste Zielfunktion minimiert den spätesten Belieferungszeitpunkt, während die zweite die Gesamtkosten, bestehend aus Energieverbrauch und der Anzahl der Drohnen, reduziert. Zur Lösung wird eine *Simulated-Annealing-Heuristik* vorgestellt, die den Energieverbrauch der Drohnen in Abhängigkeit vom Gewicht modelliert. Ein Modell mit mehreren stationären Depots wird in Song et al. (2018)

präsentiert. Hervorzuheben ist, dass die Drohnen nach der Belieferung eines Kunden das Depot wechseln können. Die Flugzeit hängt linear vom Gewicht der transportierten Sendung ab. Zur Lösung werden ein MIP und eine Heuristik vorgestellt. In der Arbeit von Cheng et al. (2020) wird eine Problemstellung mit einem Depot untersucht. Jedem Kunden ist ein Zeitfenster für die Belieferung zugeordnet. Der Energieverbrauch der Drohnen wird in Abhängigkeit von der Transportlast gemessen. Ein Branch-and-Cut-Algorithmus wird zur Lösung genutzt.

Kirschstein (2020) stellt ein detailliertes Modell zur Messung des Energieverbrauchs von Drohnen vor. Das Modell erfasst neben dem Energieverbrauch während des Fluges auch den Verbrauch beim Steig- und Sinkflug sowie im Schwebestand. Darüber hinaus werden Wetterbedingungen in die Betrachtung einbezogen. Der Autor präsentiert eine Simulation, bei der Kunden von einem zentralen Depot aus beliefert werden. Dabei vergleicht er den Energieverbrauch der Drohnen mit dem von Zustellfahrzeugen, die entweder mit Diesel oder elektrisch betrieben werden. Dukkanci et al. (2021) beschreiben ein Problem, bei dem Kunden von Drohnen beliefert werden. Dabei müssen für die Belieferung der Kunden geeignete Standorte ausgewählt werden, von denen die Drohnen starten können. Die Drohnen werden zu den gewählten Standorten transportiert und liefern von dort aus ihre Sendungen. Das Energieverbrauchsmodell für Drohnen von (Zeng et al., 2019) wird von den Autoren verwendet, um ihr Optimierungsmodell in Abhängigkeit von den Fluggeschwindigkeiten der Drohnen zu formulieren. Jeder Kunde hat einen festgelegten spätesten Belieferungszeitpunkt. Zur Lösung des Problems wird ein nichtlineares Programm vorgestellt.

Auch für andere Transportmittel wurde die Geschwindigkeitsoptimierung betrachtet. So untersuchen Fagerholt et al. (2010) die Geschwindigkeitsoptimierung für Schiffe, die eine gegebene Reihenfolge von Häfen anfahren. Das zugrunde liegende Problem wird in der Literatur als *Speed Optimization Problem* (SOP) bezeichnet. Das Ziel besteht darin, für eine Reihenfolge der Häfen optimale Geschwindigkeiten zu ermitteln, sodass die Zeitfenster der Häfen eingehalten und der Kraftstoffverbrauch des Schiffs minimiert wird. In der Publikation wird angenommen, dass der Bunkerverbrauch eines Schiffs quadratisch von der Geschwindigkeit abhängt. Das Problem wird als nichtlineares Programm formuliert und eine diskretisierte Variante als Kürzestes-Wege-Problem gelöst. Der erste exakte Algorithmus für das SOP, der in Polynomialzeit läuft, wird von Norstad et al. (2011) präsentiert. Als Kostenfunktion wird eine konvexe und monoton steigende Funktion angenommen. Allerdings erfolgt die Einführung ohne den Nachweis der Korrektheit des Algorithmus. Der Algorithmus wird von Hvattum et al. (2013) analysiert. Die Autoren betrachten den Algorithmus für eine konvexe und stetige Kostenfunktion, die über das Intervall $[v_{min}, v_{max}]$ definiert ist. Diese Kostenfunktion wird in eine konvexe, stetige und monoton steigende Funktion transformiert, die konstant von 0 bis zum Minimum der Funktion über dem Intervall ist. Ein Beispiel für diese Transformation ist in Abbildung 5.3 dargestellt. Anschließend wird die Korrektheit der transformierten Kostenfunktion bewiesen. Allerdings wird der Nachweis für die ursprüngliche konvexe, stetige Kostenfunktion über ein beliebiges Intervall nicht erbracht.

Eine Variante des SOP, die zusätzlich den ersten Abfahrtszeitpunkt vom Depot berücksichtigt, wird von Kramer et al. (2015) untersucht. Die Autoren analysieren eine spezifische konvexe Funktion, die sowohl den Treibstoffverbrauch als auch das Gehalt des Fahrers berücksichtigt. Daher

muss neben dem Verbrauch auch die Dauer einer Tour minimiert werden. Zur Lösung wird ein exaktes dynamisches Programm vorgestellt, dessen Laufzeit polynomiell begrenzt ist, und es wird die Korrektheit des Algorithmus bewiesen. Dabei wird angenommen, dass die Kostenfunktion auf allen Kanten identisch ist. Diese Einschränkung wird von He et al. (2017) aufgehoben, indem für jede Kante eine kontinuierliche, differenzierbare und strikt konvexe Funktion verwendet wird.

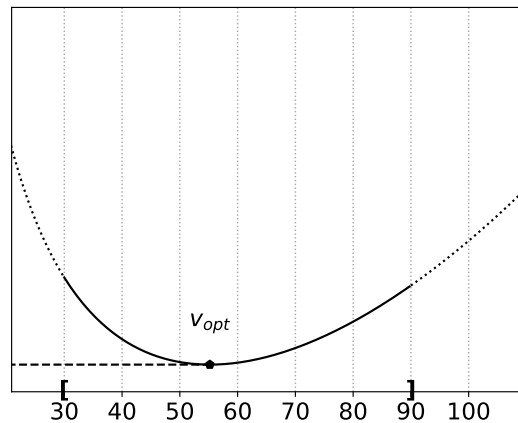


Abbildung 5.3: Strikt konvexe Funktion (durchgezogen) transformiert in eine monoton steigende konvexe Funktion (gestrichelt und durchgezogen)

Neben der Betrachtung einer festgelegten Kundenreihenfolge wird die Geschwindigkeitsoptimierung auch für das TSP und das VRP untersucht. Der Ursprung dieses Forschungsfelds liegt in der Transportlogistik. Fagerholt et al. (2010); Norstad et al. (2011) untersuchen den Seeverkehr, während Bektaş und Laporte (2011) den Straßenverkehr analysieren. Letztere definieren das sogenannte *Pollution-Routing Problem* (PRP). Hierbei wird angenommen, dass die Kunden über Zeitfenster verfügen und die Geschwindigkeit variabel ist. Die Zielfunktion umfasst die Gehälter der Fahrer und die Kraftstoffkosten, die vom Verbrauch der Transportfahrzeuge abhängen. Der Verbrauch hängt von der Geschwindigkeit, der Strecke und dem transportierten Gewicht ab. Um das Problem zu lösen, diskretisieren die Autoren die Geschwindigkeiten und präsentieren ein MIP. In einer weiteren Publikation der Autoren wird zusätzlich eine *Adaptive-Large-Neighborhood-Search*-Heuristik für das PRP präsentiert (Demir et al., 2012). Aufbauend darauf werden das PRP sowie die Geschwindigkeitsoptimierung von mehreren Autoren untersucht (Kramer et al., 2015; Fukasawa et al., 2016; Dabia et al., 2017; Fukasawa et al., 2018). Zwei Publikationen, die eng mit dem in diesem Kapitel beschriebenen Branch-and-Price-Algorithmus verwandt sind, werden von Dabia et al. (2017) und Fukasawa et al. (2018) vorgestellt. Dabei präsentieren Dabia et al. (2017) einen Branch-and-Price-Algorithmus für das PRP. Hierbei wird angenommen, dass die Geschwindigkeiten aller Kanten innerhalb einer Sequenz gleich gewählt sein müssen. Ein Branch-Cut-and-Price-Algorithmus zur Lösung des PRP wird von Fukasawa et al. (2018) präsentiert. Die Autoren untersuchen eine strikt konvexe, differenzierbare Funktion und formulieren das Problem als ein Partitionsproblem. Es ist anzumerken, dass in der Publikation und in dieser Arbeit unterschiedliche Formulierungen des Problems verwendet werden. Die Ergebnisse dieser Publikation werden in Kapitel 6 zum Vergleich genutzt.

5.3 Notationen

Eine Übersicht über die in diesem Kapitel verwendeten Notationen ist in Tabelle 5.2 dargestellt.

Allgemeine Notationen	
V	Menge der Knoten
C	Menge der Kunden
u_i	Kunde mit $u_i \in C$
u_0	Depot
m	Anzahl der verfügbaren Drohnen
$[a_i, b_i]$	Zeitfenster von Kunde $u_i \in C$
t_i	Ankunftszeitpunkt an Knoten $u_i \in V$
T	Zeithorizont
τ_i	Servicezeit von Kunde $u_i \in C$
d_{ij}	Distanz zwischen Knoten $u_i \in V$ und Knoten $u_j \in V$
v_{ij}	Geschwindigkeit von $u_i \in V$ nach $u_j \in V$
v_i	Geschwindigkeit zu $u_i \in V$
f	Energieverbrauchsfunktion/Kostenfunktion
$\mathbf{v}$	Geschwindigkeitsvektor
v_{min}, v_{max}	minimale und maximale Geschwindigkeit
s	Sequenz von Kunden mit $s = (u_1, u_2, \dots, u_h)$
u_h	letzter Kunde einer Sequenz
$c_f(s)$	Minimaler Energieverbrauch/ minimale Kosten einer Sequenz s unter der Kostenfunktion f
e_{is}	binärer Koeffizient, der angibt, ob Kunde u_i in Sequenz s enthalten ist
Algorithmus 6 (Unterkapitel 5.5)	
d_i	Distanz und Geschwindigkeit zu Kunden $u_i \in C$
$\tilde{f}$	monoton steigende, konvexe Kostenfunktion
$\tilde{\mathcal{V}}$	Menge der kostenminimalen Geschwindigkeitsvektoren
arr	Funktion, die die Belieferungszeitpunkte des letzten Kunden einer Sequenz beschreibt
$cost$	Funktion, die die Kosten einer Sequenz für verschiedene Belieferungszeitpunkte des letzten Kunden einer Sequenz beschreibt
$\tilde{s}(\mathbf{v}), \bar{s}_i(\mathbf{v})$	Teilsequenzen von Sequenz s basierend auf einem Geschwindigkeitsvektor $\mathbf{v}$ mit $i = 1, \dots, p$
u_l	letzter Kunde von $\tilde{s}(\mathbf{v})$
Branch-and-Price-Algorithmus (Unterkapitel 5.6)	
Ω, Ω'	Menge der zulässigen Sequenzen und reduzierte Menge mit $\Omega' \subseteq \Omega$
$\bar{c}_f(s, \mathbf{y}^*)$	reduzierte Kosten der Sequenz s für die optimale duale Lösung $\mathbf{y}^*$ unter Funktion f
$\mathcal{V}$	Menge der Knoten in Hilfsgraph $\mathcal{G}$
W, W^{tw}	Weg mit $W = (w_0, u_1, u_2, \dots, u_h, w_{n+1})$ bzw. Teilweg mit $W^{tw} = (w_0, u_1, u_2, \dots, u_h)$
w_0, w_{n+1}	Quelle und Senke der Wege
L	Label

Tabelle 5.2: Notationen für Kapitel 5

5.4 Problembeschreibung

Im Folgenden wird das Optimierungsproblem für ein Zustellnetzwerk mit Paketdrohnen mathematisch beschrieben. Anschließend wird das *Vehicle Routing Problem with Time Windows and Speed Optimization* (VRPTWSO) vorgestellt und gezeigt, wie das Optimierungsproblem für ein Zustellnetzwerk mit Paketdrohnen auf eine Instanz des VRPTWSO reduziert werden kann. Im letzten Abschnitt dieses Unterkapitels wird ein IP für das VRPTWSO präsentiert.

5.4.1 Optimierung eines Zustellnetzwerks mit Paketdrohnen

Gegeben ist ein gerichteter Graph $\tilde{G} = (V, \tilde{A})$, wobei $V = C \cup \{u_0\}$ die Menge der Knoten beschreibt. Dabei stellt u_0 den Depotknoten und C die Menge der Kunden dar. Die Menge der gerichteten Kanten ist $\tilde{A} = \{(0, i) : u_i \in C\} \cup \{(i, 0) : u_i \in C\}$, wobei jeder Kunde mit dem Depot verbunden ist. Zur Belieferung der Kunden stehen im Depot m Drohnen zur Verfügung. Jeder Kunde $u_i \in C$ muss innerhalb seines Zeitfensters $[a_i, b_i]$ beliefert werden, wobei stets $a_i \leq b_i$ gilt. Die Servicezeit am Kunden $u_i \in C$ beträgt τ_i Zeiteinheiten. Das Depot hat eine Servicezeit von Null ($\tau_0 = 0$) und ein Zeitfenster von $[0, T]$, wobei T ein nicht restriktiver Zeithorizont ist. Die Länge einer Kante $(i, j) \in \tilde{A}$ ist gegeben als d_{ij} . Die Geschwindigkeit auf einer Kante $(i, j) \in \tilde{A}$ ist v_{ij} , wobei $v_{ij} \in [v_{\min}, v_{\max}]$ gilt. Die Flugzeit für die Kante $(i, j) \in \tilde{A}$ mit Geschwindigkeit v_{ij} beträgt d_{ij}/v_{ij} . Der Energieverbrauch auf einer Kante $(i, j) \in \tilde{A}$ hängt von der Geschwindigkeit v_{ij} ab und beträgt $d_{ij}f(v_{ij})$, wobei f eine strikt konvexe Energieverbrauchsfunction ist. Im Folgenden wird der Energieverbrauch als Kosten und die Funktion f als Kostenfunktion bezeichnet. Die Geschwindigkeit, die im Intervall $[v_{\min}, v_{\max}]$ liegt und die Funktion f minimiert, wird als v_{opt} bezeichnet. Es gilt

$$v_{\text{opt}} = \arg \min_{v_{\min} \leq v \leq v_{\max}} f(v). \quad (5.5)$$

Dabei ist v_{opt} eindeutig. Für die Einführung des Optimierungsproblems wird die Definition einer Sequenz benötigt. Diese ist identisch mit der Definition aus Kapitel 4.

Definition 1. Eine Sequenz $s = (u_1, u_2, \dots, u_h)$ beschreibt eine Reihenfolge der Kunden, mit $u_1, u_2, \dots, u_h \in C$.

Es ist zu beachten, dass Sequenzen leer sein können. Die Kosten einer Sequenz $s = (u_1, u_2, \dots, u_h)$ mit einem Vektor $\mathbf{v} = (v_{0,1}, v_{1,0}, v_{0,2}, v_{2,0}, \dots, v_{0,h}, v_{h,0})$ berechnen sich für eine Funktion f wie folgt:

$$\sum_{i=1}^h (d_{0i}f(v_{0i}) + d_{i0}f(v_{i0})).$$

Die Belieferungszeitpunkte der Kunden für eine Sequenz lassen sich wie folgt berechnen. Gegeben sei eine Sequenz $s = (u_1, u_2, \dots, u_h)$ sowie ein dazugehöriger Geschwindigkeitsvektor

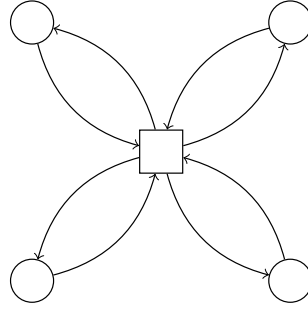


Abbildung 5.4: Gerichteter Graph mit einem Depot (Quadrat) und vier Kunden (Kreise)

$\mathbf{v} = (v_{0,1}, v_{1,0}, v_{0,2}, v_{2,0}, \dots, v_{0h}, v_{h0})$. Der Belieferungszeitpunkt t_i eines Kunden $u_i \in s$ wird rekursiv definiert als

$$t_i = \max(a_i, t_{i-1} + \tau_{i-1} + d_{i-1,0}/v_{i-1,0} + d_{0i}/v_{0i}),$$

wobei $t_0 = 0$, $d_{0,0} = 0$ und $v_{0,0} = 0$ gilt.

Das Ziel des Optimierungsproblems besteht darin, m Sequenzen mit dazugehörigen Geschwindigkeitsvektoren zu finden, sodass die Summe der Kosten, bezogen auf eine Kostenfunktion f , minimal ist und folgende Restriktionen eingehalten werden:

- Jeder Kunde ist in genau einer Sequenz enthalten,
- Jeder Kunde wird in seinem Zeitfenster beliefert,
- Die Geschwindigkeiten der Vektoren bewegen sich im Intervall $[v_{min}, v_{max}]$.

Zur Veranschaulichung der Problemstruktur wird in Abbildung 5.4 ein abstraktes Netzwerk mit vier Kunden und einem Depot dargestellt.

Im Folgenden wird erläutert, wie die Geschwindigkeiten aufeinanderfolgender Kanten gewählt werden sollten. Es ist grundsätzlich möglich, dass die Geschwindigkeit entlang einer einzelnen Kante zu variieren. Allerdings sollte in einer kostenminimalen Lösung die Geschwindigkeit konstant sein. Diese Behauptung lässt sich durch eine Analyse der Funktion f belegen. Dabei werden die Kosten beim Angleichen zweier unterschiedlicher Geschwindigkeiten betrachtet.

Beobachtung 2. Seien v_i, v_j , mit $v_i > v_j$, zwei Geschwindigkeiten. Für jedes δ , mit $0 < \delta < v_i - v_j$, gilt

$$\frac{f(v_j + \delta) - f(v_j)}{\delta} < \frac{f(v_i) - f(v_i - \delta)}{\delta} \quad (5.6)$$

$$\Leftrightarrow f(v_j + \delta) + f(v_i - \delta) < f(v_j) + f(v_i). \quad (5.7)$$

Diese Beobachtung gilt, da f strikt konvex ist, wodurch die Wachstumsrate ansteigt. Außerdem wird δ so gewählt, dass es kleiner als die Differenz zwischen den Geschwindigkeiten ist.

Die Aussage der vorhergehenden Beobachtung wird durch Lemma 8 erweitert. Das Lemma besagt, dass durch das Angleichen der Geschwindigkeiten sowohl die Kosten als auch die Gesamtzeit einer Sequenz reduziert werden können. Diese Aussage ist zentral für die Geschwindigkeitsoptimierung im Unterkapitel 5.5. Der Beweis des Lemmas folgt der Argumentation von Hvattum et al. (2013). Dabei ist zu beachten, dass das Lemma die Zeitfenster nicht berücksichtigt. Die Aussagen des Lemmas ermöglichen jedoch die Formulierung der Lemmata 9 und 10, die eine Beziehung zwischen den Belieferungszeitpunkten und den Zeitfenstergrenzen herstellen.

Im Folgenden werden eine strikt konvexe Funktion f sowie zwei Kanten betrachtet. Es wird angenommen, dass die erste Kante eine Länge d_i und die zweite Kante eine Länge d_j hat. Die Geschwindigkeiten auf den Kanten seien v_i und v_j . Seien $\tilde{d}_i$ mit $0 < \tilde{d}_i \leq d_i$ und $\tilde{d}_j$ mit $0 < \tilde{d}_j \leq d_j$ Teile der Distanzen, sodass $\tilde{d} = \tilde{d}_i = \tilde{d}_j$ gilt.

Lemma 8. *Seien zwei Kanten gegeben, wie vorab beschrieben. Angenommen, es gilt $v_i > v_j$, dann können die Kosten gesenkt werden, wenn v_i auf dem Teilstück $\tilde{d}_i$ um δ gesenkt wird und v_j auf $\tilde{d}_j$ um δ erhöht wird, wobei $0 < \delta < v_i - v_j$ gilt.*

Beweis. Sei $\bar{d}_i$ die verbleibende Distanz auf der ersten Kante, wobei $d_i = \bar{d}_i + \tilde{d}_i$ gilt, und sei $\bar{d}_j$ die verbleibende Distanz auf der zweiten Kante mit $d_j = \bar{d}_j + \tilde{d}_j$. Die Kosten für beide Kanten mit den Ausgangsgeschwindigkeiten sind gegeben als

$$c = d_i f(v_i) + d_j f(v_j) = (\bar{d}_i + \tilde{d}_i) f(v_i) + (\bar{d}_j + \tilde{d}_j) f(v_j).$$

Die resultierende Gesamtzeit berechnet sich als

$$\tilde{T} = \frac{\bar{d}_i + \tilde{d}_i}{v_i} + \frac{\bar{d}_j + \tilde{d}_j}{v_j} = \frac{\bar{d}_i}{v_i} + \frac{\bar{d}_j}{v_j} + \frac{\tilde{d}(v_i + v_j)}{v_i v_j}.$$

Wird die Geschwindigkeit v_i auf dem Teilstück $\tilde{d}_i$ um δ reduziert und die Geschwindigkeit v_j auf $\tilde{d}_j$ um δ erhöht, beträgt die neue Gesamtzeit

$$\tilde{T}_\delta = \frac{\bar{d}_i}{v_i} + \frac{\tilde{d}_i}{v_i - \delta} + \frac{\tilde{d}_j}{v_j + \delta} + \frac{\bar{d}_j}{v_j} = \frac{\bar{d}_i}{v_i} + \frac{\tilde{d}(v_i + v_j)}{v_i v_j + (v_i - v_j)\delta - \delta^2} + \frac{\bar{d}_j}{v_j}.$$

Es folgt $\tilde{T}_\delta < \tilde{T}$, da $0 < \delta < v_i - v_j$.

Die Kosten nach der Anpassung der Geschwindigkeiten sind

$$c_\delta = \bar{d}_i f(v_i) + \tilde{d}_i f(v_i - \delta) + \bar{d}_j f(v_j) + \tilde{d}_j f(v_j + \delta).$$

Es gilt $c_\delta \stackrel{\text{Ungl. (5.7)}}{<} c$. □

Iterativ folgt aus Lemma 8, dass eine konstante Geschwindigkeit, ohne Berücksichtigung der Zeitfenster, die Gesamtkosten minimiert.

Beobachtung 3. Seien $u_i, u_j \in C$ zwei aufeinanderfolgende Kunden. Aus Lemma 8 folgt, dass in einer optimalen Lösung die Geschwindigkeiten auf den Kanten $(i, 0)$ und $(0, j)$ gleich sein sollten.

Diese Beobachtung gilt, da angenommen wurde, dass das Zeitfenster des Depots nicht restriktiv ist. Daraus folgt, dass zur Minimierung der Gesamtkosten bei einer strikt konvexen Funktion nicht nur die Geschwindigkeit auf einer Kante konstant gehalten werden sollte, sondern auch zwischen zwei Kunden. Das bedeutet, die Geschwindigkeit sollte sowohl auf dem Rückflug vom Kunden zum Depot als auch auf dem Hinflug zum nächsten Kunden konstant sein.

Mit diesen Erkenntnissen lässt sich das Problem auf eine Instanz des VRPTWSO reduzieren. Im Unterschied zur bisherigen Problemstellung werden hierbei auch direkte Kanten zwischen zwei Kunden betrachtet. Die Transformation in eine Instanz des VRPTWSO hat den Vorteil, dass in einer Lösung weniger Kanten genutzt werden. Außerdem wird in Kapitel 6 ein Vergleich mit Benchmark-Instanzen des VRPTWSO ermöglicht. Im Folgenden wird das VRPTWSO beschrieben und die Reduzierung erläutert.

5.4.2 Vehicle Routing Problem With Time Windows and Speed Optimization

Gegeben ist ein gerichteter Graph $G = (V, A)$. Die Knotenmenge V bleibt gleich zur Knotenmenge der Problembeschreibung aus Unterkapitel 5.4.1. Die Kantenmenge ist definiert als $A = \{(i, j) : u_i, u_j \in V\}$. Am Depot stehen zur Belieferung der Kunden m Drohnen beziehungsweise Fahrzeuge zur Verfügung. Die Definitionen der Zeitfenster sowie der Servicezeiten der Kunden und des Depots bleiben unverändert. Die Länge einer Kante $(i, j) \in A$ ist als d_{ij} definiert. Die Geschwindigkeit auf einer Kante $(i, j) \in A$ ist als v_{ij} definiert, wobei $v_{ij} \in [v_{min}, v_{max}]$ gelten muss. Die Zeit für eine Kante $(i, j) \in A$ mit Geschwindigkeit v_{ij} beträgt weiterhin d_{ij}/v_{ij} . Der Energieverbrauch auf einer Kante $(i, j) \in A$ in Abhängigkeit von der Geschwindigkeit v_{ij} beträgt $d_{ij}f(v_{ij})$, wobei f eine strikt konvexe Energieverbrauchsfunktion ist. Die Definition der optimalen Geschwindigkeit v_{opt} bleibt gleich zur Gleichung (5.5).

Die Berechnung der Belieferungszeitpunkte der Kunden einer Sequenz wird an den neuen Graphen angepasst. Gegeben sei eine Sequenz $s = (u_1, u_2, \dots, u_h)$ und ein dazugehöriger Geschwindigkeitsvektor $\mathbf{v} = (v_{0,1}, v_{1,2}, \dots, v_{h,0})$. Der Belieferungszeitpunkt t_i eines Kunden $u_i \in s$ kann rekursiv definiert werden als

$$t_i = \max(a_i, t_{i-1} + \tau_{i-1} + d_{i-1,i}/v_{i-1,i}),$$

wobei $t_0 = 0$ gilt.

Die Kosten einer Sequenz $s = (u_1, u_2, \dots, u_h)$ mit einem Vektor $\mathbf{v} = (v_{0,1}, v_{1,2}, v_{2,0}, \dots, v_{h,0})$ für eine Funktion f berechnen sich wie folgt

$$\sum_{i=1}^h d_{i-1,i}f(v_{i-1,i}) + d_{h,0}f(v_{h,0}).$$

Im Folgenden werden die Definitionen eines zulässigen Geschwindigkeitsvektors und einer unzulässigen Sequenz sowie eines optimalen Geschwindigkeitsvektors vorgestellt.

Definition 14. Ein Geschwindigkeitsvektor $\mathbf{v}$ zu einer Sequenz s wird als *zulässig* bezeichnet, wenn alle Kunden aus s innerhalb ihrer Zeitfenster beliefert werden und die Geschwindigkeiten des Vektors $\mathbf{v}$ im Intervall $[v_{\min}, v_{\max}]$ liegen.

Definition 15. Eine Sequenz s wird als *unzulässig* bezeichnet, wenn es keinen zulässigen Geschwindigkeitsvektor $\mathbf{v}$ zu s gibt.

Definition 16. Ein Geschwindigkeitsvektor $\mathbf{v}^* = (v_{0,1}^*, v_{1,2}^*, \dots, v_{h,0}^*)$ zu einer Sequenz $s = (u_1, u_2, \dots, u_h)$ und einer Funktion f wird als *optimal* bezeichnet, wenn $\mathbf{v}$ zulässig ist und es gilt

$$\begin{aligned} & \sum_{i=1}^h d_{i-1,i} f(v_{i-1,i}^*) + d_{h,0} f(v_{h,0}^*) \\ & \leq \min \left\{ \sum_{i=1}^h d_{i-1,i} f(v_{i-1,i}) + d_{h,0} f(v_{h,0}) \mid \mathbf{v} = (v_{0,1}, v_{1,2}, \dots, v_{h,0}) \text{ ist zulässig zu } s \right\}. \end{aligned}$$

Damit lässt sich das VRPTWSO wie folgt beschreiben. Es müssen m Sequenzen mit dazugehörigen zulässigen Geschwindigkeitsvektoren gefunden werden, sodass die Summe der Kosten der Sequenzen minimal ist und die folgenden Restriktionen eingehalten werden:

- (i) jeder Kunde ist in genau einer Sequenz enthalten und
- (ii) alle Geschwindigkeitsvektoren sind zulässig.

Nach der Einführung des VRPTWSO wird die Reduzierung beschrieben. Damit eine optimale Lösung des VRPTWSO in eine optimale Lösung für die Problemstellung aus Unterkapitel 5.4.1 überführt werden kann, sind folgende Schritte nötig. Die Distanzen in G müssen wie folgt, angepasst werden. Es gilt $d_{ij} = d_{i0} + d_{0j}$, wobei d_{i0} und d_{0j} die Distanzen gemäß Unterkapitel 5.4.1 beschreiben. Die Länge der Kanten $(i, 0), (0, i) \in A$ mit $u_i \in C$ bleibt gleich zum Graph $\tilde{G}$. Die Notation der Geschwindigkeit wird an den neuen Graphen angepasst. Die Geschwindigkeit v_{ij} auf der Kante $(i, j) \in A$, mit $u_i, u_j \in C$, kann für die Kanten $(i, 0), (0, j) \in \tilde{A}$ übernommen werden. Dies ist möglich, da nach Lemma 8 und Beobachtung 3 gilt, dass die Geschwindigkeit zwischen zwei Kunden in einer optimalen Lösung einheitlich sein sollte. Für die Kanten $(i, 0), (0, i) \in A$ kann die Geschwindigkeit auf $(i, 0), (0, i) \in \tilde{A}$ direkt übernommen werden. Damit entspricht eine optimale Lösung des VRPTWSO auch einer optimalen Lösung des Ausgangsproblems. Im folgenden Abschnitt wird das Problem als ganzzahliges Programm formuliert. Im weiteren Verlauf dieses Kapitels werden die Notationen des VRPTWSO verwendet.

5.4.3 Ganzzahliges lineares Programm

In Anlehnung an Fukasawa et al. (2018) wird das VRPTWSO als IP formuliert. Sei Ω die Menge aller zulässigen Sequenzen. Der binäre Koeffizient e_{is} gibt an, ob Kunde $u_i \in C$ in Sequenz $s \in \Omega$ enthalten ist. Der Koeffizient $c_f(s)$ gibt die Kosten einer Sequenz $s \in \Omega$ mit einem optimalen Geschwindigkeitsvektor für die Funktion f an. Die Entscheidungsvariable x_s gibt an, ob Sequenz $s \in \Omega$ Teil der Lösung ist. Damit lässt sich das Problem wie folgt formulieren:

$$\text{minimiere} \quad \sum_{s \in \Omega} c_f(s) x_s \quad (5.8)$$

$$\text{u.d.N.} \quad \sum_{s \in \Omega} e_{is} x_s = 1 \quad \forall u_i \in C, \quad (5.9)$$

$$\sum_{s \in \Omega} x_s \leq m, \quad (5.10)$$

$$x_s \in \{0, 1\} \quad \forall s \in \Omega. \quad (5.11)$$

Die Zielfunktion (5.8) minimiert die Gesamtkosten der ausgewählten Sequenzen. Die Nebenbedingungen (5.9) stellen sicher, dass jeder Kunde in genau einer ausgewählten Sequenz enthalten ist. Mit Ungleichung (5.10) wird die Anzahl der verfügbaren Drohnen berücksichtigt. Die Ganzzahligkeit der Entscheidungsvariablen x_s , für $s \in \Omega$, wird mit Nebenbedingung (5.11) garantiert. Im nächsten Unterkapitel wird gezeigt, wie sich für eine Sequenz ein optimaler Geschwindigkeitsvektor bestimmen lässt.

5.5 Geschwindigkeitsoptimierung für Sequenzen

In diesem Unterkapitel wird die Minimierung des Energieverbrauchs für eine Sequenz betrachtet. Dazu werden zwei spezifische Probleme genauer untersucht. Im Unterkapitel 5.5.1 wird die Berechnung der optimalen Geschwindigkeiten für eine Sequenz betrachtet. Dazu erfolgt zuerst die Vorstellung eines nichtlinearen Programms (NLP). Anschließend wird gezeigt, dass diese Berechnung auch mit einem exakten dynamischen Programm durchgeführt werden kann. Das dynamische Programm wird in Anlehnung an Hvattum et al. (2013) vorgestellt. Das Programm berücksichtigt die untere Geschwindigkeitsbegrenzung v_{min} nicht. Daher wird eine Erweiterung des Programms vorgestellt, die v_{min} in Bezug auf die optimale Geschwindigkeit v_{opt} berücksichtigt. Diese Erweiterung wird von Hvattum et al. (2013) nicht beschrieben. Für die Erweiterung erfolgt im Anschluss der Beweis der Korrektheit.

Im Unterkapitel 5.6.4 wird ein Algorithmus vorgestellt, der für eine vorgegebene Sequenz alle relevanten Belieferungszeitpunkte des letzten Kunden berechnet. Zudem werden die dazugehörigen optimalen Geschwindigkeitsvektoren und die entstehenden Kosten berechnet. Die Korrektheit dieses Algorithmus wird ebenfalls im Anschluss bewiesen. Beide Algorithmen sind von zentraler Bedeutung für den Branch-and-Price-Algorithmus, der im Unterkapitel 5.6 vorgestellt wird.

Da in diesem Unterkapitel Sequenzen betrachtet werden, wird die Notation wie folgt vereinfacht. Sei s eine Sequenz und seien $u_{i-1}, u_i \in s$ zwei aufeinanderfolgende Kunden in s . Die Distanz von u_{i-1} nach u_i wird als $d_i = d_{i-1,i}$ definiert. Für den ersten Kunden u_1 in s gilt $d_1 = d_{0,1}$. Auch die Notation der Geschwindigkeiten wird vereinfacht. Die Geschwindigkeit von u_{i-1} nach u_i wird als $v_i = v_{i-1,i}$ definiert, und es gilt $v_1 = v_{0,1}$. Außerdem wird die Geschwindigkeit auf der Kante $(i, j) \in A$ mit $u_j \in C$ als die Geschwindigkeit des Kunden u_j bezeichnet. Außerdem gilt für einen optimalen Geschwindigkeitsvektor $\mathbf{v} = (v_{0,1}, v_{1,2}, \dots, v_{h-1,h}, v_{h,0})$, dass der letzte Eintrag v_{opt} sein muss, da das Zeitfenster des Depots nicht restriktiv ist. Dies bedeutet, dass die Drohnen nach der Belieferung des letzten Kunden mit der optimalen Geschwindigkeit zum Depot fliegen können. Im Folgenden wird der letzte Eintrag von $\mathbf{v}$ zur Vereinfachung nicht explizit aufgeführt.

5.5.1 Optimale Geschwindigkeiten für Sequenzen

Die Berechnung der optimalen Geschwindigkeiten für eine Sequenz kann als NLP formuliert werden. Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und seien t_i , für $i = 1, \dots, h$, kontinuierliche Variablen, die den Belieferungszeitpunkt von Kunden $u_i \in s$ beschreiben. Es wird angenommen, dass die Drohne zum Zeitpunkt $t_0 = 0$ vom Depot startet. Des Weiteren sind kontinuierliche Variablen v_i , für $i = 1, \dots, h$, gegeben, die die Geschwindigkeit der Drohne auf der Kante von Knoten u_{i-1} nach u_i beschreiben. Alle weiteren Parameter des NLP entsprechen denjenigen aus Unterkapitel 5.4. Das NLP lässt sich somit wie folgt formulieren:

$$\begin{array}{ll} \text{minimiere} & \sum_{i=1}^h d_i f(v_i) \end{array} \quad (5.12)$$

$$\text{u.d.N.} \quad t_{i-1} + \tau_{i-1} + d_i/v_i \leq t_i \quad \forall i = 1, \dots, h, \quad (5.13)$$

$$a_i \leq t_i \leq b_i \quad \forall i = 1, \dots, h, \quad (5.14)$$

$$v_{min} \leq v_i \leq v_{max} \quad \forall i = 1, \dots, h. \quad (5.15)$$

Die Zielfunktion (5.12) beschreibt den Energieverbrauch der Drohne für die Sequenz s . Die Nebenbedingungen (5.13) garantieren, dass die Belieferungszeitpunkte von zwei aufeinanderfolgenden Kunden korrekt gesetzt werden. Die Nebenbedingungen (5.14) stellen sicher, dass die Belieferung der Kunden innerhalb ihrer Zeitfenster erfolgt, und Nebenbedingungen (5.15) gewährleisten, dass die Drohne nur mit den erlaubten Geschwindigkeiten fliegt.

Zur Beschreibung des dynamischen Programms wird eine Kostenfunktion $\tilde{f}$ eingeführt. Dabei ist $\tilde{f}$ monoton steigend und konvex. Die Funktion $\tilde{f}$ wird aus der Funktion f abgeleitet und berechnet sich wie folgt:

$$\tilde{f}(v) = \begin{cases} f(v_{opt}) & \text{falls } v \leq v_{opt}, \\ f(v) & \text{falls } v > v_{opt}. \end{cases} \quad (5.16)$$

In Unterkapitel 5.2 ist in Abbildung 5.3 ein Beispiel für die Funktion f (durchgezogene Kurve) und die daraus abgeleitete Funktion $\tilde{f}$ (zuerst gestrichelt und dann durchgezogene Kurve) dargestellt.

Die Darstellung des dynamischen Programms erfolgt in Anlehnung an Hvattum et al. (2013). Neben der Vorstellung des Programms wird in dieser Veröffentlichung auch der Beweis der Korrektheit des Algorithmus für eine konvexe, monoton steigende Funktion $\tilde{f}$ erbracht. Die zentrale Idee des dynamischen Programms besteht darin, die Geschwindigkeiten innerhalb der Sequenz möglichst gleich zu halten. Das Programm basiert auf einem sogenannten *Teile-und-herrsche-Verfahren*. Im ersten Schritt werden die Geschwindigkeiten der betrachteten Sequenz gleichgesetzt. Im nächsten Schritt wird das Zeitfenster mit der größten Verletzung betrachtet. Eine Verletzung bedeutet in diesem Zusammenhang, dass die Drohne entweder zu früh oder zu spät am Kunden ankommt und damit das erlaubte Zeitfenster überschreitet. Die Größe der Verletzung wird als zeitliche Differenz zwischen dem tatsächlichen Eintreffen am Kunden und dem Beginn oder Ende des Zeitfensters berechnet. Anschließend wird die Sequenz am Kunden mit dem verletzten Zeitfenster aufgeteilt und das Programm für die Teilsequenzen erneut ausgeführt. Dabei wird der Belieferungszeitpunkt des verletzten Kunden für beide Teilsequenzen auf die nächste Zeitfenstergrenze gesetzt. Die näher gelegene Zeitfenstergrenze ist diejenige, bei der die obere Grenze bei zu später Belieferung und die untere Grenze bei zu früher Belieferung gewählt wird. Für die Teilsequenzen werden neue, einheitliche Geschwindigkeiten festgelegt und deren Zulässigkeit überprüft. Auf diese Weise wird die Zulässigkeit schrittweise erzeugt.

Im Folgenden wird der Pseudocode, der in Algorithmus 4 dargestellt ist, beschrieben. Die Eingabe des Algorithmus sind die maximale Geschwindigkeit v_{max} , der Start- und Endzeitpunkt sowie die Distanzen, Zeitfenster und Servicezeiten der Kunden. Die Ausgabe des Algorithmus ist entweder ein optimaler Geschwindigkeitsvektor $\mathbf{v}$ für Sequenz s oder die Sequenz wird als *unzulässig* bewertet. Die Geschwindigkeiten werden aus dem Intervall $[0, v_{max}]$ gewählt.

In Zeile 4 wird eine einheitliche Durchschnittsgeschwindigkeit ermittelt. Diese berechnet sich als Summe der Distanzen, geteilt durch die Differenz zwischen Start- und Endzeitpunkt, abzüglich der Summe der Servicezeiten. Wenn die Durchschnittsgeschwindigkeit kleiner oder gleich 0 oder größer als v_{max} ist, wird in Zeile 3 *unzulässig* zurückgegeben. Anschließend werden die Belieferungszeitpunkte auf Grundlage des Startzeitpunkts und der Durchschnittsgeschwindigkeit berechnet. Dabei wird überprüft, ob Zeitfenster verletzt werden (siehe Zeilen 6 bis 13). Kommt es zu einer Verletzung eines Zeitfensters, wird der Index des Kunden mit der größten Verletzung sowie sein Belieferungszeitpunkt gespeichert. Ist der Zeitpunkt der Belieferung später als die obere Zeitfenstergrenze, wird der Belieferungszeitpunkt auf die obere Grenze gesetzt. Bei zu früher Belieferung wird der Zeitpunkt auf die untere Grenze gesetzt. Im Anschluss wird die Sequenz am verletzten Kunden aufgeteilt und der Algorithmus für beide Teilsequenzen erneut ausgeführt (siehe Zeilen 14 bis 22). Die erste Teilsequenz reicht dabei bis zum verletzten Kunden und endet an der korrigierten Zeitfenstergrenze. Die zweite Teilsequenz beginnt nach dem verletzten Kunden zum Zeitpunkt der korrigierten Zeitfenstergrenze zuzüglich der Servicezeit. Dieser Teile-und-Herrsche-Ansatz wird fortgesetzt, bis alle Zeitfenster eingehalten werden oder

Algorithmus 4 Berechnung eines optimalen Geschwindigkeitsvektors für eine Sequenz

Eingabe: Maximale Geschwindigkeit v_{max} , Startzeit t_s , Endzeit t_e , Distanzen $d_1, \dots, d_h$,
Zeitfenster $[a_i, b_i]$, Servicezeiten τ_i für alle $i = 1, \dots, h$
Ausgabe: Geschwindigkeitsvektor $\mathbf{v} = (v_1, v_2, \dots, v_h)$

```

1:  $v \leftarrow (\sum_{i=1}^h d_i) / (t_e - t_s - \sum_{i=1}^h \tau_i)$ 
2: falls  $v < 0$  or  $v_{max} < v$  dann
3:   Rückgabe unzulässig
4: ende falls
5:  $\Delta \leftarrow 0, j \leftarrow 0, t_0 \leftarrow t_s$ 
6: für alle  $i = 1$  to  $h$  mache
7:    $t_i \leftarrow t_{i-1} + \tau_{i-1} + d_i / v$ 
8:    $v_i \leftarrow v$ 
9:    $\delta_i \leftarrow \max\{0, t_i - b_i, a_i - t_i\}$ 
10:  falls  $\delta_i > \Delta$  dann
11:     $\Delta \leftarrow \delta_i, j \leftarrow i$ 
12:  ende falls
13: ende für alle
14: falls  $\Delta > 0$  dann
15:   falls  $t_j > b_j$  dann
16:     $t_j \leftarrow b_j$ 
17:   else falls  $t_j < a_j$  dann
18:     $t_j \leftarrow a_j$ 
19:   ende falls
20:   Algorithmus 4 ( $v_{max}, t_s, t_j, d_i, a_i, b_i, \tau_i$  für alle  $i = 1, \dots, j$ )
21:   Algorithmus 4 ( $v_{max}, t_j + \tau_j, t_e, d_i, a_i, b_i, \tau_i$  für alle  $i = j + 1, \dots, h$ )
22: ende falls

```

eine Unzulässigkeit festgestellt wird.

Zur Veranschaulichung des Algorithmus wird im Folgenden ein Beispiel zur Geschwindigkeitsoptimierung einer Sequenz mit fünf Kunden präsentiert.

Beispiel 3. Es werden ein Depot und eine Sequenz $s = (u_1, u_2, u_3, u_4, u_5)$ mit fünf Kunden betrachtet. Die Kunden werden der Reihe nach beliefert und der früheste Abflugzeitpunkt vom Depot ist 0. In diesem Beispiel beträgt die maximale Geschwindigkeit 25. Die minimale Geschwindigkeit wird gemäß den Vorgaben des Algorithmus auf Null festgelegt. Die Kostenfunktion wird durch $f(v) = v^2/1.000$ beschrieben. In Tabelle 5.3 werden die weiteren Parameter übersichtlich dargestellt. In der ersten Spalte sind die Indizes der Kunden aufgelistet. Spalte 2 enthält die Distanzen zu den Kunden und zwischen diesen, während Spalten 3 und 4 die jeweiligen Zeitfenster abbilden. In diesem Beispiel wird von einer Servicezeit von Null für alle Kunden ausgegangen. Die weiteren Spalten sind paarweise angeordnet, wobei die linke Spalte den Geschwindigkeitsvektor und die rechte die resultierenden Ankunftszeitpunkte darstellt. Die unterste Zeile zeigt die resultierenden Kosten der verschiedenen Geschwindigkeitsvektoren.

Da sich der Algorithmus selbst aufruft und bei einer Verletzung die Sequenz in zwei Teilsequenzen aufteilt, wird sein Vorgehen in einem binären Baum dargestellt. Abbildung 5.5 zeigt diesen Baum. Jeder Knoten im Baum enthält eine Sequenz oder Teilsequenz, die durch Start- und Endzeitpunkt definiert ist. Die Berechnungen des Algorithmus erfolgen innerhalb eines Knotens im Baum. Für

i	d_i	a_i	b_i	Tiefe 0		Tiefe 1		Tiefe 2	
				v_i	t_i	v_i	t_i	v_i	t_i
1	600	100	150	6	100	10	60	6	100
2	1.200	150	200	6	300	10	180	12	200
3	1.200	220	300	6	500	10	300	12	300
4	800	250	500	6	633,33	4	500	4	500
5	1.600	270	900	6	900	4	900	4	900
Kosten				194,4		338,4		405,6	

Tabelle 5.3: Übersicht zu Beispiel 3

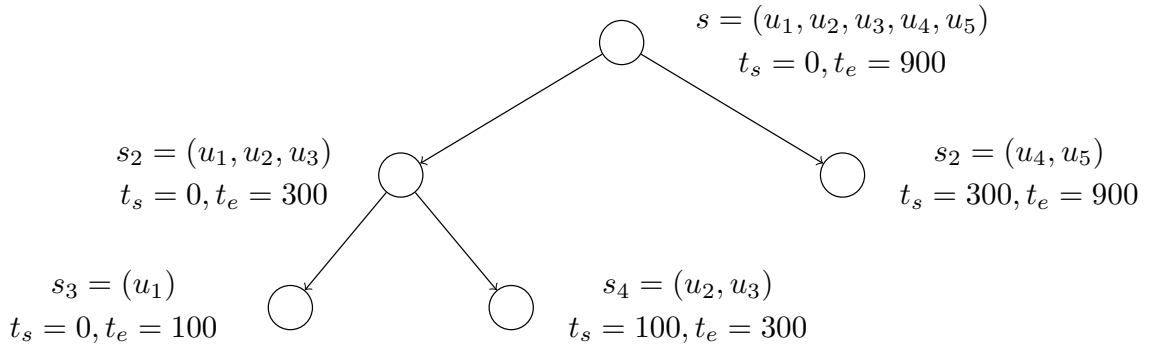


Abbildung 5.5: Binärer Baum zur Visualisierung der Teilprobleme

die Berechnung des optimalen Geschwindigkeitsvektors sind in diesem Beispiel fünf Iterationen erforderlich. In der Wurzel des Baums liegen der Startzeitpunkt bei 0 und der Endzeitpunkt bei 900. Letzterer entspricht der oberen Zeitfenstergrenze von Kunde u_5 . Es wird die gesamte Sequenz s berücksichtigt. Die resultierende Durchschnittsgeschwindigkeit beträgt 6. Bei dieser Geschwindigkeit findet die Belieferung der Kunden u_2 , u_3 und u_4 nicht innerhalb ihrer Zeitfenster statt. Die größte Verletzung liegt bei u_3 vor, mit einem Ankunftszeitpunkt von 500 bei einer oberen Zeitfenstergrenze von 300. Daher wird die Sequenz bei u_3 in $s_1 = (u_1, u_2, u_3)$ und $s_2 = (u_4, u_5)$ aufgeteilt. Der Belieferungszeitpunkt für Kunde u_3 wird auf 300 festgelegt.

Für die erste Teilsequenz s_1 sind der Startzeitpunkt 0 und der Endzeitpunkt 300. Für s_1 beträgt die Durchschnittsgeschwindigkeit 4. Bei dieser Geschwindigkeit wird die untere Zeitfenstergrenze von Kunde u_1 unterschritten, da die Drohne zum Zeitpunkt 60 eintrifft. Das Zeitfenster von Kunde u_1 beginnt jedoch erst bei 100. Daher wird s_1 erneut in die Teilsequenzen $s_3 = (u_1)$ und $s_4 = (u_2, u_3)$ aufgeteilt. Anschließend wird Algorithmus 4 für die Teilsequenzen erneut aufgerufen. Für die Teilsequenz $s_2 = (u_4, u_5)$ liegen der Startzeitpunkt bei 300 und der Endzeitpunkt bei 900. Die Durchschnittsgeschwindigkeit beträgt 10. Bei dieser Geschwindigkeit werden beide Kunden innerhalb ihrer Zeitfenster beliefert. Die Ergebnisse für die Sequenzen s_1 und s_2 sind in Tabelle 5.3 in den Spalten sieben und acht dargestellt.

In Tiefe 2 des Baums werden die Sequenzen s_3 und s_4 betrachtet. Für $s_3 = (u_1)$ gilt, dass der Startzeitpunkt bei 0 liegt und der Endzeitpunkt bei 100 liegt. Daraus ergibt sich eine Geschwindigkeit von 6 und der Kunde u_1 wird am unteren Rand seines Zeitfensters beliefert. Für die Teilsequenz $s_4 = (u_2, u_3)$ gilt, dass der Startzeitpunkt bei 100 und der Endzeitpunkt bei 300 liegt.

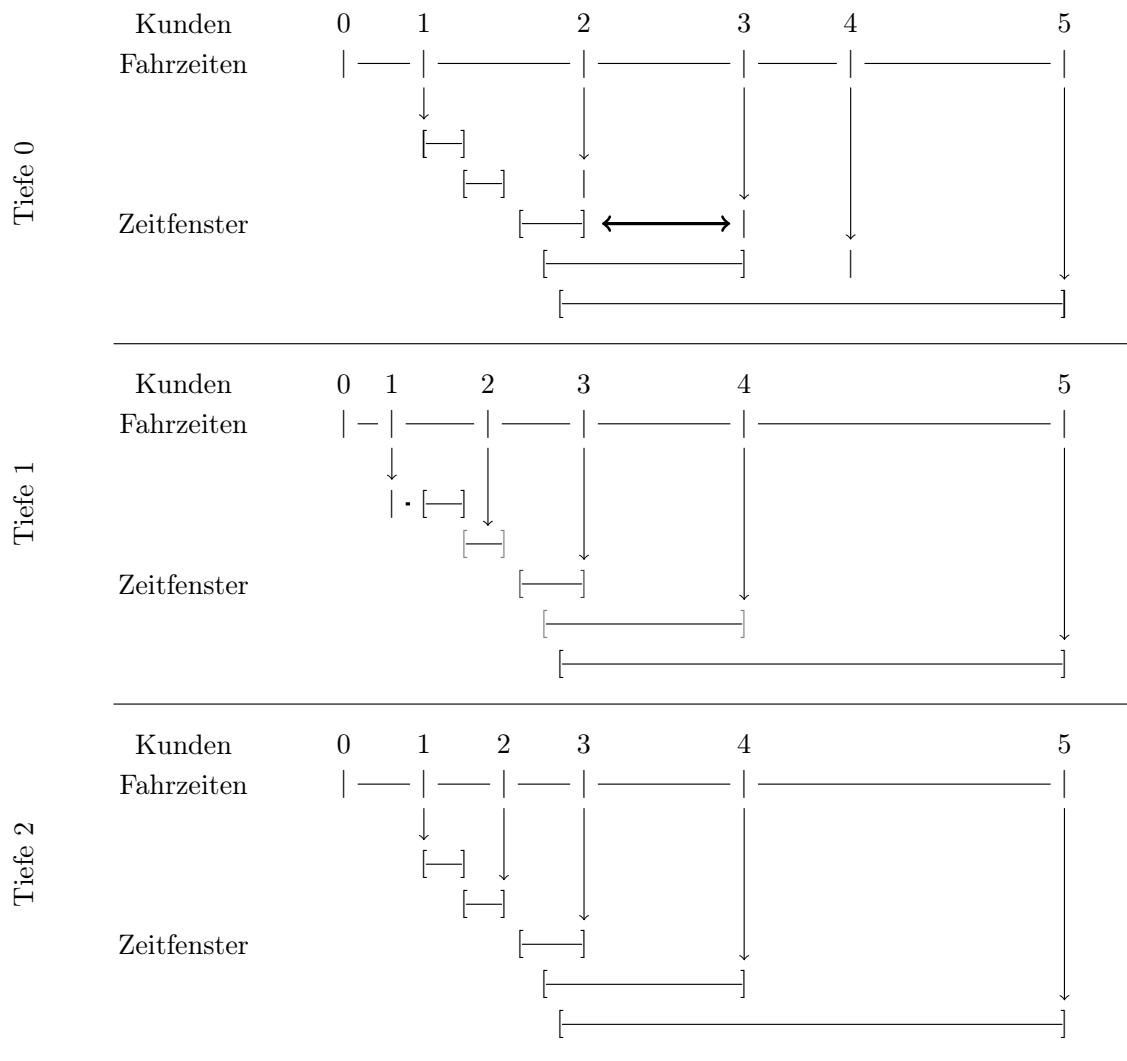


Abbildung 5.6: Visualisierung der Ergebnisse des Geschwindigkeitsalgorithmus

Die resultierende Geschwindigkeit beträgt 12 und beide Kunden werden innerhalb ihrer Zeitfenster beliefert. Infolgedessen wird der Algorithmus beendet. Der endgültige Geschwindigkeitsvektor lautet somit $\mathbf{v} = (6, 12, 12, 4, 4)$ und führt zu Kosten in Höhe von 405,6 (siehe Spalte neun und zehn von Tabelle 5.3).

In Abbildung 5.6 sind die einzelnen Schritte visualisiert. Die Abbildung ist in die drei Tiefen des Baums gegliedert, wobei jede Tiefe nach dem gleichen Muster aufgebaut ist. Im oberen Teil sind die Indizes der Kunden dargestellt, gefolgt von den Fahrzeiten und Ankunftszeitpunkten. Im unteren Bereich jeder Tiefe sind die Zeitfenster sowie die Ankunftszeitpunkte der Kunden eingezeichnet. Damit ist die Vorstellung des Beispiels abgeschlossen.

Algorithmus 5 Geschwindigkeitsoptimierung mit oberer und unterer Geschwindigkeitsgrenze**Eingabe:** Maximale Geschwindigkeit v_{max} , optimale Geschwindigkeit v_{opt} , Startzeit t_s ,Endzeit t_e , Distanzen d_i , Zeitfenster $[a_i, b_i]$, Servicezeiten τ_i für alle $i = 1, \dots, h$ **Ausgabe:** Geschwindigkeitsvektor $\mathbf{v}^* = (v_1^*, v_2^*, \dots, v_h^*)$

- 1: **falls** Algorithmus 4 *unzulässig* zurückgibt **dann**
- 2: **Rückgabe** Kein zulässiger Geschwindigkeitsvektor existiert
- 3: **ende falls**
- 4: $(\tilde{v}_1, \tilde{v}_2, \dots, \tilde{v}_h) \leftarrow \text{Algorithmus 4 } (v_{max}, t_s, t_e, d_i, a_i, b_i, \tau_i \text{ für alle } i = 1, \dots, h)$
- 5: $v_i^* \leftarrow \max\{v_{opt}, \tilde{v}_i\}$ für alle $i = 1, \dots, h$
- 6: **Rückgabe** $\mathbf{v}^*$

Wie eingangs erwähnt, berücksichtigt Algorithmus 4 nicht die minimale Geschwindigkeit v_{min} und ist nur für monoton steigende Funktionen geeignet. Daher wird eine Erweiterung beschrieben, die die strikt konvexe Kostenfunktion f und die Mindestgeschwindigkeit v_{min} berücksichtigt. Die Idee des Algorithmus besteht darin, Algorithmus 4 zu verwenden, um einen optimalen Geschwindigkeitsvektor für die Funktion $\tilde{f}$ zu berechnen. Anschließend werden die Einträge des Vektors einzeln überprüft, um daraus einen optimalen Geschwindigkeitsvektor für die Funktion f zu erstellen.

Der Pseudocode dieser Erweiterung wird in Algorithmus 5 dargestellt. Die Eingabe des Algorithmus ist identisch mit der von Algorithmus 4, mit der Ausnahme, dass zusätzlich die optimale Geschwindigkeit v_{opt} übergeben wird. Die Ausgabe des Algorithmus ist ein optimaler Geschwindigkeitsvektor $\mathbf{v}^*$ für die Funktion f , der nur Geschwindigkeiten im Intervall $[v_{min}, v_{max}]$ enthält. Im ersten Schritt wird in Zeile 1 mit Algorithmus 4 überprüft, ob eine zulässige Lösung existiert. Falls dies nicht der Fall ist, wird in Zeile 2 zurückgegeben, dass keine zulässige Lösung vorliegt. Andernfalls wird in Zeile 4 mit Algorithmus 4 ein optimaler Geschwindigkeitsvektor für die Funktion $\tilde{f}$, im Bereich $[0, v_{max}]$ berechnet. Dann wird in Zeile 6 jeder Eintrag $\tilde{v}_i$ des Vektors für alle $u_i \in s$ geprüft und auf das Maximum von $\tilde{v}_i$ und v_{opt} gesetzt. Im letzten Schritt wird der Vektor zurückgegeben.

Nach der Einführung des Algorithmus wird der Korrektheitsbeweis durchgeführt. Im Folgenden wird ein gegebener optimaler Geschwindigkeitsvektor $\mathbf{v}$ für die Funktion f , der nur Geschwindigkeiten aus dem Intervall $[v_{min}, v_{max}]$ enthält, mit der Rückgabe von Algorithmus 5 verglichen. Für den Korrektheitsbeweis wird angenommen, dass alle Servicezeiten null sind. Diese Annahme ist ohne Beschränkung der Allgemeinheit möglich, da die Reihenfolge der Kunden in s fest ist und eine Lösung ohne Servicezeiten durch Anpassung der Zeitfenster in eine Lösung mit Servicezeiten überführt werden kann. Hierzu werden die unteren und oberen Zeitfenstergrenzen jedes Kunden um die Summe der vorangegangenen Servicezeiten verkleinert. Diese Transformation hat weder Einfluss auf den Geschwindigkeitsvektor noch auf die Kosten der optimalen Lösung.

Zur Durchführung des Korrektheitsbeweises des Algorithmus werden im Folgenden einige Hilfsaussagen vorgestellt. Diese sind von zentraler Bedeutung für die Beweisführung. Die dazugehörigen Beweise finden sich im Anhang C. Zunächst werden die Lemmata 9 bis 10 sowie Lemma 11 eingeführt. Die Beweise der Lemmata 9 bis 10 orientieren sich an Hvattum et al. (2013), die die Korrektheit für die Kostenfunktion $\tilde{f}$ gezeigt haben. Die folgenden Lemmata formulieren zentrale

Aussagen zum Korrektheitsbeweis sowie zum Zusammenhang zwischen Geschwindigkeiten und Belieferungszeitpunkten innerhalb einer Kundensequenz. Tritt in einem optimalen Geschwindigkeitsvektor ein Wechsel der Geschwindigkeit zwischen zwei aufeinanderfolgenden Kunden auf, so erfolgt die Belieferung des ersten dieser beiden Kunden exakt auf einer seiner Zeitfenstergrenzen. Der Grund hierfür ist, dass somit die Geschwindigkeiten am weitesten angeglichen werden können.

Lemma 9. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $v = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt beliefert und dabei bezüglich f kostenminimal ist. Enthält die Sequenz zwei aufeinanderfolgende Kunden u_i und u_{i+1} , für die $v_i > v_{i+1}$ gilt, dann gilt $t_i = b_i$.*

Eine vergleichbare Aussage gilt auch für die untere Zeitfenstergrenze.

Lemma 10. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $v = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt beliefert und dabei bezüglich f kostenminimal ist. Enthält die Sequenz zwei aufeinanderfolgende Kunden u_i und u_{i+1} , für die $v_i < v_{i+1}$ gilt, dann gilt $t_i = a_i$.*

Nach der Einführung der Lemmata kann die Korrektheit der Erweiterung bewiesen werden. Die folgenden beiden Lemmata zeigen, dass ein von Algorithmus 5 berechneter Geschwindigkeitsvektor *zulässig* ist.

Lemma 11. *Sei $s = (u_1, \dots, u_h)$ eine Sequenz. Algorithmus 5 berechnet zulässige Geschwindigkeitsvektoren und die Belieferungszeitpunkte von Algorithmus 4 sind auch für den Vektor von Algorithmus 5 zulässig.*

Gibt der Algorithmus hingegen *unzulässig* zurück, existiert kein Geschwindigkeitsvektor für die Kundensequenz, der die Geschwindigkeitsgrenzen erfüllt.

Lemma 12. *Sei $s = (u_1, \dots, u_h)$ eine Sequenz. Gibt Algorithmus 5 unzulässig zurück, dann kann kein Geschwindigkeitsvektor zu s , der die Geschwindigkeitsgrenzen $[v_{\min}, v_{\max}]$ erfüllt, existieren.*

Damit lässt sich die Korrektheit des Algorithmus nachweisen.

Theorem 2. *Sei $s = (u_1, \dots, u_h)$ eine Sequenz. Algorithmus 5 berechnet einen optimalen Geschwindigkeitsvektor für s bezüglich der Funktion f für die Geschwindigkeitsgrenzen $[v_{\min}, v_{\max}]$, sofern ein solcher existiert.*

Nach der Einführung eines Algorithmus zur Berechnung eines optimalen Geschwindigkeitsvektors für eine Sequenz sowie dessen Erweiterung zur Berücksichtigung strikt konvexer Funktionen wird im nächsten Unterkapitel ein Algorithmus zur Berechnung aller relevanten Belieferungszeitpunkte des letzten Kunden einer Sequenz vorgestellt.

5.5.2 Berechnung aller Belieferungszeitpunkte des letzten Kunden

Dieser Abschnitt stellt einen Algorithmus zur Berechnung aller Belieferungszeitpunkte des letzten Kunden einer Sequenz vor. Zusätzlich zu den Belieferungszeitpunkten berechnet der Algorithmus auch die dazugehörigen Geschwindigkeitsvektoren sowie die minimalen Kosten basierend auf der strikt konvexen Kostenfunktion f . Es gelten weiterhin die Geschwindigkeitsgrenzen $[v_{min}, v_{max}]$. Der Algorithmus wird im Unterkapitel 5.6.4 zur Erstellung einer Dominanzregel im Pricing-Problem des Branch-and-Price-Algorithmus verwendet. Es wird weiterhin davon ausgegangen, dass die Belieferungszeitpunkte so früh wie möglich gewählt werden.

Vor der detaillierten Beschreibung des Algorithmus wird ein naiver Ansatz vorgestellt, bei dem lediglich die Geschwindigkeit des letzten Kunden erhöht wird. Dieser führt jedoch nicht zwangsläufig zu einer kostenminimalen Lösung. Abbildung 5.7 veranschaulicht die Kostenentwicklung im Vergleich zum Algorithmus 6 anhand der Sequenz aus Beispiel 3. Dabei zeigt sich, dass der naive Ansatz nicht nur höhere Kosten verursacht, sondern auch einen deutlich späteren Belieferungszeitpunkt (knapp unter 600 statt unter 400) liefert.

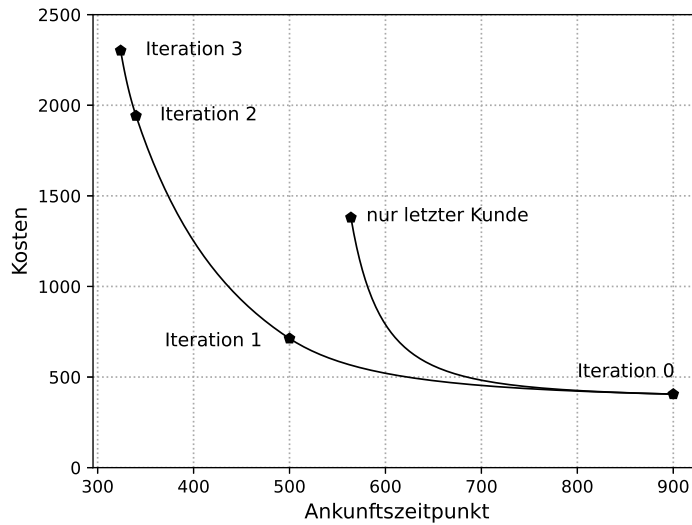


Abbildung 5.7: Verlauf der Kosten in Abhängigkeit des Belieferungszeitpunkts

Die Idee von Algorithmus 6 besteht darin, ausgehend von einem optimalen Geschwindigkeitsvektor (berechnet mit Algorithmus 5) den Belieferungszeitpunkt des letzten Kunden schrittweise zu reduzieren. Statt lediglich die Geschwindigkeit des letzten Kunden zu erhöhen, wird eine längste Teilsequenz s^* identifiziert, die gleichmäßig beschleunigt wird. Sie umfasst den letzten Kunden sowie alle vorhergehenden Kunden mit identischer Geschwindigkeit, die nicht auf ihrer unteren Zeitfenstergrenze beliefert werden. Während dieses Prozesses werden die Belieferungszeitpunkte, resultierenden Geschwindigkeitsvektoren und zugehörigen Kosten fortlaufend gespeichert. Die Geschwindigkeitserhöhung wird solange fortgesetzt, bis eine Abbruchbedingung erfüllt ist. Diese wird im folgenden Abschnitt detailliert im Pseudocode beschrieben. Danach wird s^* erneut bestimmt und der Vorgang wiederholt. Der Algorithmus terminiert, wenn der letzte Kunde seine

untere Zeitfenstergrenze erreicht oder die maximale Geschwindigkeit überschritten würde. Dabei gilt: Solange das Abbruchkriterium nicht greift, ist s^* stets nicht leer. Der vollständige Ablauf ist in Algorithmus 6 dargestellt und wird im nächsten Abschnitt erläutert.

Algorithmus 6 Berechnung aller relevanten Belieferungszeitpunkte des letzten Kunden einer Sequenz

Eingabe: Sequenz $s = (u_1, u_2, \dots, u_h)$, optimaler Geschwindigkeitsvektor $\mathbf{v}^* = (v_1^*, v_2^*, \dots, v_h^*)$, strikt konvexe Funktion f , Belieferungszeitpunkte t_i , Distanzen d_i , Zeitfenster $[a_i, b_i]$, Servicezeiten τ_i für alle $i = 1, \dots, h$

Ausgabe: Belieferungszeitfunktion arr , Kostenfunktion $cost$, Menge der Geschwindigkeitsvektoren $\bar{\mathcal{V}}$

```

1:  $v_j \leftarrow v_j^*$  für alle  $j = 1, \dots, h$ 
2:  $arr(v) = \begin{cases} t_h, & \text{für } v = v_h \\ \infty, & \text{andernfalls} \end{cases}$ 
3:  $cost(v) = \begin{cases} \sum_{j=1}^h d_j f(v_j), & \text{für } v = v_h \\ \infty, & \text{andernfalls} \end{cases}$ 
4:  $\bar{\mathcal{V}} \leftarrow \mathbf{v}^*$ 
5: während  $v_h < v_{max}$  und  $a_h < t_h$  mache
6:    $s^* \leftarrow u_h$ 
7:    $i \leftarrow h - 1$ 
8:   während  $v_i = v_h$  und  $t_i \neq a_i$  und  $i \neq 0$  mache
9:      $s^* \leftarrow u_i \cup s^*$ 
10:     $i \leftarrow i - 1$ 
11:  ende während
12:   $\tilde{v}_k \leftarrow \sum_{j=i+1}^k d_j / (a_k - t_i - \sum_{j=i}^{k-1} \tau_j)$  für alle  $u_k \in s^*$ 
13:   $\tilde{v} \leftarrow \min_{u_k \in s^*} \{\tilde{v}_k, v_{max}\}$ 
14:  falls  $i > 0$  und  $t_i > a_i$  und  $v_i < \tilde{v}$  dann
15:     $\tilde{v} \leftarrow v_i$ 
16:  ende falls
17:   $t_k \leftarrow t_i + \sum_{j=i+1}^k (\tau_{j-1} + d_j / \tilde{v})$  für alle  $u_k \in s^*$ 
18:   $arr(v) = \begin{cases} t_i + \sum_{j=i+1}^h (\tau_{j-1} + d_j / v), & \text{für } v \in (v_h, \tilde{v}] \\ arr(v), & \text{andernfalls} \end{cases}$ 
19:   $cost(v) = \begin{cases} \sum_{j=1}^i d_j f(v_j) + \sum_{j=i+1}^h d_j f(v), & \text{für } v \in (v_h, \tilde{v}] \\ cost(v), & \text{andernfalls} \end{cases}$ 
20:   $\bar{\mathcal{V}} \leftarrow \bar{\mathcal{V}} \cup \{\bar{\mathbf{v}} : \bar{\mathbf{v}}_j = v_j \ \forall u_j \in s \setminus s^* \wedge \bar{\mathbf{v}}_j = v \ \forall u_j \in s^*, v \in (v_h, \tilde{v}]\}$ 
21:   $v_j \leftarrow \tilde{v}$  für alle  $j = i + 1, \dots, h$ 
22: ende während
23: Rückgabe  $arr, cost, \bar{\mathcal{V}}$ 

```

Die Eingabe des Algorithmus umfasst eine Sequenz $s = (u_1, u_2, \dots, u_h)$, einen optimalen Geschwindigkeitsvektor $\mathbf{v}^*$ für s bezüglich f , die Kostenfunktion f sowie die frühesten Belieferungszeitpunkte t_i für jeden Kunden $u_i \in s$, die sich aus Vektor $\mathbf{v}^*$ ergeben. Zusätzlich werden die Distanzen zwischen den Kunden in s , deren Zeitfenster und die Servicezeiten als Eingabeparameter übergeben. Die Ausgabe des Algorithmus umfasst die zwei Funktionen arr und $cost$ sowie die Menge aller kostenminimalen Geschwindigkeitsvektoren $\bar{\mathcal{V}}$ für die Funktion f . Dabei beschreibt $arr(v)$ den Belieferungszeitpunkt des Kunden u_h in Abhängigkeit von dessen Geschwindigkeit v , während $cost(v)$ die resultierenden Kosten der Sequenz in Abhängigkeit von

der Geschwindigkeit des Kunden u_h angibt. In jeder Iteration wird ein Geschwindigkeitsvektor $\mathbf{v} = (v_1, v_2, \dots, v_h)$ betrachtet, der die zu Beginn der Iteration vorliegenden Geschwindigkeiten beschreibt. In den Zeilen 1 bis 4 findet die Initialisierung von $\mathbf{v}$, den Funktionen arr und $cost$ und der Menge $\bar{\mathcal{V}}$ statt. Dabei wird $\mathbf{v}$ auf den optimalen Geschwindigkeitsvektor $\mathbf{v}^*$ der Eingabe gesetzt, die Funktion arr enthält nur den Ankunftszeitpunkt an u_h mit $\mathbf{v}^*$, die Funktion $cost$ enthält die entsprechenden Kosten mit $\mathbf{v}^*$ und die Menge $\bar{\mathcal{V}}$ enthält $\mathbf{v}^*$. In den Zeilen 6 bis 11 erfolgt die Berechnung der Teilsequenz s^* . Die Berechnung basiert auf den folgenden Regeln:

- Der letzte Kunde ist in der Teilsequenz.
- Die in der Teilsequenz enthaltenen Kunden sind zusammenhängend, das bedeutet, Kunde u_i ist nur enthalten, wenn auch Kunde u_{i+1} enthalten ist.
- Alle Kunden in der Teilsequenz haben die gleiche Geschwindigkeit.
- Kein Kunde in der Teilsequenz wird auf seiner unteren Zeitfenstergrenze beliefert.

In den Zeilen 12 bis 16 wird die neue Geschwindigkeit $\tilde{v}$ für die Teilsequenz s^* berechnet. Diese ergibt sich als Minimum der drei folgenden Kriterien:

- Ein Kunde in s^* wird auf seiner unteren Zeitfenstergrenze beliefert.
- Die neue Geschwindigkeit entspricht der höchsten Geschwindigkeit v_{max} .
- Die neue Geschwindigkeit entspricht der Geschwindigkeit des Kunden, der der Teilsequenz unmittelbar vorausgeht.

Im letzten Teil der Iteration, in den Zeilen 17 bis 21, werden die Ankunftszeitpunkte aller Kunden aus s^* basierend auf $\tilde{v}$ neu berechnet. Zusätzlich werden die Funktionen arr und $cost$ für alle Geschwindigkeiten zwischen der zu Beginn der Iteration geltenden Geschwindigkeit v_h und $\tilde{v}$ erweitert. Abschließend werden alle Geschwindigkeitsvektoren dieser Iteration zu $\bar{\mathcal{V}}$ hinzugefügt und der Vektor $\mathbf{v}$ wird an die neue Geschwindigkeit angepasst. Dabei entspricht der neue Geschwindigkeitsvektor dem vorherigen, wobei alle Einträge, die den Kunden aus s^* entsprechen, auf $\tilde{v}$ gesetzt werden.

Der Algorithmus terminiert, wenn der letzte Kunde entweder mit der maximalen Geschwindigkeit beliefert wird oder auf seiner unteren Zeitfenstergrenze. Falls keines dieser Kriterien erfüllt ist, wird eine weitere Iteration durchgeführt. Der Algorithmus gibt die Funktionen arr und $cost$ sowie die Menge der Geschwindigkeitsvektoren $\bar{\mathcal{V}}$ zurück.

Es ist zu beachten, dass die Funktionen arr und $cost$ als Argument nur eine Geschwindigkeit v verwenden. Diese Geschwindigkeit v ist die Geschwindigkeit des letzten Kunden. Dabei gilt, dass für jede Geschwindigkeit v ein eindeutiger und kostenminimaler Geschwindigkeitsvektor für die Funktion f bestimmt werden kann, der in der Menge $\bar{\mathcal{V}}$ gespeichert ist. Dies ist möglich, weil in $\bar{\mathcal{V}}$ keine zwei Vektoren denselben letzten Eintrag haben. Zur Veranschaulichung von Algorithmus 6 wird im Folgenden ein Beispiel präsentiert, das auf Beispiel 3 aufbaut.

Beispiel 4. Gegeben sind ein Depot und eine Sequenz $s = (u_1, u_2, u_3, u_4, u_5)$ mit fünf Kunden. Die Kostenfunktion lautet $f(v) = v^2/1.000$, das zulässige Geschwindigkeitsintervall ist $[4, 25]$. Tabelle 5.4 gibt einen Überblick über die relevanten Parameter. Die erste Spalte enthält die Kundenindizes, die zweite die Distanzen zwischen den Knoten, Spalte drei und vier die jeweiligen Zeitfenster. Es wird angenommen, dass alle Servicezeiten null betragen. Die folgenden Spalten sind paarweise aufgebaut: links der jeweilige Geschwindigkeitsvektor, rechts die resultierenden Belieferungszeitpunkte. Das erste Paar zeigt die mit Algorithmus 5 berechnete optimale Lösung. Die übrigen Spalten enthalten die Ergebnisse der Iterationen von Algorithmus 6.

i	d_i	l_i	u_i	Iteration 0		Iteration 1		Iteration 2		Iteration 3	
				v_i	t_i	v_i	t_i	v_i	t_i	v_i	t_i
1	600	100	150	6	100	6	100	6	100	6	100
2	1.200	150	200	12	200	12	200	20	160	20	160
3	1.200	220	300	12	300	12	300	20	220	20	220
4	800	250	500	4	500	12	366,67	20	260	25	260
5	1.600	270	900	4	900	12	500	20	340	25	324

Tabelle 5.4: Übersicht zu Beispiel 4

Abbildung 5.8 zeigt die einzelnen Iterationen, aufgebaut analog zu Beispiel 3. Jede Iteration folgt demselben Schema und ist durch eine durchgehende Linie abgetrennt. Im oberen Bereich sind die Kunden dargestellt, darunter die jeweiligen Fahrzeiten und Belieferungszeitpunkte. Im unteren Teil finden sich die Zeitfenster sowie die zugehörigen Ankunftszeiten.

Zu Beginn des Algorithmus (Iteration 0) wird Kunde u_5 an der oberen Grenze seines Zeitfensters beliefert, der zugehörige Geschwindigkeitsvektor lautet $\mathbf{v}_0 = (6, 12, 12, 4, 4)$. Die erste Teilsequenz ist $s^* = (u_4, u_5)$, da beide Kunden mit Geschwindigkeit 4 angefahren werden. Diese wird auf 12 erhöht, entsprechend der Geschwindigkeit von u_3 , wodurch sich $\mathbf{v}_1 = (6, 12, 12, 12, 12)$ ergibt. In Iteration 2 umfasst s^* die Kunden u_2 bis u_5 , da sie alle dieselbe Geschwindigkeit haben und nicht auf der unteren Zeitfenstergrenze beliefert werden (siehe Abbildung 5.8). Die Geschwindigkeit wird auf 20 erhöht, da somit Kunde u_3 auf seiner unteren Zeitfenstergrenze beliefert wird. Dadurch entfällt u_3 aus der nächsten Teilsequenz, sodass $s^* = (u_4, u_5)$ gilt. Diese wird abschließend auf die Maximalgeschwindigkeit 25 erhöht, woraus $\mathbf{v}_3 = (4, 20, 20, 25, 25)$ resultiert. Mit diesem Vektor wird u_5 zum Zeitpunkt 324 beliefert. Eine weitere Reduktion ist nicht möglich, da u_4 und u_5 mit maximaler Geschwindigkeit angefahren werden und u_3 bereits auf der unteren Zeitfenstergrenze beliefert wird. Somit ist 324 der frühestmögliche Belieferungszeitpunkt für u_5 .

Abbildung 5.7 zeigt die Kosten der Sequenz in Abhängigkeit vom Belieferungszeitpunkt des letzten Kunden. Die Kosten steigen streng monoton mit jedem früheren Zeitpunkt und bilden eine konvexe Funktion. Abbildung 5.9 stellt die Geschwindigkeit des letzten Kunden in Abhängigkeit von dessen Belieferungszeitpunkt dar. Auch hier steigt die Geschwindigkeit streng monoton, jedoch ist die Funktion nicht konvex. Ursache sind die unterschiedlichen Weglängen, die in den Iterationen beschleunigt werden: In Iteration 1 beträgt die Weglänge 2.400, in Iteration 2 bereits 4.800 (siehe Tabelle 5.4). Daher sinkt der erforderliche Geschwindigkeitszuwachs pro Zeiteinheit bei späteren Iterationen. Damit ist das Beispiel abgeschlossen.

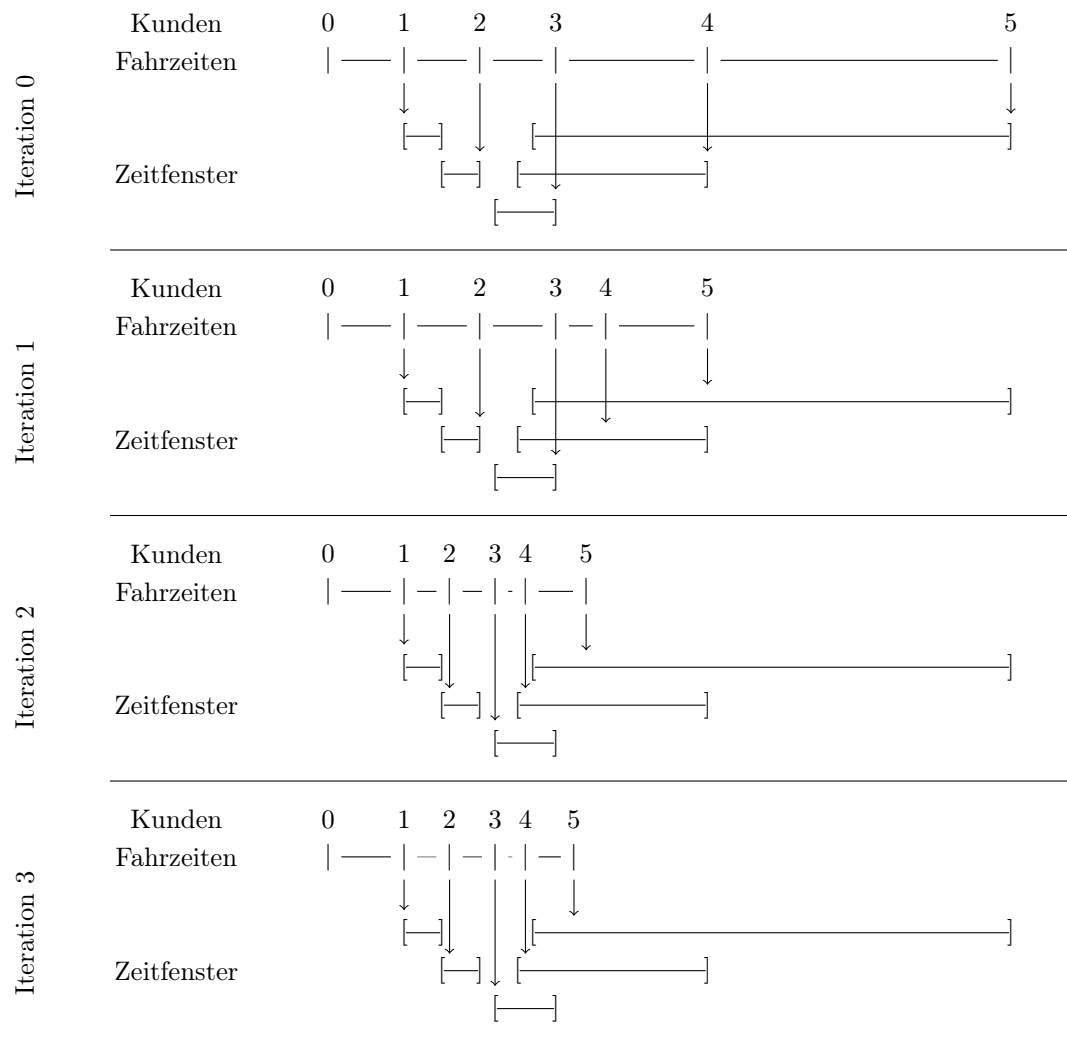


Abbildung 5.8: Visualisierung der Ergebnisse des Algorithmus 6

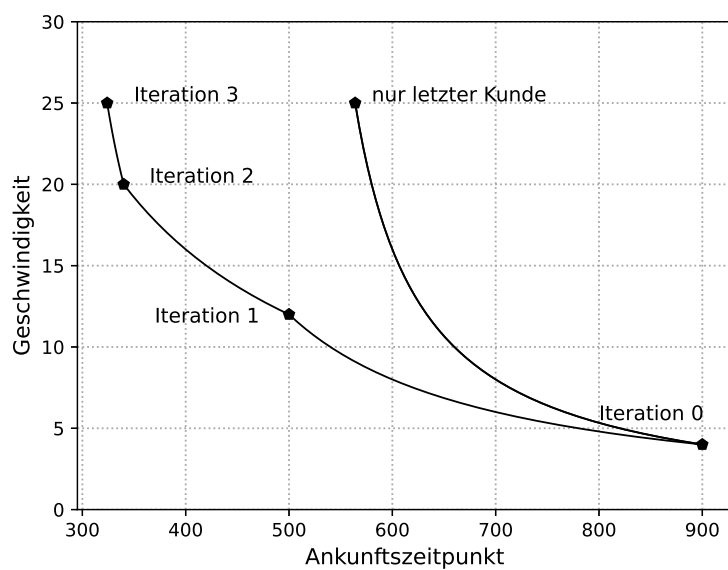


Abbildung 5.9: Entwicklung der Geschwindigkeit mit dem naiven Ansatz und Algorithmus 6

Es gilt noch zu zeigen, dass Algorithmus 6 korrekt ist. Zur Beweisführung der Korrektheit werden zunächst Hilfsaussagen eingeführt. Die Beweise der folgenden Aussagen befinden sich im Anhang C.

Beobachtung 4. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz. Gibt es Wartezeiten zwischen zwei Kunden u_i und $u_{i+1} \in s$, dann gilt $t_{i+1} = a_{i+1}$.*

Diese Beobachtung gilt, da sie andernfalls der Annahme widersprechen würde, dass die Belieferungszeiten der Kunden minimal gewählt sind. In den beiden folgenden Lemmata wird die Umkehrung der Aussagen von Lemmata 9 und 10 gezeigt.

Lemma 13. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und $\mathbf{v} = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt beliefert und bezüglich f kostenminimal ist. Falls für einen Kunden $u_i \in s \setminus u_h$ gilt $t_i = b_i$ und $a_i \neq b_i$, dann gilt $v_i \geq v_{i+1}$.*

Lemma 14. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und $\mathbf{v} = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt beliefert und bezüglich f kostenminimal ist. Falls für einen Kunden $u_i \in s \setminus u_h$ gilt $t_i = a_i$ und $a_i \neq b_i$, dann gilt $v_i \leq v_{i+1}$.*

Für den Korrektheitsbeweis werden zusätzliche Notationen eingeführt. Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und $\mathbf{v} = (v_1, \dots, v_h)$ ein Geschwindigkeitsvektor. Angenommen, die Sequenz s kann zusammen mit $\mathbf{v}$ wie folgt partitioniert werden. Sei $\bar{s}(\mathbf{v}) = (u_{l+1}, \dots, u_h)$ die längste zusammenhängende Teilsequenz von s , für die gilt:

- $u_h \in \bar{s}(\mathbf{v})$,
- $v_i \geq v_{i+1}$ für $u_i, u_{i+1} \in \bar{s}(\mathbf{v})$,
- $t_i \neq a_i$ für $u_i \in \bar{s}(\mathbf{v})$.

Die Teilsequenz $\bar{s}(\mathbf{v})$ lässt sich weiter unterteilen in Teilsequenzen $\bar{s}_i(\mathbf{v})$, für $i = 1, \dots, p$. Dabei ist p minimal und die Geschwindigkeit innerhalb jeder Teilsequenz ist einheitlich.

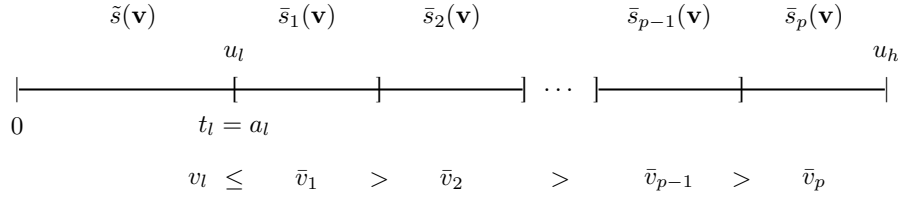
Sei $\tilde{s}(\mathbf{v}) = (u_1, \dots, u_l)$ die verbleibende Teilsequenz von s , sodass

$$s = \tilde{s}(\mathbf{v}) \cup \bar{s}(\mathbf{v}) = \tilde{s}(\mathbf{v}) \cup \bigcup_{i=1}^p \bar{s}_i(\mathbf{v})$$

gilt. Sei u_l der letzte Kunde aus $\tilde{s}(\mathbf{v})$. Es gilt $t_l = a_l$. Falls $\bar{s}(\mathbf{v})$ nicht leer ist und $a_l \neq b_l$, dann gilt $v_l \leq v_{l+1}$. Es sei angemerkt, dass sowohl $\tilde{s}(\mathbf{v})$ als auch $\bar{s}(\mathbf{v})$ leer sein können.

Beobachtung 5. *Es gilt, dass die Teilsequenz $\bar{s}_p(\mathbf{v})$ und s^* gleich sind.*

In Abbildung 5.10 ist die Unterteilung der Sequenz s in die Teilsequenzen visualisiert.


 Abbildung 5.10: Aufteilung einer Sequenz s in Teilsequenzen $\tilde{s}(\mathbf{v})$ und $\bar{s}_1(\mathbf{v}), \dots, \bar{s}_p(\mathbf{v})$

Definition 17. Sei s eine Sequenz und sei $\mathbf{v}$ ein Geschwindigkeitsvektor zu s . Sei $\bar{s}_i(\mathbf{v})$, $i = 1, \dots, p$ und $\tilde{s}(\mathbf{v})$ eine Partitionierung von s basierend auf $\mathbf{v}$, wie vorab definiert. Falls für eine Sequenz und einen Geschwindigkeitsvektor eine solche Partitionierung existiert, dann wird dies als *Match* bezeichnet.

Beispiele für ein *Match* sind in Abbildung 5.11 dargestellt. Dabei stellen die ersten drei Beispiele ein Match dar und für die letzten beiden Beispiele liegt kein Match vor. Im ersten Beispiel wird die Sequenz s in $\tilde{s}(\mathbf{v})$ und $\bar{s}_1(\mathbf{v}), \dots, \bar{s}_p(\mathbf{v})$ unterteilt, siehe Abbildung 5.11(a). Der letzte Kunde u_l aus $\tilde{s}(\mathbf{v})$ wird auf seiner unteren Zeitfenstergrenze beliefert und es gilt $v_l \leq v_{l+1}$. Für die Geschwindigkeiten innerhalb einer Teilsequenz $\bar{s}_i(\mathbf{v})$, für $i = 1, \dots, p$, gilt, dass sie gleich sind und zwischen zwei Teilsequenzen absteigend. Das folgende Beispiel zeigt eine Partitionierung, bei der $\tilde{s}(\mathbf{v})$ leer ist (siehe Abbildung 5.11(b)). Daraus ergibt sich, dass in der Sequenz s kein Kunde auf seiner unteren Zeitfenstergrenze beliefert wird. Im Beispiel 5.11(c) gilt $s = \tilde{s}(\mathbf{v})$ und der letzte Kunde in s wird auf seiner unteren Zeitfenstergrenze beliefert. Ein Beispiel ohne Match wird in Abbildung 5.11(d) dargestellt. Angenommen, u_l ist der letzte Kunde, der auf seiner unteren Zeitfenstergrenze beliefert wird, und es gilt $a_l \neq b_l$. Da jedoch $v_l > v_{l+1}$ gilt, kann kein Match vorliegen. Auch in Abbildung 5.11(e) liegt kein Match vor. Angenommen, u_l ist der letzte Kunde, der auf seiner unteren Zeitfenstergrenze beliefert wird und sei $u_{l'}$ ein Kunde, der später beliefert wird. Angenommen, es gilt $v_{l'} < v_{l'+1}$, dann kann es kein Match geben.

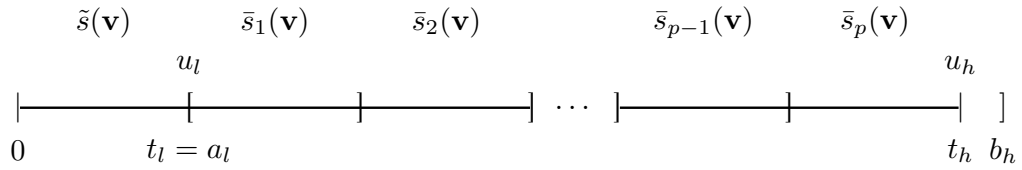
Lemma 15. Sei s eine Sequenz und $\mathbf{v}$ ein Geschwindigkeitsvektor für s . Falls $\mathbf{v}$ kostenminimal ist, dann bilden s zusammen mit $\mathbf{v}$ ein Match und es existiert eine Partitionierung.

Lemma 16. Sei $s = (u_1, u_2, \dots, u_k)$ eine Sequenz und $\bar{\mathcal{V}}$ die Menge der von Algorithmus 6 berechneten Geschwindigkeitsvektoren. Es gilt, dass alle Vektoren aus $\bar{\mathcal{V}}$ ein Match mit s bilden.

Die folgende Proposition zeigt, dass $\bar{s}(\mathbf{v})$ keine Wartezeiten enthält.

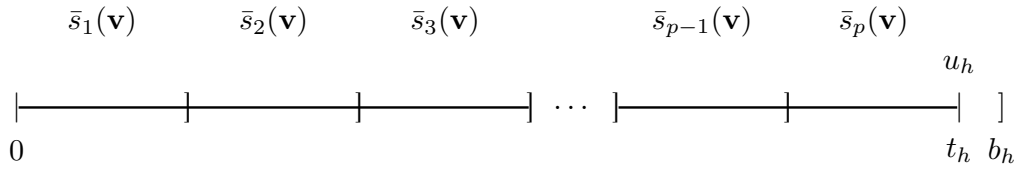
Proposition 2. Sei $s = (u_1, u_2, \dots, u_k)$ eine Sequenz und $\mathbf{v}$ ein Geschwindigkeitsvektor, der zusammen mit s ein Match bildet. Es gilt, dass die Teilsequenzen $\bar{s}_i(\mathbf{v})$, für $i = 1, \dots, p$, keine Wartezeiten enthalten und es gilt

$$t_i = t_{i-1} + \tau_i + d_i/v_i, \quad u_i \in \bar{s}(\mathbf{v}).$$



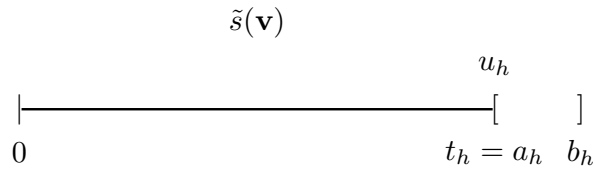
$$v_l \leq \bar{v}_1 > \bar{v}_2 > \bar{v}_{p-1} > \bar{v}_p$$

(a) Die Sequenz wird aufgeteilt in die Teilsequenzen $\tilde{s}(\mathbf{v}), \bar{s}_1(\mathbf{v}), \dots, \bar{s}_p(\mathbf{v})$

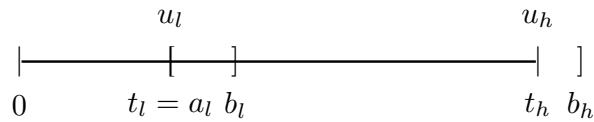


$$\bar{v}_1 > \bar{v}_2 > \bar{v}_3 > \bar{v}_{p-1} > \bar{v}_p$$

(b) Die Sequenz wird aufgeteilt in die Teilsequenzen $\bar{s}_1(\mathbf{v}), \dots, \bar{s}_p(\mathbf{v})$

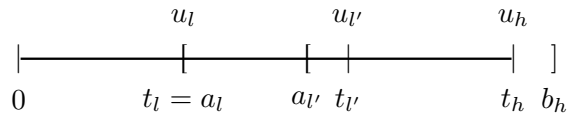


(c) Die Sequenz wird aufgeteilt in Teilsequenz $\tilde{s}(\mathbf{v})$



$$v_l > v_{l+1}$$

(d) Die Sequenz bildet zusammen mit dem Geschwindigkeitsvektor kein Match



$$v_{l'} < v_{l'+1}$$

(e) Die Sequenz bildet zusammen mit dem Geschwindigkeitsvektor kein Match

Abbildung 5.11: Beispiel für drei Sequenzen, die zusammen mit ihrem Geschwindigkeitsvektor ein Match bilden und zwei Sequenzen mit Geschwindigkeitsvektoren ohne Match

Mit dem nächsten Lemma wird gezeigt, dass Algorithmus 6 ausschließlich zulässige Geschwindigkeitsvektoren berechnet.

Lemma 17. *Alle Geschwindigkeitsvektoren aus $\bar{\mathcal{V}}$, berechnet von Algorithmus 6 für die Sequenz $s = (u_1, u_2, \dots, u_h)$, sind zulässig.*

Das nächste Lemma beschreibt, wie sich die Geschwindigkeiten der Kunden in $\tilde{s}(\mathbf{v})$ ändern, wenn der Belieferungszeitpunkt des letzten Kunden reduziert wird.

Lemma 18. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $\mathbf{v}^* = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt t_h^* beliefert und bezüglich f kostenminimal ist. Sei $\tilde{s}(\mathbf{v}^*)$ und $\tilde{s}_i(\mathbf{v}^*)$, $i = 1, \dots, p$, die Partitionierung von s basierend auf $\mathbf{v}^*$. Sei t'_h ein zulässiger Belieferungszeitpunkt von u_h , wobei $t'_h < t_h^*$ gilt. Dann existiert ein kostenminimaler Geschwindigkeitsvektor $\mathbf{v}$, für den $t_h = t'_h$ und $v_i = v_i^*$, für $u_i \in \tilde{s}(\mathbf{v}^*)$ gilt.*

Als Nächstes wird gezeigt, dass alle von Algorithmus 6 berechneten Geschwindigkeitsvektoren bezüglich der Funktion f kostenminimal sind.

Lemma 19. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $\bar{\mathcal{V}}$ die Menge der von Algorithmus 6 berechneten Geschwindigkeitsvektoren. Alle Vektoren in $\bar{\mathcal{V}}$ sind bezüglich f und dem jeweiligen Belieferungszeitpunkt von u_h kostenminimal.*

Das nächste Lemma sagt aus, dass Algorithmus 6 alle Belieferungszeitpunkte am letzten Kunden berechnet.

Lemma 20. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $\mathbf{v}^*$ der optimale Geschwindigkeitsvektor der Eingabe von Algorithmus 6. Sei arr die Funktion, die die Belieferungszeitpunkte von u_h beschreibt. Die Funktion arr ist korrekt und beschreibt alle zulässigen Belieferungszeitpunkte von u_h , die kleiner oder gleich t_h^* sind.*

Lemma 21. *Die Laufzeit von Algorithmus 6 ist polynomiell beschränkt durch $O(|s|^2)$.*

Damit kann der Korrektheitsbeweis für Algorithmus 6 geführt und gezeigt werden, dass die Laufzeit polynomiell in der Eingabe beschränkt ist.

Theorem 3. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz. Algorithmus 6 berechnet in polynomieller Zeit alle relevanten Belieferungszeitpunkte für u_h und die dazugehörigen kostenminimalen Geschwindigkeitsvektoren.*

Damit ist der Korrektheitsbeweis abgeschlossen. Im Anschluss folgt die Vorstellung eines Branch-and-Price-Algorithmus zur Lösung der Problemstellung aus Unterkapitel 5.4.

5.6 Branch-and-Price-Algorithmus

Dieses Unterkapitel stellt einen Branch-and-Price-Algorithmus zur Lösung der Problemstellung aus Unterkapitel 5.4 vor und formuliert das Problem als IP. Das duale Programm des relaxierten IPs wird in Unterkapitel 5.6.1 beschrieben. Anschließend wird das Pricing-Problem definiert. Zur Lösung wird in Unterkapitel 5.6.2 ein Labeling-Algorithmus vorgestellt. Abgeschlossen wird das Unterkapitel mit der Beschreibung der Branching-Regeln und Beschleunigungsverfahren für den Labeling-Algorithmus (siehe Unterkapitel 5.6.3 und 5.6.4).

5.6.1 Duales Programm und Pricing-Problem

Der Branch-and-Price-Algorithmus soll das IP (5.8)-(5.11) lösen. Im Folgenden wird das DP der relaxierten Formulierung des IPs (5.8)-(5.11) beschrieben. Für die Relaxation wird die Nebenbedingung (5.11) durch $x_s \geq 0$ für alle $s \in \Omega$ ersetzt. Die obere Beschränkung der Variablen ist nicht erforderlich, da diese bereits durch die Nebenbedingungen (5.9) gewährleistet wird. Das DP umfasst für jeden Kunden $u_i \in C$ eine duale Variable y_i sowie für Nebenbedingung (5.10) die duale Variable $\tilde{y}$. Das duale Programm lässt sich somit wie folgt formulieren:

$$\begin{array}{ll} \text{maximiere} & \sum_{u_i \in C} y_i - m\tilde{y} \end{array} \quad (5.17)$$

$$\begin{array}{ll} \text{u.d.N.} & \sum_{u_i \in C} e_{is}y_i - \tilde{y} \leq c_f(s) \quad \forall s \in \Omega, \end{array} \quad (5.18)$$

$$\tilde{y} \geq 0. \quad (5.19)$$

Die Zielfunktion (5.17) maximiert die Summe der dualen Variablen, abzüglich der mit der Anzahl der Drohnen multiplizierten dualen Variable $\tilde{y}$. In den Nebenbedingungen (5.18) werden für jede Sequenz $s \in \Omega$ die dualen Variablen der Kunden aufsummiert und die Variable $\tilde{y}$ subtrahiert. Dieser Wert muss kleiner oder gleich den Kosten $c_f(s)$ der Sequenzen sein. Die Nichtnegativität von $\tilde{y}$ wird durch die Nebenbedingung (5.19) sichergestellt.

Die Anzahl möglicher Sequenzen wächst exponentiell mit der Kundenzahl. Daher wird nur eine Teilmenge der Variablen berücksichtigt. Sei Ω' eine Teilmenge der Variablen, wobei $\Omega' \subseteq \Omega$ gilt. Es ist zu beachten, dass Ω' auch die leere Menge sein kann. Die Relaxation des IPs (5.8)-(5.11) auf Basis der Menge Ω' wird als EMP bezeichnet. Die Beschränkung der Menge Ω erfordert jedoch die Lösung des Pricing-Problems. Dafür wird eine Lösung des DP für das EMP berechnet. Anschließend wird das Pricing-Problem anhand der Werte der dualen Variablen initialisiert. Daraufhin kann entschieden werden, welche Entscheidungsvariablen zu Ω' hinzugefügt werden. Eine Einführung in das allgemeine Pricing-Problem findet sich in Unterkapitel 3.2.

Sei $\mathbf{y}^*$ eine optimale Lösung des LPs (5.17)-(5.19), eingeschränkt auf Ω' . Die reduzierten Kosten $\bar{c}_f(s, \mathbf{y}^*)$ für eine Sequenz $s = (u_1, u_2, \dots, u_h)$, für $s \in \Omega$, mit einem optimalen Geschwindigkeits-

vektor $\mathbf{v} = (v_1, v_2, \dots, v_h, v_{opt})$ für die Funktion f berechnen sich wie folgt:

$$\bar{c}_f(s, \mathbf{y}^*) = c_f(s) - \sum_{u_i \in C} e_{is} y_i^* + \tilde{y}^* = \sum_{i=1}^h (d_{i-1,i} f(v_i) - y_i^*) + d_{h0} f(v_{opt}) + \tilde{y}^*. \quad (5.20)$$

Es ist zu beachten, dass bei der Berechnung der reduzierten Kosten der Rückflug zum Depot nach dem letzten Kunden mit berücksichtigt wird. Die optimale Lösung des Pricing-Problems wird definiert als $\bar{c}_f^*(\mathbf{y}^*) = \min_{s \in \Omega} \bar{c}_f(s, \mathbf{y}^*)$. Es ist möglich, dass das EMP anfangs keine zulässige Lösung besitzt. In diesem Fall wird das sogenannte *Farkas-Pricing* angewendet (Lübbecke, 2011). Bei dem Verfahren wird eine zulässige Lösung durch die Aufnahme zusätzlicher Variablen angestrebt. Dazu wird das Pricing-Problem ohne Berücksichtigung der Sequenzkosten $c_f(s)$ und der Rückflugkosten zum Depot initialisiert und gelöst.

Zur Lösung des Pricing-Problems wird dieses zunächst auf ein Kürzeste-Wege-Problem mit Beschränkung der Ressourcen reduziert (siehe Unterkapitel 3.3). Dieses Problem wird mit einem Labeling-Algorithmus gelöst, der auch im Folgenden vorgestellt wird.

5.6.2 Labeling-Algorithmus

Im Folgenden wird der Hilfsgraph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ zur Aufstellung des Kürzeste-Wege-Problems mit Beschränkung der Ressourcen beschrieben. Dabei bezeichnet $\mathcal{V} = C \cup \{w_0, w_{n+1}\}$ die Knotenmenge und $\mathcal{A} = \{(i, j) : u_i, u_j \in C\} \cup \{(0, i) : u_i \in C\} \cup \{(i, n+1) : u_i \in C\}$ die Kantenmenge. Dabei stellen w_0 und w_{n+1} die Quelle bzw. Senke eines jeden Weges im Graphen dar und können mit dem Depot assoziiert werden. Eine Kante $(i, j) \in \mathcal{A}$, $u_i, u_j \in C$, hat eine Länge von d_{ij} . Für die Kanten $(0, i)$ und $(i, n+1)$, wobei $u_i \in C$, beschreibt d_{0i} bzw. d_{i0} die Länge der jeweiligen Kante. Jeder Kunde $u_i \in C$ hat eine Servicezeit von τ_i und ein Zeitfenster $[a_i, b_i]$. Die Knoten w_0, w_{n+1} haben keine Servicezeit. Für beide Knoten wird ein Zeitfenster $[0, T]$ angenommen, wobei T nicht restriktiv ist. Bei der Belieferung eines Kunden $u_i \in C$ fallen Kosten in Höhe von $-y_i^*$ an. Dabei beschreibt $\mathbf{y}^* = (y_1^*, \dots, y_h^*, \tilde{y}^*)$ die optimale Lösung des DP (5.17)-(5.19) auf Ω' . Beim Start von Knoten w_0 fallen Kosten in Höhe von $\tilde{y}^*$ an. Sei v_{ij} die Geschwindigkeit auf Kante $(i, j) \in \mathcal{A}$, mit $v_{ij} \in [v_{min}, v_{max}]$. Die Flugzeit auf einer Kante $(i, j) \in \mathcal{A}$ mit Geschwindigkeit v_{ij} beträgt d_{ij}/v_{ij} . Sei f eine strikt konvexe Kostenfunktion. Jede Kante $(i, j) \in \mathcal{A}$ verursacht Kosten in Höhe von $d_{ij} f(v_{ij})$, die von der Geschwindigkeit v_{ij} abhängen. Die Geschwindigkeit, die die Funktion f minimiert, wird mit v_{opt} bezeichnet. Zur Berechnung eines kürzesten Weges in $\mathcal{G}$ mit beschränkten Ressourcen wird die Definition eines Weges eingeführt. Diese ist gleich zu Kapitel 4.

Definition 18. *Ein Weg $W = (w_0, u_1, u_2, \dots, u_h, w_{n+1})$ beschreibt eine Reihenfolge von Kunden, wobei $u_1, u_2, \dots, u_h \in C$ gilt. Ein Weg startet an der Quelle w_0 und endet an der Senke w_{n+1} .*

Sei $\mathcal{W}$ die Menge aller zulässigen Wege in $\mathcal{G}$. Die Definition 14 der Zulässigkeit eines Geschwindigkeitsvektors für eine Sequenz lässt sich auf Wege übertragen. Damit gilt, dass ein Weg $W = (w_0, u_1, u_2, \dots, u_h, w_{n+1})$, $W \in \mathcal{W}$, zusammen mit einem Geschwindigkeitsvektor

$\mathbf{v} = (v_1, v_2, \dots, v_h, v_{opt})$ zulässig ist, falls

- jeder Kunde nur maximal einmal in W enthalten ist,
- jeder Kunde auf W mit $\mathbf{v}$ in seinem Zeitfenster beliefert wird,
- alle Einträge aus $\mathbf{v}$ sind im Intervall $[v_{min}, v_{max}]$.

Ein Vektor $\mathbf{v} = (v_1, v_2, \dots, v_h, v_{opt})$ zu einem Weg $W = (w_0, u_1, u_2, \dots, u_h, w_{n+1})$ wird bezüglich Funktion f als optimal bezeichnet, falls $\mathbf{v}$ ein optimaler Geschwindigkeitsvektor für die Sequenz $s = (u_1, u_2, \dots, u_h)$ ist (siehe Definition 16). Die Kosten eines Weges $W = (w_0, u_1, \dots, u_h, w_{n+1})$ mit einem optimalen Geschwindigkeitsvektor $\mathbf{v} = (v_1, v_2, \dots, v_h, v_{opt})$ und der optimalen Lösung des DPs auf Ω' sind definiert als

$$\bar{c}_f(W, \mathbf{y}^*) = \tilde{y}^* + \sum_{i=1}^h (d_{i-1,i} f(v_i) - y_i^*) + d_{h,n+1} f(v_{opt}). \quad (5.21)$$

Die Kosten $\bar{c}_f(W, \mathbf{y}^*)$ zu einem Weg $W \in \mathcal{W}$ sind gleich den reduzierten Kosten $\bar{c}_f(s, \mathbf{y}^*)$ der Sequenz $s \in \Omega$, die in W enthalten ist. Ein Weg mit minimalen Kosten stellt außerdem eine Lösung für das Pricing-Problem dar. Zur Lösung des Kürzeste-Wege-Problems mit Beschränkung der Ressourcen wird im folgenden Abschnitt ein Labeling-Algorithmus vorgestellt.

Der Algorithmus zielt darauf ab, den kürzesten Weg mit Beschränkung der Ressourcen in $\mathcal{G}$ von w_0 nach w_{n+1} zu berechnen. Dazu wird die Definition eines Teilwegs eingeführt.

Definition 19. Ein Teilweg $W^{tw} = (w_0, u_1, u_2, \dots, u_h)$ beschreibt eine Reihenfolge von Kunden, wobei $u_1, u_2, \dots, u_{h-1} \in C$ und $u_h \in C \cup \{w_{n+1}\}$ gilt. Ein Weg startet an der Quelle w_0 und endet entweder an einem Kunden oder an der Senke w_{n+1} .

Die Definition der Zulässigkeit von Wegen kann auf Teilwege übertragen werden. Ein Geschwindigkeitsvektor $\mathbf{v} = (v_1, v_2, \dots, v_h)$ zu einem Teilweg $W^{tw} = (w_0, u_1, u_2, \dots, u_h)$ wird in Bezug auf Funktion f als optimal bezeichnet, falls $\mathbf{v}$ ein optimaler Geschwindigkeitsvektor zur Sequenz $s = (u_1, u_2, \dots, u_h)$ ist (siehe Definition 16). Die Kosten eines Teilwegs $W^{tw} = (w_0, u_1, u_2, \dots, u_h)$ mit einem optimalen Geschwindigkeitsvektor $\mathbf{v} = (v_1, v_2, \dots, v_h)$ und der optimalen Lösung des dualen Problems sind definiert als

$$\bar{c}_f(W^{tw}, \mathbf{y}^*) = \tilde{y}^* + \sum_{i=1}^h (d_{i-1,i} f(v_i) - y_i^*). \quad (5.22)$$

Jeder Teilweg im Graphen wird durch ein Label repräsentiert. Die Bestandteile eines Labels $L = (u_i, W^{tw}, \mathbf{v}, \bar{c}_f(W^{tw}, \mathbf{y}^*))$ sind der aktuelle Knoten $u_i \in \mathcal{V}$, der Teilweg $W^{tw} = (w_0, u_1, u_2, \dots, u_h)$, ein optimaler Geschwindigkeitsvektor $\mathbf{v}$ zu W^{tw} und die reduzierten Kosten $\bar{c}_f(W^{tw}, \mathbf{y}^*)$ des Teilwegs mit $\mathbf{v}$. Der Algorithmus wird mit dem Label $L_0 = (w_0, (), \emptyset, \tilde{y}^*)$ initialisiert. Da das Vorgehen eines Labeling-Algorithmus bereits in Kapitel 4 vorgestellt wurde, erfolgt hier lediglich eine kurze Zusammenfassung. Der Algorithmus führt eine Liste der Labels, die noch nicht

	Label $L = (u_h, W^{tw} = (w_0, u_1, u_2, \dots, u_h), \mathbf{v}, \bar{c}_f(W^{tw}, \mathbf{y}^*))$
Zulässigkeit Erweiterung zu $u_j \in \mathcal{V}$ zulässig, falls	$u_h \neq w_{n+1}, u_j \notin W^{tw},$ $(h, i) \in \mathcal{A}$ Algorithmus 5 berechnet einen Geschwindigkeitsvektor für $W \cup \{u_j\}$
Erweiterung Neues Label $L' = (u_j, \tilde{W}^{tw}, \tilde{\mathbf{v}}, \bar{c}_f(\tilde{W}^{tw}, \mathbf{y}^*))$ ist definiert als	aktueller Knoten = u_j , $\tilde{W}^{tw} = (w_0, u_1, u_2, \dots, u_h, u_j),$ $\tilde{\mathbf{v}}$ = optimaler Geschwindigkeitsvektor für $\tilde{W}$ mit Algorithmus 5, $\bar{c}_f(\tilde{W}^{tw}, \mathbf{y}^*) = \tilde{y}^* + \sum_{i=1}^h (d_{i-1,i} f(\tilde{v}_i) - y_i^*) + d_{hj} f(\tilde{v}_j) - y_j^*$

Tabelle 5.5: Zulässigkeits- und Erweiterungsregel für den Labeling-Algorithmus

erweitert wurden. Zu Beginn enthält die Liste nur L_0 . Der Algorithmus wiederholt die folgenden Schritte, bis die Liste leer ist:

- (1) Entferne das erste Label aus der Liste.
- (2) Prüfe, ob Erweiterung zu Knoten aus $\mathcal{V}$ zulässig ist.
- (3) Für jede mögliche Erweiterung erstelle ein neues, erweitertes Label.
- (4) Füge das neue Label zur Liste hinzu.

Für Schritt (2) sind die folgenden Bedingungen zu prüfen:

- (2.1) Der aktuelle Knoten darf nicht der Senke entsprechen.
- (2.2) Der nächste Knoten ist nicht im Teilweg enthalten.
- (2.3) Es existiert eine Kante vom aktuellen Knoten zum nächsten Knoten in $\mathcal{A}$.
- (2.4) Es existiert ein optimaler Geschwindigkeitsvektor für den erweiterten Teilweg.

Für die Erstellung eines neuen Labels in Schritt (3) sind folgende Schritte durchzuführen:

- (3.1) Der aktuelle Knoten wird zum nächsten Knoten.
- (3.2) Der Teilweg wird um diesen Knoten erweitert.
- (3.3) Ein optimaler Geschwindigkeitsvektor wird für den erweiterten Teilweg berechnet.
- (3.4) Die Kosten für den erweiterten Teilweg werden berechnet.

Die Schritte (2) und (3) sind in Tabelle 5.5 zusammengefasst. Damit ist die Vorstellung des Labeling-Algorithmus abgeschlossen.

5.6.3 Branching-Regel

Eine Möglichkeit, die Laufzeit des Branch-and-Price-Algorithmus zu reduzieren, ist die Wahl einer geeigneten Branching-Strategie. Ein naiver Ansatz, das Branching auf den Binärvariablen x_s durchzuführen, würde jedoch zu einem Ungleichgewicht im Branch-and-Bound-Baum führen. Daher wird das *Kanten-Branching* eingeführt, das von Desrosiers et al. (1984) für das VRP mit Zeitfenstern entwickelt wurde. Bei diesem Verfahren wird die Nutzung einer Kante $(i, j) \in \mathcal{A}, u_i, u_j \in C$ entweder erzwungen oder ausgeschlossen. Um die Kante zu erzwingen, werden alle Variablen, die u_i oder u_j enthalten, aber nicht (i, j) nutzen, aus Ω' entfernt, und alle ausgehenden bzw. eingehenden Kanten von u_i und u_j bis auf (i, j) aus $\mathcal{A}$ gelöscht. Um eine Kante auszuschließen, werden alle Variablen, die sie nutzen, sowie die Kante selbst aus Ω' und $\mathcal{A}$ entfernt. Die Anpassungen des Graphen und der Menge Ω' dürfen nur lokal im Knoten des Branch-and-Bound-Baums und in seinen Kindern durchgeführt werden. Der Algorithmus berechnet für jede Kante die Nutzungshäufigkeit und wählt die Kante mit einem Wert nahe 0,5. Nach Auswahl der Kante werden zwei Teilprobleme erstellt.

5.6.4 Beschleunigungsverfahren

Im Folgenden werden vier Verfahren zur Beschleunigung des Labeling-Algorithmus vorgestellt. Mit diesen Verfahren wird grundsätzlich versucht, die Menge der zu betrachtenden Labels zu reduzieren. Zunächst wird der Hilfsgraph $\mathcal{G}$ reduziert, und alle Kanten, die zu unzulässigen Lösungen führen, entfernt. Eine Kante $(i, j) \in \mathcal{A}, u_i, u_j \in C$, wird aus $\mathcal{G}$ entfernt, falls $a_i + \tau_i + d_{ij}/v_{\max} > b_j$ gilt. Diese Reduzierung kann vor der Ausführung des Branch-and-Price-Algorithmus angewendet werden und muss daher nur einmal durchgeführt werden.

Dominanz

Eine Möglichkeit, die Anzahl der Labels zu reduzieren, ist die Einführung einer Dominanzregel. Diese vergleicht zwei Labels mit demselben aktuellen Knoten und prüft, ob eines als besser (d.h. dominierend) identifiziert werden kann. Ein Label gilt als dominierend, wenn es bei allen Fortsetzungen des Teilwegs keine höheren reduzierten Kosten als das dominierte Label erzielt. Die Dominanz wird anhand der reduzierten Kosten, der besuchten Kunden und des Belieferungszeitpunkts des aktuellen Kunden beurteilt. Dabei hängt der Belieferungszeitpunkt des letzten Kunden von der Fortsetzung des Teilwegs ab, was die reduzierten Kosten verändert. Zur Definition der Dominanzregel wird jedes Label um die Menge $\bar{\mathcal{V}}$ sowie die Funktionen arr und $cost$ erweitert. $\bar{\mathcal{V}}$ enthält alle kostenminimalen Geschwindigkeitsvektoren, die die Belieferungszeitpunkte festlegen. Die Funktion arr beschreibt die Belieferungszeitpunkte in Abhängigkeit von der Geschwindigkeit des aktuellen Kunden, und $cost$ gibt die Kosten des Teilwegs für verschiedene Belieferungszeitpunkte des letzten Kunden an. Zur Berechnung der Einträge kann Algorithmus 6 verwendet werden.

Nach der Erweiterung der Labels lässt sich eine Dominanzregel definieren. Seien L^A und L^B zwei Labels, deren aktueller Knoten der Kunde u_h ist. Seien W^A bzw. W^B die jeweiligen Teilwege

der beiden Labels. Seien X^A und X^B die Mengen der Kunden, die auf den Teilwegen W^A und W^B besucht werden. Sei

$$Y := \left\{ u_j \in X^A \setminus X^B \mid a_h + \tau_h + \frac{d_{hj}}{v_{\max}} < b_j \right\}.$$

Diese Menge beschreibt die Kunden, die ausschließlich von W^A besucht werden und die vom aktuellen Knoten unter Berücksichtigung der Zeitfenster, Entfernung und maximalen Geschwindigkeit noch erreichbar sind. Damit lässt sich eine Dominanzregel wie folgt formulieren:

Definition 20. Seien $L^A = (u_h, W^A, v^A, \bar{c}_f(W^A, y^*), \bar{v}^A, arr^A, cost^A)$ und $L^B = (u_h, W^B, v^B, \bar{c}_f(W^B, y^*), \bar{v}^B, arr^B, cost^B)$ zwei Labels, die denselben aktuellen Kunden $u_h \in C$ besitzen. Label L^A dominiert Label L^B , falls die folgenden Bedingungen erfüllt sind:

$$\begin{aligned} Y &= \emptyset \wedge \\ \forall v^B \in \bar{v}^B \exists v^A \in \bar{v}^A : arr^A(v_h^A) &\leq arr^B(v_h^B) \wedge \\ cost^A(v_h^A) - \sum_{u_j \in W^A \setminus w_0} y_j^* &\leq cost^B(v_h^B) - \sum_{u_j \in W^B \setminus w_0} y_j^*. \end{aligned}$$

Dies bedeutet, dass keine Kunden, die noch erreichbar sind, ausschließlich von L^A besucht werden. Die zweite Bedingung fordert, dass zu jedem Belieferungszeitpunkt von u_h in L^B ein gleicher oder früherer Zeitpunkt in L^A existiert, bei dem die reduzierten Kosten in L^A kleiner oder gleich denen von L^B sind.

Das folgende Beispiel veranschaulicht den Vergleich von Belieferungszeitpunkten und reduzierten Kosten: In Abbildung 5.12 liegt keine Dominanz vor, da sich die Kurven schneiden und die gestrichelte Kurve bei einem Belieferungszeitpunkt von 400 niedrigere Kosten aufweist. Damit ist die Dominanzregel eingeführt.

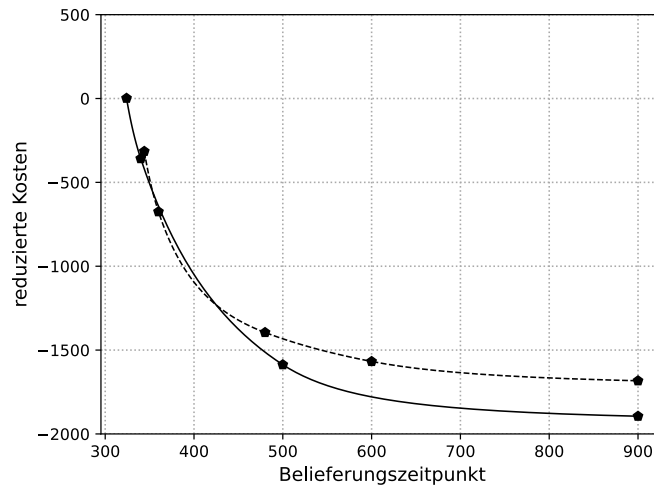


Abbildung 5.12: Zwischen den Kurven besteht keine Dominanz

Variation der Anzahl der aufgenommenen Variablen

Eine weitere Einflussmöglichkeit betrifft die Anzahl der Variablen, die pro Pricing-Lösung in das EMP aufgenommen werden. Zwar liefert der Labeling-Algorithmus alle nicht-dominierten Wege (und damit alle Variablen mit negativen reduzierten Kosten), doch ist es möglich, mehr als nur eine Variable zu übernehmen. Dazu werden die gefundenen Variablen nach ihren reduzierten Kosten sortiert, und eine vordefinierte Anzahl wird in das EMP integriert. Hierbei gilt es, ein Gleichgewicht zwischen der Anzahl der Pricing-Aufrufe und der Größe des EMPs zu finden.

Heuristik

Eine weitere Möglichkeit besteht darin, das Pricing-Problem heuristisch zu lösen, wie von Desaulniers et al. (2008) beschrieben. Dazu kann etwa der Ausgangsgrad jedes Kunden in $\mathcal{G}$ reduziert werden, wodurch sich die Anzahl der zu betrachtender Labels im Labeling-Algorithmus verringert. Allerdings besteht die Gefahr, dass dabei keine Variable mit negativen reduzierten Kosten gefunden wird. In diesem Fall muss das Pricing-Problem erneut auf dem vollständigen Graphen gelöst werden, um die Optimalität sicherzustellen. Daraus ergeben sich zwei gegenläufige Ziele: Der Ausgangsgrad sollte einerseits möglichst klein sein, um die Rechenzeit zu reduzieren, andererseits groß genug, um Variablen mit negativen reduzierten Kosten nicht auszuschließen.

Zur Berechnung der Kanten, die temporär aus $\mathcal{G}$ entfernt werden, wird ein Ranking erstellt. Der Wert der Kante $(i, j) \in \mathcal{A}$ wird nach folgender Formel berechnet:

$$\tilde{c}_{ij} = \frac{\min_{(i,k) \in \mathcal{A}} d_{ik}/(b_k - a_i)}{d_{ij}/(b_j - a_i)} y_j^*.$$

Dabei wird das Minimum aller kleinsten Geschwindigkeiten von Kunde $u_i \in C$ zu einem Kunden $u_k \in C$ durch die kleinste mögliche Geschwindigkeit der Kante (i, j) geteilt und mit dem Wert der dualen Variable y_j^* multipliziert.

Zudem wird die Dominanzregel gelockert, sodass die Menge Y nicht zwingend leer sein muss; in diesem Fall wird gefordert, dass die reduzierten Kosten von L^A größer oder gleich den reduzierten Kosten von L^B zuzüglich der maximal möglichen reduzierten Kosten der Menge Y sind. Diese Lockerung wird auch im Wurzelknoten des Branch-and-Price-Verfahrens verwendet.

Damit ist die Vorstellung des Branch-and-Price-Algorithmus und der dazugehörigen Beschleunigungsverfahren abgeschlossen. Im nächsten Kapitel werden die Ergebnisse der Optimierung vorgestellt, die mit diesem Algorithmus und dem Algorithmus aus Kapitel 4 erzielt wurden.

Kapitel 6

Analyse und Diskussion der Optimierungsergebnisse

Dieses Kapitel präsentiert die Rechenergebnisse der vorgestellten Branch-and-Price-Algorithmen aus den Kapiteln 4 und 5. Die Algorithmen wurden in der Programmiersprache C++ entwickelt. Für die Berechnung der Lösung kommt die *SCIP Optimization Suite 6.0.1* zum Einsatz (Gleixner et al., 2018). Für die Berechnungen wurde ein Zeitlimit von einer Stunde festgelegt.

Dieses Kapitel gliedert sich wie folgt: Zunächst werden die Ergebnisse für die Zustellroboter aus Kapitel 4 vorgestellt. Die Berechnungen wurden auf einem einzelnen Kern einer 3,3 GHz CPU unter Windows 10 durchgeführt. Anschließend werden die Ergebnisse für die Paketdrohnen aus Kapitel 5 dargestellt. Die Berechnungen wurden auf einem einzelnen Kern einer 2,9 GHz CPU unter Windows 10 durchgeführt. Die Darstellung der Ergebnisse unterscheidet sich, da für den Algorithmus aus Kapitel 5 Vergleichswerte aus der Literatur vorliegen. Für den Algorithmus aus Kapitel 4 existieren keine Vergleichswerte. Aus diesem Grund erfolgt der Vergleich mit einer Implementierung des MIPs (4.1)-(4.11). Darüber hinaus wird der Einfluss von Parameterveränderungen analysiert. Simulationsbasierte Ergebnisse für die Zustellung mit den innovativen Zustellkonzepten werden in Kapitel 7 vorgestellt.

6.1 Ergebnisse für Zustellroboter

Die Leistungsfähigkeit des Branch-and-Price-Algorithmus aus Kapitel 4 wird auf Benchmark-Instanzen von Cordeau et al. (2001) untersucht. Diese Instanzen wurden ursprünglich für das *Multiple-Depot Vehicle Routing Problem with Time Windows* konzipiert. Die Verwendung dieser Instanzen bietet sich an, da für die Optimierung von Zustellrobotern bislang keine spezifischen Benchmark-Instanzen verfügbar sind. Die originalen Benchmark-Instanzen von Cordeau et al. umfassen 20 Instanzen, die unter <https://bernabe.dorrnsoro.es/vrp/> verfügbar sind. Die 20 Instanzen sind aufgeteilt in zehn Instanzen mit kurzen Zeitfenstern und zehn Instanzen mit längeren Zeitfenstern. In der ersten Hälfte der Instanzen wird die Zeitfensterlänge zwischen 90

und 180 Minuten variiert, wobei die Startzeitpunkte gleichmäßig zwischen 60 und 480 Minuten verteilt sind. Für die zweite Hälfte liegt die Zeitfensterlänge zwischen 180 und 360 Minuten, während der Startzeitpunkt zwischen 60 und 300 Minuten liegt. Die Anzahl der Kunden variiert für beide Hälften zwischen 48 und 288. Die Zahl der Depots liegt zwischen vier und sechs. Es wird angenommen, dass alle Depots sowohl als Zwischenlager für Sendungen dienen als auch mit Lademöglichkeiten für die Roboter ausgestattet sind. Da die Instanzen ursprünglich für den Straßengüterverkehr entworfen worden sind, sind Anpassungen für die Anwendung von Zustellrobotern erforderlich. Im Folgenden werden diese Anpassungen im Detail beschrieben. Die Koordinaten der ursprünglichen Instanzen waren zufällig innerhalb des Rechtecks $[-100; 100]^2$ verteilt. Für den Einsatz von Robotern muss dieses Rechteck skaliert werden. In früheren Veröffentlichungen (Boysen et al., 2018; Sonneberg et al., 2019; Ostermeier et al., 2022; Heimfarth et al., 2022) wurden die Seitenlängen des Rechtecks als Meter interpretiert, wobei Werte zwischen 2 und 5 km angenommen wurden. Aus diesem Grund erfolgt eine Skalierung des eben genannten Rechtecks um den Faktor 20. Damit ergibt sich eine Seitenlänge von 4 km. Die Berechnung der Distanzen zwischen zwei Knoten erfolgt anhand der Manhattan-Metrik, da diese ein Straßennetz im urbanen Raum besser abbildet. Damit ergibt sich eine durchschnittliche Distanz zwischen einem Kunden und seinem nächstgelegenen Depot von 612 Metern. Die durchschnittliche Distanz zwischen zwei Kunden, gemessen über das Depot mit minimalem Fahrtweg, beträgt 1.846 Meter. Die maximale Entfernung zwischen zwei Depots beträgt 2.191 Meter. In allen Berechnungen wurde sichergestellt, dass jeder Kunde vom jeweils nächstgelegenen Depot aus erreichbar ist. Ob ein Kunde erreichbar ist, hängt von der maximalen Reichweite der Roboter ab. Diese wird im nächsten Abschnitt vorgestellt. Ist ein Kunde nicht erreichbar, wird er entweder durch einen anderen Kunden ersetzt oder die Instanz wird ausgeschlossen. Hiervon sind lediglich 3,76 % der Kunden betroffen.

Die Analyse basiert auf den Daten des Roboters von Starship Technologies. Dieser Roboter wurde gewählt, da der Hersteller Marktführer in diesem Bereich ist und Daten zum Roboter veröffentlicht wurden. Den Daten zufolge erreicht der Roboter eine maximale Geschwindigkeit von 6 km/h und eine durchschnittliche Geschwindigkeit von 3 km/h. Zur Vereinfachung der Berechnungen wird eine Geschwindigkeit von 3,6 km/h (entspricht 1 m/s) angenommen. Die maximale Reichweite des Roboters liegt bei 6 km. In der Problembeschreibung 4.3 wurde angenommen, dass die Akkukapazität nicht vollständig ausgeschöpft wird. Aus diesem Grund wird die Kapazität des Akkus auf 60% begrenzt. Damit ergibt sich eine maximale Reichweite von 3,6 km, also eine Stunde Fahrzeit. Die daraus resultierende Ladezeit beläuft sich auf 27 Minuten. Nach Angaben von Starship Technologies beträgt die Laderate 2,22. Das bedeutet, dass die Reichweite des Roboters pro Ladeinheit um 2,22 Einheiten an Fahrzeit zunimmt. Zur Vereinfachung der Berechnungen und für eine konservative Schätzung wird $\alpha = 2$ angenommen. Es wird angenommen, dass alle Kunden die gleiche Zahlungsbereitschaft (Profite) aufweisen. Die Belieferungszeit am Kunden (Servicezeit) beträgt für alle Kunden 5 Minuten. Obwohl die Servicezeit im Algorithmus erst nach dem Eintreffen des Roboters am Kunden beginnt, kann diese auf die Beladung am Depot und die Entladung beim Kunden aufgeteilt werden, da das Zeitfenster des Depots als nicht restriktiv gilt. Wie bereits in der Problembeschreibung 4.3 beschrieben, besteht das Ziel darin, die Anzahl der belieferten Kunden zu maximieren.

Abbildung 6.1 zeigt eine optimale Lösung für eine Instanz mit 30 Kunden (Kreise), sechs Depots (Quadrate) und zwei Zustellrobotern. Die Touren der Fahrzeuge werden mit durchgezogenen Linien visualisiert. Ist ein Kunde nur über eine einzige Kante mit einem Depot verbunden, kehrt der Zustellroboter nach der Belieferung zu diesem Depot zurück. Isoliert dargestellte Kunden werden in der optimalen Lösung nicht eingeplant und folglich nicht beliefert. In der dargestellten Lösung werden 27 Kunden von einem Roboter beliefert. Die Visualisierung verdeutlicht außerdem, dass die Roboter nicht nur von einem Depot aus agieren, sondern dieses nach der Zustellung in einigen Fällen wechseln.

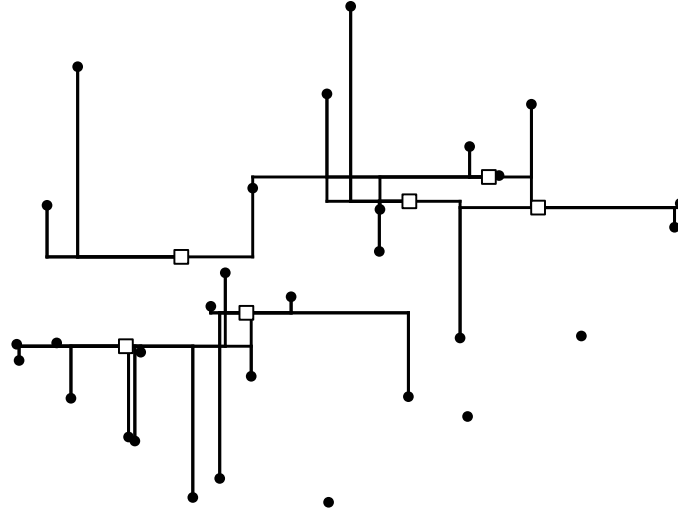


Abbildung 6.1: Lösung mit 30 Kunden, sechs Depots und zwei Zustellrobotern

In Unterkapitel 4.6.4 werden Verfahren zur Beschleunigung des Branch-and-Price-Algorithmus vorgestellt. Für diese Verfahren müssen geeignete Parameterwerte festgelegt werden. Zahlreiche Versuche und Parameterstudien ergaben, dass die Anzahl der im Pricing hinzugefügten Variablen auf 500 festgelegt werden sollte. Zudem wird die relative Anzahl der ausgehenden Kanten von jedem Kunden im Graphen $\mathcal{G}$ für das heuristische Pricing auf 30% gesetzt. Es ist zu beachten, dass diese Werte instanz- und problemspezifisch sind und diese gegebenenfalls nicht auf weitere Problemstellungen oder Instanzen übertragen werden können.

Die Präsentation der Ergebnisse ist in zwei Teile aufgeteilt. Im ersten Teil erfolgt ein Vergleich der Ergebnisse des Branch-and-Price-Algorithmus mit der MIP-Formulierung (4.1)-(4.11). Im zweiten Teil wird die Performance des Algorithmus für unterschiedliche Anzahlen von Kunden und Robotern sowie für verschiedene Zeitfensterlängen untersucht. Für alle Experimente wird eine Rechenzeitbegrenzung von einer Stunde angesetzt. Falls eine Instanz nicht innerhalb der vorgegebenen Zeit gelöst werden kann, wird die Optimalitätslücke (Gap) angegeben.

Zum besseren Vergleich wurde das MIP zusätzlich in der SCIP Optimization Suite 6.0.1 implementiert. Die Algorithmen werden anhand von Instanzen mit einem bzw. zwei Zustellrobotern verglichen. Bei Instanzen mit einem Roboter werden die ersten 15 erreichbaren Kunden, bei Instanzen mit zwei Robotern die ersten 30 erreichbaren Kunden berücksichtigt. Zur Gewährleistung einer einheitlichen Verteilung der Zeitfensterlängen werden lediglich die ersten 10 Instanzen

verwendet. Die zugehörigen Ergebnisse sind in Tabelle 6.1 bzw. in Tabelle 6.2 dargestellt. Beide Tabellen sind gleich aufgebaut. Die erste Spalte enthält den Namen der Instanz (Inst). Dieser setzt sich aus der Kundenanzahl, der Anzahl der Depots und dem Index der Instanz zusammen. Die Spalten zwei bis vier zeigen die Ergebnisse des MIPs. Dabei entspricht Spalte zwei der besten Lösung (BLB), gefolgt von der Rechenzeit (T) in Sekunden und der Optimalitätslücke (G(%)). Falls die Instanz nicht innerhalb der vorgegebenen Rechenzeit von einer Stunde gelöst werden kann, wird die Rechenzeit mit *tl* gekennzeichnet. Die Optimalitätslücke beschreibt die Differenz zwischen der besten relaxierten Lösung und der besten zulässigen Lösung, geteilt durch die beste zulässige Lösung. Die Spalten fünf bis sieben weisen denselben Aufbau auf und zeigen die Ergebnisse des Branch-and-Price-Algorithmus.

Für beide Kundenanzahlen zeigt sich, dass der Branch-and-Price-Algorithmus deutlich kürzere Rechenzeiten als das MIP erfordert. Das MIP kann 6 von 10 Instanzen mit 15 Kunden optimal lösen, jedoch keine einzige Instanz mit 30 Kunden. Im Gegensatz dazu löst der Branch-and-Price-Algorithmus alle Instanzen optimal. Im Durchschnitt beträgt die Rechenzeit für den Branch-and-Price-Algorithmus für 15 Kunden 0,49 Sekunden und für 30 Kunden 410,03 Sekunden. Im Vergleich dazu liegt der Durchschnitt für das MIP bei 1.690,97 Sekunden für 15 Kunden und bei einer Stunde für 30 Kunden, da keine der Instanzen optimal gelöst werden konnte. Damit ist gezeigt, dass der Branch-and-Price-Algorithmus auf den Benchmark-Instanzen deutlich bessere Ergebnisse erzielt. Im Folgenden wird untersucht, wie sich die Veränderung der Kundenanzahl, der Anzahl an Robotern und der Zeitfensterlängen auf die Laufzeit des Branch-and-Price-Algorithmus auswirkt.

Inst	MIP			Branch-and-Price		
	BLB	T	G(%)	BLB	T	G(%)
15-4-1	13	565,57	0	13	0,52	0
15-4-2	13	tl	15,38	13	0,57	0
15-4-3	14	138,36	0	14	0,52	0
15-4-4	13	609,45	0	13	0,45	0
15-4-5	14	127,76	0	14	0,43	0
15-4-6	13	360,7	0	13	0,6	0
15-6-7	12	tl	25	12	0,4	0
15-6-8	13	tl	7,69	13	0,51	0
15-6-9	11	708,52	0	11	0,34	0
15-6-10	12	tl	25	13	0,55	0

Tabelle 6.1: Vergleich zwischen MIP und Branch-and-Price-Algorithmus für 15 Kunden und einem Roboter

Zur Überprüfung der Auswirkungen von Änderungen der Eingabeparameter werden nacheinander die Kundenanzahl, die Roboteranzahl und die Zeitfensterlängen variiert. Die Ergebnisse werden jeweils in einheitlich aufgebauten Graphen dargestellt: Die x-Achse zeigt den variierenden Parameter, die y-Achse die Rechenzeit sowie, bei nicht optimal gelösten Instanzen, die Optimalitätslücke. Jede Instanz wird zehnmal mit zufällig gewählten Kunden gelöst und der Durchschnitt der Rechenzeiten wird abgebildet (Kreise). Falls nicht genügend erreichbare Kunden für zehn Samples vorhanden sind, wird für diese Instanz kein Durchschnitt berechnet. Die Rechenzeit ergibt sich aus der tatsächlichen Laufzeit plus $\text{Gap} \cdot 60$. Die Mediane werden durch eine Linie

Inst	MIP			Branch-and-Price		
	BLB	T	G(%)	BLB	T	G(%)
30-4-1	20	tl	50	26	15,51	0
30-4-2	22	tl	36,36	28	3.014,15	0
30-4-3	26	tl	20	29	437,3	0
30-4-4	20	tl	50	27	18,46	0
30-4-5	27	tl	11,11	29	241,77	0
30-4-6	23	tl	30,44	29	64,22	0
30-6-7	22	tl	36,36	29	49,74	0
30-6-8	20	tl	50	28	25,03	0
30-6-9	20	tl	50	27	194,43	0
30-6-10	22	tl	36,36	29	39,70	0

Tabelle 6.2: Vergleich zwischen MIP und Branch-and-Price-Algorithmus für 30 Kunden und zwei Robotern

verbunden.

Im ersten Experiment wird untersucht, wie sich die Kundenanzahl auf die Laufzeit auswirkt. Dafür wurde die Anzahl der Roboter auf drei gesetzt und alle weiteren Parameter wurden wie bereits beschrieben gewählt. Die Anzahl der Kunden variiert zwischen 34 und 50 mit einer Schrittweite von zwei Kunden. Da die Zeitfensterlängen der 20 Instanzen variieren, wurden die ersten zehn Benchmark-Instanzen mit 90 Minuten Zeitfensterlänge ausgewählt. Zu beachten ist zudem, dass die erste Instanz lediglich 47 erreichbare Kunden umfasst. Daher wurden die Ergebnisse bis einschließlich 46 Kunden auf allen zehn Instanzen berechnet und ab 48 Kunden wurden die restlichen neun Instanzen berücksichtigt.

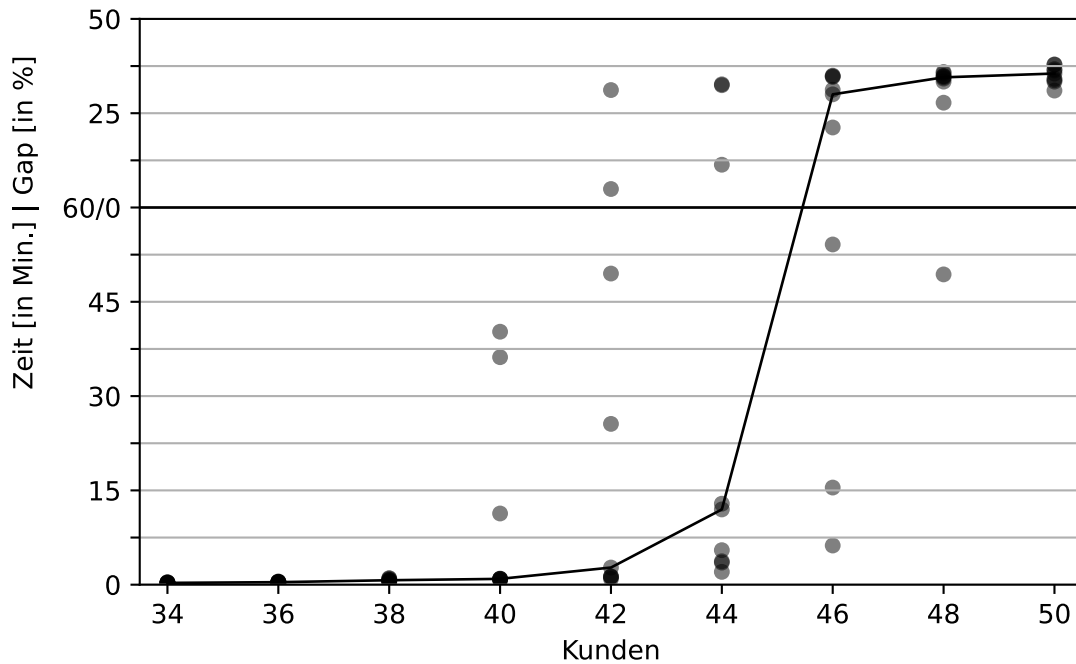


Abbildung 6.2: Ergebnisse für Instanzen mit 34 bis 50 Kunden und drei Robotern

Die Ergebnisse des ersten Experiments sind in Abbildung 6.2 dargestellt. Es zeigt sich, dass In-

stanzen bis zu 38 Kunden sehr schnell gelöst werden können. Ein möglicher Grund hierfür ist, dass bei Instanzen mit 38 Kunden in 99,55 % der Fälle alle Kunden bedient werden konnten. Für Instanzen mit mehr als 38 Kunden steigt die Laufzeit stark an und schon ab 46 Kunden überschreitet der Median eine Stunde. Ab 50 Kunden überschreiten alle durchschnittlichen Laufzeiten das Zeitlimit. Dieser Trend wird durch die Linie, die die Mediane verbindet, deutlich sichtbar. Insgesamt zeigt sich, dass die Lösbarkeit einer Instanz stark von der Kundenanzahl abhängt. Ein Erklärungsansatz für dieses Ergebnis ist, dass der Graph $\mathcal{G}$ mit der Kundenanzahl wächst, die Anzahl möglicher kürzester Wege durch den Graphen jedoch exponentiell mit der Größe von $\mathcal{G}$ zunimmt.

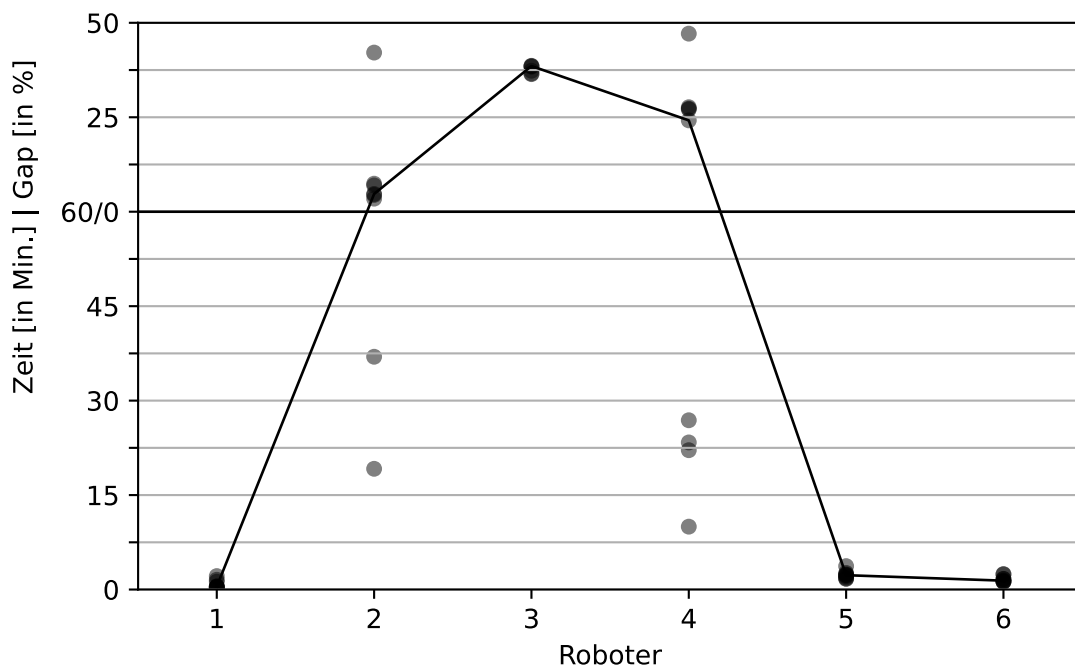


Abbildung 6.3: Ergebnisse für Instanzen mit 60 Kunden und einem bis sechs Robotern

Im zweiten Experiment wird der Einfluss der Änderung der Roboteranzahl auf die Rechenzeit untersucht. Die Roboteranzahl variiert zwischen eins und sechs. Die Kundenanzahl wurde auf 60 festgelegt, während alle weiteren Parameter unverändert blieben. Wie im ersten Experiment wurden die Instanzen zwei bis zehn untersucht. Die Ergebnisse dieses Experiments sind in Abbildung 6.3 dargestellt. Auch hier zeigt sich, dass die Roboteranzahl einen starken Einfluss auf die Rechenzeit hat. Bei Instanzen mit einem Roboter sind die Rechenzeiten sehr niedrig. Dies gilt ebenfalls für Instanzen mit fünf oder sechs Robotern. Beim Übergang von einem auf zwei Roboter steigt die Rechenzeit stark an, und bereits bei drei Robotern überschreiten alle Durchschnitte das Zeitlimit. Bei der Hinzunahme eines weiteren Roboters sinkt die Rechenzeit wieder etwas ab. Ein möglicher Erklärungsansatz ist, dass es bei einer geringen Roboteranzahl einfacher scheint, diesen die besten Kunden zuzuweisen. Bei vielen Robotern können wahrscheinlich alle Kunden bedient werden. Es zeigt sich, dass dies für alle Instanzen mit fünf oder sechs Robotern sowie für alle lösbaren Instanzen mit vier Robotern zutrifft. Interessant ist, dass es ein Verhältnis zwischen Robotern und Kunden gibt, welches das Lösen einer Instanz mit 60 Kunden innerhalb von einer Stunde für den Algorithmus unmöglich macht.

Im letzten Experiment wurde der Einfluss der Zeitfensterlänge auf die Lösbarkeit untersucht. Dazu wurden Zeitfenster mit Längen zwischen zehn Minuten und einer Stunde in Schritten von zehn Minuten betrachtet. Die Zeitfenster wurden wie folgt berechnet: Die obere Grenze blieb unverändert, während sich die untere Grenze aus der Differenz zwischen der oberen Grenze und der Zeitfensterlänge ergab. Es wurden Instanzen mit 60 Kunden und vier Robotern betrachtet. Diese Wahl gewährleistet ein ausgewogenes Verhältnis zwischen Kunden und Robotern. Es wurden die Instanzen zwei bis zehn mit mehr als 48 Kunden betrachtet. Die Ergebnisse dieses Experiments sind in Abbildung 6.4 dargestellt. Insgesamt zeigt sich ein ähnlicher Verlauf der Median-Linie wie im ersten Experiment. Es wird deutlich, dass größere Zeitfenster zu längeren Rechenzeiten führen. Ein möglicher Grund ist, dass größere Zeitfenster die Anzahl der Kanten im Hilfsgraphen $\mathcal{G}$ erhöhen können, was wiederum die Anzahl der möglichen kürzesten Wege durch den Graphen vergrößert.

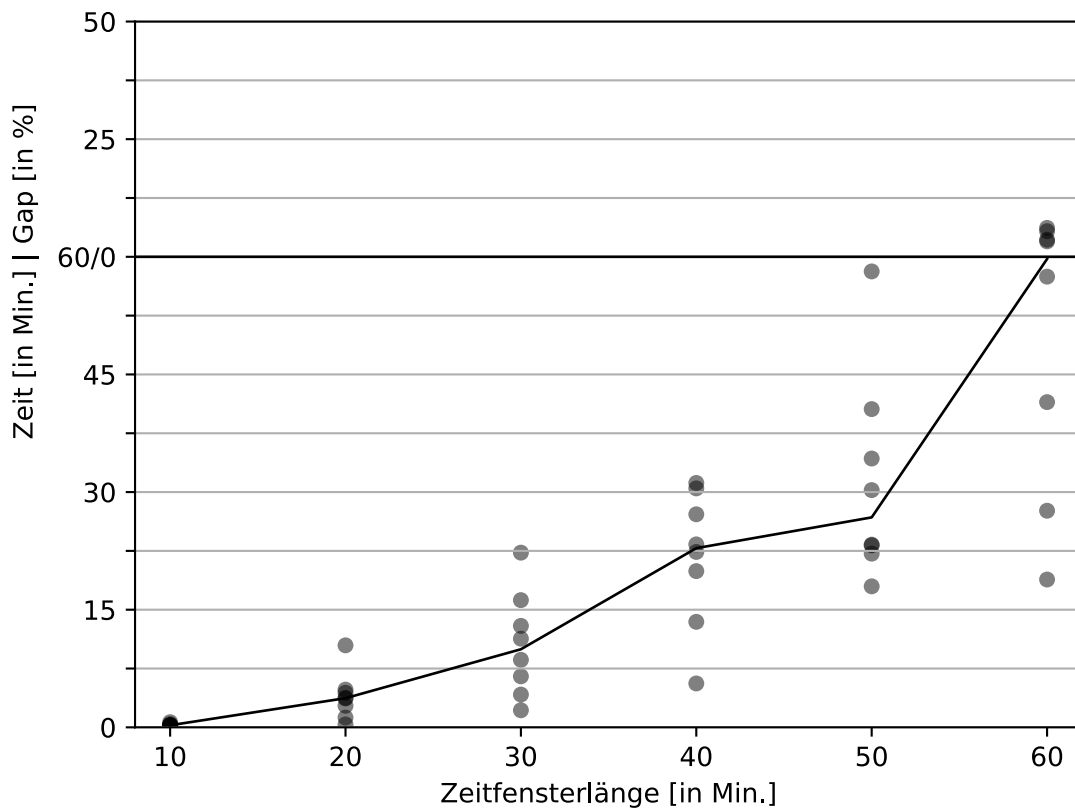


Abbildung 6.4: Ergebnisse für Instanzen mit 60 Kunden, vier Robotern und Zeitfensterlängen zwischen 10 und 60 Minuten

Zusammenfassend wurde gezeigt, dass der in Kapitel 4 beschriebene Branch-and-Price-Algorithmus im Vergleich zum MIP-Modell signifikant bessere Laufzeiten erzielt und viele Instanzen mit bis zu 60 Kunden lösen kann. Darüber hinaus wurde analysiert, wie sich eine Änderung der Parameter auf die Rechenzeit auswirkt. Damit ist die Diskussion der Ergebnisse für die Optimierung der Zustellroboter abgeschlossen.

6.2 Ergebnisse für die Geschwindigkeitsoptimierung

In diesem Abschnitt wird die Leistungsfähigkeit des Branch-and-Price-Algorithmus aus Kapitel 5 anhand verschiedener Benchmark-Instanzen untersucht. Dieses Unterkapitel basiert auf einer Arbeit mit dem Titel *Sequence and Speed Optimisation*, die im *INFORMS Journal on Computing* erschienen ist (Schaudt und Bektaş, 2026). Der Fokus liegt hierbei nicht auf dem Transport mit Paketdrohnen, sondern auf der Geschwindigkeitsoptimierung für Transportmittel auf Wasser- und Landwegen. Grund hierfür ist, dass für diese Anwendungsbereiche bereits Vergleichsergebnisse in der Literatur vorliegen.

Für den Transport zu Wasser und auf der Straße werden die Einstellungen des Branch-and-Price-Algorithmus aus Unterkapitel 5.6 gleichermaßen gewählt. Anpassbar sind die Anzahl der Variablen, die nach der Lösung des Pricing-Problems aufgenommen werden, sowie der Ausgangsgrad jedes Kunden im Hilfsgraph $\mathcal{G}$. Intensive Tests zeigten, dass die maximale Anzahl der aufzunehmenden Variablen auf 50% der Kundenanzahl begrenzt und die maximale Anzahl ausgehender Kanten auf 60% festgelegt werden sollte. Es ist anzumerken, dass diese Werte spezifisch sind und nicht auf andere Problemstellungen oder Instanzen übertragen werden können.

6.2.1 Seetransport

Hemmati et al. (2014) erstellten Instanzen für das Routing- und Planungsproblem von Industrie- und Trampschiffen (RPIT). Diese wurden von Fukasawa et al. (2018) transformiert, um sie für ein kapazitiertes Vehicle Routing Problem mit Zeitfenstern und Geschwindigkeitsoptimierung nutzbar zu machen. Die Transformation der Instanzen wird im Folgenden beschrieben, da auf deren Basis neue Instanzen mit 60, 70 und 80 Knoten erstellt wurden. Jede Instanz enthält ein Depot und eine Menge von Kunden, die beliefert werden müssen. Das ursprüngliche RPIT enthält eine Liste von Aufträgen, die Start- und Zielhäfen zugeordnet sind. Für eine feste Relation können dabei mehrere Aufträge vorliegen. Fukasawa et al. (2018) setzen die maximale Anzahl der Knoten gleich der Anzahl der Häfen im RPIT. Jeder Knoten wird jedoch, wie für das VRP üblich, nur einmal angefahren. Die nachgefragte Menge eines Knotens im kapazitierten Vehicle Routing Problem mit Zeitfenstern und Geschwindigkeitsoptimierung entspricht der Nachfrage des ersten Auftrags im RPIT, bei dem der Hafen als Starthafen angegeben ist. Zeitfenster und Servicezeiten für den Knoten werden aus demselben Auftrag gewählt.

Der Bunkerverbrauch der Schiffe wird durch die Funktion $f(v) = 0,0036v^2 - 0,1015v + 0,8848$ modelliert, wobei die Geschwindigkeit v in Knoten angegeben ist. Diese Funktion basiert auf realistischen Daten, die von Fagerholt et al. (2010) veröffentlicht wurden. Die untere und obere Geschwindigkeitsgrenze für f sind 14 bzw. 20 Knoten. Die untere Grenze wird auf $v_{min} = 1015/72 = 14,097\bar{2}$ Knoten gesetzt, da v_{min} die Funktion f minimiert.

Hemmati et al. (2014) unterteilen die RPIT-Instanzen in *deep-sea shipping* und *short-sea shipping*. Dabei beschreibt *deep-sea shipping* die Hochseeschifffahrt, also den interkontinentalen

Transport über mindestens einen der großen Ozeane. Im Gegensatz dazu bezieht sich *short-sea shipping* auf den Kurzstreckenseeverkehr, also Schiffstransporte über kürzere Distanzen, etwa im Mittelmeer. Die *short-sea shipping*-Instanzen umfassen 30 und 39 Knoten und die *deep-sea shipping*-Instanzen 30, 40 und 50 Knoten. Für jede Kombination aus Knotenanzahl und Teilmenge wurden jeweils fünf Instanzen erstellt. Es ist zu beachten, dass die *short-sea shipping*-Instanzen 2 und 4 mit eigentlich 39 Knoten nur 38 Knoten enthalten.

Basierend auf den existierenden Instanzen von Fukasawa et al. (2018) wurden neue Instanzen für das *deep-sea shipping* mit 60, 70 und 80 Knoten erstellt. Die *short-sea shipping*-Instanzen können nicht erweitert werden, da in den RPIT nicht mehr als 39 Häfen vorhanden sind. Für die neuen Knotenanzahlen wurden ebenfalls jeweils fünf Instanzen erstellt. Die neuen Instanzen für das *deep-sea shipping* können unter <https://zenodo.org/records/7153039> abgerufen werden. Eine Instanz wird referenziert mit *Typ-n-i*, wobei der Typ entweder *deep* oder *short* ist, n die Menge der Knoten angibt und i einen Wert zwischen eins und fünf annimmt. Für alle Instanzen ist die Menge der Schiffe auf fünf gesetzt.

Vor der Präsentation der Ergebnisse für den Seetransport werden zunächst die Modifikationen am Branch-and-Price-Algorithmus beschrieben. Die Problemstellung aus Unterkapitel 5.4 ist zur Lösung des kapazitierten Vehicle Routing Problems mit Zeitfenstern und Geschwindigkeitsoptimierung nicht geeignet, da diese von einer Kapazität in Höhe von einer Einheit ausgeht. Das Problem besteht darin, dass sowohl die Kapazität der Schiffe als auch die Bedarfe der Häfen größer sind. Für die Problemstellung kann der Graph G aus dem Abschnitt 5.4 verwendet werden. Die Berechnung der Distanz muss jedoch angepasst werden, da die Schiffe direkt von einem Knoten zum nächsten fahren können und der Depotbesuch zwischen zwei Knoten entfällt. Zusätzlich muss die größere Kapazität berücksichtigt werden. Es wird angenommen, dass jedes Schiff eine Kapazität von Q Einheiten hat und jeder Knoten $u_i \in V$ einen Bedarf in Höhe von q_i Einheiten aufweist. Jede Tour ist durch die Kapazität Q begrenzt, das heißt, die Summe der Bedarfe der Knoten darf Q nicht überschreiten.

Auch der Labeling-Algorithmus muss angepasst werden. Dazu wird die Länge einer jeden Kante, wie zuvor beschrieben, angepasst. Zusätzlich müssen die Kapazitäten der Schiffe und die Bedarfe der Kunden berücksichtigt werden. Hierzu wird jedes Label um einen Eintrag q erweitert. Dieser Eintrag gibt die Summe der Bedarfe des Teilwegs bis zum aktuellen Knoten an (siehe Definition 18 aus Unterkapitel 5.6.2). Um zu prüfen, ob eine Erweiterung zu einem Knoten $u_j \in \mathcal{V}$ zulässig ist, muss die Regel für die *Erweiterung* eines Labels aus Tabelle 5.5 erweitert werden. Dazu wird geprüft, ob $q + q_i \leq Q$ gilt, also ob der gesamte Bedarf des Teilwegs zusammen mit dem Bedarf von u_j die Kapazität des Schiffs überschreitet. Ist die Erweiterung zulässig, wird der Eintrag q' des neuen Labels auf $q + q_i$ gesetzt. Auch die Dominanzregel aus Unterkapitel 5.6.4 muss angepasst werden. Es gilt, dass ein Label L^A mit einem Gesamtbedarf von q^A ein Label L^B mit einem Gesamtbedarf von q^B nur dominieren kann, falls $q^A \leq q^B$ gilt. Damit sind alle Anpassungen des Labeling-Algorithmus beschrieben und die Ergebnisse der Optimierung können vorgestellt werden.

Die Ergebnisse des beschriebenen Problems werden mit den Ergebnissen von Fukasawa et al.

(2018) verglichen. Die Autoren haben zur Lösung der Problemstellung einen Branch-Cut-and-Price-Algorithmus (BCP-Algorithmus) verwendet. Die Resultate für die *deep-sea*- und *short-sea shipping*-Instanzen sind in Tabelle 6.3 dargestellt. Die erste Spalte gibt den Instanznamen an und die Spalten zwei bis vier zeigen das Ergebnis von Fukasawa et al. (2018). Die Ergebnisse wurden direkt aus der Publikation übernommen. Spalte zwei gibt die beste obere Schranke (BUB) an, Spalte drei die Rechenzeit in Sekunden (T) und Spalte vier die prozentuale Lücke (Gap, G(%)) für Instanzen, die nicht innerhalb des Zeitlimits gelöst werden konnten. Gleichmaßen sind die Spalten fünf bis sieben gestaltet. Diese Spalten repräsentieren die Resultate des Branch-and-Price-Algorithmus. Instanzen, die nicht innerhalb des Zeitlimits gelöst wurden, sind mit *tl* gekennzeichnet. Eine Zelle wird gestrichen, wenn weder eine beste obere Schranke noch eine Lücke (Gap) gefunden wurde.

Zunächst werden die Ergebnisse des Branch-and-Price-Algorithmus präsentiert und anschließend mit denen des BCP-Algorithmus verglichen. Der Branch-and-Price-Algorithmus konnte insgesamt 34 von 40 Instanzen optimal lösen und für unlösbare bleibt die Optimalitätslücke klein. Es zeigt sich, dass die Rechenzeit mit zunehmender Knotenanzahl ansteigt. Beispielsweise steigt die durchschnittliche Rechenzeit beim Schritt von 60 auf 70 Knoten von 98,65 auf 225,43 Sekunden, wobei ungelöste Instanzen ausgeschlossen werden. Ein Vergleich zwischen *deep-sea shipping* und *short-sea shipping* zeigt, dass die *deep-sea shipping*-Instanzen schneller gelöst werden. So beträgt beispielsweise die durchschnittliche Rechenzeit für *deep-sea shipping* mit 40 Knoten 3,35 Sekunden und für *short-sea shipping* liegt sie bei 24,19 Sekunden. Ein möglicher Grund dafür sind die kürzeren Distanzen im *short-sea shipping*, die zu einer größeren Anzahl gleichwertiger Touren führen könnten.

Im Vergleich der beiden Algorithmen zeigt sich, dass der Branch-and-Price-Algorithmus über fast alle Instanzen hinweg schneller ist. Für *deep-sea shipping* mit 50 Knoten wird ersichtlich, dass der BCP-Algorithmus nur eine Instanz lösen kann. Im Gegensatz dazu löst der hier vorgestellte Algorithmus alle Instanzen optimal bei einer durchschnittlichen Rechenzeit von 22,94 Sekunden. Ein ähnliches Ergebnis ergibt sich für die *short-sea shipping*-Instanzen mit 39 Knoten. Hier löst der BCP-Algorithmus drei der fünf Instanzen optimal, wobei die durchschnittliche Rechenzeit für die gelösten Instanzen 239 Sekunden beträgt. Im Gegensatz dazu löst der Branch-and-Price-Algorithmus alle Instanzen mit 39 Knoten und hat dabei eine durchschnittliche Rechenzeit von 24,19 Sekunden. Damit ist die Vorstellung der Ergebnisse für die Seeschifffahrt abgeschlossen und es folgen die Ergebnisse für den Straßengüterverkehr.

Inst	Fukasawa et al. (2018)			Branch-and-Price		
	BUB	T	G(%)	BUB	T	G(%)
deep-30-1	14.356,3	2	0	14.356,3	0,49	0
deep-30-2	17.943,5	94	0	17.943,5	0,81	0
deep-30-3	10.812,3	0,2	0	10.812,3	0,31	0
deep-30-4	15.166,7	1	0	15.166,7	0,45	0
deep-30-5	16.268,1	1	0	16.268,1	0,88	0
deep-40-1	15.738,3	5,3	0	15.738,3	1,71	0
deep-40-2	14.859,3	60	0	14.859,3	6,31	0
deep-40-3	18.877,4	200	0	18.877,4	2,26	0
deep-40-4	18.408,4	366	0	18.408,4	3,38	0
deep-40-5	18.065,4	3	0	18.065,4	3,08	0
deep-50-1	17.605,6*	tl	-	19.107	23,98	0
deep-50-2	17.918,1	tl	-	17.918,1	28,81	0
deep-50-3	19.650,6*	3,459	0	19.602,4	11,71	0
deep-50-4	19.538,9	tl	-	19.538,9	6,92	0
deep-50-5	17.605,9	tl	-	17.603,1	43,26	0
deep-60-1				19.686,3	66,31	0
deep-60-2				19.396,6	258,45	0
deep-60-3				20.579,5	53,74	0
deep-60-4				19.953	19,5	0
deep-60-5				20.179,1	95,26	0
deep-70-1				20.035,3	403,17	0
deep-70-2				-	tl	-
deep-70-3				21.613,5	163,62	0
deep-70-4				20.989,5	109,49	0
deep-70-5				19.500,8	tl	0,13
deep-80-1				21.664	tl	0
deep-80-2				-	tl	-
deep-80-3				23.135,3	315,45	0
deep-80-4				21.693,2	tl	0,35
deep-80-5				22.036,2	tl	0,44
short-30-1	3.091,8	1,2	0	3.091,78	0,67	0
short-30-2	2.814,9	2,4	0	2.814,86	1,48	0
short-30-3	3.508,4	336	0	3.508,42	6,02	0
short-30-4	3.137,5	5	0	3.137,5	0,75	0
short-30-5	3.081,6	1	0	3.081,6	0,76	0
short-39-1	3.643,1	42	0	3.643,06	3,42	0
short-39-2	4.087,6	tl	18,4	3.569,39	68,12	0
short-38-3	-	tl	-	3.620,2	32,08	0
short-39-4	3.663,4	595	0	3.663,39	14,84	0
short-38-5	3.437,3	80	0	3.437,28	2,47	0

* Es wurde versucht, die veröffentlichten Optimalwerte von (Fukasawa et al., 2018) zu reproduzieren und die Diskrepanz zu den Ergebnissen des Branch-and-Price-Algorithmus zu erklären. Die Originalautoren wurden kontaktiert, konnten jedoch die Dateien zur Verifizierung der ursprünglich berichteten Optimalwerte für diese Instanzen nicht mehr auffinden.

Tabelle 6.3: Resultate des Branch-and-Price-Algorithmus im Vergleich zu Fukasawa et al. (2018) auf den deep-sea shipping-Instanzen und short-sea shipping-Instanzen

6.2.2 Straßengüterverkehr

Die Untersuchung der optimalen Geschwindigkeiten im Straßenverkehr geht auf (Bektaş und Laporte, 2011) zurück. Die Autoren führten das sogenannte *Pollution-Routing Problem* (PRP) ein, eine Variante des kapazitierten Vehicle Routing Problems mit Zeitfenstern und Geschwindigkeitsoptimierung. Im PRP werden die Emissionen der Fahrzeuge berücksichtigt und über den Kraftstoffverbrauch abgebildet. Der Kraftstoffverbrauch ist abhängig von der Geschwindigkeit der Zustellfahrzeuge und ihrem Gewicht. Das Gewicht setzt sich aus dem Fahrzeuggewicht und den Bedarfen der Kunden zusammen und ist somit auch von deren Reihenfolge abhängig. Neben dem Kraftstoffverbrauch wird in der Zielfunktion auch der Fahrerlohn berücksichtigt, der sich aus der Dauer einer Tour ergibt.

Für das PRP wurden Instanzen veröffentlicht, die auf realen Entfernungen zwischen Städten des Vereinigten Königreichs basieren. Die Instanzen gliedern sich in die drei Teilmengen: UK-A, UK-B und UK-C. Die Teilmengen unterscheiden sich ausschließlich in der Länge der Zeitfenster. Die Instanzen aus UK-A haben die längsten Zeitfenster mit einer durchschnittlichen Länge von 20.000 Sekunden (5,55 Stunden). Die Zeitfensterlängen von UK-B und UK-C liegen gleichmäßig verteilt zwischen 2.000 und 5.000 Sekunden bzw. 2.000 und 15.000 Sekunden. Die Teilmengen sind weiter unterteilt, wobei die Instanzen unterschiedliche Anzahlen an Städten zwischen 10 und 200 umfassen. Für jede feste Anzahl an Städten wurden 20 Instanzen erstellt. Die Teilmenge UK-A wurde von Bektaş und Laporte (2011) vorgestellt, während die Erweiterungen UK-B und UK-C von Kramer et al. (2015) veröffentlicht wurden. Alle Instanzen (UK-A,UK-B,UK-C) können unter <http://www.vrp-rep.org> abgerufen werden.

Die Problemstellung des PRP ähnelt derjenigen aus dem Unterkapitel 6.2.1. Die Kraftstoffverbrauchsfunktion für eine gegebene Distanz d mit einer Geschwindigkeit v ist gegeben als:

$$f(d, v, q) = \bar{\lambda}(\bar{k}NV_H + (w + q)\gamma\bar{\alpha}v + \tilde{\beta}\gamma v^3)d/v,$$

wobei $\bar{\lambda} = \zeta/(\kappa\Psi)$, $\gamma = 1/(10000n_{tf}\eta)$ und $\bar{\alpha} = \bar{\tau} + g\sin\Theta + gC_r\cos\Theta$ konstant sind. Zur Vereinfachung wird angenommen, dass die Beschleunigung $\bar{\tau}$ und die Steigung der Straße Θ gleich null sind. Die Konstante $\tilde{\beta} = 0,5C_d\bar{\rho}\bar{A}$ beschreibt den Luftwiderstand, welcher abhängig vom Fahrzeug ist. Dabei bezeichnet w das Leergewicht des Fahrzeugs und q das Transportgewicht. Alle Parameter und Werte sind in Tabelle 6.4 aufgeführt. Eine detaillierte Herleitung der Kraftstoffverbrauchsfunktion findet sich in (Bektaş und Laporte, 2011).

Die Zielfunktion berücksichtigt nicht nur die Kraftstoffkosten, sondern auch die Lohnkosten des Fahrers. Die Gesamtkosten für das Zurücklegen einer Distanz d mit Geschwindigkeit v und Gewicht q werden wie folgt berechnet:

$$F(d, v) = f_d d/v + f_c f(d, v, q),$$

wobei f_d die Lohnkosten des Fahrers und f_c Kraftstoffkosten inklusive Abgaben für die CO_2 -Emissionen sind.

Für einen Teilweg $W^{tw} = (w_0, u_1, u_2, \dots, u_{h-1}, u_h)$ mit dem Geschwindigkeitsvektor $\mathbf{v}$ können die Kosten nicht direkt als Summe der Kantenkosten berechnet werden. Dies liegt an möglichen Wartezeiten bei den Kunden, da diese bei einer Summation der Kanten nicht eingerechnet werden und daher zu einer Unterschätzung der Gesamtkosten führen könnten. Im Folgenden wird eine Formel zur Berechnung der Kosten eines Teilwegs W^{tw} vorgestellt. Sei dazu t_h der Ankunftszeitpunkt am Kunden u_h . Es ist zu beachten, dass sich das Transportgewicht des Fahrzeugs von Kante zu Kante ändert, da die Bedarfe der Kunden abgearbeitet werden. Auf der ersten Kante (w_0, u_1) des Teilwegs entspricht das Transportgewicht der Summe der Bedarfe aller Kunden auf diesem Teilweg. Auf der zweiten Kante wird das Transportgewicht durch das Gewicht der ersten Kante abzüglich des Bedarfs des ersten Kunden bestimmt. Dies setzt sich fort bis zum letzten Kunden. Das Transportgewicht für den Teilweg W^{tw} kann daher als Vektor $\mathbf{q} = (q_1, q_2, \dots, q_h)$ beschrieben werden. Die Kosten des Teilwegs W^{tw} ergeben sich als

$$f_d(t_h + \tau_h) + f_c \sum_{i=1}^h f(d_{i-1,i}, \mathbf{v}_i, \mathbf{q}_i).$$

Notation	Beschreibung	Wert
w	Leergewicht (<i>Kilogramm</i>)	6350
ζ	Verbrennungsluftverhältnis	1
$\bar{k}$	Reibungsfaktor des Motors (<i>Kilojoule/Umdrehungen/Liter</i>)	0,2
N	Motordrehzahl (<i>Umdrehungen/Sekunde</i>)	33
V_H	Hubraum (<i>Liter</i>)	5
g	Erdanziehungskraft (<i>Meter/Sekunde</i> ²)	9,81
C_d	Strömungswiderstandskoeffizient	0,7
$\bar{\rho}$	Luftdichte (<i>Kilogramm/Meter</i> ³)	1,2041
$\bar{A}$	Frontale Fläche (<i>Meter</i> ²)	3,912
C_r	Rollwiderstandskoeffizient	0,01
n_{tf}	Wirkungsgrad des Antriebsstrangs	0,4
η	Effizienz Parameter für einen Dieselmotor	0,9
f_c	Kraftstoffkosten inkl. Kosten für CO ₂ -Emissionen pro Liter (£)	1,4
f_d	Kosten des Fahrers pro Sekunde (£/Sekunden)	0,0022
κ	Brennwert von Dieseldieselkraftstoff (<i>Kilojoule/Gramm</i>)	44
Ψ	Umrechnungsfaktor	373

Tabelle 6.4: Parameter des Pollution-Routing Problems (Demir et al., 2012)

Die Kosten eines Teilwegs setzen sich aus Fahrzeit- und Kraftstoffkosten zusammen. Abbildung 6.5 zeigt deren Verlauf für 100 km in Abhängigkeit von der Geschwindigkeit. Die Fahrerkosten sinken mit steigender Geschwindigkeit, während die Kraftstoffkosten eine konvexe Funktion mit einem Minimum bei v_{min}^* bilden. Folglich erhöht langsames Fahren den Verbrauch. Die Gesamtkosten, bestehend aus beiden Anteilen, sind ebenfalls konvex und minimal bei v_{min} . Ohne Zeitfenster wäre es optimal mit Geschwindigkeit v_{min} zu fahren. Doch durch mögliche Wartezeiten bei zu früher Ankunft kann eine geringere Geschwindigkeit günstiger sein, solange sie nicht unter v_{min} fällt oder die Wartezeit eliminiert.

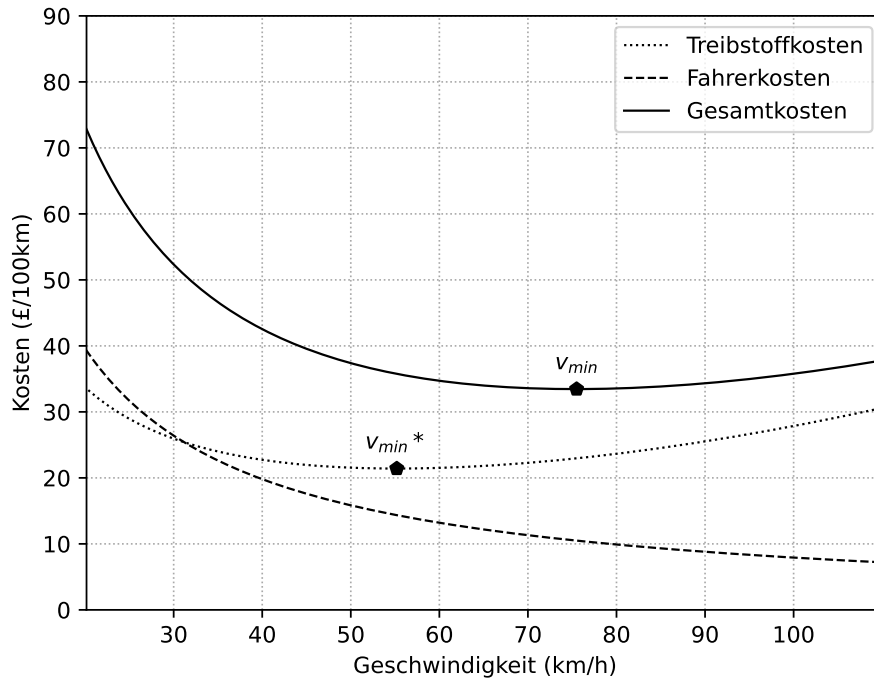


Abbildung 6.5: Kostenfunktion des Pollution-Routing-Problems in Abhängigkeit von der Geschwindigkeit

Für das PRP wird zunächst ein optimaler Geschwindigkeitsvektor mit Algorithmus 5 bei minimaler Geschwindigkeit v_{min}^* berechnet. Anschließend wird Algorithmus 6 mit maximaler Geschwindigkeit v_{min} ausgeführt. Dadurch wird der letzte Kunde frühestmöglich beliefert, ohne dass unnötige Wartezeiten durch höhere Geschwindigkeiten entstehen.

Die reduzierten Kosten eines Teilwegs $W^{tw} = (w_0, u_1, \dots, u_h)$ können unter Verwendung eines optimalen Geschwindigkeitsvektors $\mathbf{v} = (v_1, \dots, v_h)$, des daraus resultierenden Belieferungszeitpunkts t_h am letzten Kunden u_h sowie einer optimalen Lösung des dualen Problems wie folgt definiert werden:

$$\bar{c}_f(W^{tw}) = \tilde{y}^* + f_d(t_h + \tau_h + d_{h,n+1}/v_{min}) + f_{cf}(d_{h,n+1}, v_{min}, 0) + \sum_{i=1}^h (f_{cf}(d_{i-1,i}, v_i, \mathbf{q}_i) - y_i^*).$$

Damit ist die Berechnung der reduzierten Kosten eines Teilwegs abgeschlossen. Der Labeling-Algorithmus aus Unterkapitel 5.6.2 muss wie folgt angepasst werden. Die Längen der Kanten werden, wie im vorherigen Unterkapitel 6.2.1, der Problemstellung entsprechend angepasst. Jedes Label wird um die Einträge q und $\mathbf{q}$ erweitert, wobei q die Summe der Bedarfe des entsprechenden Teilwegs und $\mathbf{q} = (q_1, \dots, q_h)$ die Transportgewichte auf dem Teilweg repräsentiert. Analog zum vorherigen Abschnitt muss die Zulässigkeitsregel für die Erweiterung zu einem Knoten $u_j \in \mathcal{V}$ erweitert werden. Sei q'_j der Bedarf von Kunde u_j , dann ist die Erweiterung zulässig, falls $q + q'_j \leq Q$ gilt. Falls die Erweiterung zulässig ist, wird der Eintrag q' des neuen Labels auf $q + q'_j$ gesetzt. Zusätzlich gilt $\mathbf{q}' = (q_1 + q'_j, q_2 + q'_j, \dots, q_h + q'_j, q'_j)$. Der neue Geschwindigkeitsvektor wird wie

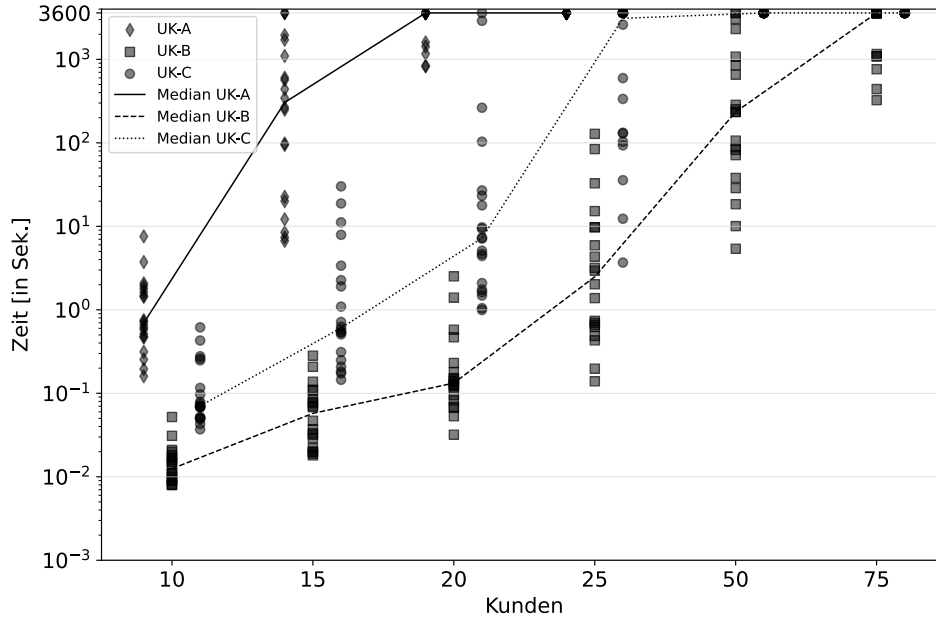


Abbildung 6.6: Ergebnisse für das PRP auf den Instanzen der Teilmengen UK-A, UK-B und UK-C

vorab beschrieben berechnet. Die Anpassungen an der Dominanzregel können aus Abschnitt 6.2.1 übernommen werden. Es ist anzumerken, dass Algorithmus 6 für die neue Zielfunktion nicht angepasst werden muss. Damit ist die Beschreibung der Anpassungen für das PRP abgeschlossen und im nächsten Schritt werden die Ergebnisse diskutiert.

Der folgende Abschnitt ist analog zum vorherigen aufgebaut: Zunächst werden die Ergebnisse des Branch-and-Price-Algorithmus vorgestellt, anschließend mit den Resultaten aus Fukasawa et al. (2017, 2018) verglichen. Die Ergebnisse wurden aus den beiden Publikationen direkt übernommen. Die Autoren haben die Fahrzeuganzahl als Anzahl der Kunden geteilt durch vier (abgerundet) festgelegt und nur die ersten zehn Instanzen je Teilmenge betrachtet. Daher erfolgen die Auswertungen sowohl mit als auch ohne fixierte Fahrzeuganzahl.

Die Ergebnisse des Branch-and-Price-Algorithmus sind in Abbildung 6.6 dargestellt. Auf der x-Achse sind die verschiedenen Kundenanzahlen abgetragen, während die y-Achse die Rechenzeiten in logarithmischer Skalierung zeigt. Die drei Teilmengen sind durch unterschiedliche Markierungen gekennzeichnet. Rauten markieren die Ergebnisse der Teilmenge UK-A, Kreise diejenigen der Teilmenge UK-C und Quadrate kennzeichnen die Ergebnisse der Teilmenge UK-B. Alle Markierungen haben 50% Transparenz. Die Abbildung enthält außerdem drei Linien, die für jede Teilmenge die Mediane der Ergebnisse verbinden. Die durchgezogene Linie verbindet die Mediane für UK-A Instanzen, eine gestrichelte Linie die Mediane der UK-B Instanzen und eine gepunktete Linie die Mediane der UK-C Instanzen. Für die Teilmenge UK-A wurden Instanzen mit 10, 15, 20 und 25 Kunden betrachtet. Für die Teilmengen UK-B und UK-C wurden zusätzlich Instanzen mit 50 und 75 Kunden berücksichtigt.

Für 10 Kunden sind die Rechenzeiten sehr gering und alle Instanzen können optimal gelöst

Inst	m	Fukasawa et al. (2017, 2018)		Branch-and-Price		$\varnothing T / T'$
		$\varnothing T$	Opt/No-Opt	$\varnothing T'$	Opt/No-Opt	
A-20	5	550,57	9/1	1.988,63	5/5	0,3
A-25	6	3.283,2	2/8	3.600	0/10	0,91
B-25	6	17,25	10/0	2,01	10/0	6,96
B-50	12	2.326,94	5/5	569,5	9/1	23,39
C-20	5	21,44	10/0	19,92	10/0	10,14
C-25	6	1.939,65	5/5	1.460,02	8/2	4,62
C-50	12	3.600	0/10	3.295,04	1/9	1,55

Tabelle 6.5: Vergleich zwischen dem Branch-and-Price-Algorithmus und dem Algorithmus von Fukasawa et al. (2017, 2018) auf den UK-A, UK-B und UK-C Instanzen

werden. Zudem zeigt sich, dass die Teilmenge UK-A am schwierigsten zu lösen ist, gefolgt von UK-C und UK-B. Der Grund hierfür sind die unterschiedlichen Zeitfensterlängen, die die Anzahl der Kanten im Hilfsgraphen $\mathcal{G}$ und damit die Zahl der möglichen Wege direkt beeinflussen. Insgesamt ist für alle drei Teilmengen ein gleicher Trend erkennbar. Betrachtet man drei aufeinanderfolgende Kundenanzahlen, dann gilt, dass für die erste Anzahl an Kunden der Median nahe 0 liegt. Von dort steigt der Median etwas an für die nächste Anzahl an Kunden und liegt bei der dritten nahe bei 3.600 Sekunden. Für die Teilmenge UK-A sind die drei Anzahlen 10, 15 und 20 Kunden. Damit ist die Vorstellung der Ergebnisse für den Branch-and-Price-Algorithmus abgeschlossen.

Ein Vergleich der Resultate mit Fukasawa et al. (2017, 2018) ist in Tabelle 6.5 abgebildet. Die erste Spalte enthält die Bezeichnung der Teilmenge sowie die Kundenanzahl. Die zweite Spalte gibt die Anzahl der Fahrzeuge an. Die Ergebnisse von Fukasawa et al. (2017, 2018) finden sich in den Spalten drei und vier, während die Spalten fünf und sechs die Resultate des Branch-and-Price-Algorithmus zeigen. Spalte drei zeigt die durchschnittliche Rechenzeit (T) in Sekunden, während Spalte vier die Anzahl der optimalen Lösungen und der Instanzen, die das Zeitlimit erreichen, enthält. Ebenso sind die Spalten fünf und sechs aufgebaut. Spalte sieben gibt den durchschnittlichen Beschleunigungsfaktor des Branch-and-Price-Algorithmus gegenüber dem BCP-Algorithmus an. Ergebnisse für zehn Kunden wurden vom Vergleich ausgeschlossen, da die Rechenzeiten nahezu null betragen und Fukasawa et al. diese nur auf eine Nachkommastelle gerundet haben. Aus dem gleichen Grund wurden auch die Ergebnisse für 20 Kunden der Teilmenge UK-B ausgeschlossen.

Der BCP-Algorithmus löst im Vergleich 41 von 70 Instanzen optimal, während der hier beschriebene Algorithmus insgesamt 43 optimale Lösungen erzielt. Der Branch-and-Price-Algorithmus berechnet generell schneller eine optimale Lösung. Über alle Instanzen ist der Branch-and-Price-Algorithmus im Durchschnitt 6,84-mal schneller. Zur Transparenz und Vergleichbarkeit sind alle Ergebnisse für das PRP im Anhang C dargestellt.

Kapitel 7

Simulationsstudie

Dieses Kapitel beschreibt ein Simulationsmodell, das die Zustellung von Paketen durch Roboter und Drohnen simuliert und die Problemstellungen aus den Kapiteln 4 und 5 realitätsnah abbildet. Das Modell verfolgt zwei Ziele: Erstens, Erkenntnisse über den Einsatz innovativer Zustellkonzepte auf der letzten Meile zu gewinnen, und zweitens, den Mehrwert der Optimierungen zu zeigen. Zu diesem Zweck werden modifizierte Szenarien vorgestellt.

Das Kapitel ist wie folgt aufgebaut. Im Unterkapitel 7.1 wird das allgemeine Simulationsmodell beschrieben, wobei der Fokus auf den Agenten *Zustellroboter* und *Drohne* liegt. Für beide Transportmittel werden weiterführende Problemstellungen definiert, die Abwandlungen der Problemstellungen aus den Kapiteln 4 und 5 darstellen. Diese Abwandlungen dienen dem Vergleich von Optimierungsansätzen sowie der Analyse ihres Mehrwerts. Im Unterkapitel 7.2 werden die Ergebnisse der Simulation präsentiert. Die Ergebnisse für Zustellroboter und Paketdrohnen werden in den Unterkapiteln 7.2.1 bzw. 7.2.2 dargestellt.

Das vorgestellte Modell ist eine Weiterentwicklung des Modells aus den Veröffentlichungen von Poeting et al. (2019a,c). Das ursprüngliche Modell simulierte die Paketzustellung mit Zustellrobotern in der Stadt Köln. Das hier präsentierte Modell baut auf dem ursprünglichen Modell auf und erweitert es um die folgenden Aspekte: Integration von Paketdrohnen, Berechnung exakter Lösungen, Aufhebung der Zuweisung zu Mikrodepots, Berücksichtigung des Akkus, Modellierung von Ladevorgängen sowie die Einbeziehung variabler Geschwindigkeiten.

7.1 Simulationsmodell

Die Einführung des Simulationsmodells erfolgt mit einer Beschreibung der zugrunde liegenden Problemstellungen. Das Simulationsmodell bildet die Zustellung von Paketsendungen mit Zustellrobotern und Drohnen ab. Die Zustellung der Sendungen erfolgt in einer zweistufigen letzten Meile, wie im Unterkapitel 2.1 beschrieben. Hierbei wird der zweite Teil von den Zustellrobotern bzw. Paketdrohnen übernommen. Es wird angenommen, dass sich die Sendungen zu Beginn

eines Tages in einem Depot befinden, das nahe den Zustellgebieten liegt. Im Depot werden die Sendungen nach Zustellgebieten aufgeteilt. Transporter übernehmen den ersten Teil der letzten Meile, indem sie die Sendungen gebündelt in die Zustellgebiete transportieren. Nach ihrer Ankunft werden die Sendungen auf Mikrodepots verteilt. Die Mikrodepots dienen als Zwischenlager für die Pakete. Im zweiten Schritt der letzten Meile transportieren Zustellroboter oder Drohnen die Sendungen einzeln von den Mikrodepots zu den Kunden. Es wird angenommen, dass die Zustellung beim Kunden nur in einem vordefinierten Zeitfenster stattfinden darf. Nach erfolgreicher Zustellung kehren die Transportmittel zu einem Mikrodepot zurück. Es gilt, dass Zustellroboter auch mehrere Mikrodepots anfahren können, bevor die Zustellung der nächsten Sendung erfolgt.

Die Mikrodepots dienen jedoch nicht nur als Lagerorte für die Sendungen sowie Roboter bzw. Drohnen, sondern bieten auch die notwendige Infrastruktur, um den Akku zu laden oder den Akkuwechsel durchzuführen. Nach der Belieferung eines Kunden erfolgt bei Zustellrobotern entweder ein Ladevorgang am Mikrodepot oder die Beladung mit einer weiteren Sendung. Bei Paketdrohnen wird nach der Rückkehr immer der Akku gewechselt, sodass eine weitere Zustellung erfolgen kann. Wie auch schon im Kapitel 4 wird für beide Transportmittel angenommen, dass nicht alle Sendungen ausgeliefert werden müssen. Das Ziel besteht jedoch darin, die Anzahl der zugestellten Sendungen zu maximieren. Diese Annahme hat den Vorteil, dass stets eine zulässige Lösung gefunden werden kann. Sollte es nicht möglich sein, eine Sendung zuzustellen, wird angenommen, dass der Kunde die Sendung selbst an einem Mikrodepot abholt. Neben der zweistufigen letzten Meile wird im Fall der Paketdrohnen auch die direkte Zustellung vom Depot aus betrachtet. Hierbei entfällt der erste Schritt der letzten Meile und die Drohnen stellen die Sendungen direkt vom Depot aus zu.

Neben dem reinen Zustellprozess wird im Simulationsmodell auch der Energieverbrauch der Roboter und Drohnen berücksichtigt. Für alle Prozesse, bei denen sich die Transportmittel in Bewegung befinden, wird der Energieverbrauch modelliert. Der Stromverbrauch während Wartezeiten wird nicht modelliert. Zudem werden die Ladevorgänge der Zustellroboter abgebildet. Eine detaillierte Beschreibung der Modellierung des Energieverbrauchs erfolgt in der Beschreibung der einzelnen Agenten. Für die Paketdrohnen wird angenommen, dass die Geschwindigkeit variabel ist und so gewählt wird, dass an erster Stelle die Anzahl der belieferten Kunden maximiert und als sekundäres Ziel der Energieverbrauch minimiert wird.

Die Problemstellungen werden mit einer Discrete-Event-Simulation mit einem agentenbasierten Modellierungsansatz abgebildet, die auf der Simulationssoftware *AnyLogic 8 University 8.8.0* basiert. Die Paketzustellung wird für eine vorgegebene Anzahl aufeinanderfolgender Tage abgebildet. Die Tage werden im Folgenden als Simulationstage bezeichnet. Jeder Simulationstag wird einzeln betrachtet und ist, abgesehen von der Verteilung der Zustellroboter bzw. Paketdrohnen auf die Mikrodepots, unabhängig vom vorherigen Simulationstag. Es ist zu beachten, dass die Experimente mit Zustellrobotern und Paketdrohnen getrennt voneinander durchgeführt werden. Da die beiden Modelle bis auf das jeweilige Transportmittel sehr ähnlich sind, erfolgt die Beschreibung in zusammengefasster Form.

Zur Umsetzung der Problemstellung werden Annahmen getroffen, die im Folgenden beschrie-

ben werden. Die Annahmen sind auch in Tabelle 7.1 zusammengefasst. Es wird angenommen, dass der Transporter, der die Sendungen in die Zustellgebiete transportiert, keine Kapazitätsbeschränkung besitzt. Außerdem wird angenommen, dass in den Mikrodepots ausreichend Fläche für die Lagerung der Sendungen sowie die Unterbringung der Zustellroboter bzw. Drohnen vorhanden ist. Für die Roboter wird angenommen, dass eine ausreichende Anzahl an Ladeplätzen vorhanden ist, während für die Paketdrohnen angenommen wird, dass in jedem Mikrodepot eine ausreichende Anzahl an geladenen Akkus für den Austausch vorgehalten wird. Der Austausch benötigt dabei eine konstante Zeit. Zustellroboter und Paketdrohnen haben eine Kapazität von jeweils einem Paket. Es wird angenommen, dass alle Sendungen dasselbe Gewicht haben und grundsätzlich für beide Transportmittel geeignet sind. Für die Roboter wird angenommen, dass die Zunahme und Abnahme des Ladestandes linear verläuft und zu Beginn eines jeden Simulationstags alle Akkus vollständig geladen sind. Der Energieverbrauch der Paketdrohnen wird mit dem Modell aus Unterkapitel 5.1 abgebildet.

Für die Roboter gilt, dass jeder Kunde mit einem vollständig geladenen Akku von mindestens einem Mikrodepot aus erreichbar ist. Zudem gilt, dass zwischen jedem Paar von Mikrodepots ein Weg existiert, der im Hinblick auf die Akkukapazität zulässig ist. Für die Paketdrohnen gilt, dass die Entfernungen im Modell nicht restriktiv sind. Dies bedeutet, dass jeder Kunde von jedem Mikrodepot aus mit maximaler Geschwindigkeit beliefert werden kann.

Bezogen auf die Lage der Kundenzeitfenster wird angenommen, dass diese nicht in Konflikt mit den Öffnungszeiten der Mikrodepots stehen. Dasselbe gilt für die Belieferung der Mikrodepots durch den Transporter. Dieser Zeitpunkt liegt früh am Simulationstag. Die Simulation berücksichtigt keine Unsicherheiten, was bedeutet, dass zu Beginn eines jeden Simulationstags alle relevanten Informationen für den Tag vorliegen.

Agent	Annahme
Transporter	Keine Kapazitätsbeschränkung
Mikrodepot	Keine Kapazitätsbeschränkung für Sendungen, Zustellroboter, Paketdrohnen, Ladestationen, geladene Akkus
Zustellroboter / Paketdrohnen	Kapazität von einer Sendung Jede Sendung für Transport geeignet Alle Sendungen haben das gleiche Gewicht
Zustellroboter	Zunahme und Abnahme des Ladestands ist eine lineare Funktion Akkus vollständig geladen zu Beginn des Tages Jeder Kunde von mindestens einem Mikrodepot erreichbar Zwischen je zwei Mikrodepots existiert ein Weg
Paketdrohne	Tausch des Akkus in konstanter Zeit
Kunde	Reichweite des Akkus ist nicht restriktiv Zeitfenster sind nicht restriktiv

Tabelle 7.1: Annahmen für das Simulationsmodell

Im Simulationsmodell wird für die Lokalisierung der Agenten und die Berechnung der Entfernungen das Geoinformationssystem von AnyLogic verwendet. Es basiert auf Daten von OpenStreetMap (OpenStreetMap contributors, 2017). Die Software ermöglicht es, für die einzelnen Transportmittel Routen und Weglängen zu berechnen. Dabei bewegen sich das Zustellfahrzeug auf dem Straßennetz, die Zustellroboter auf Gehwegen und die Drohnen durch die Luft.

Nach der Einführung des Simulationsmodells werden im Folgenden die Entscheidungen erläutert, die an jedem Simulationstag getroffen werden müssen:

1. Erstellung einer Auftragsliste für die Zustellroboter bzw. Drohnen.
2. Erstellung einer Tour für die Belieferung der Mikrodepots.

Zu 1.: Die Auftragslisten enthalten eine festgelegte Reihenfolge von Kunden und Mikrodepots. Die Reihenfolge bestimmt, welche Sendung von welchem Mikrodepot aus geliefert wird. Im Fall der Zustellroboter enthält die Auftragsliste zusätzliche Informationen, die die Ladezeiten an den Mikrodepots festlegen. Für die Paketdrohnen sind Informationen über die Fluggeschwindigkeiten hinterlegt. Der genaue Aufbau dieser Listen wird in der Beschreibung der Agenten erläutert. Auf Basis der Auftragslisten lässt sich ermitteln, welcher Roboter bzw. welche Drohne welche Sendung zustellt. Aus den Listen geht zudem hervor, welche Sendungen nicht zugestellt werden können. Kann eine Sendung nicht zugestellt werden, wird sie dem Mikrodepot zugeordnet, das dem jeweiligen Kunden am nächsten liegt. Wie bereits beschrieben, wird angenommen, dass der Kunde seine Sendung im Mikrodepot abholen muss. Zur Erstellung einer Auftragsliste können die Lösungsalgorithmen aus den Kapiteln 4 und 5 verwendet werden. Dabei ist zu beachten, dass für die Drohnen der Algorithmus angepasst werden muss, sodass nicht zwingend alle Kunden beliefert werden müssen. Die Modifikationen werden in Unterkapitel 7.1.3 vorgestellt.

Zu 2.: Auf Grundlage der Verteilung der Sendungen auf die Mikrodepots kann eine Tour zur Belieferung dieser berechnet werden. Das zugrunde liegende Problem kann als *Problem des Handlungsreisenden* formuliert werden. Dies ist möglich, da die Mikrodepots keine Kapazität aufweisen und die Öffnungszeiten nicht restriktiv sind. Mit der Lösung des Problems kann die Entfernung, die ein Zustellfahrzeug zur Belieferung der Mikrodepots zurücklegen muss, minimiert werden.

Eine weitere Annahme, die für die Simulation getroffen wird, ist, dass ausschließlich optimale Lösungen berücksichtigt werden. Findet die Optimierung innerhalb einer vorgegebenen Zeit keine optimale Lösung, wird der Simulationstag nicht in die Auswertung einbezogen. In diesem Fall wird die Anzahl der simulierten Tage jedoch um einen Tag erhöht. Die Einschränkung auf ausschließlich optimale Lösungen ist nötig, um die Vergleichbarkeit der verschiedenen Szenarien sicherzustellen. Die Laufzeit der Optimierungsläufe kann vor dem Start der Simulation festgelegt werden. Im Folgenden werden die zentralen Agenten der Simulation beschrieben.

Die Agenten *Depot* und *Transporter* spielen in der Simulation eine untergeordnete Rolle, da sie lediglich dafür sorgen, dass die Mikrodepots mit den Sendungen beliefert werden bzw. die Sendungen bis zur Auslieferung aufbewahren. Daher wird auf eine detaillierte Beschreibung dieser

Agenten verzichtet und stattdessen auf Poeting et al. (2019c) verwiesen. Der Agent *Kunde* entspricht dem Empfänger einer Sendung und ist durch seine Adresse sowie ein Zeitfenster definiert. Die Übergabe der Sendungen an den Kunden kann ausschließlich im festgelegten Zeitfenster erfolgen. Der Agent *Mikrodepot* lagert auch im Simulationsmodell die Sendungen. Zusätzlich werden auch die nicht zugestellten Sendungen gelagert. Diese werden gezählt und dienen der Ermittlung des erreichten Servicegrads. In den folgenden zwei Unterkapiteln werden die Agenten *Zustellroboter* und *Drohne* sowie die Beschreibung der angepassten Optimierungsmodelle für diese Agenten vorgestellt. Zusätzlich werden alternative Probleme und Lösungskonzepte beschrieben, die zum Vergleich mit der hier dargestellten Problemstellung dienen.

7.1.1 Modellierung des Agenten Zustellroboter

Der Ablauf der Handlungen jedes Zustellroboters basiert auf seiner Auftragsliste. Diese Liste wird mit dem Branch-and-Price-Algorithmus aus Kapitel 4 berechnet. Eine beispielhafte Auftragsliste für einen Zustellroboter ist in Tabelle 7.2 dargestellt. In der ersten Spalte sind die anzufahrenden Knoten aufgelistet. Dabei sind d die Mikrodepots und c die Kunden. In der zweiten Spalte sind die Ankunftszeitpunkte an den Knoten angegeben, und in der letzten Spalte sind die Ladezeiten an den Mikrodepots in Sekunden aufgeführt. Im Beispiel befindet sich der Roboter zu Beginn am Mikrodepot d_0 und wechselt zu Beginn des Simulationstages zum Mikrodepot d_1 . Dieser Wechsel erfolgt, um eine Startverteilung der Roboter auf die Mikrodepots zu realisieren, die von der Optimierung berechnet wurde. Diese Umverteilung der Roboter ist notwendig, da die Optimierung die Verteilung der Roboter des Vortages nicht in die Berechnung einbezieht. Nach der Ankunft am Mikrodepot wird der Akku wieder vollständig aufgeladen und die Lösung des Branch-and-Price-Algorithmus wird ausgeführt. In der letzten Zeile ist das finale Mikrodepot aufgeführt, das das Ende des Simulationstages für den Roboter markiert. An diesem Mikrodepot wird der Akku wieder vollständig aufgeladen.

Knoten	Ankunftszeitpunkt	Ladezeit[s]
d_0	-	0,0
d_1	06:05	131,25
c_1	09:00	-
d_1	09:32	0,0
c_2	10:08	-
d_2	10:48	283,0
d_0	11:06	444,0
c_2	13:07	-
d_1	13:30	338,0

Tabelle 7.2: Beispielhafte Auftragsliste für einen Zustellroboter

Neben der Auftragsliste verfügt jeder Roboter über ein Zustandsdiagramm, das die Handlungsmöglichkeiten und Übergänge beschreibt. In Abbildung 7.1 ist das Zustandsdiagramm für die Zustellroboter dargestellt. Zu Beginn befindet sich der Agent im Zustand *Warten* in einem Mikrodepot. Von diesem Zustand aus gibt es drei Möglichkeiten:

1. Laden des Akkus,
2. Wechsel des Mikrodepots,
3. Belieferung eines Kunden.

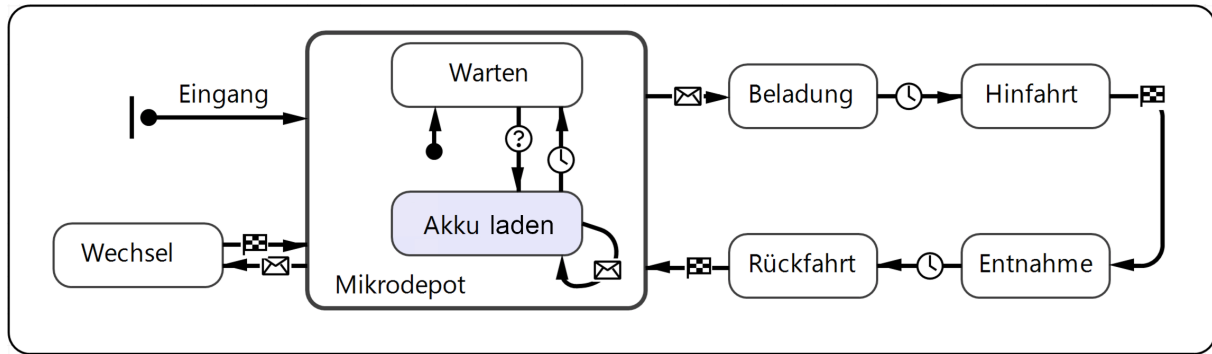


Abbildung 7.1: Zustandsdiagramm des Agenten Zustellroboter

Zu 1.: Das Laden des Akkus wird durch den Zustand *Akku laden* repräsentiert. Der Roboter gelangt in diesen Zustand nach der Rückkehr in ein Mikrodepot und lädt für eine in der Auftragsliste angegebene Dauer. Nach Ablauf der Ladezeit wechselt der Roboter zurück in den Zustand *Warten*.

Zu 2.: Der Wechsel des Mikrodepots ist im linken Abschnitt des Zustandsdiagramms abgebildet. Der Übergang in den Zustand *Wechsel* wird entweder durch zwei direkt aufeinanderfolgende Mikrodepots in der Auftragsliste des Zustellroboters oder durch die Umsetzung der Startverteilung zu Beginn eines Simulationstags ausgelöst. In diesem Zustand bewegt sich der Roboter von einem Mikrodepot zum nächsten.

Zu 3.: Die letzte Möglichkeit, den Zustand *Warten* zu verlassen, besteht in der Belieferung eines Kunden. Der Ablauf der Zustellung einer Sendung ist im rechten Abschnitt des Zustandsdiagramms dargestellt. Der Startzeitpunkt am Mikrodepot kann anhand des Ankunftszeitpunkts beim Kunden, der in der Auftragsliste aufgeführt ist, zurückgerechnet werden. Im ersten Schritt wird der Roboter im Zustand *Beladung* für eine vordefinierte Zeit mit der entsprechenden Sendung im Mikrodepot beladen. Im Anschluss wechselt der Agent in den Zustand *Hinfahrt* und fährt zum Kunden. Beim Kunden angekommen, wird der Zustand *Entnahme* aufgerufen und der Roboter wird für eine vordefinierte Zeit entladen. Im letzten Zustand *Rückfahrt* fährt der Roboter zurück in ein Mikrodepot, welches der Auftragsliste entnommen wird. Bei den Fahrten zum Kunden und zurück zum Depot oder beim Depotwechsel verringert sich der Akku entsprechend der Fahrzeit. Die Entladung des Akkus während Wartezeiten oder bei der Be- und Entladung des Roboters ist nicht abgebildet.

Damit ist die Vorstellung des Agenten *Zustellroboter* abgeschlossen. Es folgt die Beschreibung weiterer Varianten des Problems, die mit der zuvor beschriebenen Problemstellung verglichen werden sollen.

7.1.2 Problemvarianten für Zustellroboter

Wie bereits zu Beginn des Kapitels dargelegt, besteht das Ziel darin, den Mehrwert der Optimierung zu evaluieren. Im Folgenden werden daher zwei Problemvarianten näher untersucht:

1. Feste Zuweisung der Roboter zu den Mikrodepots,
2. Einfache Ladestrategie.

Beide Varianten stellen Erweiterungen bzw. Modifikationen des Problems aus Kapitel 4 dar. Die verwendeten Notationen orientieren sich an denen aus Kapitel 4.

Zu 1.: Diese Variante ähnelt dem Problem, das von Poeting et al. (2019a) untersucht wurde. Zu Beginn eines jeden Simulationstages wird entschieden, von welchem Mikrodepot aus wie viele Zustellroboter fahren sollen, wobei die Roboter das Mikrodepot während des Tages nicht wechseln dürfen. Ziel dieser Problemvariante ist es, einen Vergleich zu ermöglichen, der den Mehrwert der Auflösung der festen Zuweisung evaluiert. Für die Umsetzung dieser Variante wird angenommen, dass jedem Kunden das nächstgelegene Mikrodepot zugeordnet ist. Für jedes Mikrodepot wird im Anschluss für jede Anzahl an Robotern eine optimale Lösung berechnet, sodass insgesamt $m|D|$ Zielfunktionswerte gegeben sind. Dabei beschreibt m die Anzahl der Zustellroboter und $|D|$ die Anzahl der Mikrodepots. Zur Berechnung der Zielfunktionswerte kann der Branch-and-Price-Algorithmus aus Kapitel 4 verwendet werden. Dazu wird die Menge der Kunden und der Graph entsprechend angepasst.

Die optimale Verteilung der Zustellroboter auf die Mikrodepots lässt sich als folgendes ganzzahliges Programm formulieren. Sei dazu $M = \{0, 1, 2, \dots, m\}$ die Menge der Zahlen von 0 bis m und sei D die Menge der Mikrodepots. Die binäre Variable z_{ij} gibt an, ob einem Mikrodepot $u_j \in D$ genau $i \in M$ Roboter zugewiesen sind. Der Koeffizient c_{ij} repräsentiert die Anzahl an erfolgreichen Zustellungen bei dieser Zuweisung. Damit lässt sich die optimale Verteilung der Roboter auf die Mikrodepots als das folgende IP formulieren:

$$\text{maximiere} \quad \sum_{i \in M} \sum_{u_j \in D} c_{ij} z_{ij} \quad (7.1)$$

$$\text{u.d.N.} \quad \sum_{i \in M} \sum_{u_j \in D} i z_{ij} \leq m, \quad (7.2)$$

$$\sum_{i \in M} z_{ij} = 1 \quad \forall u_j \in D, \quad (7.3)$$

$$z_{ij} \in \{0, 1\} \quad \forall i \in M, u_j \in D. \quad (7.4)$$

Die Zielfunktion (7.1) maximiert die Anzahl der Zustellungen. Die Nebenbedingung (7.2) stellt sicher, dass höchstens m Roboter auf die Mikrodepots verteilt werden. Mit den Nebenbedingungen (7.3) wird garantiert, dass jedem Mikrodepot genau eine spezifische Anzahl an Robotern zugewiesen wird. Die Ganzzahligkeit der Entscheidungsvariablen wird durch Bedingung (7.4) sichergestellt.

Nach der Berechnung der Verteilung der Zustellroboter auf die Mikrodepots erfolgt die Neuzuweisung der Roboter zu den Mikrodepots. Dabei wird die Verteilung des Vortags berücksichtigt, und die Roboter werden gemäß der Lösung des IP neu zugewiesen. Im Anschluss werden die berechneten Lösungen an den Mikrodepots in der Simulation umgesetzt. Damit ist die Vorstellung dieser Problemvariante abgeschlossen und es folgt die Diskussion der vereinfachten Ladestrategie.

Zu 2.: Um den Mehrwert der optimalen Ladezeit zu evaluieren, wird eine zusätzliche Problemvariante eingeführt. In dieser Variante wird angenommen, dass die Akkus automatisch geladen werden, sobald eine vordefinierte Nachladegrenze unterschritten wird. Allerdings dürfen die Akkus auch erst ab diesem Ladestand geladen werden. Es wird angenommen, dass nur vollständige Ladezyklen möglich sind. Eine Ausnahme gilt zu Beginn und am Ende eines Simulationstages. Damit ist garantiert, dass die Roboter den Tag mit einem vollständig geladenen Akku beginnen und die Zuweisung zu Beginn des Tages umgesetzt werden kann. Zur Formulierung des Problems wird zur Vereinfachung angenommen, dass zwischen jedem Paar von Kunden höchstens ein Mikrodepot angefahren wird. Dabei wird immer das Mikrodepot gewählt, das die Entfernung zwischen den Kunden minimiert.

Im Folgenden wird ein MIP für das beschriebene Problem formuliert. Durch die vorgegebene Wahl des Mikrodepots zwischen jedem Kundenpaar werden die Mikrodepots in der Formulierung implizit berücksichtigt. Ein Dummy-Depot wird benötigt, das für jeden Kunden das nächstgelegene Mikrodepot repräsentiert und gleichzeitig das erste und letzte Mikrodepot eines jeden Roboters an einem Simulationstag darstellt. Sei $V = \{u_0\} \cup \{C\}$ die Menge der Knoten, wobei u_0 das Dummy-Depot und C die Menge der Kunden ist. Die Binärvariable x_{ij} , mit $u_i, u_j \in V$, ist gleich eins, falls ein Roboter von Knoten u_i nach u_j fährt. Die Variable t_i , mit $u_i \in C$, beschreibt den Belieferungszeitpunkt vom Kunden u_i und die Variable λ_i , mit $u_i \in C$, beschreibt die Ladezeit am Mikrodepot, das auf Kunde u_i in der Lösung folgt. Die Variable β_i , mit $u_i \in C$, beschreibt den Ladestand des Akkus bei Ankunft beim Kunden u_i und die Variable y_i gibt an, ob an dem Mikrodepot, das auf Kunde u_i in einer Lösung folgt, geladen wird. Der Parameter τ_i bezeichnet die Servicezeit an u_i und α entspricht der Laderate der Roboter. Die Fahrzeit zwischen den Knoten u_i und u_j , mit $u_i, u_j \in C$, wird mit d_{ij} bezeichnet und d_i bezeichnet die Fahrzeit von u_i zu seinem nächstgelegenen Mikrodepot. Die Fahrzeit von Kunde u_i zum Mikrodepot zwischen den Knoten u_i und u_j wird mit $\tilde{d}_{ij}$ bezeichnet. Die Kapazität des Akkus wird mit B bezeichnet, während der Ladestand, ab dem aufgeladen werden muss, mit $\underline{\beta}$ angegeben ist. Damit lässt sich das MIP wie folgt formulieren:

$$\text{maximiere} \quad \sum_{u_i \in C} \sum_{u_j \in V} x_{ij} \quad (7.5)$$

$$\text{u.d.N.} \quad \sum_{u_i \in V} x_{ij} \leq 1 \quad \forall u_j \in C, \quad (7.6)$$

$$\sum_{u_i \in V} (x_{ij} - x_{ji}) = 0 \quad \forall u_j \in C, \quad (7.7)$$

$$\sum_{u_i \in C} x_{0i} \leq m. \quad (7.8)$$

$$t_i + \tau_i + \lambda_i + d_{ij} \leq t_j + (1 - x_{ij})M_1 \quad \forall u_i, u_j \in C, \quad (7.9)$$

$$d_i \leq t_i \quad \forall u_i \in C, \quad (7.10)$$

$$\beta_i + \alpha\lambda_i - d_{ij} \geq \beta_j - (1 - x_{ij})M_2 \quad \forall u_i, u_j \in C, \quad (7.11)$$

$$\beta_i - \sum_{u_j \in C} \tilde{d}_{ij}x_{ij} \geq \underline{\beta}(1 - y_i) \quad \forall u_i \in C, \quad (7.12)$$

$$\underline{\beta} - (\beta_i - \sum_{u_j \in C} \tilde{d}_{ij}x_{ij}) + B \geq B(1 - y_i) \quad \forall u_i \in C, \quad (7.13)$$

$$\beta_i - \sum_{u_j \in C} \tilde{d}_{ij}x_{ij} + \alpha\lambda_i \geq By_i \quad \forall u_i \in C, \quad (7.14)$$

$$\beta_i - \sum_{u_j \in C} \tilde{d}_{ij}x_{ij} + \alpha\lambda_i \leq B \quad \forall u_i \in C, \quad (7.15)$$

$$0 \leq \lambda_i \leq 1/\alpha By_i \quad \forall u_i \in C, \quad (7.16)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A, \quad (7.17)$$

$$y_i \in \{0, 1\} \quad \forall u_i \in C, \quad (7.18)$$

$$a_i \leq t_i \leq b_i \quad \forall u_i \in C, \quad (7.19)$$

$$d_i \leq \beta_i \leq B - d_i \quad \forall u_i \in C. \quad (7.20)$$

Die Zielfunktion (7.5) maximiert die Anzahl der belieferten Kunden. Die Nebenbedingungen (7.6) stellen sicher, dass jeder Kunde höchstens einmal beliefert wird. Die Flusserhaltung wird durch die Nebenbedingungen (7.7) garantiert. Die Nebenbedingung (7.8) begrenzt die maximale Anzahl an Robotern auf m . Die Berechnung der korrekten Ankunftszeitpunkte zwischen zwei aufeinanderfolgenden Kunden wird mit den Nebenbedingungen (7.9) sichergestellt. Hierbei beschreibt M_1 eine ausreichend große Zahl. Die Ungleichungen (7.10) stellen sicher, dass der Ankunftszeitpunkt bei einem Kunden mindestens der Fahrzeit vom nächstgelegenen Mikrodepot zum Kunden entspricht. Die Forderungen an den Ladestand des Akkus sind mit den Nebenbedingungen (7.11) bis (7.16) abgebildet. Die Nebenbedingungen (7.11) beschreiben das Verhältnis der Ladestände von Kunde u_i zu Kunde u_j , $u_i, u_j \in C$, falls die beiden Kunden aufeinanderfolgen. Dabei bezeichnet M_2 eine ausreichend große Zahl. Die Nebenbedingungen (7.12) stellen sicher, dass der Ladestand des Akkus bei der Ankunft an einem Mikrodepot entweder größer als $\underline{\beta}$ ist oder dass eine Aufladung am Mikrodepot erforderlich ist. Die Nebenbedingungen (7.13) stellen sicher, dass an einem Mikrodepot nur geladen werden darf, wenn der Ladestand am Mikrodepot unterhalb der Ladegrenze ist. Falls an einem Mikrodepot geladen werden muss, dann stellen die Nebenbedingungen (7.14) bis (7.16) sicher, dass der Akku vollständig geladen wird. Die Wertebereiche der Entscheidungsvariablen werden mit den Bedingungen (7.17) bis (7.20) gesetzt.

Es ist zu beachten, dass der Ladevorgang am letzten Mikrodepot im MIP nicht berücksichtigt wird, da dieser für die Lösung nicht relevant ist. Die Ladezeit nach dem letzten Kunden wird manuell festgelegt. Diese entspricht einer vollständigen Aufladung des Akkus. Zusätzlich müssen nach der Umsetzung der Startverteilung die Ladezeiten gesetzt werden, sodass jeder Roboter den Simulationstag mit einem vollständig geladenen Akku beginnen kann. Damit ist die Beschreibung der Problemvarianten abgeschlossen und die Vorstellung des Agenten *Drohne* erfolgt.

7.1.3 Modellierung des Agenten Drohne

Die Vorstellung des Agenten Drohne gliedert sich in zwei Teile: Zunächst wird beschrieben, wie die Optimierungsalgorithmen aus Kapitel 5 an die Anforderungen der Simulation angepasst wurden. Anschließend folgt die Darstellung, wie der Agent Drohne im Simulationsmodell technisch umgesetzt wurde. Die Modellierung basiert auf einer zweistufigen letzten Meile, im späteren Verlauf wird jedoch auch die Zustellung per Drohne ab dem Depot betrachtet. In diesem Fall übernimmt das Depot die Rolle des Mikrodepots, und die Sendungen verbleiben nach der Sortierung dort, anstatt an einen Transporter übergeben zu werden.

Zur Erstellung einer Auftragsliste muss ein angepasstes Optimierungsproblem gelöst werden, das sich von der in Kapitel 5 beschriebenen Problemstellung unterscheidet. Statt den Energieverbrauch zu minimieren, wird nun primär die Anzahl der belieferten Kunden maximiert und sekundär der Energieverbrauch minimiert. Diese Änderung der Zielstellung stellt sicher, dass überhaupt eine Lösung gefunden wird, da nicht mehr jeder Kunde zwingend beliefert werden muss. Im Folgenden wird das angepasste Problem vorgestellt.

Das IP (5.8)-(5.11) aus Unterkapitel 5.6 muss wie folgt umformuliert werden. Sei Ω die Menge aller möglichen Sequenzen und C die Menge der Kunden. Die Entscheidungsvariable x_s ist gleich eins, falls die Sequenz $s \in \Omega$ ausgeführt wird, und null sonst. Der minimale Energieverbrauch für die Ausführung der Sequenz s unter Verwendung der Funktion f sei $c_f(s)$. Der Koeffizient e_{is} gibt an, ob der Kunde $u_i \in C$ in der Sequenz s enthalten ist. Jeder belieferte Kunde reduziert die Zielfunktion um c_M , wobei gilt, dass $c_M > c_f(s)$ für alle $s \in \Omega$. Es stehen m Drohnen zur Belieferung der Kunden zur Verfügung. Das Problem lässt sich daher als das folgende IP formulieren:

$$\text{minimiere} \quad \sum_{s \in \Omega} x_s (c_f(s) - \sum_{u_i \in s} c_M e_{is}) \quad (7.21)$$

$$\text{u.d.N.} \quad \sum_{s \in \Omega} e_{is} x_s \leq 1 \quad \forall u_i \in C, \quad (7.22)$$

$$\sum_{s \in \Omega} x_s \leq m, \quad (7.23)$$

$$x_s \in \{0, 1\} \quad \forall s \in \Omega. \quad (7.24)$$

Die Zielfunktion (7.21) maximiert an erster Stelle die Anzahl der belieferten Kunden und minimiert für die größte Anzahl an belieferten Kunden die dazu benötigte Energie. Die Nebenbedingungen (7.22) stellen sicher, dass jeder Kunde in höchstens einer ausgewählten Sequenz vertreten ist. Die Ungleichung (7.23) garantiert, dass höchstens m Entscheidungsvariablen gewählt werden. Die Ganzzahligkeit der Entscheidungsvariablen $x_s, s \in \Omega$, wird durch die Bedingungen (7.24) gewährleistet.

Durch die Anpassung des IP muss auch das Pricing-Problem des Branch-and-Price-Algorithmus angepasst werden. Sei dazu $\mathbf{y}^*$ eine optimale Lösung des DP zum EMP. Die reduzierten Kosten

einer zulässigen Sequenz $s \in \Omega$ ergeben sich wie folgt:

$$\bar{c}_f(s, \mathbf{y}^*) = c_f(s) - \sum_{u_i \in s} (c_M + y_i^*) e_{is} - \tilde{y}^*.$$

Die optimale Lösung des Pricing-Problems ergibt sich daher als $\bar{c}^*(\mathbf{y}^*) = \max_{s \in \Omega} \bar{c}_f(s, \mathbf{y}^*)$. Die weiteren Anpassungen sind analog zur Beschreibung des Labeling-Algorithmus für das ursprüngliche IP, wie im Unterkapitel 5.6 beschrieben, und werden daher nicht im Detail erläutert. Der neue Algorithmus wird im Folgenden als angepasster Branch-and-Price-Algorithmus bezeichnet.

Zur Berechnung der Auftragsverteilung der Sendungen auf die Drohnen sowie der entsprechenden Reihenfolgen und Geschwindigkeiten kann der angepasste Branch-and-Price-Algorithmus verwendet werden. Dabei sollte zwischen je zwei Kunden das Mikrodepot gewählt werden, das die Entfernung minimiert, da die Entfernungen nicht restriktiv sind.

Die Lösung des Branch-and-Price-Algorithmus wird als Auftragsliste dargestellt. Ein Beispiel zeigt Tabelle 7.3. Spalte eins enthält die anzufahrenden Knoten (d für Mikrodepots, c für Kunden), Spalte zwei die Ankunftszeiten und Spalte drei die Abfluggeschwindigkeiten. Im Beispiel startet die Drohne am Mikrodepot d_0 und wechselt zur Startverteilung nach d_1 , wobei sie mit energieoptimaler Geschwindigkeit fliegt (gemäß Modell aus Unterkapitel 5.1). Der anschließende Zustellflug zeigt, dass Hin- und Rückflug mit minimalem Energieverbrauch erfolgen, da erste Zeitfenster und Depot-Schlusszeit als nicht restriktiv angenommen werden.

Knoten	Ankunftszeitpunkt	Geschwindigkeit [km/h]
d_0	-	72,4
d_1	06:15	72,4
c_1	09:00	90,0
d_1	09:32	90,0
c_2	10:08	80,0
d_2	10:48	80,0
c_2	13:07	72,4
d_1	13:30	-

Tabelle 7.3: Beispielhafte Auftragsliste für eine Paketdrohne

Anschließend wird die Implementierung des Agenten *Drohne* in der Simulation beschrieben. Die möglichen Handlungen werden in einem Zustandsdiagramm beschrieben, das in Abbildung 7.2 dargestellt ist. Das Zustandsdiagramm der Drohne weist im Vergleich zu den Zustellrobotern mehr Zustände auf, wobei die Grundfunktionen jedoch ähnlich sind. Auch die Drohne befindet sich zu Beginn im Zustand *Warten*. Ausgehend von diesem Zustand gibt es für den Agenten drei Möglichkeiten:

1. Wechsel des Akkus,
2. Wechsel des Mikrodepots,
3. Belieferung eines Kunden.

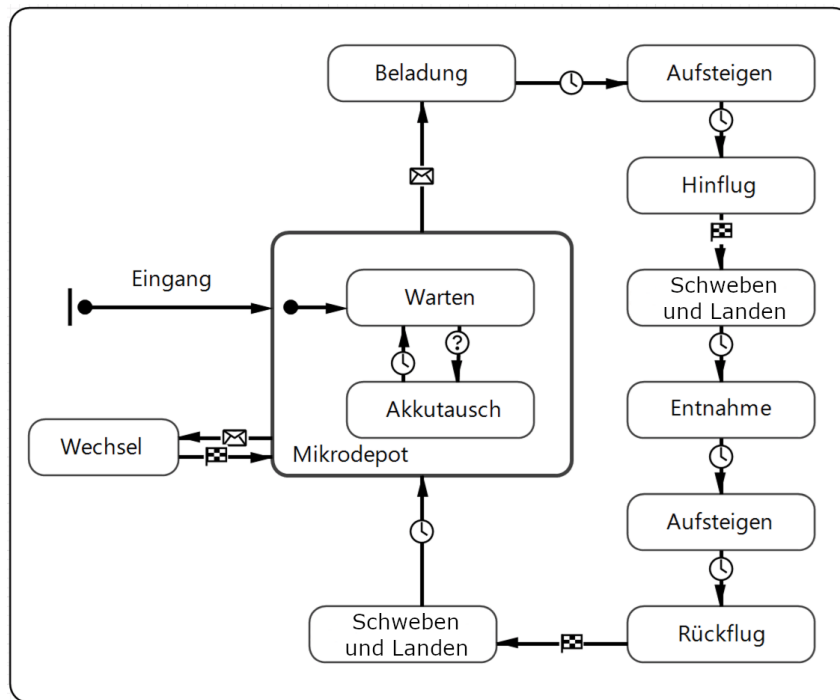


Abbildung 7.2: Zustandsdiagramm des Agenten Drohne

Zu 1.: Der Zustand *Akkutausch* erfolgt bei jeder Rückkehr ins Mikrodepot. Der Akku wird in konstanter Zeit gewechselt, sodass jeder Flug mit vollem Akku beginnt.

Zu 2.: Der Zustand *Wechsel* wird nur zu Beginn eines Simulationstages aufgerufen und setzt die Startverteilung der Optimierung um. Er umfasst Aufstieg, Flug, Schweben und Laden am neuen Mikrodepot. Der Energieverbrauch wird vom Simulationsmodell erfasst.

Zu 3.: Für die Belieferung eines Kunden durchläuft die Drohne nacheinander die Zustände *Beladung*, *Aufsteigen*, *Hinflug*, *Schweben und Landen*, *Entnahme* sowie die gleichen Zustände in umgekehrter Reihenfolge für den Rückflug. Geschwindigkeit und Zielknoten lassen sich der Auftragsliste entnehmen. Nach Rückkehr ins Mikrodepot erfolgt ein Akkuwechsel. Der Energieverbrauch wird in der Simulation erfasst. Abbildung 7.3 visualisiert die Zustände über Zeit (x-Achse) und Flughöhe (y-Achse).

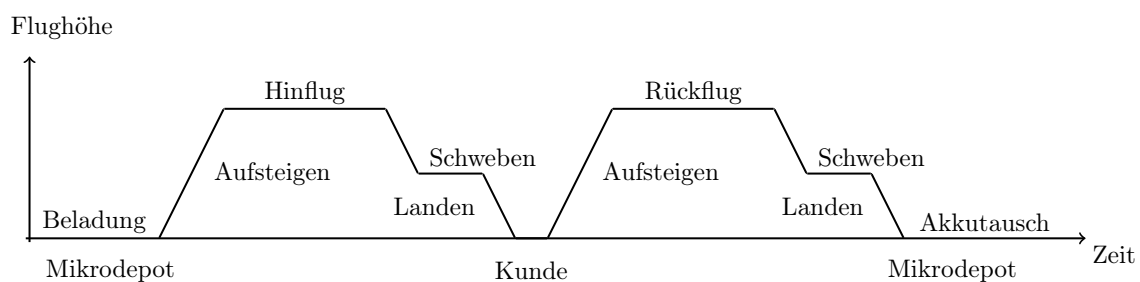


Abbildung 7.3: Flugmuster einer Paketdrohne (Kirschstein, 2020)

7.1.4 Problemvarianten für Paketdrohnen

Zur Evaluierung des Mehrwerts eines optimierten Zustellnetzwerks mit Paketdrohnen wird eine alternative Problemvariante untersucht. In dieser Variante wird angenommen, dass die Flugeschwindigkeiten diskret verteilt sind und lediglich eine begrenzte Anzahl von möglichen Geschwindigkeiten betrachtet wird.

Die Notation folgt Kapitel 5. Die Diskretisierung der Geschwindigkeiten hat den Vorteil, dass sich ein einfaches, lineares MIP formulieren lässt. Der Nachteil dieser Annahme ist, dass die Drohnen gegebenenfalls nicht mit der optimalen, sondern nur mit der nächsthöheren verfügbaren Geschwindigkeit fliegen müssen. Der Grad der Diskretisierung entscheidet, wie weit die verfügbaren Geschwindigkeiten von den optimalen abweichen.

Sei Φ die Menge der Geschwindigkeiten und $v_l \in \Phi$ eine beliebige Geschwindigkeit. Es wird ein Dummy-Depot genutzt, das dem ersten und letzten Mikrodepot einer jeden Paketdrohne an einem Simulationstag entspricht. Sei $V = \{u_0\} \cup C$ die Menge der Knoten, die sich aus der Menge der Kunden C und dem Dummy-Depot u_0 zusammensetzt. Die binäre Variable x_{ij}^l , für $u_i, u_j \in V$, und $v_l \in \Phi$, ist eins, falls eine Paketdrohne von Knoten u_i nach u_j mit Geschwindigkeit v_l fliegt. Die kontinuierliche Variable t_i beschreibt den Belieferungszeitpunkt des Kunden u_i . Das Zeitfenster von Kunde u_i ist $[a_i, b_i]$. Der Energieverbrauch von Knoten u_i nach $u_j \in V$ mit einer Geschwindigkeit $v_l \in \Phi$ wird durch $d_{ij}f(v_l)$ beschrieben, wobei d_{ij} die Distanz zwischen den Knoten und f die Kostenfunktion ist. Jeder belieferte Kunde reduziert die Zielfunktion um c_M , wobei c_M so gewählt wird, dass es größer als der gesamte Energieverbrauch einer Lösung ist. Damit lässt sich diese Problemvariante als das MIP folgende formulieren:

$$\text{minimiere} \quad \sum_{u_i, u_j \in V} \sum_{v_l \in \Phi} d_{ij}f(v_l)x_{ij}^l - \sum_{u_i \in V} \sum_{u_j \in C} \sum_{v_l \in \Phi} c_M x_{ij}^l \quad (7.25)$$

$$\text{u.d.N.} \quad \sum_{u_i \in V} \sum_{v_l \in \Phi} x_{ij}^l \leq 1 \quad \forall u_j \in C, \quad (7.26)$$

$$\sum_{u_j \in V} \sum_{v_l \in \Phi} x_{ji}^l - x_{ij}^l = 0 \quad \forall u_i \in C, \quad (7.27)$$

$$\sum_{u_i \in C} \sum_{v_l \in \Phi} x_{0i}^l \leq m, \quad (7.28)$$

$$t_i + \tau_i + \sum_{v_l \in \Phi} x_{ij}^l d_{ij}/v_l \leq t_j + (1 - \sum_{v_l \in \Phi} x_{ij}^l)M_3 \quad \forall u_i, u_j \in C, \quad (7.29)$$

$$\sum_{v_l \in \Phi} x_{0i}^l d_i/v_l \leq t_i + (1 - \sum_{v_l \in \Phi} x_{0i}^l)M_4 \quad \forall u_i \in C, \quad (7.30)$$

$$a_i \leq t_i \leq b_i \quad \forall u_i \in C, \quad (7.31)$$

$$x_{ij}^l \in \{0, 1\} \quad \forall u_i, u_j \in V, v_l \in \Phi. \quad (7.32)$$

Die Zielfunktion (7.25) maximiert als primäres Ziel die Anzahl der belieferten Kunden und minimiert als sekundäres Ziel den Energieverbrauch der Paketdrohnen. Die Nebenbedingungen (7.26) stellen sicher, dass jeder Kunde höchstens einmal besucht wird. Die Nebenbedingungen (7.27)

garantieren die Flusserhaltung. Die Nebenbedingung (7.28) stellt sicher, dass höchstens m Paketdrohnen vom Dummy-Depot aus starten. Die Nebenbedingungen (7.29) garantieren, dass die Belieferungszeitpunkte zweier aufeinanderfolgender Kunden korrekt gesetzt werden. Dabei stellt M_3 eine ausreichend große Zahl dar. Der Belieferungszeitpunkt des ersten Kunden jeder Tour wird durch die Nebenbedingungen (7.30) festgelegt. Die Bedingungen (7.31) und (7.32) legen die Grenzen der Variablen fest. Damit ist die Betrachtung der Problemvarianten abgeschlossen.

7.1.5 Verifikation des Simulationsmodells

Für das Simulationsmodell ist es erforderlich, die Korrektheit des Gesamtsystems sicherzustellen. Dabei wird zwischen den Optimierungsalgorithmen und dem Simulationsmodell unterschieden, das die Ergebnisse der Optimierung umsetzt. Die Korrektheit der Optimierungsalgorithmen wird durch zwei unabhängige Verifikationen sichergestellt. Zum einen dient der Vergleich der Algorithmen aus Kapitel 6 der Verifizierung, zum anderen wurden im Simulationsmodell zusätzliche Routinen implementiert, die die Korrektheit einer Lösung der Optimierung überprüfen. In Kapitel 6 wurde der Branch-and-Price-Algorithmus aus Kapitel 4 mit dem MIP (4.1)-(4.11) verglichen und der Branch-and-Price-Algorithmus aus Kapitel 5 wurde mit Ergebnissen aus der Literatur verglichen. In beiden Fällen wurden die gleichen optimalen Zielfunktionswerte ermittelt. Daher kann davon ausgegangen werden, dass die Optimierungsalgorithmen korrekt sind. In der Simulation wurden in die Zustandsdiagramme der Agenten *Zustellroboter* und *Drohne* zeitliche Überprüfungen integriert, die die Ankunftszeiten an den Mikrodepots und bei den Kunden mit den Ergebnissen der Optimierung vergleichen und etwaige Abweichungen aufzeigen. Dadurch kann neben der korrekten Synchronisierung der Optimierung mit dem Simulationsmodell auch die zeitliche Zulässigkeit einer Lösung gewährleistet werden. Dies gilt ebenfalls für den Ladestand der Zustellroboter.

Zur Sicherstellung der korrekten Funktionsweise des Simulationsmodells wurden alle Modellkomponenten einzeln auf ihre Korrektheit überprüft. Dabei wurde überprüft, ob die Komponenten den festgelegten Anforderungen entsprechen. Zur Identifikation von Fehlern im Modell kam die Methode der *Animation* zum Einsatz. Dabei wird die grafische Darstellung des Modells betrachtet. Aus den Beobachtungen können über kurze Phasen Beurteilungen über die Korrektheit der ablaufenden Prozesse getroffen werden (Rabe et al., 2008, S. 95f.) und (Law, 2014, S.252). Zur Begutachtung des Modells wurde die Animation der Prozesse zweidimensional aufbereitet. Zusätzlich ist es möglich, die Wege der Transportmittel im Modell anzuzeigen. Dies erleichtert die Nachvollziehbarkeit des Prozessdurchlaufs. Zudem werden alle Prozessschritte aufgezeichnet und nach Abschluss der Simulation in einer separaten Datei gespeichert. Dies ermöglicht die *Trace-Analyse*, die die Überprüfung der korrekten Bewegung der Agenten durch das System erlaubt (Rabe et al., 2008, S. 97,104f.).

7.2 Experimente und Ergebnisse

Im Folgenden werden die Annahmen des Simulationsmodells erläutert. Die Paketzustellung mittels Zustellrobotern und Paketdrohnen wird in der Stadt Köln simuliert. Die Verteilung der Sendungen auf die Stadtgebiete basiert auf den Daten des sogenannten Paketradars, das vom Paketdienstleister *Hermes Germany GmbH* veröffentlicht wurde (Bertram, 2018). Die Daten liefern Informationen zur Sendungsmenge auf Stadtteilebene für den Zeitraum von November 2015 bis Oktober 2016. Zusätzlich wurde von der Stadt Köln ein Straßenverzeichnis mit allen Straßen und Hausnummern zur Verfügung gestellt. Damit lässt sich die Verteilung der Sendungen auf die Adressen simulieren. Zur Vereinfachung wird angenommen, dass die Empfänger in einem Zustellgebiet gleichmäßig über die Adressen und Hausnummern verteilt sind. Die Standorte der Mikrodepots entsprechen realen Hermes-Paketshops. Es wird angenommen, dass die Mikrodepots zwischen 6 und 18 Uhr geöffnet sind. Die Zeitfenster der Kunden beginnen frühestens um 9 Uhr und enden spätestens um 15 Uhr. Dies gewährleistet, dass der Weg zum Kunden sowie der Rückweg nicht durch die Öffnungszeiten der Mikrodepots eingeschränkt werden. Der zeitliche Ablauf eines Simulationstags wird in Abbildung 7.4 dargestellt.

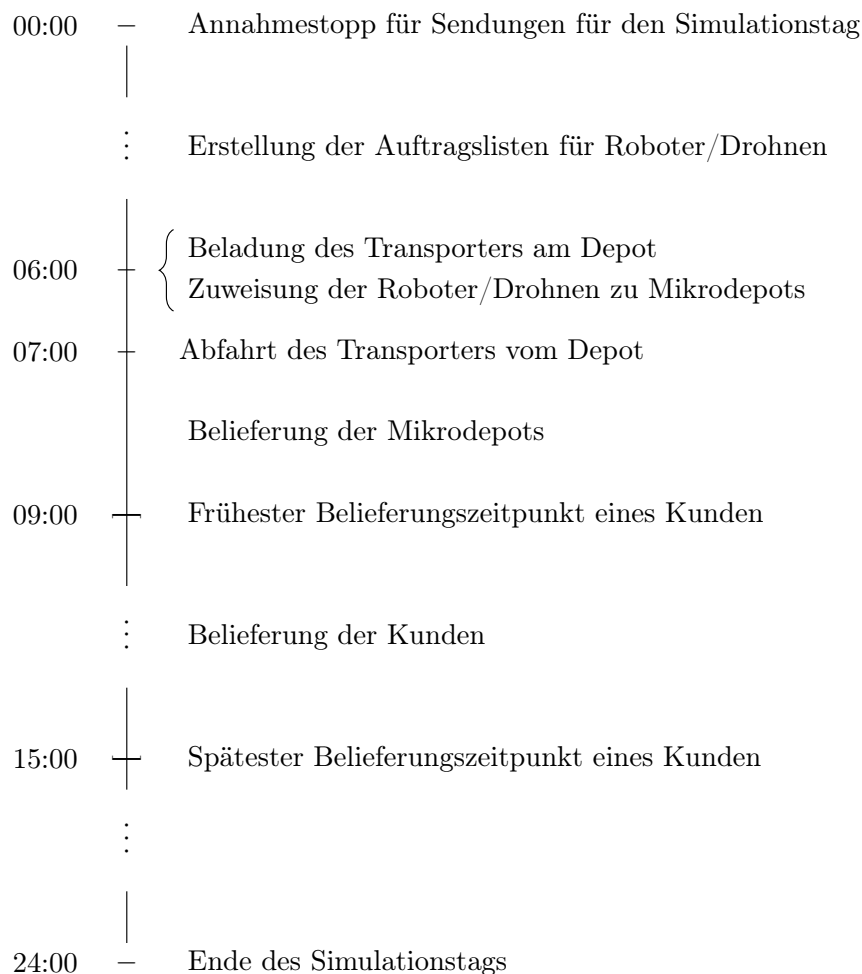


Abbildung 7.4: Zeitlicher Ablauf eines Simulationstages

Um Mitternacht erfolgt der Aufnahmestopp für Sendungen für den neuen Simulationstag. Jede generierte Sendung hat eine Adresse und ein Zeitfenster. Zwischen 0:00 und 6:00 Uhr wird die Lösung für die Auslieferung der Sendungen mit den Robotern und Drohnen berechnet. In diesem Zeitraum werden die Optimierungsprobleme der Unterkapitel 7.1.1 bis 7.1.4 gelöst. Um 6 Uhr wird die Startverteilung gemäß der Optimierung umgesetzt. Dazu bewegen sich die Roboter und Drohnen zu ihren jeweils zugewiesenen Mikrodepots. Zeitgleich beginnt um 06:00 Uhr die Beladung des Transporters im Depot und um 07:00 Uhr erfolgt die Abfahrt des Transporters in das Zustellgebiet. Die Sendungen werden auf die Mikrodepots verteilt. Zur Minimierung der Fahrtstrecke wird für die Tour des Transporters das Handlungsreisenden-Problem gelöst.

Nach erfolgreicher Verteilung der Sendungen auf die Mikrodepots erfolgen die Zustellungen an die Kunden. Es ist anzumerken, dass der früheste und späteste Belieferungszeitpunkt eines Kunden durch den Benutzer der Simulationssoftware eingestellt werden kann. Der Dropfaktor beträgt für beide Transportmittel 1,0.

Nach der Zustellung der jeweiligen letzten Sendung fährt jeder Roboter beziehungsweise jede Drohne zum Mikrodepot, das am nächsten am letzten Kunden liegt. Dort wird der Akku wieder vollständig geladen oder getauscht und der nächste Simulationstag abgewartet. Parallel zu den Zustellungen werden die Sendungen für den folgenden Simulationstag generiert.

Jeder Simulationslauf umfasst 15 aufeinanderfolgende Simulationstage. An jedem Simulationstag wird die gleiche Anzahl an Sendungen erzeugt und versucht zuzustellen. Es ist anzumerken, dass die Anzahl der Sendungen pro Tag vom Nutzer eingestellt werden kann. Die Zeitfenster der Kunden sind gleichmäßig über den Tag verteilt. Alle Zeitfenster haben die gleiche Länge. Auch diese kann vom Nutzer festgelegt werden.

Für alle durchgeführten Experimente werden verschiedene Schlüsselkennzahlen betrachtet, die die Güte der gefundenen Lösung bewerten. Dazu zählen beispielsweise der erreichte Servicegrad, die Auslastung der Zustellroboter und Drohnen, die zurückgelegte Distanz des Transportmittels sowie die verbrauchte Energie. Der Servicegrad berechnet sich als Verhältnis der Summe der erfolgreich durchgeführten Zustellungen zur Gesamtzahl der Sendungen. Die Auslastung wird wie folgt berechnet:

$$\text{Auslastung} = \frac{\text{Summe der aktiven Zeiten der Roboter/Drohnen}}{\#\text{Roboter/Drohnen} \cdot (\text{späteste Rückkehr} - \text{früheste Abfahrt})}. \quad (7.33)$$

Der Zähler beschreibt die Summe der Zeiten, in denen die Transportmittel nicht im Zustand *Warten* oder *Wechsel* sind. Dazu zählen das Fahren, Laden, die Be- und Entladung. Im Nenner wird die Anzahl der Roboter und Drohnen mit der maximal möglichen Beschäftigungszeit multipliziert. Die maximal mögliche Beschäftigungszeit entspricht der Differenz zwischen der spätesten Rückkehr zu einem Mikrodepot und dem frühesten Abfahrtszeitpunkt von einem Mikrodepot. Die Auslastung stellt nur eine untere Schranke dar, da der Zeitpunkt der spätesten Rückkehr und der frühesten Abfahrt von jedem Roboter bzw. jeder Drohne abhängt. Diese Ungenauigkeit gleicht sich jedoch für viele Empfänger aus, da die Zeitfenster gleichmäßig verteilt sind.

7.2.1 Ergebnisse für Zustellroboter

Die Zustellroboter werden auf Basis der Daten des Herstellers Starship Technologies simuliert (Schweizerische Post AG, 2017). Die maximale Geschwindigkeit des Roboters beträgt 6 km/h. Es wird eine durchschnittliche Geschwindigkeit von 3,6 km/h (1 m/s) angenommen. Die Kapazität des Akkus beträgt 8.000 mAh und die Leistung beträgt 148 Wh. Es wird davon ausgegangen, dass 60 % der angegebenen Kapazität genutzt werden können. Dies bedeutet, dass die unteren und oberen 20% als Reserve verbleiben. Diese Maßnahme dient einerseits der Langlebigkeit des Akkus und andererseits als Reserve für unerwartete Situationen. Der Energieverbrauch beim Fahren beträgt 4.800 mA. Daraus ergibt sich eine maximale Reichweite des Roboters von 3,6 km. Der Energieverbrauch während der Be- und Entladung des Roboters sowie sonstige Verbräuche werden vernachlässigt. Die Beladung am Mikrodepot wird auf 3 Minuten geschätzt, während für die Entnahme der Sendung beim Empfänger 7 Minuten veranschlagt werden. Laut Starship Technologies beträgt die Ladezeit für eine vollständige Ladung des Akkus 45 Minuten. Für die reduzierte Kapazität ergibt sich daraus eine maximale Ladezeit von etwa 27 Minuten. Somit sind alle relevanten Daten zur Abbildung der Zustellroboter eingeführt. Im nächsten Abschnitt werden das Zustellgebiet sowie die erzielten Ergebnisse beschrieben.

Vergleich feste und flexible Zuweisung zu den Mikrodepots

In diesem Experiment wird der Einfluss einer flexiblen Zuweisung der Roboter zu den Mikrodepots untersucht. Die Ergebnisse der beschriebenen Problemstellung werden mit denen der Problemvariante mit fester Zuweisung verglichen. Eine detaillierte Erläuterung dieser Problemvariante ist in Unterkapitel 7.1.2 zu finden. Die Untersuchung umfasst zwei Szenarien: eines mit zwei Mikrodepots und vier Zustellrobotern sowie eines mit fünf Mikrodepots und fünf Zustellrobotern. In beiden Szenarien liegen die Zeitfenster im Zeitraum zwischen 9 Uhr und 15 Uhr mit einer einheitlichen Länge von 30 Minuten.



Abbildung 7.5: Zustellgebiet im Stadtteil Altstadt Süd mit 2 Mikrodepots

Für das erste Szenario wird als Zustellgebiet die Kölner Altstadt Süd (siehe Abbildung 7.5) betrachtet. Dieser Stadtteil hat eine Fläche von $2,36 \text{ km}^2$. Die Mikrodepots sind gleichmäßig über das Zustellgebiet verteilt. Die Anzahl der Kunden variiert in diesem Experiment zwischen 20 und 48, wobei eine Schrittweite von zwei Kunden gewählt wurde. Für jede feste Anzahl von Kunden werden 20 Simulationsläufe durchgeführt. Damit ergibt sich eine Gesamtzahl von 600 Simulationsläufen für das Experiment. Die Anzahl der Roboter ist auf vier festgelegt. Die Ergebnisse des Experiments sind in Abbildung 7.6 zusammengefasst. Die Abbildung beschreibt die Entwicklung des Servicegrads in Abhängigkeit von der Kundenanzahl. Der Servicegrad ergibt sich dabei als Durchschnitt der 20 Läufe.

Der Graph zeigt, dass eine Erhöhung der Kundenanzahl, unabhängig von der Variante, zu einer Verringerung des Servicegrads führt. Zusätzlich ist ersichtlich, dass die Aufhebung der Zuweisung zu den Mikrodepots zu einer signifikanten Steigerung des Servicegrads führt. Dies deutet darauf hin, dass die Verteilung der Aufträge auf die Zustellroboter ohne feste Zuweisung effizienter ist. Im Durchschnitt steigt der Servicegrad um 3,3 %. Auch die Auslastung der Roboter ist ohne feste Zuweisung zu den Mikrodepots höher. Beispielsweise liegt für 48 Kunden die Auslastung bei durchschnittlich 96 %, während sie im Fall einer festen Zuweisung bei 90,4 % liegt. Im Durchschnitt steigt die Auslastung durch die Aufhebung der festen Zuweisung über alle Kundenanzahlen hinweg um 8,9 %. Dies zeigt, dass die Roboter ohne feste Zuweisung zu den Mikrodepots besser ausgelastet werden können.

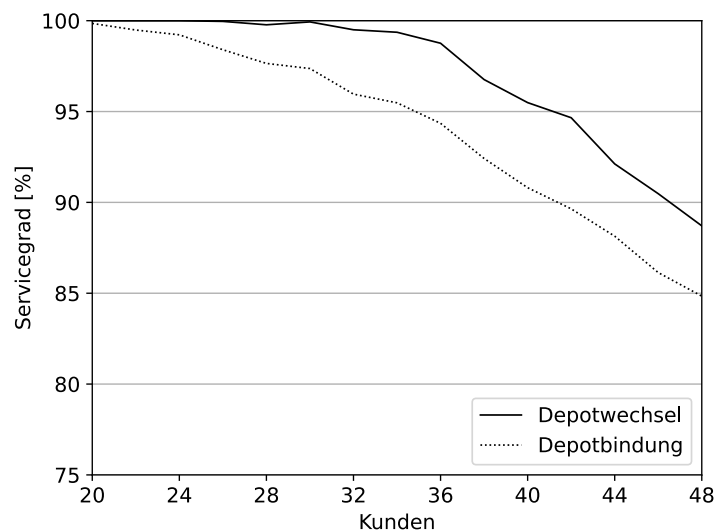


Abbildung 7.6: Servicegrade für vier Zustellroboter mit und ohne fester Zuweisung zu den Mikrodepots

Im Fall ohne feste Zuweisung wurden 50,3 % der Sendungen vom südlicheren Mikrodepot und 49,7 % vom nördlichen Mikrodepot geliefert. Nach der Belieferung eines Kunden wird durchschnittlich in 15,66 % der Fälle ein anderes Mikrodepot angesteuert. Zusätzlich werden in 0,06 % der Depot-Pfade beide Depots zwischen zwei aufeinanderfolgenden Kunden angefahren. Im Fall mit fester Zuweisung zu den Mikrodepots liegt die Verteilung der Sendungen auf die Mikrodepots bei 50,1 % und 49,9 %.

Für das zweite Experiment wird die gesamte Altstadt von Köln (Altstadt Süd und Nord) betrachtet, die eine Gesamtfläche von $4,82 \text{ km}^2$ umfasst. In diesem Experiment werden fünf Mikrodepots gleichmäßig auf das Zustellgebiet verteilt. Das Zustellgebiet sowie die Verteilung der Mikrodepots sind in Abbildung 7.7 dargestellt.

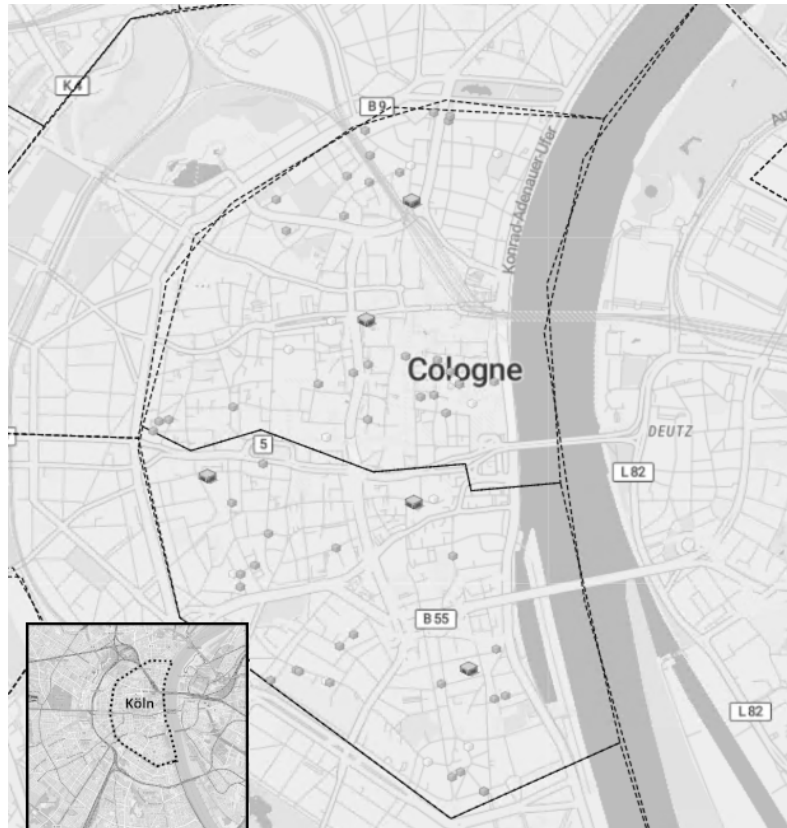


Abbildung 7.7: Zustellgebiet in den Stadtteilen Altstadt Nord und Süd mit 5 Mikrodepots

Die Kundenanzahl variiert zwischen 30 und 54 mit einer Schrittweite von zwei Kunden, mit 20 Simulationsläufen pro Kundenanzahl. Insgesamt ergeben sich 520 Läufe bei konstant fünf Robotern. Abbildung 7.8 zeigt den durchschnittlichen Servicegrad beider Varianten in gewohnter Darstellungsweise.

Mit zunehmender Kundenanzahl sinkt der Servicegrad. Durch die Aufhebung der festen Zuweisung zu den Mikrodepots steigt der Servicegrad im Schnitt um 14 % und die Roboterauslastung um 33,5 %. Auch in diesem Experiment verbessert sich somit beides deutlich.

Ohne feste Zuweisung ist die Sendungsverteilung auf die Mikrodepots nahezu gleichmäßig (19–22 %), bei durchschnittlich 36,1 % Depotwechseln nach einer Zustellung. Zusätzlich gilt, dass 0,24 % aller Depot-Pfade mehr als ein Depot umfassen. Auch mit fester Zuweisung ist die Verteilung ausgeglichen (19–21,3 %).

Insgesamt lässt sich anhand der beiden Experimente zeigen, dass die Aufhebung der festen Zuweisung zu den Mikrodepots den Servicegrad deutlich steigert. Zusätzlich lässt sich die Auslastung der Zustellroboter für eine feste Anzahl an Kunden deutlich verbessern. Im nächsten Experiment wird der Vorteil einer optimierten Ladestrategie näher untersucht.

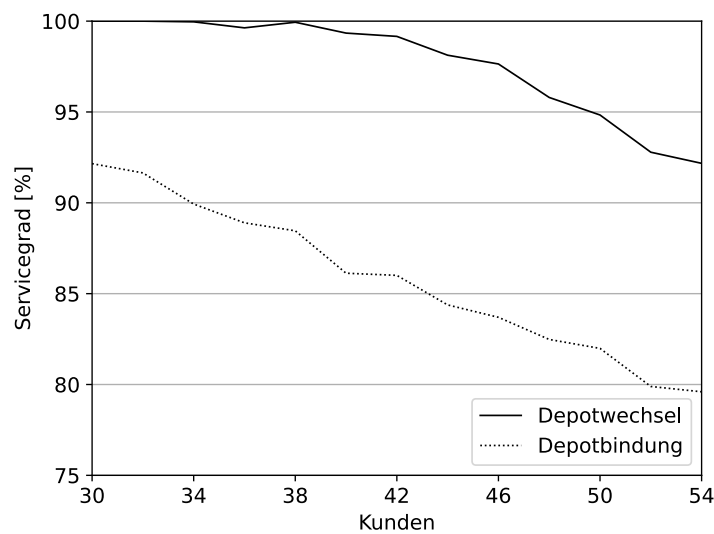


Abbildung 7.8: Gegenüberstellung des Servicegrads mit fünf Zustellrobotern und 30 bis 54 Kunden mit und ohne fester Zuweisung zu den Mikrodepots

Ergebnisse der Optimierung der Ladezeit

Im folgenden Experiment werden die optimalen Ladezeiten mit der Variante mit festgelegten Nachladegrenzen verglichen. In dieser Strategie ist es nur möglich, einen Zustellroboter bei Unterschreitung eines vordefinierten Ladestands zu laden. Dabei dürfen ausschließlich vollständige Ladevorgänge durchgeführt werden. Im vorangegangenen Abschnitt wurde gezeigt, dass die Aufhebung der festen Zuweisung zu den Mikrodepots einen großen Einfluss auf den Servicegrad hat. Um die verschiedenen Strategien besser vergleichen zu können, wird die Anzahl der Mikrodepots auf eins begrenzt. Hierdurch soll sichergestellt werden, dass Nebeneffekte, die durch die Wahl des Mikrodepots entstehen, eliminiert werden. Als Zustellgebiet wird der Kölner Stadtteil Bayenthal gewählt, der im Süden von Köln liegt und eine nahezu rechteckige Form aufweist (siehe Abbildung 7.9). Das Zustellgebiet umfasst rund $1,28 \text{ km}^2$. Das gewählte Mikrodepot liegt im Zentrum des Stadtteils.

Das Experiment wird mit drei Robotern durchgeführt, wobei die Anzahl der Sendungen zwischen 10 und 32 variiert und eine Schrittweite von zwei Kunden verwendet wird. Die Zeitfenster der Kunden liegen im Zeitraum zwischen 9 und 15 Uhr und die Länge der Zeitfenster beträgt 20 Minuten. Die optimale Ladestrategie wird mit der optimalen Lösung des Modells (7.5)-(7.20) verglichen. Dabei wurden drei verschiedene Nachladegrenzen getestet, nämlich 25 %, 50 % und 75 %. Dies bedeutet beispielsweise, dass der Roboter den Akku am Mikrodepot ab einem Ladestand von 50 % wieder vollständig aufladen muss. Allerdings darf das Nachladen auch erst ab 50 % erfolgen. Die maximale Ladezeit beträgt 27 Minuten. Dementsprechend liegen die Ladezeiten bei den jeweiligen Nachladegrenzen bei 20,25 Minuten, 13,5 Minuten und 6,75 Minuten. Für jede Kombination aus Kundenanzahl und Variante werden 20 Simulationsläufe durchgeführt. In Summe werden für das Experiment 960 Läufe durchgeführt. Damit ist die Vorstellung der Parameter abgeschlossen und es folgt die Diskussion der erzielten Ergebnisse.



Abbildung 7.9: Zustellgebiet im Stadtteil Bayenthal mit einem Mikrodepot

Zunächst werden die erreichten Servicegrade für die verschiedenen Varianten und Kundenanzahlen verglichen. Die Ergebnisse der Experimente sind in Abbildung 7.10 zusammenfassend dargestellt. Die Abbildung ist analog zu den vorherigen aufgebaut. Der Servicegrad für die optimale Berechnung der Ladezeiten ist dabei durch eine durchgezogene Linie gekennzeichnet. Die Servicegrade für die Ergebnisse mit Nachladegrenzen sind durch gestrichelte, gepunktete und Punkt-Strich-Linien gekennzeichnet.

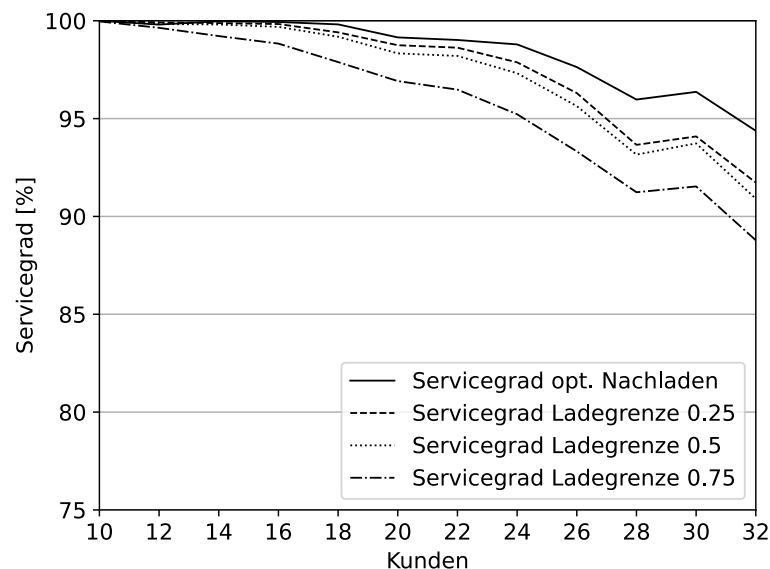


Abbildung 7.10: Servicegrade für verschiedene Kundenanzahlen und Ladestrategien

Der Graph zeigt, dass die Variante mit optimierten Ladezeiten alle drei Nachladegrenzen übertrifft. Für die festen Nachladegrenzen gilt, dass eine Grenze von 25 % zu den besten Ergebnissen führt. Im Durchschnitt steigt der Servicegrad durch die Optimierung der Ladezeiten um 1,7 %.

Dabei fällt auf, dass die Steigerung des Servicegrads mit steigender Kundenanzahl auch steigt. Beispielsweise liegt sie für 32 Kunden bei 4,2 %. Allerdings zeigt sich, dass die Steigerung dieser Kennzahlen nicht so ausgeprägt ist wie bei der Aufhebung der festen Zuweisung zu den Mikrodepots.

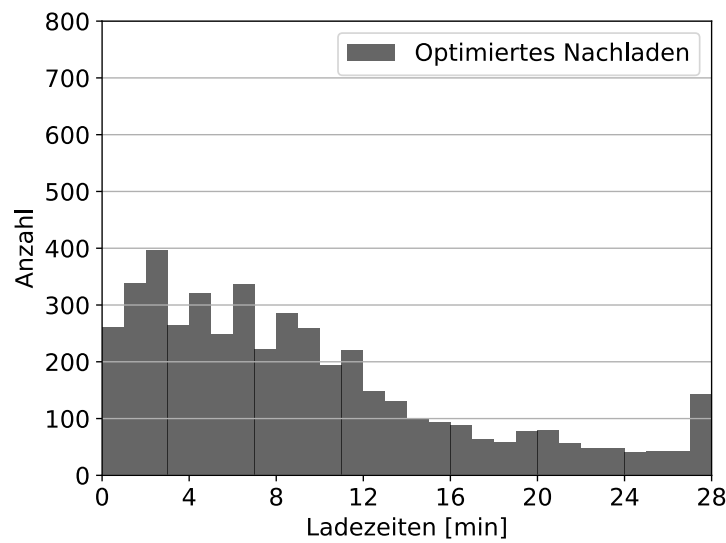
Im Folgenden werden für die größte Kundenanzahl die Ladezeiten detaillierter untersucht. Die Häufigkeiten der verschiedenen Ladezeiten sind in Histogrammen dargestellt. Histogramm 7.11(a) zeigt die Häufigkeiten der optimierten Ladezeiten. Im Vergleich dazu zeigt Histogramm 7.11(b) die Verteilung der Ladezeiten für die drei Nachladegrenzen. Die einzelnen Klassen haben eine Breite von einer Minute, und die Anzahl der Elemente pro Klasse wird angegeben. Den Histogrammen liegen für jede Variante bzw. Nachladegrenze 20 Simulationsläufe mit 15 Simulationstagen zugrunde. Die Ladevorgänge am Ende des Simulationstages wurden in den Histogrammen nicht berücksichtigt, da sie vom Modell restriktiv vorgegeben sind und nicht direkt durch die Optimierung beeinflusst werden können. Es ist zu beachten, dass zu Beginn des Tages kein Wechsel des Mikrodepots stattfinden kann, weshalb diese potentiellen Ladezeiten entfallen.

Zunächst wird das Histogramm 7.11(a) beschrieben. Die Stichprobengröße umfasst 4.607 Datenpunkte. Ein Roboter lädt durchschnittlich 5,12-mal täglich den Akku am Mikrodepot, bzw. 6,12-mal, wenn man die Ladung am Ende des Tages hinzuzieht. Die Daten sind asymmetrisch und weisen eine rechtsschiefe Verteilung auf. Die durchschnittliche Ladezeit beträgt etwa 9 Minuten. Ein Ausreißer mit 143 Datenpunkten tritt bei einer Ladedauer von 27 Minuten auf. Dies sind vollständige Ladevorgänge. Der Grund hierfür liegt darin, dass die Lösung auf Algorithmus 2 basiert. Dabei wird stets die minimale Ladezeit und -menge verwendet. Außer es liegen Wartezeiten vor, dann werden diese zum Laden genutzt. Dadurch entsteht diese Häufung. In 60,8 % der Fälle wird nach der Zustellung bei einem Kunden ein Ladevorgang durchgeführt.

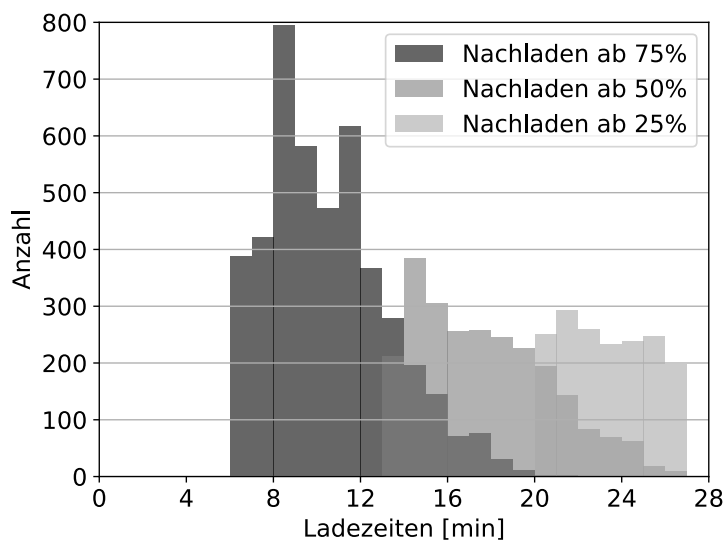
Bei der Betrachtung des Histogramms 7.11(b) ist es offensichtlich, dass die drei Nachladegrenzen eingehalten werden. Die Stichprobengröße betragen 1.724 (25 %), 2.470 (50 %) und 4.463 (75 %). Somit laden die Roboter bei einer Ladegrenze von 25 % durchschnittlich 1,92-mal täglich und 4,96-mal bei einer Nachladegrenze von 75 %. Im Durchschnitt wird bei der Rückkehr zum Mikrodepot in 29,8 % (25 %), 38,6 % (50 %) bzw. 62,9 % (75 %) der Fälle der Akku geladen.

Im Folgenden wird der Verlauf des Ladestandes über einen Simulationstag für die optimale Ladestrategie betrachtet. Hierzu wurde eine Simulation mit 20 Kunden und einem Roboter durchgeführt. Der Verlauf des Ladezustands ist in Abbildung 7.12 dargestellt. Auf der x-Achse ist die Uhrzeit und auf der y-Achse der Ladezustand in mAh abgebildet. Die verschiedenen Aktivitäten des Roboters werden durch unterschiedliche Linienarten visualisiert. Die Be- und Entladung des Roboters wird durch eine durchgezogene Linie, das Fahren durch eine gepunktete Linie, das Laden durch eine Strich-Punkt-Linie und das Warten durch eine gestrichelte Linie visualisiert. Die Reserven des Akkus sind durch graue Bereiche gekennzeichnet.

Die Abbildung zeigt, dass der Ladestand während der Fahrt linear abnimmt und während der Ladevorgänge linear zunimmt. Im Untersuchungszeitraum lädt der Zustellroboter den Akku fünfmal am Mikrodepot und beliefert insgesamt elf Kunden. Vor der ersten Auslieferung wartet der



(a) Ladezeiten für 32 Kunden, drei Zustellroboter und optimales Nachladen



(b) Ladezeiten für 32 Kunden, drei Zustellroboter und drei Nachladegrenzen

Abbildung 7.11: Gegenüberstellung der Ladezeiten für verschiedene Ladestrategien

Roboter im Mikrodepot. Weitere Wartezeiten treten um 10 Uhr, zwischen 13 und 14 Uhr sowie nach 15 Uhr auf. Es ist ersichtlich, dass diese Zeiträume zum Laden des Akkus genutzt werden. Damit ist die Analyse des Mehrwerts der Optimierung abgeschlossen. Im nächsten Abschnitt werden die Simulationsergebnisse detaillierter betrachtet.

Paketzustellung in der Innenstadt von Köln mit Zustellrobotern

In diesem Abschnitt wird die Paketzustellung durch Zustellroboter in der Innenstadt von Köln eingehender analysiert. Das Ziel dieser Untersuchung besteht darin, die Simulation zu verwenden, um Erkenntnisse über die letzte Meile der Paketzustellung mittels Zustellrobotern zu gewinnen. Dazu wird zunächst das Basisszenario vorgestellt. Im Anschluss erfolgt eine Variation der Para-

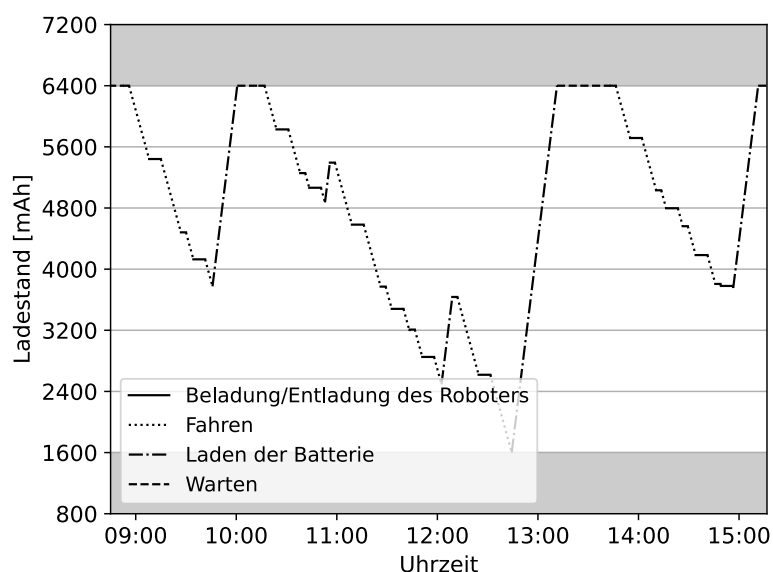


Abbildung 7.12: Verlauf des Ladestandes eines Zustellroboters zwischen 9 und 15 Uhr

meter des Basisszenarios. Es wird untersucht, welchen Einfluss die Anpassung der Zeitfensterlängen und die Anzahl der Mikrodepots haben. Für die verschiedenen Parameter werden jeweils 100 Simulationsläufe durchgeführt. Die Zeitfenster der Kunden liegen in einem Zeitraum zwischen 9 und 15 Uhr. Als Zustellgebiet wird die komplette Innenstadt der Stadt Köln betrachtet.

Das Basisszenario entspricht dem Szenario des zweiten Experiments, jedoch ohne feste Zuweisung zu den Mikrodepots. Es kommen fünf Zustellroboter und fünf Mikrodepots zum Einsatz. Abbildung 7.13 zeigt die Verteilung der fünf Mikrodepots in der Innenstadt sowie die Aufteilung der Sendungen auf diese Mikrodepots. Die Zeitfensterlänge der Kunden beträgt eine halbe Stunde. Die Anzahl der Sendungen wird so gewählt, dass ein durchschnittlicher Servicegrad von mehr als 95 % erreicht wird. Zur Berechnung der maximalen Anzahl an Sendungen werden für jede Anzahl 100 Simulationsläufe durchgeführt. Aus den Berechnungen resultiert eine Anzahl von 50 Sendungen mit einem durchschnittlichen Servicegrad von 95,1 %.

Im Durchschnitt sind die Sendungen nahezu gleichmäßig auf die fünf Mikrodepots verteilt (siehe Abbildung 7.13). Ein Zustellroboter beliefert 9,5 Kunden pro Tag und legt dabei 11,3 km zurück. Für die Belieferung eines Kunden werden im Durchschnitt 1,2 km zurückgelegt. In 30,1 % der Fälle wechseln die Roboter nach der Belieferung eines Kunden das Mikrodepot. In 0,12 % aller Depot-Pfade werden zwei Mikrodepots zwischen zwei Kunden angefahren. Die Dauer einer Tour eines Roboters, das heißt die Beladung am Mikrodepot, die Fahrt zum Kunden, die Entnahme der Sendung, die Rückfahrt zum Mikrodepot und die Aufladung des Akkus, beträgt im Durchschnitt etwa 43 Minuten. Ohne Berücksichtigung der Wiederaufladung ergibt sich eine durchschnittliche Belieferungszeit von etwa einer halben Stunde je Kunde. Der Stromverbrauch für die Belieferung eines Kunden beträgt 29,2 Wh. In den 15 Tagen benötigen die fünf Zustellroboter durchschnittlich 20,8 kWh. Die Auslastung der Zustellroboter beträgt 92,1 %.

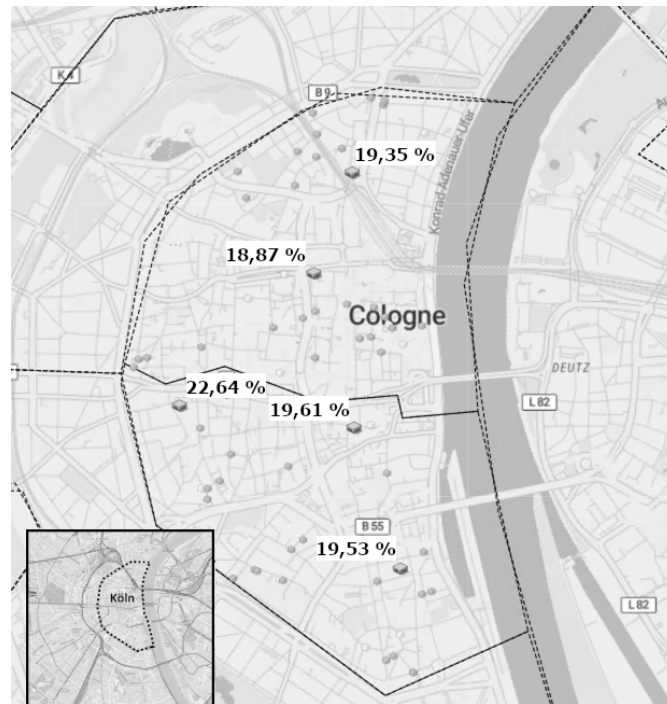


Abbildung 7.13: Verteilung auf fünf Mikrodepots für die Altstadt Nord und Altstadt Süd

Im nächsten Schritt wird das Basisszenario mit den anderen Szenarien verglichen. Zunächst werden die Ergebnisse der Szenarien mit kürzeren und längeren Zeitfenstern präsentiert. Die Längen der Zeitfenster betragen hierbei 15 und 45 Minuten. Im zweiten Teil wird überprüft, welche Auswirkung eine Reduzierung der Mikrodepots hat. Dazu wird die Anzahl der Mikrodepots auf drei gesetzt.

Für beide Zeitfensterlängen gilt, dass sich die Verteilung der Sendungen auf die Mikrodepots im Vergleich zum Basisszenario nicht wesentlich verändert. Bei einer Zeitfensterlänge von 15 Minuten sinkt der erreichte Servicegrad jedoch um 6,4 % auf 89 %. Damit sinkt auch die Anzahl der belieferten Kunden auf durchschnittlich 8,9 Kunden je Roboter und Simulationstag. Betrachtet man die Entfernung, die für die Belieferung eines einzelnen Kunden zurückgelegt wird, stellt sich ein Anstieg von 5,5 % auf 1,3 km heraus. Dies spiegelt sich auch in der Häufigkeit der Depotwechsel nach der Belieferung eines Kunden wider, die bei 57,6 % liegt. Damit steigt auch der Energiebedarf je beliefertem Kunden auf 29,2 Wh. Die Auslastung der Roboter sinkt leicht auf 89,4 %.

Im Folgenden werden die Ergebnisse für eine Zeitfensterlänge von 45 Minuten beschrieben. Der Servicegrad erhöht sich nur marginal auf 97 %. Damit bleibt die Anzahl der durchschnittlich belieferten Kunden je Roboter nahezu unverändert bei 9,7. Auch die durchschnittliche Distanz, die ein Roboter für die Belieferung eines Kunden zurücklegt, sinkt lediglich marginal auf 1,2 km. Auch die anderen Kennzahlen verändern sich durch die Erhöhung des Zeitfensters nur geringfügig.

Dies zeigt, dass kürzere Zeitfensterlängen im Vergleich zum Basisszenario einen erheblichen Einfluss auf die wesentlichen Kennzahlen haben, sodass von 15-minütigen Zeitfenstern abgeraten werden kann. Eine Erhöhung der Zeitfenster auf 45 Minuten steigert im Vergleich die Kennzah-

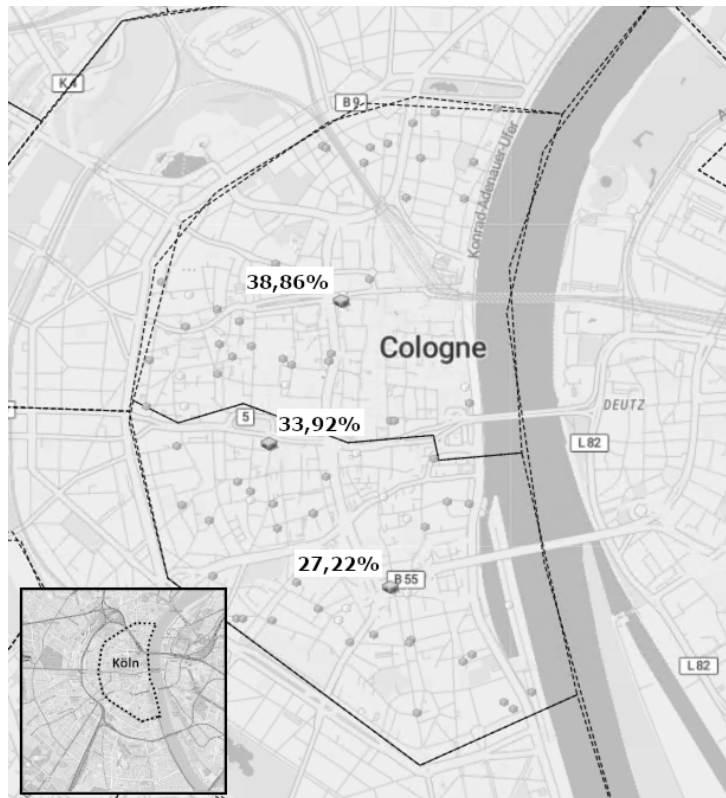


Abbildung 7.14: Verteilung auf drei Mikrodepots für die Altstadt Nord und Altstadt Süd

len nicht wesentlich, weshalb auch von dieser Anpassung abgeraten wird.

Abgeschlossen wird die Vorstellung mit der Auswertung des Szenarios mit drei Mikrodepots. Die Verteilung der drei Mikrodepots ist in Abbildung 7.14 dargestellt. Es zeigt sich eine leicht ungleichmäßigere Sendungsverteilung (Abweichung 5 %). Im Vergleich zum Basisszenario sinkt der Servicegrad deutlich auf 87,7 %. Die durchschnittlich belieferten Kunden pro Roboter sinken auf 8,77. Wie im Szenario mit kurzen Zeitfenstern steigt die Zustelldistanz um 14 % auf 1,4 km. Die Depotwechselrate fällt auf 16,9 %, der Energieverbrauch pro Kunde steigt auf 33,4 Wh. Insgesamt werden in den 15 Tagen 22 kWh Strom verbraucht. Die Reduktion auf drei Mikrodepots wirkt sich somit klar negativ aus.

Eine übersichtliche Zusammenfassung der erzielten Ergebnisse ist in den Abbildungen 7.15 bis 7.18 dargestellt. Jede Abbildung zeigt vier Box-Plots, die die entsprechenden Kennzahlen für das Basisszenario, das Szenario mit 15-minütigen Zeitfenstern, das Szenario mit 45-minütigen Zeitfenstern und das Szenario mit drei Mikrodepots darstellen. Die betrachteten Kennzahlen umfassen die durchschnittliche Distanz für die Belieferung eines Kunden, die gesamte Distanz pro Simulationslauf, den Energiebedarf für die Belieferung eines Kunden sowie den gesamten Energieverbrauch der Zustellroboter pro Simulationslauf. Damit ist die Vorstellung der Ergebnisse für die Paketzustellung mit autonomen Robotern abgeschlossen. Im Folgenden werden die Ergebnisse der Simulation der Zustellung mit Paketdrohnen präsentiert.

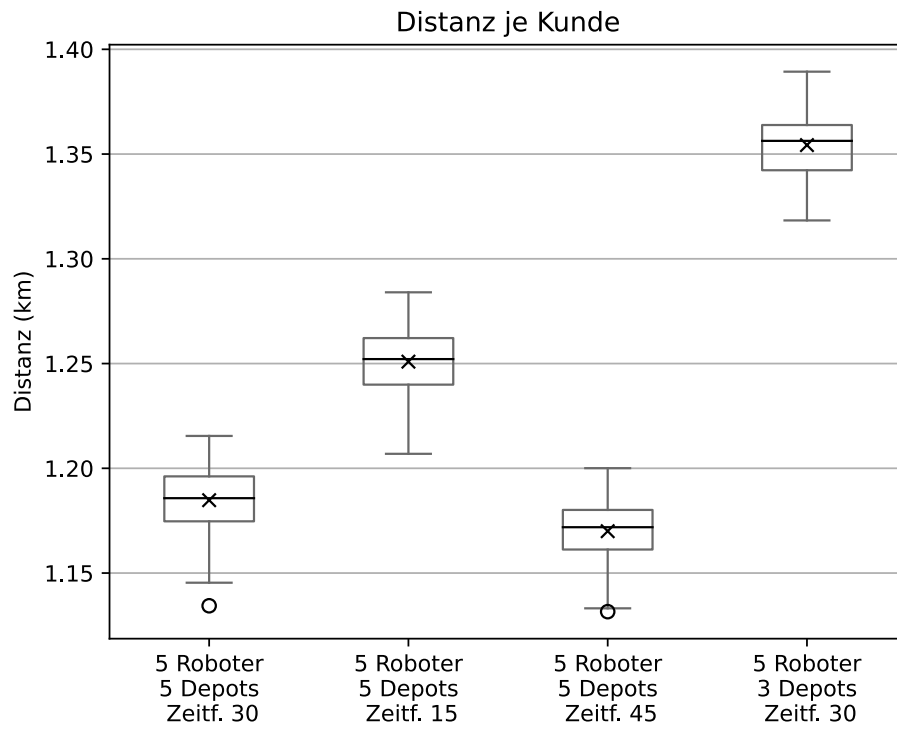


Abbildung 7.15: Verteilung der Distanz je Belieferung

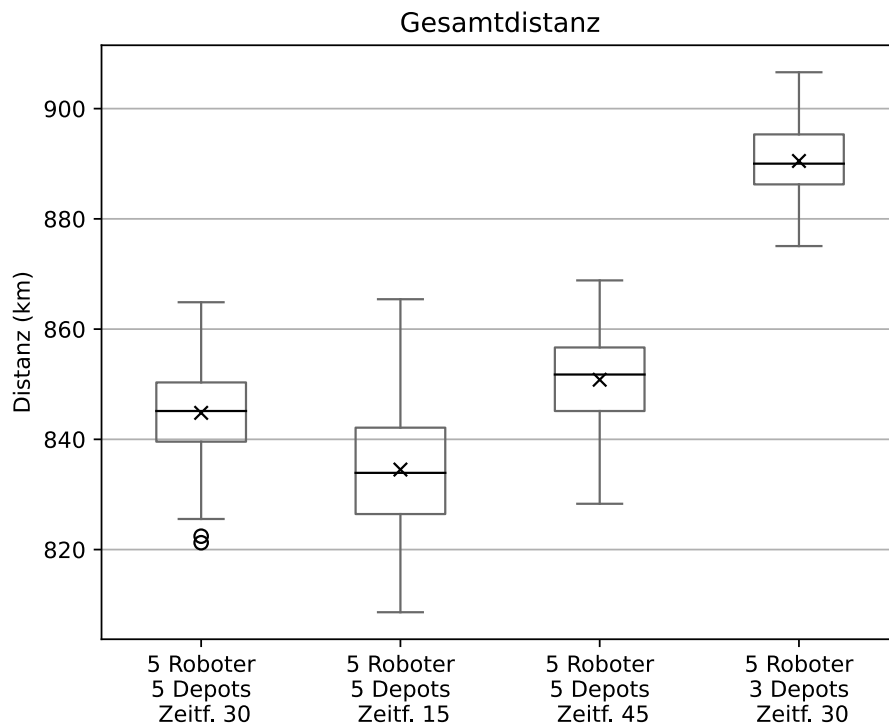


Abbildung 7.16: Verteilung der Gesamtdistanz für 5 Roboter über 15 Tage

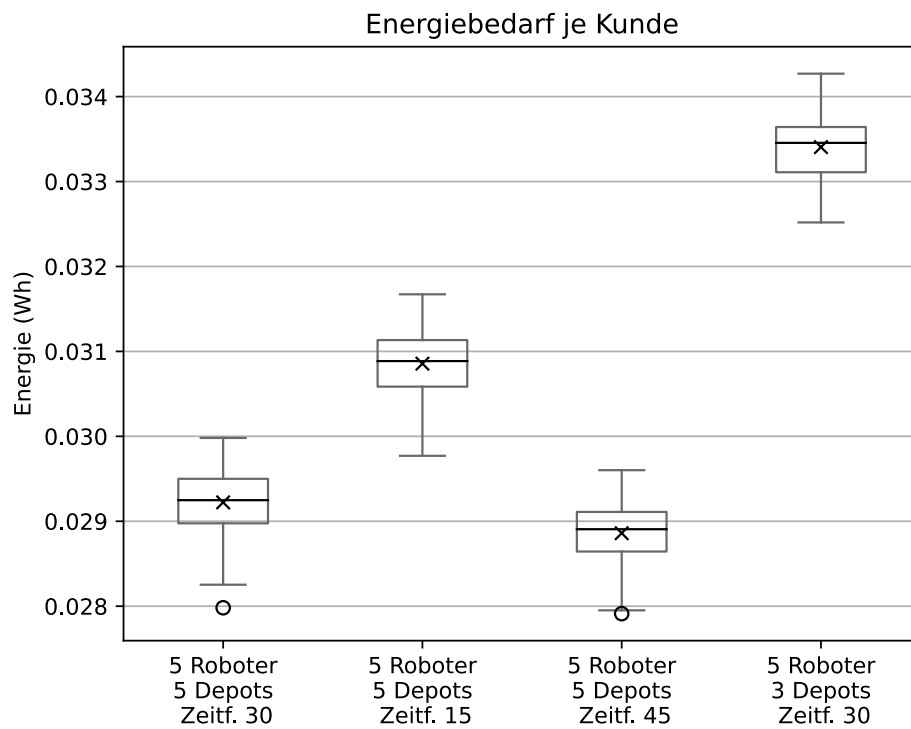


Abbildung 7.17: Verteilung des Energieverbrauchs je Belieferung

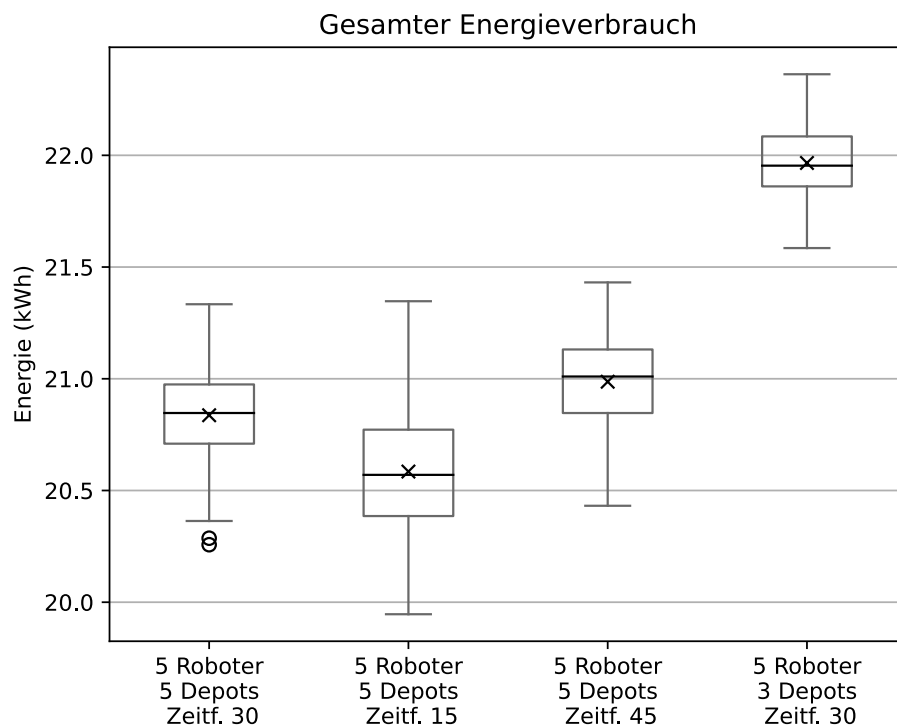


Abbildung 7.18: Verteilung des Gesamtenergieverbrauchs für fünf Roboter über 15 Tage

7.2.2 Ergebnisse für Paketdrohnen

In diesem Abschnitt werden die Ergebnisse der Simulation zur Zustellung mit Paketdrohnen präsentiert. Zunächst werden die getroffenen Annahmen und Parameter beschrieben. Die Bewertung der verschiedenen Ansätze erfolgt unter anderem anhand des gemessenen Energieverbrauchs. Hierfür wird das Modell verwendet, das in Unterkapitel 5.1 beschrieben ist. In diesem Modell wird eine Funktion definiert, die den Energieverbrauch der Drohnen in Abhängigkeit von Gewicht, Geschwindigkeit und Distanz beschreibt. Es wird angenommen, dass die Summe der Energieverbräuche für das Aufsteigen und Absteigen der Paketdrohne dem Energieverbrauch im Schwebезustand entspricht (Liu et al., 2017). Der Energieverbrauch in Abhängigkeit von der Flugzeit wird mit der Funktion gemessen, die in Unterkapitel 5.1 beschrieben ist. Alle für die Aufstellung der Funktion erforderlichen Daten sind in Tabelle 5.1 angegeben. Da die Algorithmen zur Geschwindigkeitsoptimierung aus Kapitel 5 von einer konstanten Kostenfunktion über alle Kanten ausgehen, wird angenommen, dass die Paketdrohnen sowohl auf dem Flug zum Kunden als auch auf dem Rückflug das gleiche Gewicht tragen. Es wird angenommen, dass die Paketdrohnen stets ein zusätzliches Gewicht von 0,5 kg transportieren. Das Gesamtgewicht der Paketdrohne, einschließlich Akku und Sendung, beträgt 3,43 kg. Die Kapazität der Akkus der Drohnen beträgt 10.000 mAh, mit einer Leistung von 150 Wh (Dukkanci et al., 2021). Wie bereits bei den Zustellrobotern wird angenommen, dass nur 60 % der Akkukapazität genutzt werden. Dies bedeutet, dass die unteren und oberen 20% als Reserve verbleiben. Dies verlängert einerseits die Lebensdauer des Akkus und ermöglicht andererseits eine Reaktion auf unerwartete Situationen. Damit ergibt sich eine verfügbare Energie von 90 Wh. Es wird angenommen, dass die Paketdrohnen in einer Höhe von 50 Metern fliegen. Der Aufstieg dauert 6,41 Sekunden und die Landung dauert 12,82 Sekunden (Murray und Chu, 2015). Außerdem werden folgende Annahmen getroffen:

- Die Beladung der Drohne mit einer Sendung am Depot/Mikrodepot benötigt 3 Minuten.
- Die Schwebezeit je Belieferung eines Kunden benötigt 5 Minuten.
- Die Abgabe der Sendung beim Kunden benötigt 1 Minute.
- Der Wechsel des Akkus benötigt 3 Minuten.

Kontinuierliche und diskrete Geschwindigkeiten

In diesem Abschnitt werden die Ergebnisse der exakten Geschwindigkeitsoptimierung mit den Ergebnissen verglichen, die sich aus der Diskretisierung der möglichen Fluggeschwindigkeiten ergeben. Um eine Lösung für eine Instanz mit diskreten Geschwindigkeiten zu erhalten, wird das MIP (7.25)-(7.32) verwendet. Im Folgenden wird das Szenario beschrieben.

Es werden 20 Simulationsläufe je Variante durchgeführt. Jeder Simulationslauf umfasst 15 Simulationstage. Die Zeitfenster der Kunden sind dabei gleichmäßig über den Zeitraum von 9 bis 15 Uhr verteilt und haben eine Länge von 30 Minuten. Als Zustellgebiet wird, wie auch schon in den vorherigen Experimenten, die Altstadt Nord und Süd der Stadt Köln betrachtet. Die



Abbildung 7.19: Depot und Zustellgebiet in der Altstadt Nord und Süd der Stadt Köln

Drohnen starten und beenden jede Belieferung von einem Depot, das am Randbereich von Köln liegt. Die Entfernung zu den Kunden im Zustellgebiet liegt zwischen 7 und 10 km. Eine Darstellung des Zustellgebiets einschließlich des Depots ist in Abbildung 7.19 gegeben. Das Depot liegt am unteren Rand der Abbildung. Für das Szenario werden zwei Paketdrohnen eingesetzt, die mit einer Geschwindigkeit zwischen 50 und 108 km/h fliegen. Es ist anzumerken, dass bei einer Geschwindigkeit von 72,4 km/h (optimale Geschwindigkeit) die Reichweite maximiert und der Energieverbrauch minimiert wird. Daher bewegen sich die Drohnen nicht langsamer.

Für den Vergleich werden verschiedene Anzahlen an verfügbaren Geschwindigkeiten im diskreten Fall betrachtet. Es werden insgesamt vier Anzahlen betrachtet, die zwischen zwei und fünf variieren. Die Geschwindigkeiten sind gleichmäßig im Intervall der zulässigen Geschwindigkeiten verteilt. Dies bedeutet, dass bei drei verfügbaren Geschwindigkeiten beispielsweise 72,4 km/h, 90,2 km/h und 108 km/h berücksichtigt werden. Es ist zu beachten, dass der Servicegrad unabhängig von der Anzahl der verfügbaren Geschwindigkeiten ist, da es für alle Diskretisierungen möglich ist, mit der höchsten Geschwindigkeit zu fliegen und das übergeordnete Ziel die Maximierung der belieferten Kunden darstellt. Damit sind die Parameter vorgestellt und es folgt die Präsentation der Ergebnisse.

Für das hier betrachtete Szenario wird die Anzahl der Kunden so gewählt, dass der Servicegrad über 95 % liegt. Damit ergibt sich ein durchschnittlicher Servicegrad von 95,8 %. An einem Simulationstag fliegt eine Drohne 174 km und beliefert etwa 10,5 Kunden.

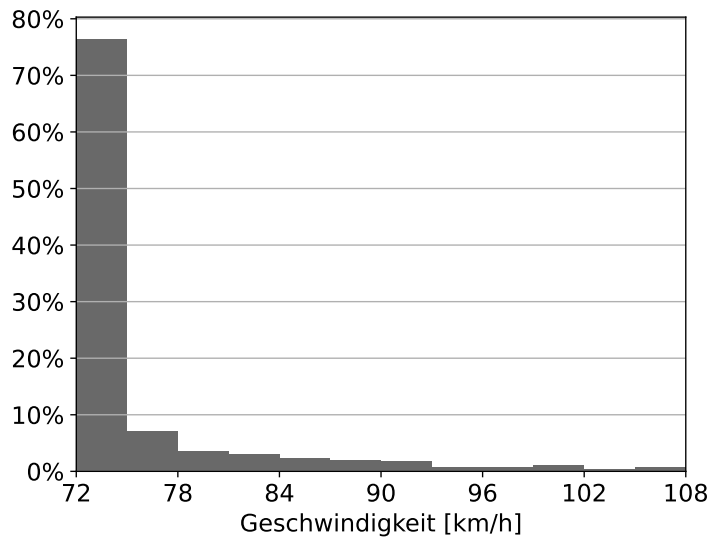


Abbildung 7.20: Verteilung der optimalen Geschwindigkeiten für zwei Drohnen

Zunächst werden die Ergebnisse für den Fall kontinuierlicher Geschwindigkeiten vorgestellt. Der durchschnittliche Energiebedarf für die Belieferung eines Kunden beträgt 71,9 Wh. Insgesamt verbrauchen die beiden Drohnen über die 15 Simulationstage 22,7 kWh. Die Verteilung der Geschwindigkeiten, die sich aus den 20 Simulationsläufen ergeben haben, ist im Histogramm 7.20 dargestellt. Im Histogramm sind die Geschwindigkeiten in 12 Klassen unterteilt. Jede Klasse hat eine Breite von 3 km/h, beginnend bei 72 km/h. Die relative Häufigkeit jeder Klasse ist auf der y-Achse abgetragen. Es ist ersichtlich, dass drei Viertel aller Geschwindigkeiten im Intervall von 72 bis 75 km/h liegen. Es zeigt sich, dass 71,4 % der Flüge mit der optimalen Geschwindigkeit durchgeführt werden. Zudem ist ersichtlich, dass die relative Häufigkeit mit höheren Geschwindigkeiten abnimmt.

Wie bereits dargelegt, werden für die diskreten Geschwindigkeiten vier Varianten mit unterschiedlichen Anzahlen an Geschwindigkeiten untersucht. Im Folgenden werden die Ergebnisse für das Szenario mit zwei verfügbaren Geschwindigkeiten präsentiert. Der durchschnittliche Energiebedarf für die Belieferung eines Kunden beträgt 72,9 Wh. Insgesamt verbrauchen die beiden Drohnen über die 15 Simulationstage durchschnittlich 23,1 kWh. Im Vergleich zum kontinuierlichen Fall entspricht dies einer Steigerung von etwa 1,4 %. Die Ergebnisse für die anderen Fälle sind der Tabelle 7.4 zu entnehmen. Die einzelnen Spalten zeigen den durchschnittlichen Energieverbrauch für die Belieferung eines Kunden sowie den durchschnittlichen Verbrauch für zwei Paketdrohnen über 15 Simulationstage für die verschiedenen Varianten. Die zweite Spalte zeigt die Verbräuche für den kontinuierlichen Fall, während die Spalten drei bis sechs die Ergebnisse der Diskretisierung der Geschwindigkeiten darstellen. Es ist erkennbar, dass mit einer zunehmenden Anzahl verfügbarer Geschwindigkeiten der Energiebedarf sinkt oder gleich bleibt.

Neben den Energieverbräuchen wird auch die Verteilung der Geschwindigkeiten im diskreten Fall untersucht. In Abbildung 7.21 ist die relative Häufigkeit der verschiedenen Geschwindigkeiten für die verschiedenen Diskretisierungen in Form von Kreisdiagrammen dargestellt. Die Abbildung

# Geschwindigkeiten	optimal	2	3	4	5
Energieverbrauch je Zustellung	71,9 Wh	72,9 Wh	72,3 Wh	72 Wh	72 Wh
Energieverbrauch für 15 Simulationstage	22,7 kWh	23 kWh	22,8 kWh	22,8 kWh	22,8 kWh

Tabelle 7.4: Energieverbrauch mit kontinuierlichen und diskreten Geschwindigkeiten

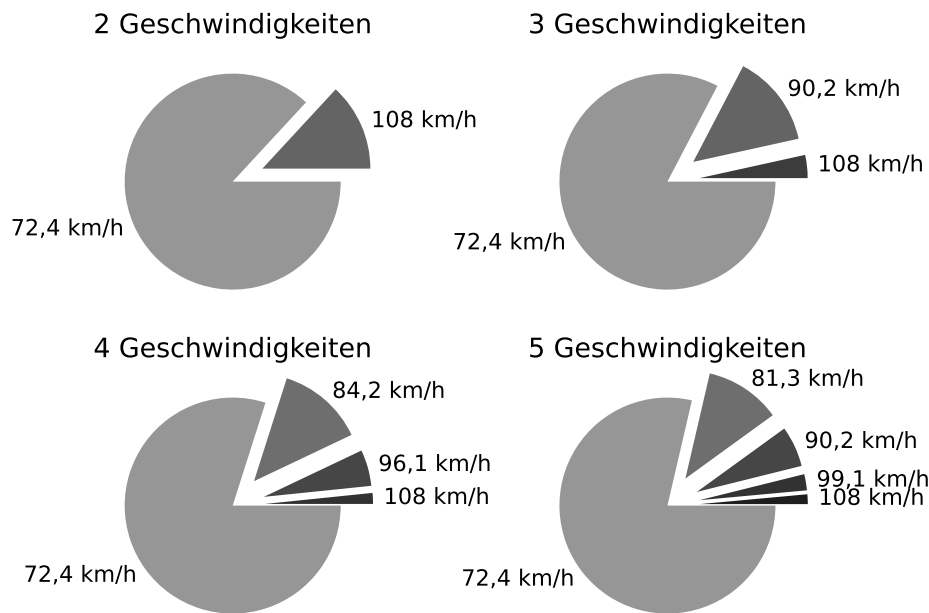


Abbildung 7.21: Verteilung diskreter Geschwindigkeiten für zwei Drohnen bei unterschiedlicher Anzahl an Diskretisierungsschritten

umfasst insgesamt vier Kreisdiagramme. Dabei entspricht die relative Größe eines Kreisstücks in einem Diagramm der relativen Häufigkeit der jeweiligen Geschwindigkeit. Es ist ersichtlich, dass die optimale Geschwindigkeit, unabhängig von der Anzahl der verfügbaren Geschwindigkeiten, den größten Anteil ausmacht. Zudem gilt, dass der Anteil der höchsten Geschwindigkeit mit einer Erhöhung der verfügbaren Geschwindigkeiten sinkt. Dasselbe gilt auch für den Anteil der optimalen Geschwindigkeit.

Abschließend lässt sich festhalten, dass die kontinuierliche Wahl der Geschwindigkeiten zu einem geringeren Energiebedarf führt. Dieser Vorteil ist jedoch für das betrachtete Szenario auf den einstelligen Prozentbereich begrenzt. Die Ausweitung auf größere Szenarien mit mehr Kunden ist nicht möglich, da das MIP (7.25)-(7.32) keine optimalen Lösungen innerhalb einer akzeptablen Rechenzeit liefert. Damit ist der Vergleich zwischen diskreten und kontinuierlichen Geschwindigkeiten abgeschlossen. Im nächsten Abschnitt wird die Paketzustellung mit Paketdrohnen von einem Depot in ein Zustellgebiet untersucht und ein Basisszenario mit verschiedenen Parameter-Variationen wird verglichen.

Paketzustellung im urbanen Raum mit Paketdrohnen

In diesem Abschnitt wird der Einsatz von Paketdrohnen für die Zustellung in der Kölner Innenstadt simuliert. Ziel ist es dabei, die Simulation zu verwenden, um Einblicke in die letzte Meile mit Paketdrohnen zu gewinnen. Ausgangspunkt ist ein Basisszenario mit 38 Kunden pro Tag und drei Drohnen. Es folgen Varianten dieses Szenarios zur Analyse verschiedener Einflussfaktoren. Für jedes Szenario werden 20 Simulationen durchgeführt. Die übrigen Parameter entsprechen dem vorherigen Abschnitt.

Im Basisszenario ergibt sich für die 38 Kunden ein durchschnittlicher Servicegrad von 97,4 %. Dabei beliefert eine Paketdrohne am Tag etwa 12,3 Kunden und legt eine Strecke von 204,2 km zurück. Für die Belieferung eines einzelnen Kunden werden im Durchschnitt 16,6 km zurückgelegt. Die Dauer einer Tour beträgt im Durchschnitt 24,51 Minuten, wobei der Batteriewechsel von drei Minuten mit eingerechnet ist. Die durchschnittliche Geschwindigkeit beträgt 79,4 km/h und liegt damit knapp über der optimalen Geschwindigkeit von 72,4 km/h. Der Energieverbrauch für die Belieferung eines einzelnen Kunden beträgt 72,8 Wh. Der Stromverbrauch für drei Drohnen über 15 Tage beträgt insgesamt 40,3 kWh.

Im nächsten Schritt wird das Basisszenario mit weiteren Szenarien verglichen. Hierbei wird der Einfluss der Zeitfensterlänge untersucht. Die Zeitfensterlänge wird auf 25 und 35 Minuten gesetzt. Im zweiten Vergleich wird ein Szenario betrachtet, in dem die letzte Meile aufgeteilt wird und die Pakete zunächst mit einem Transporter zu einem Mikrodepot transportiert und von dort mit Paketdrohnen zugestellt werden.

Die Reduzierung der Zeitfensterlänge von 30 Minuten auf 25 Minuten führt zu einer Reduzierung des Servicegrads um 3,5 % auf 94 %. Infolgedessen sinkt die Anzahl der belieferten Kunden auf durchschnittlich 11,9 pro Drohne und Simulationstag. Die durchschnittliche Geschwindigkeit verringert sich auf 77,5 km/h. Daraus ergibt sich ein Stromverbrauch von 38,7 kWh für die drei Drohnen über die 15 Tage. Grundsätzlich zeigt sich, dass eine Reduzierung der Länge der Zeitfenster den Servicegrad senkt und somit auch die Auslastung der Drohnen verringert. Durch die geringere Auslastung fliegen die Drohnen um 2,3 % langsamer und haben infolgedessen einen geringeren Energiebedarf.

Eine Erhöhung der Länge des Zeitfensters führt zu einem Anstieg des Servicegrads auf 98 %. Dies geht mit einem leichten Anstieg der Anzahl der belieferten Kunden einher. In Summe werden je Paketdrohne 12,4 Kunden beliefert. Auch die Gesamtdistanz für die drei Drohnen in den 15 Tagen steigt geringfügig auf 9.276 km. Die weiteren Kennzahlen ändern sich nur geringfügig, da die durchschnittliche Geschwindigkeit leicht auf 78,7 km/h sinkt. Damit ist die Darstellung der Veränderungen abgeschlossen, die durch die Anpassung der Länge der Zeitfenster hervorgerufen werden. Im Anschluss erfolgt eine Betrachtung der Veränderungen, die durch die Verschiebung des Abflugorts der Drohnen entstehen.

Neben der Zustellung vom Depot wird auch die Zustellung von einem Mikrodepot untersucht, das sich zwischen dem Depot und dem Zustellgebiet befindet. Die Lage des Mikrodepots ist in



Abbildung 7.22: Zustellgebiet in der Altstadt Nord und Altstadt Süd mit einem Mikrodepot

Abbildung 7.22 dargestellt. Das Mikrodepot ist in der Mitte zwischen Depot und Zustellgebiet positioniert. Zunächst werden die Sendungen mit einem Zustellfahrzeug zum Mikrodepot transportiert, von dort erfolgt die Zustellung ins Zustellgebiet mit den Paketdrohnen. Durch die Verlagerung des Abflugorts verkürzt sich die Entfernung zum Zustellgebiet um 4 km. Die Verlagerung des Abflugorts erfordert die Anmietung eines Mikrodepots am Stadtrand sowie den Transport der Sendungen mit einem Transporter dorthin. Allerdings verringert sich die Flugstrecke, die für die Belieferung der Kunden benötigt wird.

Durch kürzere Flugzeiten steigt der Servicegrad deutlich auf 99,5 %, sodass eine Drohne täglich 12,6 Kunden beliefern kann. Die zurückgelegte Gesamtstrecke der drei Drohnen sinkt in 15 Tagen auf 4.546 km und beträgt somit weniger als die Hälfte. Auch die durchschnittliche Distanz pro Zustellung reduziert sich auf 8 km. Mit 72,7 km/h liegt die Durchschnittsgeschwindigkeit nur knapp über dem Optimum. Der Energieverbrauch sinkt weniger stark, da distanzunabhängige Komponenten wie Auf-/Abstieg und Schweben weiterhin bestehen. Insgesamt verbrauchen die drei Drohnen 27,2 kWh in 15 Tagen, also durchschnittlich 47,85 Wh pro Zustellung.

Damit ist die Analyse der verschiedenen Szenarien abgeschlossen. Es lässt sich festhalten, dass die Verlagerung des Abflugorts erheblichen Einfluss hat: Der Energieverbrauch sinkt um über 30 %, während der Servicegrad hoch bleibt. Eine Anpassung der Zeitfenster ist nicht erforderlich, da eine Verkürzung den Servicegrad um 3,5 % senkt und eine Verlängerung kaum Wirkung zeigt. Damit ist die Ergebnispräsentation für die Simulation mit Paketdrohnen abgeschlossen.

Kapitel 8

Zusammenfassung und Ausblick

Die nachhaltige und kosteneffiziente Gestaltung der letzten Meile stellt eine der größten Herausforderungen für Logistikdienstleister dar. Daher werden alternative Konzepte erforscht, um die letzte Meile sowohl umweltfreundlicher als auch effizienter zu gestalten. Ein wesentlicher Fokus liegt dabei auf der Automatisierung und Autonomisierung der Zustellprozesse.

Zwei innovative Transportmittel, die autonom agieren und gleichzeitig einen hohen Servicegrad für die Empfänger bieten, sind Zustellroboter und Paketdrohnen. Im Vergleich zur traditionellen Zustellung per Transporter führt der Einsatz dieser Technologien zu einer grundlegenden Umstrukturierung der letzten Meile, die in zwei Abschnitte unterteilt wird. Zuerst erfolgt der Transport der Sendungen von einem zentralen Depot zu Mikrodepots mit einem Transporter. Anschließend übernehmen Zustellroboter oder Paketdrohnen den Weitertransport zu den Empfängern. Charakteristisch für beide Konzepte ist, dass jeweils nur eine einzelne Sendung transportiert und anschließend zugestellt wird, bevor das Fahrzeug zum Mikrodepot zurückkehrt.

Diese Arbeit fokussiert sich auf die Optimierung von Zustellnetzwerken mit Zustellrobotern und Paketdrohnen. Ziel ist die Entwicklung von Algorithmen zur effizienten Gestaltung der letzten Meile mit diesen innovativen Zustellkonzepten. Im Folgenden werden die zentralen Ergebnisse der Arbeit zusammengefasst, gefolgt von einem Ausblick auf zukünftige Forschungsfragen. Zuerst werden die Ergebnisse für die Zustellroboter vorgestellt und im Anschluss werden die Ergebnisse für die Paketdrohnen beschrieben.

Für die Zustellroboter gilt es, zu entscheiden, welche Sendung von welchem Mikrodepot mit welchem Roboter transportiert wird und in welcher Reihenfolge die Zustellroboter die Sendungen ausliefern. Die Roboter können nach einer Zustellung das Mikrodepot wechseln. Darüber hinaus können zwischen zwei Empfängern mehrere Mikrodepots angesteuert werden. Zusätzlich wird der Energieverbrauch der Roboter modelliert, sodass Ladezeiten an Mikrodepots mit einbezogen werden. Das Problem umfasst somit die simultane Planung der Paketzuteilung, der Routenplanung sowie der Ladezeiten. Das Ziel der Optimierung ist es, einen Plan zu erstellen, der die Anzahl der ausgelieferten Sendungen maximiert, wobei Empfänger nur innerhalb vorgegebener Zeitfenster beliefert werden dürfen und die Ladezeit der Roboter berücksichtigt werden muss.

Die Problemstellung wird zunächst auf einen einzelnen Roboter und eine gegebene Reihenfolge der Empfänger beschränkt. Es gilt zu entscheiden, ob zwischen je zwei Empfängern ein Pfad von Mikrodepots mit Ladezeiten existiert, sodass jeder Empfänger im Zeitfenster beliefert werden kann. Hierzu wird ein Algorithmus entwickelt, der dieses Entscheidungsproblem in polynomieller Zeit löst. Auf dieser Grundlage wird ein Branch-and-Price-Algorithmus vorgestellt, der für das Ausgangsproblem eine optimale Lösung berechnet.

Die Leistungsfähigkeit des Branch-and-Price-Algorithmus wird durch Experimente evaluiert. Zum Vergleich wird eine MIP-Formulierung implementiert und auf modifizierten Benchmark-Instanzen getestet. Da keine direkten Vergleichslösungen aus der Literatur vorliegen, dient dieser Ansatz als Referenz. Die Experimente zeigen, dass der Branch-and-Price-Algorithmus im Durchschnitt innerhalb einer Sekunde eine Lösung berechnet, während die MIP-Implementierung rund 30 Minuten benötigt.

Es wird auch das Laufzeitverhalten des Branch-and-Price-Algorithmus bei Variation der Eingabeparameter untersucht. Dabei zeigt sich, dass sowohl die Anzahl der Empfänger als auch die Länge der Zeitfenster einen signifikanten Einfluss auf die Rechenzeit haben. Zudem existiert ein bestimmtes Verhältnis zwischen der Anzahl an Zustellrobotern und Empfängern, das die Lösbarkeit der Instanzen erschwert.

Um den Mehrwert der Optimierung nicht nur auf Laufzeitebene zu analysieren, wird im letzten Abschnitt der Arbeit eine Discrete-Event-Simulation mit einem agentenbasierten Modellierungsansatz vorgestellt. Diese untersucht die Auswirkungen der flexiblen Mikrodepot-Zuweisung und der Ladezeitenoptimierung. Außerdem ermöglicht das Modell eine realistischere Modellierung der innovativen Zustellkonzepte. Die Simulationsergebnisse zeigen, dass sich durch die Möglichkeit, das Mikrodepot nach der Zustellung zu wechseln, die Anzahl der zugestellten Sendungen um 9 % bis 14 % steigern lässt. Zudem wechseln die Roboter nach einer Zustellung in 16 % bis 36 % der Fälle das Mikrodepot. Weiterhin zeigt sich, dass eine optimierte Ladestrategie die Anzahl der zugestellten Sendungen um 1,7 % erhöhen kann.

Ein weiterer zentraler Untersuchungsgegenstand dieser Arbeit ist die Optimierung eines Zustellnetzwerks für Paketdrohnen mit dem Ziel, den Energieverbrauch zu minimieren. Dieser kann über die Fluggeschwindigkeit der Drohnen gesteuert werden. Der Energieverbrauch verhält sich jedoch nichtlinear, sondern folgt einer streng konvexen Funktion.

Analog zum Problem der Zustellroboter gilt es auch hier zu entscheiden, welche Sendung mit welcher Drohne transportiert wird und in welcher Reihenfolge die Empfänger beliefert werden. Da Drohnen mit wechselbaren Akkus ausgestattet sind, entfällt die Notwendigkeit der Ladevorgänge. Ziel ist es, eine optimale Reihenfolge der Zustellungen zu bestimmen, sodass alle Empfänger innerhalb ihrer Zeitfenster beliefert werden und der Gesamtenergieverbrauch minimiert wird. Die Berechnung der optimalen Geschwindigkeiten erfolgt in polynomieller Zeit mithilfe eines aus der Literatur bekannten Algorithmus, der in dieser Arbeit um Mindest- und Höchstgeschwindigkeiten erweitert wird und dessen Korrektheit ebenfalls nachgewiesen wird. Darauf aufbauend wird ein weiterer Algorithmus entwickelt, der alle relevanten Belieferungszeitpunkte des letzten

Empfängers einer gegebenen Reihenfolge berechnet. Zusätzlich werden alle optimalen Geschwindigkeitsvektoren und Energieverbräuche berechnet. Basierend auf diesen beiden Algorithmen wird ein Branch-and-Price-Algorithmus vorgestellt, der das Ausgangsproblem löst. Dabei wird der Algorithmus zur Berechnung der Ankunftszeitpunkte zur Erstellung einer Dominanzregel genutzt.

Zur Evaluierung des Algorithmus werden Performance-Analysen durchgeführt. In der Literatur existieren bereits leistungsfähige Algorithmen zur Geschwindigkeitsoptimierung, die insbesondere auf Anwendungsfälle im Seetransport ausgelegt sind. Um einen direkten Vergleich mit diesen Verfahren zu ermöglichen, wird der entwickelte Algorithmus entsprechend angepasst, sodass er auch Instanzen aus dem Bereich der maritimen Transportlogistik lösen kann. Die Ergebnisse belegen, dass der Branch-and-Price-Algorithmus die Laufzeiten aus der Literatur signifikant unterbietet und im Durchschnitt mehr als 17,5-mal schneller Lösungen berechnet. Dies ist unter anderem auf die entwickelte Dominanzregel zurückführbar.

Abschließend wird der Nutzen der Optimierung für Paketdrohnen in einem Simulationsmodell untersucht. Als Vergleich wird eine Problemstellung mit diskretisierten Geschwindigkeiten vorgestellt. Hierbei steht nur eine feste Auswahl an möglichen Geschwindigkeiten zur Verfügung. Die Ergebnisse zeigen, dass der Energieverbrauch bei kontinuierlich optimierten Geschwindigkeiten am geringsten ist, auch wenn der Vorteil im untersuchten Szenario moderat ausfällt. Dies ist unter anderem auf die geringe Steigung der Energieverbrauchsfunktion für Paketdrohnen zurückzuführen. Dennoch bestätigt die Analyse, dass eine diskrete Geschwindigkeitswahl suboptimal ist, da sie die Potenziale zur Energieeinsparung nicht vollständig ausschöpft.

Neben der Untersuchung der Geschwindigkeiten werden zwei alternative Zustellprozesse für die Paketdrohnen untersucht. Dazu zählen die direkte Zustellung vom Depot sowie eine Variante, bei der die Pakete zunächst zu einem näher am Zustellgebiet gelegenen Mikrodepot transportiert werden. Aus den Ergebnissen wird deutlich, dass durch die Verlagerung des Abflugortes die Anzahl der erfolgreichen Zustellungen signifikant steigt und die insgesamt zurückgelegte Flugstrecke nahezu halbiert wird. Dies verdeutlicht den Effizienzgewinn durch die Kombination innovativer Zustellkonzepte mit strategisch platzierten Mikrodepots.

Ausblick

Auf Basis der untersuchten Problemstellung, der entwickelten Algorithmen sowie der vorgestellten Simulation ergeben sich weitere Forschungsfragen, die im Folgenden näher betrachtet werden. Insbesondere die Weiterentwicklung der Algorithmen sowie die Verknüpfung der behandelten Problemstellungen stehen hierbei im Fokus.

Mit dem Simulationsmodell konnte gezeigt werden, dass sich die Optimierungsalgorithmen zur Planung der letzten Meile der Logistik mit innovativen Zustellkonzepten in einem realitätsnahen Umfeld eignen. Ein nächster Schritt wäre die Erprobung der Algorithmen in einem Reallabor. Hierbei könnten sie zur Erstellung eines Auslieferungsplans eingesetzt werden. Durch die praktische Umsetzung ließe sich überprüfen, inwieweit der erzeugte Plan für die tatsächliche Paket-

zustellung geeignet ist und welche Aspekte einer genaueren Modellierung bedürfen. Dies betrifft insbesondere den Einfluss externer Faktoren wie Fußgängerverkehr, Staus oder Wetterbedingungen sowie Unsicherheiten, die durch die Interaktion mit Menschen entstehen können. Neben einer verbesserten Aussagekraft hinsichtlich der Übertragbarkeit der Algorithmen und ihrer Lösungen könnte durch das Reallabor zudem der konkrete Mehrwert der Optimierung quantifiziert und ihr Potenzial zur Effizienzsteigerung untersucht werden.

Neben der praktischen Validierung der Algorithmen stellt sich die Frage, inwieweit sie erweitert werden können, um eine noch realistischere Modellierung zu ermöglichen. Beispielsweise könnten genauere Modelle zur Beschreibung des Akkuverhaltens der Zustellroboter integriert und entsprechende Anpassungen der Algorithmen analysiert werden.

Darüber hinaus wäre es von Interesse, den Entscheidungsalgorithmus für die Zustellroboter auf das Electric Vehicle Routing Problem zu übertragen. In der Literatur werden Ladestationen typischerweise als separate Knoten im Graphen modelliert. Sollte sich der Algorithmus entsprechend erweitern lassen, könnten die Ladestationen direkt durch gebrochene Geraden abgebildet werden, sodass eine separate Modellierung entfallen würde. Damit ließe sich die Anzahl der Knoten im Graphen für den Branch-and-Price-Algorithmus stark reduzieren.

Ein weiterer spannender Forschungsaspekt ist die Verknüpfung der hier untersuchten Problemstellungen. Dabei würden die Geschwindigkeiten und Ladevorgänge zusammen optimiert. Diese Fragestellung ist insbesondere für elektrische Zustellfahrzeuge relevant, die ihren Hochvoltsspeicher nicht wechseln können, oder für Drohnen mit fest verbautem Akku. Die zu lösende Problemstellung würde die Tour- und Routenplanung mit der Wahl optimaler Geschwindigkeiten kombinieren. Zusätzlich müssen Ladeorte, -zeitpunkte und -mengen bestimmt werden. Analog zu den bisherigen Problemstellungen wäre es sinnvoll, zunächst eine gegebene Kundenreihenfolge zu betrachten und darauf aufbauend ein Entscheidungsproblem mit einem passenden Lösungsalgorithmus zu formulieren. Basierend auf diesem Algorithmus könnte anschließend ein erweiterter Branch-and-Price-Algorithmus entwickelt werden. Damit ließen sich weitere Transportmittel abbilden, wie beispielsweise elektrische Zustellfahrzeuge, die keinen Akkuwechsel durchführen, aber mit hohen Geschwindigkeiten fahren.

Zusammenfassend zeigt sich, dass im Rahmen dieser Arbeit zwei leistungsfähige Optimierungsalgorithmen entwickelt wurden, die sowohl hinsichtlich ihrer Laufzeit als auch ihrer praktischen Anwendbarkeit einen bedeutenden Beitrag zur Lösung von Planungsproblemen für innovative Zustellkonzepte auf der letzten Meile leisten.

Literaturverzeichnis

- Agatz, N., Bouman, P., und Schmidt, M. (2018). Optimization Approaches for the Traveling Salesman Problem with Drone. *Transportation Science*, 52(4):965–981.
- Alfandari, L., Ljubić, I., und De Melo da Silva, M. (2022). A tailored Benders decomposition approach for last-mile delivery with autonomous robots. *EJOR*, 299(2):510–525.
- Amazon (2022). Amazon Prime Air prepares for drone deliveries. <https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries>. abgerufen am 20.05.2025.
- Azi, N., Gendreau, M., und Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *EJOR*, 202(3):756–763.
- Bakach, I., Campbell, A. M., und Ehmke, J. F. (2021). A two-tier urban delivery network with robot-based deliveries. *Networks*, 78(4):461–483.
- Barmounakis, E. N., Vlahogianni, E. I., und Golias, J. C. (2016). Unmanned Aerial Aircraft Systems for transportation engineering: Current practice and future challenges. *International Journal of Transportation Science and Technology*, 5(3):111–122.
- BBC (2017). Hermes begins trial using self-driving robots in London. <https://www.bbc.com/news/technology-39589967>. abgerufen am 20.05.2025.
- BBC (2022). Amazon to begin drone deliveries in Lockeford, California this year. <https://www.bbc.com/news/business-61790085>. abgerufen am 20.05.2025.
- Bektaş, T. und Laporte, G. (2011). The Pollution-Routing Problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250. Supply chain disruption and risk management.
- Bellman, R. (1966). Dynamic Programming. *Science*, 153(3731):34–37.
- Bertram, I. (2018). PaketRadar - Das sind Deutschlands Paket-Hotspots. <https://newsroom.hermesworld.com/paketradar-das-sind-deutschlands-paket-hotspots-11329/>. abgerufen am 20.05.2025.
- Bertram I. (2016). Starship-Roboter liefern Pakete für Hermes in Hamburg aus. Accessed February 11, 2025.

- BIEK, K. (2024). 20 JAHRE KEP: DYNAMIK. TEMPO. WACHSTUM. KEP-Studie 2024 – Analyse des Marktes in Deutschland. https://bpex-ev.de/download.html?getfile=BPEX_KEP-Studie_2024.pdf. abgerufen am 20.05.2025.
- Bogdanski, R. (2017). Innovationen auf der letzten Meile - Bewertung der Chancen für die nachhaltige Stadtlogistik von morgen. https://bpex-ev.de/files/biek/downloads/papiere/BIEK_Nachhaltigkeitsstudie_2017.pdf. abgerufen am 20.05.2025.
- Boussier, S., Feillet, D., und Gendreau, M. (2006). An exact algorithm for team orienteering problems. *4OR*, 5(3):211–230.
- Boysen, N., Fedtke, S., und Schwerdfeger, S. (2020). Last-mile delivery concepts: a survey from an operational research perspective. *ORSP*, 43(1):1–58.
- Boysen, N., Schwerdfeger, S., und Weidinger, F. (2018). Scheduling last-mile deliveries with truck-based autonomous robots. *EJOR*, 271(3):1085–1099.
- Brandt, J. C., Böker, B., Bullinger, A., Conrads, M., Duisberg, A., und Silke, S.-R. (2018). Fallstudie: Delivery Robot Hamburg für KEP- Zustellung. <https://www.bmwk.de/Redaktion/DE/Downloads/C-D/delivery-robot-hamburg.pdf>. abgerufen am 20.05.2025.
- Bundesministerium für Digitales und Verkehr (2024). Verkehr in Zahlen 2024/2025.
- Chen, C., Demir, E., und Huang, Y. (2021a). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *EJOR*, 294(3):1164–1180.
- Chen, C., Demir, E., Huang, Y., und Qiu, R. (2021b). The adoption of self-driving delivery robots in last mile logistics. *Transp. Res. Part E Logist. Transp. Rev.*, 146:102214.
- Cheng, C., Adulyasak, Y., und Rousseau, L.-M. (2020). Drone routing with energy function: Formulation and exact algorithm. *Transportation Research Part B: Methodological*, 139:364–387.
- Clausen, U. und Geiger, C. (Hrsg.) (2013). *Verkehrs- und Transportlogistik*. Springer Berlin Heidelberg.
- Clausen, U., Stütz, S., Bernsmann, A., und Heinrichmeyer, H. (2016). ZF-Zukunftsstudie 2016: Die letzte Meile. Studie, ZF Friedrichshafen AG.
- CNBC Television (2022). Amazon Prime Air prepares for drone deliveries. <https://www.youtube.com/watch?v=Ds78kHn-n84>. abgerufen am 20.05.2025.
- Congress, G. C. (2017). Mercedes-Benz Vans, Matternet and siroop start pilot project for on-demand delivery of e-commerce goods; Vans & Drones. <https://www.greencarcongress.com/2017/09/20170929-drones.html>. abgerufen am 20.05.2025.
- Cordeau, J.-F., Laporte, G., und Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936.

- Cregger, J., Machek, E., Behan, M., Epstein, A., Lennertz, T., Shaw, J., und Dopart, K. (2020). Emerging Automated Urban Freight Delivery Concepts: State of the Practice Scan. Wissenschaftliche Berichte FHWA-JPO-20-825, U.S. Department of Transportation, Volpe National Transportation Systems Center 55 Broadway, Cambridge, MA 02142.
- Dabia, S., Demir, E., und Woensel, T. V. (2017). An Exact Approach for a Variant of the Pollution-Routing Problem. *Transportation Science*, 51(2):607–628.
- Demir, E., Bektaş, T., und Laporte, G. (2012). An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. *European Journal of Operational Research*, 223(2):346–359.
- Desaulniers, G., Errico, F., Irnich, S., und Schneider, M. (2016). Exact Algorithms for Electric Vehicle-Routing Problems with Time Windows. *Oper. Res.*, 64(6):1388–1405.
- Desaulniers, G., Lessard, F., und Hadjar, A. (2008). Tabu Search, Partial Elementarity, and Generalized k-Path Inequalities for the Vehicle Routing Problem with Time Windows. *Transport. Sci.*, 42(3):387–404.
- Desrosiers, J., Soumis, F., und Desrochers, M. (1984). Routing with time windows by column generation. *Networks*, 14(4):545–565.
- DHL (2018). Rapid Response from the Air: Medicines Successfully Delivered Using a Parcel Drone in East Africa. <https://group.dhl.com/en/media-relations/press-releases/2018/rapid-response-from-the-air-medicines-successfully-delivered-using-a-parcel-drone-in-east-africa.html>. abgerufen am 20.05.2025.
- DHL Customer Solutions & Innovation (2022). Unmanned Aerial Vehicles in Logistics: a DHL Perspective on Implications and Use Cases for the Logistics Industry. www.dhl.com/discover/content/dam/dhl/downloads/interim/full/dhl-trend-report-uav.pdf. abgerufen am 20.05.2025.
- Diestel, R. (2016). *Graphentheorie, 5.Auflage*. Springer-Diestel.
- Domschke, W., Drexl, A., Klein, R., und Scholl, A. (2015). *Einführung in Operations Research*. Springer Berlin Heidelberg.
- Dorling, K., Heinrichs, J., Messier, G. G., und Magierowski, S. (2017). Vehicle Routing Problems for Drone Delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85.
- Dukkanci, O., Kara, B. Y., und Bektaş, T. (2021). Minimizing energy and cost in range-limited drone deliveries with speed optimization. *Transportation Research Part C: Emerging Technologies*, 125:102985.
- Erdelić, T. und Carić, T. (2019). A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches. *J. Adv. Transp.*, 2019:1–48.
- Europäische Agentur für Flugsicherheit (2022a). Certified Category - Civil Drones. <https://www.easa.europa.eu/en/domains/drones-air-mobility/operating-drone/certified-category-civil-drones>. abgerufen am 20.05.2025.

- Europäische Agentur für Flugsicherheit (2022b). Certified Category — Civil Drones. <https://www.easa.europa.eu/en/domains/drones-air-mobility/operating-drone/certified-category-civil-drones>. abgerufen am 20.05.2025.
- Europäische Agentur für Flugsicherheit (2022c). Open Category — Low Risk — Civil Drones. <https://www.easa.europa.eu/en/domains/drones-air-mobility/operating-drone/open-category-low-risk-civil-drones>. abgerufen am 20.05.2025.
- Fagerholt, K., Laporte, G., und Norstad, I. (2010). Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society*, 61(3):523–529.
- Federal Aviation Administration (2022). Airworthiness Criteria: Special Class Airworthiness Criteria for the Amazon Logistics, Inc. MK27-2 Unmanned Aircraft. <https://www.federalregister.gov/documents/2022/01/27/2022-01556/airworthiness-criteria-special-class-airworthiness-criteria-for-the-amazon-logistics-inc-mk27-2>. abgerufen am 20.05.2025.
- Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424.
- Feillet, D., Dejax, P., Gendreau, M., und Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.
- Ferguson, D. (2020). Introducing R2, Nuro’s Next Generation Self-Driving Vehicle. <https://medium.com/nuro/introducing-r2-nuros-next-generation-self-driving-vehicle-a9974ff6c2e0>. abgerufen am 20.05.2025.
- Figliozzi, M. A. (2017). Lifecycle modeling and assessment of unmanned aerial vehicles (Drones) CO₂e emissions. *Transportation Research Part D: Transport and Environment*, 57:251–261.
- Froger, A., Mendoza, J. E., Jabali, O., und Laporte, G. (2019). Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Comput. Oper. Res.*, 104:256–294.
- Fukasawa, R., He, Q., Santos, F., und Song, Y. (2017). A joint routing and speed optimization problem. *Unveröffentlicher Report*.
- Fukasawa, R., He, Q., Santos, F., und Song, Y. (2018). A Joint Vehicle Routing and Speed Optimization Problem. *INFORMS Journal on Computing*, 30(4):694–709.
- Fukasawa, R., He, Q., und Song, Y. (2016). A disjunctive convex programming approach to the pollution-routing problem. *Transportation Research Part B: Methodological*, 94:61–79.
- Gale, D., Kuhn, H. W., und Tucker, A. W. (1951). Linear programming and the theory of games. *Activity analysis of production and allocation*, 13:317–335.
- Gehrke, S. R., Phair, C. D., Russo, B. J., und Smaglik, E. J. (2023). Observed sidewalk autonomous delivery robot interactions with pedestrians and bicyclists. *Transportation Research Interdisciplinary Perspectives*, 18:100789.

- Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R. L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M. E., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J. M., Walter, M., Wegscheider, F., Witt, J. T., und Witzig, J. (2018). The SCIP Optimization Suite 6.0. Technical report, Optimization Online.
- Green Vision Solution (2023). CO₂e: Die Bedeutung des CO₂-Äquivalents einfach erklärt . <https://greenvisionsolutions.de/co2e-erklaert/>. abgerufen am 20.05.2025.
- Handelsverband Deutschland (HDE) (2024). Online Monitor 2024.
- He, Q., Zhang, X., und Nip, K. (2017). Speed optimization over a path with heterogeneous arc costs. *Transportation Research Part B: Methodological*, 104:198–214.
- Heid, B., Kässer, M., Klink, C., Neuhaus, C., Neuhaus, F., Schröder, J., und Tatomir, S. (2018). Fast Forwarding Last-Mile Delivery: Implications for the Ecosystem. Wissenschaftliche berichte, McKinsey.
- Heimfarth, A., Ostermeier, M., und Hübner, A. (2022). A mixed truck and robot delivery approach for the daily supply of customers. *EJOR*, 303(1):401–421.
- Hemmati, A., Hvattum, L. M., Fagerholt, K., und Norstad, I. (2014). Benchmark Suite for Industrial and Tramp Ship Routing and Scheduling Problems. *INFOR: Information Systems and Operational Research*, 52(1):28–38.
- Hiermann, G., Hartl, R. F., Puchinger, J., und Vidal, T. (2019). Routing a mix of conventional, plug-in hybrid, and electric vehicles. *EJOR*, 272(1):235–248.
- Hölter, A.-K. und Ninnemann, J. (2020). Effizienz auf der letzten Meile – Optimierung der Schnittstellen zwischen Kunde, Logistik und Onlinehandel. *Journal für Mobilität und Verkehr*, 5:30–34.
- Hossain, M. (2022). Self-Driving Robots: A Revolution in the Local Delivery. <https://cmr.berkeley.edu/assets/documents/pdf/2022-04-self-driving-robots-a-revolution-in-the-local-delivery.pdf>. abgerufen am 20.05.2025.
- Hvattum, L. M., Norstad, I., Fagerholt, K., und Laporte, G. (2013). Analysis of an exact algorithm for the vessel speed optimization problem. *Networks*, 62(2):132–135.
- Keskin, M. und Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transp. Res. Part C: Emerg. Tech.*, 65:111–127.
- Kirschstein, T. (2020). Comparison of energy demands of drone-based and ground-based parcel delivery services. *Transportation Research Part D: Transport and Environment*, 78:102209.
- Korte, B. und Vygen, J. (2018). *Kombinatorische Optimierung*. Springer-Verlag GmbH.
- Kramer, R., Subramanian, A., Vidal, T., und dos Anjos F. Cabral, L. (2015). A matheuristic approach for the Pollution-Routing Problem. *European Journal of Operational Research*, 243(2):523–539.

- Kullman, N. D., Froger, A., Mendoza, J. E., und Goodson, J. C. (2021). frvcpy: An Open-Source Solver for the Fixed Route Vehicle Charging Problem. *INFORMS J. Comput.*, 33(4):1277–1283.
- Law, A. M. (2014). *Simulation Modeling and Analysis*. McGraw-Hill Professional, New York, NY, 5. Auflage.
- Lee, D., Hess, D. J., und Heldeweg, M. A. (2022). Safety and privacy regulations for unmanned aerial vehicles: A multiple comparative analysis. *Technology in Society*, 71:102079.
- Leerkamp, B., Soteropoulos, A., und Berger, M. (2021). *Zustellroboter als Lösung für die letzte Meile in der Stadt?*, S. 107–130. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Liu, Z., Sengupta, R., und Kurzhanskiy, A. (2017). A power consumption model for multi-rotor small unmanned aircraft systems. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE.
- Lübbecke, M. E. (2011). *Column Generation*. John Wiley & Sons, Ltd.
- Marra, F., Yang, G. Y., Træholt, C., Larsen, E., Rasmussen, C. N., und You, S. (2012). Demand profile study of battery electric vehicle under different charging options. In *2012 IEEE PES General Meeting*, S. 1–7.
- Matternet (2020). Matternet Unveils the New Matternet Station. <https://www.suasnews.com/2020/03/matternet-unveils-the-new-matternet-station/>. abgerufen am 20.05.2025.
- Moshref-Javadi, M. und Winkenbach, M. (2021). Applications and Research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications*, 177:114854.
- Murray, C. C. und Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109.
- Norstad, I., Fagerholt, K., und Laporte, G. (2011). Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies*, 19(5):853–865.
- OpenStreetMap contributors (2017). Planet dump retrieved from <https://planet.osm.org>. <https://www.openstreetmap.org>.
- Osborne C. (2018). Starship robots take over package delivery on UK streets. <https://www.zdnet.com/article/autonomous-robot-package-delivery-service-launches-in-uk/>. abgerufen am 20.05.2025.
- Ostermeier, M., Heimfarth, A., und Hübner, A. (2022). Cost-optimal truck-and-robot routing for last-mile delivery. *Networks*, 79(3):364–389.
- Otto, A., Agatz, N., Campbell, J., Golden, B., und Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, 72(4):411–458.

- Park, J., Lee, J., Ahn, S., Bae, J., und Tae, H. (2017). Exact Algorithm for the Capacitated Team Orienteering Problem with Time Windows. *Math. Probl. Eng.*, 2017:1024–1230.
- Poeting, M., Schaudt, S., und Clausen, U. (2019a). A Comprehensive Case Study in Last-Mile Delivery Concepts for Parcel Robots. In Mustafee, N., Bae, K.-H. G., Lazarova-Molnar, S., Rabe, M., und Szabo, C. (Hrsg.), *Proceedings of the Winter Simulation Conference*, S. 1779–1788. IEEE Press.
- Poeting, M., Schaudt, S., und Clausen, U. (2019b). Simulation of an Optimized Last-Mile Parcel Delivery Network Involving Delivery Robots. In *Advances in Production, Logistics and Traffic*, S. 1–19. Springer International Publishing.
- Poeting, M., Schaudt, S., und Clausen, U. (2019c). Simulation of an Optimized Last-Mile Parcel Delivery Network Involving Delivery Robots. In Clausen, U., Langkau, S., und Kreuz, F. (Hrsg.), *Advances in Production, Logistics, and Traffic – Proceedings of the 4th Interdisciplinary Conference on Production Logistics and Traffic 2019*, Lecture Notes in Logistics, S. 1–19, Cham, Switzerland. Springer Nature Switzerland AG.
- Rabe, M., Wenzel, S., und Spieckermann, S. (2008). *Verifikation und Validierung für die Simulation in Produktion und Logistik*. Springer Berlin Heidelberg.
- Righini, G. und Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273.
- Roberti, R. und Wen, M. (2016). The Electric Traveling Salesman Problem with Time Windows. *Transp. Res. Part E Logist. Transp. Rev.*, 89:32–52.
- Schaudt, S. und Bektaş, T. (2026). Sequence and speed optimization. *INFORMS J. Comput.*, (ijoc.2024.1047).
- Schneider, M., Stenger, A., und Goeke, D. (2014). The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. *Transport. Sci.*, 48(4):500–520.
- Schweizerische Post AG (2017). Factsheet - Starship Delivery Robot. <https://www.post.ch/-/media/post/ueber-uns/medienmitteilungen/2017/factsheet-lieferroboter.pdf>. abgerufen am 20.05.2025.
- Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., und Guizani, M. (2019). Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access*, 7:48572–48634.
- Shankland, S. (2018). Take a look at Zipline’s new drone delivery system. <https://www.cnet.com/pictures/take-a-look-at-ziplines-new-drone-delivery-system/12/>. abgerufen am 20.05.2025.
- Song, B. D., Park, K., und Kim, J. (2018). Persistent UAV delivery logistics: MILP formulation and efficient heuristic. *Computers & Industrial Engineering*, 120:418–428.

- Sonneberg, M.-O., Leyerer, M., Kleinschmidt, A., Knigge, F., und Breitner, M. H. (2019). Autonomous Unmanned Ground Vehicles for Urban Logistics: Optimization of Last Mile Delivery Operations. In *Proc. 52nd Hawaii Int. Conf.Syst. Sci.*, S. 1538–1547.
- StarShip Technologies (2022). Starship Q&A. <https://www.starship.xyz/kit/>. abgerufen am 20.05.2025.
- Tae, H. und Kim, B.-I. (2015). A Branch-and-Price Approach for the Team Orienteering Problem with Time Windows. *Int. J. Ind. Eng. Comput.*, 22(2):243 – 251.
- Thaller, C., Telake, M., Clausen, U., Dahmen, B., und Leerkamp, B. (2017). KEP-Verkehr in urbanen Räumen. In *Innovative Produkte und Dienstleistungen in der Mobilität*, S. 443–458. Springer Fachmedien Wiesbaden.
- Thiels, C. A., Aho, J. M., Zietlow, S. P., und Jenkins, D. H. (2015). Use of Unmanned Aerial Vehicles for Medical Product Transport. *Air Medical Journal*, 34(2):104–108.
- Tittmann, P. (2021). *Graphentheorie*. Hanser Fachbuchverlag.
- Tänavsuu, T. (2017). Estonian delivery robots are transforming the world. <https://estonia.ee/delivery-robots-created-by-estonian-engineers-are-transforming-the-world/>. abgerufen am 20.05.2025.
- Umweltbundesamt (2023). Klimaschutz im Verkehr.
- Viloria, D. R., Solano-Charris, E. L., Muñoz-Villamizar, A., und Montoya-Torres, J. R. (2020). Unmanned aerial vehicles/drones in vehicle routing problems: a literature review. *Intl. Trans. in Op. Res.*, 28:1626–1657.
- Weinberg, D., Dwyer, H., Fox, S. E., und Martelaro, N. (2023). Sharing the Sidewalk: Observing Delivery Robot Interactions with Pedestrians during a Pilot in Pittsburgh, PA. *Multimodal Technologies and Interaction*, 7(5).
- Wellbrock, W., Ludin, D., und Knezevic, I. (2022). *Letzte Meile 4.0*. Springer Fachmedien Wiesbaden.
- Wingcopter (2021). Technical Details: Wingcopter 178 Heavy Lift A (Delivery Variant). <https://wingcopter.com/wp-content/uploads/2021/02/Technical-Details-Wingcopter-178-Heavy-Lift-A-Delivery-Variant-1-1.pdf>. abgerufen am 20.05.2025.
- Wong, J. C. (2017). San Francisco sours on rampant delivery robots - 'Not every innovation is great'. <https://www.theguardian.com/us-news/2017/dec/10/san-francisco-delivery-robots-laws>. abgerufen am 20.05.2025.
- Yu, S., Puchinger, J., und Sun, S. (2022). Electric van-based robot deliveries with en-route charging. *EJOR*.
- Zeng, Y., Xu, J., und Zhang, R. (2019). Energy Minimization for Wireless Communication With Rotary-Wing UAV. *IEEE Transactions on Wireless Communications*, 18(4):2329–2345.

Anhang

A Korrektheitsbeweis von Algorithmus 1

Im Folgenden wird die Korrektheit von Algorithmus 1 unter der Annahme gezeigt, dass die Servicezeiten der Kunden null sind. Diese Annahme ist ohne Beschränkung der Allgemeinheit zulässig, da sich die Zeitfenster bei gegebener Sequenz entsprechend anpassen lassen.

Der Beweis von Lemma 1 stützt sich auf die Propositionen 3 bis 10, in denen gezeigt wird, dass alle von den Algorithmen 1 und 2 berechneten Paare gültige Zeit-Batterie-Paare sind.

Proposition 3. *Die Paare $(\underline{t}, \underline{\beta})$ und $(\bar{t}, \bar{\beta})$, die in Zeile 2 von Algorithmus 1 berechnet werden, sind genau dann Zeit-Batterie-Paare, wenn die Bedingung in Zeile 3 des Algorithmus nicht erfüllt ist.*

Beweis. Es wird angenommen, dass der Zustellroboter zum Zeitpunkt null mit vollständig geladenem Akku startet. Daraus folgt, dass eine Ankunft bei Kunde u_1 vom nächstgelegenen Depot zum Zeitpunkt d_1 mit einem Ladestand von $B - d_1$ möglich ist. Der früheste Belieferungszeitpunkt ergibt sich somit als $\max(a_1, d_1)$ und der Ladestand beträgt $B - d_1$.

Zeile 3 stellt sicher, dass $\underline{t} \leq b_1$ und $\underline{\beta} \geq \frac{B}{2}$ gelten. Die erste Ungleichung gewährleistet, dass u_1 innerhalb des Zeitfensters beliefert werden kann. Die zweite Ungleichung garantiert, dass eine Rückkehr zum nächstgelegenen Depot von u_1 hinsichtlich der Akkukapazität möglich ist. Damit sind $(\underline{t}, \underline{\beta})$ und $(\bar{t}, \bar{\beta})$ Zeit-Batterie-Paare.

Umgekehrt wird angenommen, dass ein Zeit-Batterie-Paar (t, β) für u_1 gegeben ist. Daraus folgt $a_1 \leq t \leq b_1$ und $\beta \geq d_1$. Zudem muss $t \geq d_1$ und $\beta \leq B - d_1 = \underline{\beta}$ gelten, da es kein näheres Depot zu u_1 gibt. Folglich gilt $\underline{t} = \max(a_1, d_1) \leq t \leq b_1$ und $B = d_1 + B - d_1 \leq \beta + \underline{\beta} \leq 2\underline{\beta}$ und die Aussage der Proposition gilt. $\square$

Proposition 4. *Jedes Paar (t_k, β_k) , das in den Zeilen 6 und 7 von Algorithmus 2 berechnet wird, ist ein Zeit-Batterie-Paar.*

Beweis. Sei u_i der erste Kunde in der Eingabe von Algorithmus 2. Wähle $u_k \in D$ so, dass $d_{ik} \leq \bar{\beta}_i^\sigma$ gilt. Außerdem wähle $\tilde{\rho} = \min\{\rho : \bar{\beta}_i^\rho \geq d_{ik}, \rho \in \{1, \dots, \sigma\}\}$, $\lambda = \frac{1}{\alpha} \min(0, d_{ik} - \underline{\beta}_i^{\tilde{\rho}})$, $t_k = \underline{t}_i^{\tilde{\rho}} + \lambda + d_{ik}$ und $\beta_k = \underline{\beta}_i^{\tilde{\rho}} + \alpha\lambda - d_{ik}$ entsprechend den Zeilen 4 bis 7 von Algorithmus 2.

Die Wahl von $\tilde{\rho}$ impliziert, dass

$$\underline{t}_i^{\tilde{\rho}} \leq \underline{t}_i^{\tilde{\rho}} + \lambda \leq \bar{t}_i^{\tilde{\rho}} \text{ und } \underline{\beta}_i^{\tilde{\rho}} \leq \underline{\beta}_i^{\tilde{\rho}} + \alpha\lambda \leq \bar{\beta}_i^{\tilde{\rho}}. \quad (1)$$

Die Paare $(\underline{t}_i^{\rho}, \underline{\beta}_i^{\rho}), (\bar{t}_i^{\rho}, \bar{\beta}_i^{\rho})$ mit $\rho = 1, \dots, \sigma$ sind gebrochene Paare am Kunden u_i . Aus Gleichung (1) und Definition 11 folgt, dass das Paar $(t, \beta(t))$ mit $t = t_k - d_{ik} = \underline{t}_i^{\tilde{\rho}} + \lambda$ und $\beta(t) = \beta_k - d_{ik} = \underline{\beta}_i^{\tilde{\rho}} + \alpha\lambda$ ein Zeit-Batterie-Paar für Kunde u_i bildet. Das Paar wird zu (t_k, β_k) erweitert, wodurch eine zulässige Lösung von u_i nach u_k über die Kante $\{i, k\}$ entsteht.

□

Proposition 5. *Jedes Paar (t_l, β_l) , das in den Zeilen 17 und 18 von Algorithmus 2 berechnet wird, ist ein Zeit-Batterie-Paar.*

Beweis. Der Beweis erfolgt durch Induktion. Angenommen, (t_k, β_k) ist ein Zeit-Batterie-Paar am Depot u_k , wobei $k = \arg \min\{t_w : w \in D \setminus F\}$ aus Zeile 12 von Algorithmus 2 stammt.

Sei π die Lösung, die als Zertifikat für das Paar (t_k, β_k) dient (siehe Definition 7).

Wenn $d_{kl} \leq B$ gilt, kann die Lösung π über die Kante $\{k, l\}$ erweitert werden. Dafür müssen $\max(0, d_{kl} - \beta_k)$ Einheiten geladen werden. Diese erweiterte Lösung ist ein Zertifikat für das Zeit-Batterie-Paar (t_l, β_l) , wobei $t_l = t_k + \frac{1}{\alpha} \max(0, d_{kl} - \beta_k) + d_{kl}$ und $\beta_l = \beta_k + \max(0, d_{kl} - \beta_k) - d_{kl}$ wie in den Zeilen 15 bis 18 berechnet wird. □

Proposition 6. *Die Paare $(\underline{t}_k, \underline{\beta}_k)$ und $(\bar{t}_k, \bar{\beta}_k)$, die in den Zeilen 27 bis 30 von Algorithmus 2 berechnet werden, sind Zeit-Batterie-Paare.*

Beweis. Zunächst wird gezeigt, dass $\underline{t}_k, \bar{t}_k \in [a_j, b_j]$ liegen und $\underline{\beta}_k, \bar{\beta}_k \geq d_j$ gelten.

Gemäß Zeile 27 gilt $\underline{t}_k = \max(a_j, t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj})$. Gemäß der Definition der Menge D' gilt, dass $t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj} \leq b_j$ und $d_{kj} + d_j \leq B$ gilt (siehe Zeile 22). Es folgt $a_j \leq \underline{t}_k \leq b_j$, da $a_j \leq b_j$ gilt.

Gemäß Zeile 28 gilt $\bar{t}_k = \max(\underline{t}_k, \min(b_j, t_k + \frac{1}{\alpha}(B - \beta_k) + d_{kj}))$. Damit folgt $a_j \leq \underline{t}_k \leq \bar{t}_k \leq b_j$.

Es bleibt nachzuweisen, dass $[\underline{\beta}_k, \bar{\beta}_k] \geq d_j$ gilt. Dafür ist Folgendes zu beachten:

$$\begin{aligned} \underline{\beta}_k & \stackrel{\text{(Zeile 29)}}{=} \min(B - d_{kj}, \beta_k + \alpha(\underline{t}_k - t_k - d_{kj}) - d_{kj}) \\ & \stackrel{\text{(Zeile 27)}}{=} \min(B - d_{kj}, \\ & \quad \beta_k + \alpha(\max(a_j, t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj}) - t_k - d_{kj}) - d_{kj}) \\ & \geq \min(B - d_{kj}, \beta_k + \alpha(t_k + \frac{1}{\alpha}(d_{kj} + d_j - \beta_k) + d_{kj} - t_k - d_{kj}) - d_{kj}) \\ & = \min(B - d_{kj}, d_j) \stackrel{\text{(Zeile 22)}}{\geq} d_j. \end{aligned}$$

Außerdem gilt, dass

$$\begin{aligned}
\bar{\beta}_k &\stackrel{\text{(Zeile 30)}}{=} \min(B - d_{kj}, \beta_k + \alpha(b_j - t_k - d_{kj}) - d_{kj}) \\
&\stackrel{\text{(Zeile 22)}}{\geq} \min(B - d_{kj}, \beta_k + \alpha(\underline{t}_k - t_k - d_{kj}) - d_{kj}) \\
&\stackrel{\text{(Zeile 29)}}{=} \underline{\beta}_k.
\end{aligned}$$

Daraus folgt, dass $\bar{\beta}_k \geq \underline{\beta}_k \geq d_j$ gilt. Beide Ladestände $\underline{\beta}_k$ und $\bar{\beta}_k$ sind durch $B - d_{kj}$ begrenzt (siehe Zeilen 29 und 30).

Angenommen, (t_k, β_k) ist ein Zeit-Batterie-Paar und sei π sein Zertifikat. Die Lösung π kann über die Kante $\{k, j\}$ zu u_j erweitert werden, wodurch sich ein Zeit-Batterie-Paar $(\underline{t}_k, \underline{\beta}_k)$ für u_j ergibt. Dabei entspricht die Ladezeit $\max(a_j - t_k - d_{kj}, \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k))$. Zu beachten ist, dass die Ladezeit gegebenenfalls auch Wartezeiten am Depot beinhaltet. Eine einfache Rechnung zeigt, dass die erweiterte Lösung zum Zeitpunkt $\underline{t}_k$ mit einem Ladestand von $\underline{\beta}_k$ bei u_j ankommt. Da $\underline{\beta}_k \geq d_j$ gilt, folgt, dass $(\underline{t}_k, \underline{\beta}_k)$ ein Zeit-Batterie-Paar ist.

Analog kann π zu u_j erweitert werden und als Zertifikat für $(\bar{t}_k, \bar{\beta}_k)$ fungieren. Dabei wird $\max(\underline{t}_k - t_k - d_{kj}, \min(b_j - t_k - d_{kj}, \frac{1}{\alpha}(B - \beta_k)))$ Zeit geladen und gegebenenfalls gewartet, falls der Akku vollständig geladen ist. Eine einfache Rechnung zeigt, dass die erweiterte Lösung zum Zeitpunkt $\bar{t}_k$ mit einem Ladestand von $\bar{\beta}_k$ bei u_j ankommt. Da $\bar{\beta}_k \geq d_j$ gilt, ist $(\bar{t}_k, \bar{\beta}_k)$ somit ein Zeit-Batterie-Paar. $\square$

Basierend auf der vorherigen Proposition lässt sich die folgende Beobachtung treffen.

Beobachtung 6. Für obere und untere temporäre Paare gilt stets: $\underline{t}_k \leq \bar{t}_k$ und $\underline{\beta}_k \leq \bar{\beta}_k$.

Die folgende Proposition zeigt, dass die Steigung der linearen Interpolation zwischen unterem und oberem temporären Paar gleich der Laderate α ist.

Proposition 7. Alle Paare $(\underline{t}_k, \underline{\beta}_k)$ und $(\bar{t}_k, \bar{\beta}_k)$, die in den Zeilen 27 bis 30 von Algorithmus 2 berechnet werden, erfüllen die folgende Gleichung:

$$\frac{\bar{\beta}_k - \underline{\beta}_k}{\bar{t}_k - \underline{t}_k} = \alpha,$$

es sei denn, es gilt $\underline{t}_k = \bar{t}_k$. Zudem folgt, dass für $\lambda \in [0, 1]$ das Paar $\lambda(\underline{t}_k, \underline{\beta}_k) + (1 - \lambda)(\bar{t}_k, \bar{\beta}_k)$ ein Zeit-Batterie-Paar am Kunden u_j ist.

Beweis. Zunächst wird gezeigt, dass $\underline{\beta}_k \neq \bar{\beta}_k$ sein muss. Angenommen, es gilt $\underline{\beta}_k = \bar{\beta}_k$. Gemäß Proposition 6 und der Annahme gilt $\underline{t}_k < \bar{t}_k \leq b_j$. Daraus folgt, dass gemäß den Zeilen 29 und 30 $\underline{\beta}_k = \bar{\beta}_k = B - d_{kj}$ gilt. Zeile 29 impliziert, dass $B - d_{kj} \leq \beta_k + \alpha(\underline{t}_k - t_k - d_{kj}) - d_{kj}$. Somit folgt $\underline{t}_k \geq t_k + \frac{1}{\alpha}(B - \beta_k) + d_{kj}$. Aus der letzten Ungleichung zusammen mit Zeile 27 folgt, dass $\underline{t}_k = \bar{t}_k$ gelten muss. Dies steht im Widerspruch zur Annahme. Daher muss $\underline{\beta}_k \neq \bar{\beta}_k$ gelten.

Es gilt $\underline{\beta}_k = \beta_k + \alpha(\underline{t}_k - t_k - d_{kj}) - d_{kj}$, da $\underline{\beta}_k \leq \bar{\beta}_k$ (siehe Beobachtung 6) und $\underline{\beta}_k \neq \bar{\beta}_k$ gilt. Die Steigung der linearen Interpolation berechnet sich somit wie folgt:

$$\frac{\bar{\beta}_k - \underline{\beta}_k}{\bar{t}_k - \underline{t}_k} \stackrel{\text{(Zeilen 27-30)}}{=} \frac{\min(B - d_{kj}, \beta_k + \alpha(b_j - t_k - d_{kj}) - d_{kj}) - (\beta_k + \alpha(\underline{t}_k - t_k - d_{kj}) - d_{kj})}{\min(b_j, t_k + \frac{1}{\alpha}(B - \beta_k) + d_{kj}) - \underline{t}_k}.$$

Es werden vier Fälle unterschieden:

- (i) $\bar{\beta}_k = \beta_k + \alpha(b_j - t_k - d_{kj}) - d_{kj}$ und $\bar{t}_k = b_j$,
- (ii) $\bar{\beta}_k = \beta_k + \alpha(b_j - t_k - d_{kj}) - d_{kj}$ und $\bar{t}_k = t_k + \frac{1}{\alpha}(B - \beta_k) + d_{kj}$,
- (iii) $\bar{\beta}_k = B - d_{kj}$ und $\bar{t}_k = t_k + \frac{1}{\alpha}(B - \beta_k) + d_{kj}$,
- (iv) $\bar{\beta}_k = B - d_{kj}$ und $\bar{t}_k = b_j$.

Fall (i): Daraus folgt, dass die Steigung gleich α ist.

Fall (ii): Da $\bar{\beta}_k \leq B - d_{kj}$ gilt, folgt daraus $\alpha(b_j - t_k - d_{kj}) \leq B - \beta_k$. Daher gilt $\bar{t}_k \geq b_j$. Gemäß Proposition 6 folgt, dass $\bar{t}_k = b_j$ gilt, und dieser Fall entspricht somit Fall (i).

Fall (iii): Auch hier folgt, dass die Steigung gleich α ist.

Fall (iv): Es gilt $\bar{\beta}_k \leq \beta_k + \alpha(b_j - t_k - d_{kj}) - d_{kj}$. Dies ist äquivalent zu $b_j \geq t_k + \frac{1}{\alpha}(B - \beta_k) + d_{kj}$. Daher gilt $\bar{t}_k = b_j = t_k + \frac{1}{\alpha}(B - \beta_k) + d_{kj}$ und dieser Fall entspricht somit Fall (iii).

Ein Zertifikat für $\lambda(\underline{t}_k, \underline{\beta}_k) + (1 - \lambda)(\bar{t}_k, \bar{\beta}_k)$ kann durch die Zertifikate der Paare $(\underline{t}_k, \underline{\beta}_k)$ und $(\bar{t}_k, \bar{\beta}_k)$ erzeugt werden. Statt eine Abfahrt vom Depot zu Kunde u_j zum Zeitpunkt $\underline{t}_k - d_{kj}$ zu wählen, werden zusätzlich $\lambda \underline{t}_k + (1 - \lambda)\bar{t}_k - \underline{t}_k$ Zeiteinheiten zum Laden verwendet. Wenn $\lambda = 0$ ist, entspricht dies dem Zeit-Batterie-Paar $(\underline{t}_k, \underline{\beta}_k)$. Bei $\lambda = 1$ entspricht es $(\bar{t}_k, \bar{\beta}_k)$. Die Zeilen 29 und 30 implizieren, dass $\underline{\beta}_k, \bar{\beta}_k$ größer oder gleich d_j sind. Der Beweis ist damit abgeschlossen. $\square$

In der folgenden Proposition werden zwei Depot-Pfade zwischen zwei Kunden verglichen. Es wird gezeigt, dass, wenn ein Depot-Pfad zu einem kleineren frühesten Ankunftszeitpunkt beim nächsten Kunden führt, der Ladestand des zweiten Depot-Pfads zu diesem Zeitpunkt nicht größer sein kann. Mit dieser Aussage kann später gezeigt werden, dass eine gebrochene Gerade, die von Algorithmus 2 berechnet wird, keine vertikalen Sprünge besitzt.

Proposition 8. *Seien $(\underline{t}_k, \underline{\beta}_k)$ und $(\underline{t}_{k'}, \underline{\beta}_{k'})$ zwei untere temporäre Zeit-Batterie-Paare für Kunde u_j , die von Algorithmus 2 in den Zeilen 27 bis 29 berechnet werden. Falls $\underline{t}_k < \underline{t}_{k'}$ gilt, folgt $\underline{\beta}_k \geq \underline{\beta}_{k'}$.*

Beweis. Der Beweis erfolgt durch Widerspruch, indem angenommen wird, dass $\underline{\beta}_k < \underline{\beta}_{k'}$ gilt.

Da $(\underline{t}_k, \underline{\beta}_k)$ und $(\underline{t}_{k'}, \underline{\beta}_{k'})$ Zeit-Batterie-Paare sind, existiert für beide ein Zertifikat ihrer Lösung. Sei π (π' entsprechend) die Lösung, die das Zertifikat für $(\underline{t}_k, \underline{\beta}_k)$ ($(\underline{t}_{k'}, \underline{\beta}_{k'})$ entsprechend) ist. Sei β (β' entsprechend) der Ladestand bei u_i aus der Lösung π (π' entsprechend). Zur Vereinfachung

wird $pq \in \pi$ ($pq \in \pi'$ entsprechend) geschrieben, wenn die Kante $\{p, q\}$ auf dem Pfad der Lösung π (π' entsprechend) liegt.

Zunächst wird folgende Beobachtung festgehalten. Aus den Zeilen 10 bis 30 geht hervor, dass im Algorithmus 2 beziehungsweise in der Lösung π (π' entsprechend) an den Depots nur geladen wird, falls

- (i) der Ladestand zu gering ist, um das nächste Depot zu erreichen,
- (ii) der Ladestand zu gering ist, um Kunden u_j mit dem Ladestand d_j zu erreichen oder
- (iii) das Fahrzeug am letzten Depot vor u_j wartet, um dann zum Zeitpunkt a_j anzukommen.

Es gilt $\underline{t}_k < \underline{t}_{k'}$ und $a_j \leq \underline{t}_k$. Daraus folgt, dass $a_j < \underline{t}_{k'}$ gilt. Daher kann Fall (iii) für Lösung π' nicht zutreffen. Gilt Fall (ii), ergibt sich $\underline{\beta}_{k'} = d_j \leq \underline{\beta}_k$. Dies ist ein Widerspruch zur Annahme, dass $\underline{\beta}_k < \underline{\beta}_{k'}$ gilt. Damit trifft Fall (ii) nicht zu. Somit folgt, dass das Depot $u_{k'}$ mit einem Ladestand von $d_j + d_{kj'} > 0$ erreicht wird. Daher kann Fall (i) auf dem gesamten Depot-Pfad zwischen u_i und u_j nicht eintreten, und es entstehen keine Ladezeiten an Depots in π' . Es gilt daher

$$\underline{\beta}_{k'} = \beta' - \sum_{pq \in DP'} d_{pq} \text{ und } \underline{t}_{k'} = t' + \sum_{pq \in DP'} d_{pq},$$

wobei t' den Abfahrtszeitpunkt von Kunde u_i in π' beschreibt und DP (DP' entsprechend) die Menge aller Kanten zwischen u_i und u_j in Lösung π (π' entsprechend) darstellt.

Insbesondere gilt $\beta' > \sum_{pq \in DP'} d_{pq}$ und somit folgt aus den Zeilen 3 bis 7, dass $t' = \underline{t}_i^1$ und $\beta' = \underline{\beta}_i^1$ gilt.

Damit folgt, dass

$$\underline{t}_i^1 + \sum_{pq \in DP'} d_{pq} = \underline{t}_{k'} > \underline{t}_k \geq t + \sum_{pq \in DP} d_{pq} \geq \underline{t}_i^1 + \sum_{pq \in DP} d_{pq}$$

gilt, und es folgt $\sum_{pq \in DP'} d_{pq} > \sum_{pq \in DP} d_{pq}$. Folglich gilt, dass

$$\underline{\beta}_k \geq \beta - \sum_{pq \in DP} d_{pq} > \underline{\beta}_i^1 - \sum_{pq \in DP'} d_{pq} = \underline{\beta}_{k'}.$$

Der Beweis ist damit abgeschlossen. □

Die folgende Proposition zeigt, dass sich das untere gebrochene Paar als konvexe Kombination aus dem unterem und dem oberem temporären Paar darstellen lässt. Außerdem gilt, dass das obere gebrochene Paar gleich dem oberen temporären Paar ist.

Proposition 9. Für alle Paare $(\underline{t}_j^\rho, \underline{\beta}_j^\rho)$ und $(\bar{t}_j^\rho, \bar{\beta}_j^\rho)$, $\rho = 1, \dots, |D'|$, die in den Zeilen 34, 38 und 39 berechnet werden, gilt $(\underline{t}_j^\rho, \underline{\beta}_j^\rho) = \lambda(\underline{t}_k, \underline{\beta}_k) + (1 - \lambda)(\bar{t}_k, \bar{\beta}_k)$ für ein $\lambda \in [0, 1]$ und $(\bar{t}_j^\rho, \bar{\beta}_j^\rho) = (\bar{t}_k, \bar{\beta}_k)$.

Beweis. Für $\rho = 1$ ergibt sich die Bedingung direkt aus der Zeile 34. Für $\rho > 1$ ergibt sich nach Zeile 39, dass $(\bar{t}_j^\rho, \bar{\beta}_j^\rho) = (\bar{t}_k, \bar{\beta}_k)$ gilt. Gemäß Zeile 38 gilt $(\underline{t}_j^\rho, \underline{\beta}_j^\rho) = (\underline{t}_k + 1/\alpha(\beta - \underline{\beta}_k), \beta)$, wobei β den größten Ladestand der vorherigen Iteration oder der Initialisierung beschreibt. Es bleibt zu zeigen, dass $\beta \in [\underline{\beta}_k, \bar{\beta}_k]$ gilt.

Sei $\tilde{k} \in D$ das Depot, das in der vorherigen Iteration oder in der Initialisierung betrachtet wurde. Auf Grundlage der Sortierung in Zeile 31 folgt, dass $\underline{t}_{\tilde{k}} \leq \underline{t}_k$ gilt. Gemäß Proposition 8 gilt $\underline{\beta}_k \leq \underline{\beta}_{\tilde{k}}$ und damit $\underline{\beta}_k \leq \underline{\beta}_{\tilde{k}} = \bar{\beta}_j^{\rho-1}$. Gemäß Zeile 37 gilt $\bar{\beta}_k > \beta$. $\square$

Mit der folgenden Proposition wird gezeigt, dass Algorithmus 2 gebrochene Geraden berechnet.

Proposition 10. *Die in den Zeilen 34 bis 43 von Algorithmus 2 berechneten Paare $(\underline{t}_j^\rho, \underline{\beta}_j^\rho)$ und $(\bar{t}_j^\rho, \bar{\beta}_j^\rho)$ für $\rho = 1, \dots, |D'|$ sind die oberen und unteren gebrochenen Paare der gebrochenen Geraden für u_j .*

Beweis. Gemäß Beobachtung 6 gilt $\underline{t}_k \leq \bar{t}_k$ und $\underline{\beta}_k \leq \bar{\beta}_k$. In Kombination mit Proposition 9 ist Bedingung 4.16 der Definition der gebrochenen Geraden erfüllt.

Sei u_k das Depot, von dem Abschnitt ρ der gebrochenen Geraden initialisiert wird, und sei $u_{\tilde{k}}$ das Depot, das Abschnitt $\rho + 1$ initialisiert.

Der zweite Teil von Bedingung (4.17) ist erfüllt, da

$$\bar{\beta}_j^\rho \underset{\text{(Zeile 39)}}{=} \bar{\beta}_k \underset{\text{(Zeile 40)}}{=} \beta \underset{\text{(Zeile 38)}}{=} \underline{\beta}_j^{\rho+1}$$

gilt.

Gemäß der Sortierung aus Zeile 31 gilt einer der drei folgenden Fälle:

- (i) $\underline{t}_k < \underline{t}_{\tilde{k}}$,
- (ii) $\underline{t}_k = \underline{t}_{\tilde{k}}$ und $\underline{\beta}_k > \underline{\beta}_{\tilde{k}}$,
- (iii) $\underline{t}_k = \underline{t}_{\tilde{k}}$, $\underline{\beta}_k = \underline{\beta}_{\tilde{k}}$, und $\bar{\beta}_k \geq \bar{\beta}_{\tilde{k}}$.

Im Fall (i) gilt

$$\underline{t}_j^{\rho+1} \underset{\text{(Zeile 38)}}{=} \underline{t}_{\tilde{k}} + \frac{1}{\alpha}(\bar{\beta}_k - \underline{\beta}_{\tilde{k}}) \underset{\text{(i)}}{>} \underline{t}_k + \frac{1}{\alpha}(\bar{\beta}_k - \underline{\beta}_{\tilde{k}}) \underset{\text{(Prop. 8)}}{\geq} \underline{t}_k + \frac{1}{\alpha}(\bar{\beta}_k - \underline{\beta}_k) \underset{\text{(Prop. 7)}}{=} \bar{t}_k \underset{\text{(Zeile 39)}}{=} \bar{t}_j^\rho.$$

Damit ist Bedingung (4.17) erfüllt.

Im Fall (ii) gilt

$$\underline{t}_j^{\rho+1} \underset{\text{(Zeile 38)}}{=} \underline{t}_{\tilde{k}} + \frac{1}{\alpha}(\bar{\beta}_k - \underline{\beta}_{\tilde{k}}) \underset{\text{(ii)}}{>} \underline{t}_k + \frac{1}{\alpha}(\bar{\beta}_k - \underline{\beta}_k) \underset{\text{(Prop. 7)}}{=} \bar{t}_k \underset{\text{(Zeile 39)}}{=} \bar{t}_j^\rho.$$

Damit ist Bedingung (4.17) erfüllt.

Der Fall (iii) führt mit den Zeilen 37 und 40 zu einem Widerspruch.

Nun wird gezeigt, dass die Paare $\lambda(\underline{t}_j^\rho, \underline{\beta}_j^\rho) + (1 - \lambda)(\bar{t}_j^\rho, \bar{\beta}_j^\rho)$, für ein $\lambda \in [0, 1]$ und $\rho = 1, \dots, |D'|$, Zeit-Batterie-Paare sind. Für $\rho = 1$ gilt $(\underline{t}_j^1, \underline{\beta}_j^1) = (\underline{t}_1, \underline{\beta}_1)$ und $(\bar{t}_j^1, \bar{\beta}_j^1) = (\bar{t}_1, \bar{\beta}_1)$ (siehe Zeile 34). Nach Proposition 7 folgt die Behauptung. Für $\rho > 1$ gilt $(\underline{t}_j^\rho, \underline{\beta}_j^\rho) = (\underline{t}_k + \frac{1}{\alpha}(\beta - \underline{\beta}_k), \beta)$ (siehe Zeile 38) und $(\bar{t}_j^\rho, \bar{\beta}_j^\rho) = (\bar{t}_k, \bar{\beta}_k)$ (siehe Zeile 39). Gemäß Proposition 9 gilt $(\underline{t}_j^\rho, \underline{\beta}_j^\rho) = \lambda(\underline{t}_k, \underline{\beta}_k) + (1 - \lambda)(\bar{t}_k, \bar{\beta}_k)$ für ein $\lambda \in [0, 1]$, und $(\bar{t}_j^\rho, \bar{\beta}_j^\rho) = (\bar{t}_k, \bar{\beta}_k)$. Zusammen mit Proposition 7 folgt die Aussage.

Der Funktionsterm der gebrochenen Geraden wurde definiert als

$\beta : [\underline{t}^1, \bar{t}^1] \cup \bigcup_{\rho=2}^\sigma [\underline{t}^\rho, \bar{t}^\rho] \rightarrow [\underline{\beta}^1, \bar{\beta}^\sigma]$ mit

$$\beta(t) = \begin{cases} \underline{\beta}^1 + \alpha(t - \underline{t}^1), & \text{für alle } t \in [\underline{t}^1, \bar{t}^1], \\ \underline{\beta}^\rho + \alpha(t - \underline{t}^\rho), & \text{für alle } t \in [\underline{t}^\rho, \bar{t}^\rho] \text{ und } \rho \in \{2, \dots, \sigma\}, \end{cases}$$

wobei $\sigma = |D'|$ gilt. Es ist anzumerken, dass die Abschnitte der gebrochenen Geraden jeweils in der konvexen Hülle der Paare $(\underline{t}_j^\rho, \underline{\beta}_j^\rho)$ und $(\bar{t}_j^\rho, \bar{\beta}_j^\rho)$ für $\rho = 1, \dots, |D'|$ liegen.

Damit ist der Funktionsterm zulässig, da gemäß Proposition 9 gilt, dass alle Paare der konvexen Hülle, bestehend aus den unteren und oberen gebrochenen Paaren, Zeit-Batterie-Paare sind. $\square$

Damit kann Lemma 1 durch Induktion bewiesen werden.

Lemma 1. *Sei s eine Sequenz, die Algorithmus 1 als zulässig bewertet. Dann gilt, dass Algorithmus 1 zusammen mit Algorithmus 2 an jedem Kunden aus s eine gebrochene Gerade, wie in Definition 11 formuliert, berechnet.*

Beweis. Der Beweis erfolgt durch Induktion.

Induktionsanfang: Die gebrochene Gerade am ersten Kunden ist durch $(\underline{t}_1^1, \underline{\beta}_1^1), (\bar{t}_1^1, \bar{\beta}_1^1) = (\max(a_1, d_1), B - d_1)$ definiert (siehe Zeile 2 von Algorithmus 1). Gemäß Proposition 3 sind diese Paare Zeit-Batterie-Paare. Es ist anzumerken, dass bereits ein einzelnes Zeit-Batterie-Paar die Definition der gebrochenen Geraden erfüllt.

Induktionsschritt: Seien u_i und u_j zwei aufeinanderfolgende Kunden in s . Angenommen, Algorithmus 2 bewertet die Erweiterung von u_i nach u_j als zulässig. Gemäß Proposition 10 berechnet Algorithmus 2 eine gebrochene Gerade für u_j , sofern die Eingabe eine gebrochene Gerade für u_i enthält.

Daraus folgt, dass die Algorithmen 1 und 2 an jedem Kunden einer Sequenz eine gebrochene Gerade berechnen, sofern die Sequenz der Eingabe als zulässig bewertet wird. $\square$

Basierend auf dem vorangegangenen Lemma kann Lemma 2 bewiesen werden.

Lemma 2. *Falls Algorithmus 1 eine Sequenz s als zulässig bewertet, dann existiert eine zulässige Lösung.*

Beweis. Gemäß Lemma 1 wird an jedem Kunden der Sequenz eine gebrochene Gerade von Algorithmus 1 und 2 berechnet, sofern die Sequenz als zulässig bewertet wird. Folglich wird am letzten Kunden mindestens ein Zeit-Batterie-Paar berechnet. Daraus lässt sich ableiten, dass eine zulässige Lösung als Zertifikat des Paares für den letzten Kunden existiert. Gemäß der Definition 6 muss der Ladestand am Kunden mindestens ausreichen, um das nächste Depot zu erreichen. Somit kann eine zulässige Lösung am letzten Kunden zu einer zulässigen Lösung zum nächsten Depot erweitert werden. $\square$

Damit ist gezeigt, dass eine zulässige Lösung existiert, falls Algorithmus 1 zulässig zurückgibt. Im Folgenden wird gezeigt, dass die Algorithmen 1 und 2 Zeit-Batterie-Kurven an den Kunden berechnen. Dazu wird zuerst gezeigt, dass am ersten Kunden der Sequenz eine Zeit-Batterie-Kurve berechnet wird.

Proposition 1. *Gegeben ist eine Sequenz s für die eine zulässige Lösung existiert. Algorithmus 1 berechnet eine Zeit-Batterie-Kurve am ersten Kunden aus s .*

Beweis. Es wird angenommen, dass das Fahrzeug zum Zeitpunkt null mit einem Ladestand von B von einem beliebigen Depot startet. Algorithmus 1 berechnet das früheste Zeit-Batterie-Paar am ersten Kunden vom Depot aus, das dem ersten Kunden am nächsten liegt (siehe Zeile 2). Daher wird am ersten Kunden von s eine Zeit-Batterie-Kurve berechnet, da es nicht möglich ist, den Kunden früher oder mit einem größeren Ladestand zu beliefern. $\square$

Es bleibt zu zeigen, dass Algorithmen 1 und 2 an jedem Kunden eine Zeit-Batterie-Kurve berechnen. Dazu wird im Folgenden gezeigt, dass Algorithmus 2 an den Depots dominante Zeit-Batterie-Paare berechnet, falls in der Eingabe des Algorithmus eine Zeit-Batterie-Kurve übergeben wurde.

Lemma 4. *Gegeben sind eine Sequenz s , zwei aufeinanderfolgende Kunden u_i, u_j aus s und eine zulässige Lösung π zu s . Sei u_k ein Depot, welches in π zwischen u_i und u_j besucht wird und sei (t_k, b_k) das Zeit-Batterie-Paar für u_k in π . Falls Algorithmus 1 oder 2 eine Zeit-Batterie-Kurve für u_i berechnet, gilt, dass Algorithmus 2 ein Zeit-Batterie-Paar für u_k berechnet, das (t_k, b_k) dominiert.*

Beweis. Induktionsanfang: Angenommen, u_k ist das erste Depot nach Kunde u_i in Lösung π . Sei (t_i, β_i) das Zeit-Batterie-Paar für u_i und sei (t_k, β_k) das Zeit-Batterie-Paar für u_k in Lösung π .

Angenommen, Algorithmus 1 oder 2 berechnet eine Zeit-Batterie-Kurve für u_i . Gemäß Lemma 1 ist diese Zeit-Batterie-Kurve dann eine gebrochene Gerade. Somit existiert für jedes Zeit-Batterie-Paar (t_i, β_i) ein Zeit-Batterie-Paar $(\tilde{t}_i, \tilde{\beta}_i)$ auf der Zeit-Batterie-Kurve für u_i , das (t_i, β_i) dominiert. Dies ist gleichbedeutend mit $\tilde{t}_i \leq t_i$ und $\tilde{\beta}_i \geq \beta_i$. Es sei angenommen, dass Algorithmus 2 das Zeit-Batterie-Paar (t_i^*, β_i^*) für u_i berechnet, um das Depot u_k zu erreichen. Damit

gilt, dass $t_i^* \leq \tilde{t}_i \leq t_i$, da Algorithmus 2 den frühesten Zeitpunkt wählt (siehe Zeile 4). Zudem gilt nach Beobachtung 1, dass $\tilde{\beta}_i \leq \beta_i^* + \alpha(\tilde{t}_i - t_i^*)$.

Sei (t_k^*, β_k^*) das Zeit-Batterie-Paar für u_k , das von Algorithmus 2 berechnet wird. Es gilt

$$\begin{aligned} t_k &\geq t_i + d_{ik} \stackrel{(\text{Def. 8})}{\geq} \tilde{t}_i + d_{ik} \geq t_i^* + d_{ik} \stackrel{(\text{Zeile 6})}{=} t_k^* \text{ und} \\ \beta_k &= \beta_i - d_{ik} \stackrel{(\text{Def. 8})}{\leq} \tilde{\beta}_i - d_{ik} \stackrel{(\text{Beob. 1})}{\leq} \beta_i^* + \alpha(\tilde{t}_i - t_i^*) - d_{ik} \\ &\stackrel{(\text{Def. 8})}{\leq} \beta_k^* + \alpha(t_i - t_i^* + d_{ik} - d_{ik}) = \beta_k^* + \alpha(t_k^* - t_k). \end{aligned}$$

Damit gilt, dass das Zeit-Batterie-Paar (t_k^*, β_k^*) das Zeit-Batterie-Paar (t_k, β_k) dominiert.

Induktionsschritt: Gegeben sind zwei aufeinanderfolgende Depots $u_{k-1}, u_k \in D$ zwischen u_i und u_j in Lösung π . Angenommen, Algorithmus 2 berechnet ein Zeit-Batterie-Paar $(t_{k-1}^*, \beta_{k-1}^*)$ für u_{k-1} , welches das Paar (t_{k-1}, β_{k-1}) aus der Lösung π für u_{k-1} dominiert. Es ist zu zeigen, dass Algorithmus 2 ein Zeit-Batterie-Paar (t_k^*, β_k^*) für u_k berechnet, das das Zeit-Batterie-Paar (t_k, β_k) von π für u_k dominiert. Der Beweis erfolgt durch Widerspruch.

Angenommen, es gilt, $t_k < t_k^*$

$$\begin{aligned} &\Leftrightarrow \stackrel{(\text{Zeilen 15-17})}{t_{k-1} + \lambda_{k-1} + d_{k-1,k} < t_k^* \leq t_{k-1}^* + \frac{1}{\alpha} \max(0, d_{k-1,k} - \beta_{k-1}^*) + d_{k-1,k}} \\ &\Leftrightarrow t_{k-1} + \lambda_{k-1} < t_{k-1}^* + \frac{1}{\alpha} \max(0, d_{k-1,k} - \beta_{k-1}^*) \end{aligned}$$

Das Maximum wird aufgelöst und die zwei folgenden Fälle werden unterschieden:

$$\begin{aligned} &\text{Fall (i) } d_{k-1,k} - \beta_{k-1}^* \leq 0 \\ &\Rightarrow t_{k-1} + \lambda_{k-1} < t_{k-1}^*. \end{aligned}$$

Dies ist ein Widerspruch zur Induktionsannahme.

$$\begin{aligned} &\text{Fall (ii) } d_{k-1,k} - \beta_{k-1}^* > 0 \\ &\Rightarrow t_{k-1} + \lambda_{k-1} < t_{k-1}^* + \frac{1}{\alpha} (d_{k-1,k} - \beta_{k-1}^*) \\ &\stackrel{(\text{Def. 9})}{\Rightarrow} t_{k-1} + \lambda_{k-1} < t_{k-1}^* + \frac{1}{\alpha} (d_{k-1,k} - \beta_{k-1} + \alpha(t_{k-1} - t_{k-1}^*)) \\ &\Rightarrow t_{k-1} + \lambda_{k-1} < t_{k-1}^* + \frac{1}{\alpha} (d_{k-1,k} - \beta_{k-1}) + t_{k-1} - t_{k-1}^* \\ &\Rightarrow \lambda_{k-1} < \frac{1}{\alpha} (d_{k-1,k} - \beta_{k-1}) \\ &\Rightarrow \frac{1}{\alpha} (d_{k-1,k} - \beta_{k-1}) \leq \frac{1}{\alpha} \max(0, d_{k-1,k} - \beta_{k-1}) \leq \beta_{k-1} < \frac{1}{\alpha} (d_{k-1,k} - \beta_{k-1}). \end{aligned}$$

Dies ist ein Widerspruch zur Induktionsannahme und es folgt $t_k^* \leq t_k$. Es bleibt zu zeigen, dass

$\beta_k^* \leq \beta_k + \alpha(t_k - t_k^*)$ gilt. Beweis durch Widerspruch:

$$\begin{aligned}
& \text{Angenommen, es gilt, } \beta_k > \beta_k^* + \alpha(t_k - t_k^*) \\
& \Leftrightarrow \quad \text{(Zeile 18)} \quad \beta_{k-1} + \alpha\lambda_{k-1} - d_{k-1,k} > \beta_{k-1}^* + \alpha\lambda_{k-1}^* - d_{k-1,k} + \alpha(t_k - t_k^*) \\
& \Rightarrow \quad \beta_{k-1} + \alpha\lambda_{k-1} > \beta_{k-1}^* + \alpha\lambda_{k-1}^* + \alpha(t_{k-1} + \lambda_{k-1} - t_{k-1}^* - \lambda_{k-1}^*) \\
& \Rightarrow \quad \beta_{k-1} > \beta_{k-1}^* + \alpha(t_{k-1} - t_{k-1}^*).
\end{aligned}$$

Dies ist ein Widerspruch zur Induktionsannahme. Damit ist der Induktionsschritt gezeigt, und der Beweis ist abgeschlossen. $\square$

In der folgenden Proposition wird gezeigt, dass sich die Eigenschaften eines temporären Paares auf das entsprechende gebrochene Paar übertragen lassen. Sei dazu $(\underline{t}_j^\rho, \underline{\beta}_j^\rho)$ ein unteres gebrochenes Paar und sei $(\bar{t}_j^\rho, \bar{\beta}_j^\rho)$ ein oberes gebrochenes Paar, das aus den jeweiligen temporären Paaren des Depots u_k erzeugt wird.

Proposition 11. *Die gebrochenen Paare $(\underline{t}_j^\rho, \underline{\beta}_j^\rho)$ und $(\bar{t}_j^\rho, \bar{\beta}_j^\rho)$ liegen auf der linearen Interpolation der temporären Paare $(\underline{t}_k, \underline{\beta}_k)$ und $(\bar{t}_k, \bar{\beta}_k)$. Zudem gilt $[\underline{t}_j^\rho, \bar{t}_j^\rho] \subseteq [\underline{t}_k, \bar{t}_k]$ und $[\underline{\beta}_j^\rho, \bar{\beta}_j^\rho] \subseteq [\underline{\beta}_k, \bar{\beta}_k]$.*

Beweis. Es wird zwischen den Fällen $\rho = 1$ und $\rho > 1$ unterschieden. Im ersten Fall sind temporäre und gebrochene Paare identisch (siehe Zeile 34). Damit folgt die Aussage. Im Fall $\rho > 1$ sind das obere temporäre Paar und das obere gebrochene Paar gleich (siehe Zeile 39). Für das untere gebrochene Paar gilt $(\underline{t}_j^\rho, \underline{\beta}_j^\rho) = (\underline{t}_k + 1/\alpha(\beta - \underline{\beta}_k), \beta)$, wobei β dem höchsten Ladestand des vorangegangenen Abschnitts auf der gebrochenen Geraden entspricht. Gemäß Proposition 11 folgt, dass die Steigung der linearen Interpolation der temporären Paare α ist, wodurch die Aussage folgt. Damit ist bewiesen, dass die gebrochenen Paare auf der linearen Interpolation liegen. Zudem wurde gezeigt, dass $\bar{t}_k = \bar{t}_j^\rho$ und $\bar{\beta}_k = \bar{\beta}_j^\rho$ gelten. Es bleibt zu zeigen, dass $\underline{t}_k \leq \underline{t}_j^\rho \leq \bar{t}_j^\rho$ und $\underline{\beta}_k \leq \underline{\beta}_j^\rho \leq \bar{\beta}_j^\rho$. Es gilt:

$$\underline{t}_j^\rho \stackrel{\text{(Zeile 38)}}{=} \underline{t}_k + \frac{1}{\alpha}(\beta - \underline{\beta}_k) \stackrel{\text{(Zeile 37)}}{<} \underline{t}_k + \frac{1}{\alpha}(\bar{\beta}_k - \underline{\beta}_k) = \bar{t}_k \stackrel{\text{(Zeile 39)}}{=} \bar{t}_j^\rho. \quad (2)$$

Damit die Ungleichung erfüllt ist, muss $\beta > \underline{\beta}_k$ gelten. Sei dazu $\tilde{u}_k \in D'$, das Depot, das den Abschnitt $\rho - 1$ initialisiert hat. Auf Grundlage der Sortierung der temporären Paare (siehe Zeile 31) folgt $\underline{t}_{\tilde{k}} \leq \underline{t}_k$. Auf Basis von Proposition 8 folgt, dass $\underline{\beta}_{\tilde{k}} \leq \underline{\beta}_k$ gilt. Damit folgt $\underline{\beta}_k \leq \bar{\beta}_{\tilde{k}} = \bar{\beta}_j^{\rho-1} = \beta$. Gemäß Zeile 37 gilt $\bar{\beta}_k > \beta$ und damit folgt die Aussage. $\square$

Für den Induktionsschritt bleibt es zu zeigen, dass Algorithmus 2 eine Zeit-Batterie-Kurve für u_j in der Erweiterung von u_i nach u_j berechnet. Hierfür werden die Propositionen 12 und 13 eingeführt. Diese Propositionen definieren eine untere Schranke für den frühesten Belieferungszeitpunkt eines Kunden und eine obere Schranke für den Ladestand bei Ankunft am Kunden.

Proposition 12. Sei (t_k, β_k) ein Zeit-Batterie-Paar am Depot $u_k \in D$ und sei u_i ein Kunde, der direkt auf u_k folgt. Dann ist der Ankunftszeitpunkt bei u_i nach unten beschränkt durch

$$t_i \geq \max(a_i, t_k + \frac{1}{\alpha} \max(0, d_{ki} + d_i - \beta_k) + d_{ki}).$$

Beweis. Der erste Teil des Maximums ist trivial, da der Roboter die Belieferung nur innerhalb des vorgegebenen Zeitfensters durchführen kann. Der zweite Teil beschreibt den frühesten Zeitpunkt nach der minimalen Aufladung am Depot. Der Ladestand beim Verlassen des Depots muss mindestens ausreichen, um die Fahrzeit zum Kunden und anschließend zum nächstgelegenen Depot abzudecken. Diese Ladezeit wird berechnet als $\frac{1}{\alpha} \max(0, d_{ki} + d_i - \beta_k)$. Damit gilt die untere Schranke. $\square$

Proposition 13. Sei (t_k, β_k) ein Zeit-Batterie-Paar am Depot $u_k \in D$. Sei u_i ein Kunde, der direkt nach u_k angefahren wird. Der Ladestand bei u_i ist nach oben begrenzt durch

$$\beta_i \leq \min(B - d_{ki}, \beta_k + \alpha(t_i - t_k - d_{ki}) - d_{ki}).$$

Beweis. Der erste Teil des Minimums ist trivial, da der Ladestand am Depot maximal B betragen kann. Die Ladezeit für eine partielle Nachladung bei u_k kann ausgedrückt werden als $\lambda_k = t_i - t_k - d_{ki}$, wobei t_i den Belieferungszeitpunkt bei u_i beschreibt. Daraus ergibt sich der Ladestand bei u_i als $\beta_k + \alpha(t_i - t_k - d_{ki}) - d_{ki}$, was den zweiten Teil des Minimums beschreibt. $\square$

Lemma 5. Gegeben sind eine Sequenz s und eine zulässige Lösung π zu s . Seien u_i und u_j zwei aufeinanderfolgende Kunden in s . Falls Algorithmus 1 oder 2 eine Zeit-Batterie-Kurve für u_i berechnet, dann gilt, dass Algorithmus 2 eine Zeit-Batterie-Kurve für u_j berechnet.

Beweis. Sei $u_k \in D$ das letzte Depot in π vor Kunde u_j , und seien (t_k, β_k) bzw. (t_j, β_j) die Zeit-Batterie-Paare von π für u_k und u_j . Es wird gezeigt, dass auf der gebrochenen Geraden für u_j ein Paar (t_j^*, β_j^*) liegt, das vom Algorithmus 2 berechnet wurde und (t_j, β_j) dominiert. Zunächst wird gezeigt, dass $t_j \geq \underline{t}_k$ und $\beta_j \leq \bar{\beta}_k$ gilt, wobei $\underline{t}_k$ der Zeit des unteren temporären Paares für u_j entspricht (siehe Zeile 27) und $\bar{\beta}_k$ dem Ladestand des oberen temporären Paares für u_j (siehe Zeile 30). Es gilt zu zeigen, dass $t_j \geq \underline{t}_k$. Es gilt:

$$\begin{aligned} t_j &\stackrel{(\text{Prop. 12})}{\geq} \max(b_j, t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj}) \\ &\stackrel{(\text{Def. 9})}{\geq} \max(b_j, t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k^* - \alpha(t_k - t_k^*)) + d_{kj}) \\ &\geq \max(b_j, t_k^* + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k^*) + d_{kj}) \stackrel{(\text{Zeile 27})}{=} \underline{t}_k. \end{aligned}$$

Es bleibt zu zeigen, dass $\beta_j \leq \bar{\beta}_k$ gilt. Es gilt:

$$\begin{aligned}
\beta_j &\stackrel{(\text{Prop. 13})}{\leq} \min(B - d_{kj}, \beta_k + \alpha(t_j - t_k - d_{kj}) - d_{kj}) \\
&\stackrel{(\text{Def. 9})}{\leq} \min(B - d_{kj}, \beta_k^* + \alpha(t_k - t_k^*) + \alpha(t_j - t_k - d_{kj}) - d_{kj}) \\
&\leq \min(B - d_{kj}, \beta_k^* + \alpha(b_j - t_k^* - d_{kj}) - d_{kj}) \stackrel{(\text{Zeile 30})}{=} \bar{\beta}_k.
\end{aligned}$$

Es werden zwei Fälle unterschieden: Entweder gilt $\beta_j < \underline{\beta}_k$ oder $\beta_j \in [\underline{\beta}_k, \bar{\beta}_k]$. Im ersten Fall folgt, dass $(\underline{t}_k, \underline{\beta}_k)$ (t_j, β_j) dominiert, da $t_j \geq \underline{t}_k$ und $\beta_j < \underline{\beta}_k$. Im zweiten Fall gilt:

$$\begin{aligned}
t_j &\geq \max(a_j, t_k + \frac{1}{\alpha}(d_{kj} + \beta_j - \beta_k) + d_{kj}) \\
&\stackrel{(\text{Def.9})}{\geq} \max(a_j, t_k + \frac{1}{\alpha}(d_{kj} + \beta_j - \beta_k^* - \alpha(t_k - t_k^*)) + d_{kj}) \\
&= \max(a_j, t_k^* + \frac{1}{\alpha}(d_{kj} + \beta_j - \beta_k^*) + d_{kj}) \stackrel{(\diamond)}{=} t_j^*.
\end{aligned}$$

Gleichung $(\diamond)$ gilt, da β_j in dem Intervall $[\underline{\beta}_j, \bar{\beta}_j]$ liegt und die Steigung der linearen Interpolation zwischen dem unteren und oberen temporären Paar gleich α ist (siehe Proposition 7). Daher wird (t_j, β_j) von (t_j^*, β_j^*) dominiert.

Es ist zu beachten, dass das Zeit-Batterie-Paar (t_j^*, β_j^*) nicht zwingend auf der gebrochenen Geraden von u_j liegen muss, da es durch ein anderes Zeit-Batterie-Paar (t_j^{**}, β_j^{**}) auf der gebrochenen Geraden dominiert werden kann. In diesem Fall erfüllt (t_j^{**}, β_j^{**}) die geforderte Bedingung. $\square$

Damit kann gezeigt werden, dass keine zulässige Lösung existiert, falls Algorithmus 1 unzulässig zurückgibt.

Lemma 3. *Falls Algorithmus 1 eine Sequenz s als unzulässig bewertet, dann existiert keine zulässige Lösung für s .*

Beweis. Angenommen, es existiert eine zulässige Lösung π für eine Sequenz s . Dennoch bewertet Algorithmus 1 die Sequenz s als unzulässig. Dieser Fall kann entweder in Algorithmus 1 oder Algorithmus 2 auftreten.

Angenommen, Algorithmus 1 gibt als Ergebnis *unzulässig* zurück. Dies bedeutet, dass der erste Kunde u_1 in der Sequenz s nicht erreichbar ist (siehe Zeile 4). Sei $(\underline{t}_1^*, \underline{\beta}_1^*)$ das von Algorithmus 1 berechnete Paar und sei (t_1, β_1) das Zeit-Batterie-Paar der Lösung π . Gemäß Proposition 1 folgt entweder

$$b_1 \stackrel{(\text{Zeile 3})}{<} \underline{t}_1^* \stackrel{(\text{Def. 8})}{\leq} t_1 \text{ oder } B/2 \stackrel{(\text{Zeile 3})}{<} \underline{\beta}_1^* \stackrel{(\text{Def. 8})}{\leq} \beta_1.$$

Beide Fälle führen zu einem Widerspruch.

Angenommen, Algorithmus 2 gibt *unzulässig* in der Erweiterung von u_i nach u_j zurück. In diesem Fall muss D' leer sein (siehe Zeile 23). Sei u_k das letzte Depot vor u_j in der Lösung π . Sei (t_k, β_k)

das Zeit-Batterie-Paar für u_k in π , und sei (t_j, b_j) das Zeit-Batterie-Paar für u_j in π . Gemäß Lemma 5 gilt, dass Algorithmus 1 oder Algorithmus 2 für u_i eine Zeit-Batterie-Kurve berechnet. Sei (t_k^*, b_k^*) das von Algorithmus 2 berechnete Zeit-Batterie-Paar für u_k . Ein solches Zeit-Batterie-Paar muss existieren, da π zulässig ist. Gemäß Lemma 4 dominiert das Zeit-Batterie-Paar (t_k^*, b_k^*) das Paar (t_k, β_k) . Gemäß Zeile 22 gilt entweder (i) $b_j < t_k^* + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj}$ oder (ii) $d_{kj} + d_j > B$. Es wird gezeigt, dass beide Fälle in einem Widerspruch münden.

$$\begin{aligned}
\text{Fall (i): } b_j &\stackrel{\text{(Zeile 22)}}{<} t_k^* + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj} \\
&\stackrel{\text{(Def. 8)}}{\leq} t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj} \\
&\leq \max(a_j, t_k + \frac{1}{\alpha} \max(0, d_{kj} + d_j - \beta_k) + d_{kj}) \\
&\stackrel{\text{(Prop. 12)}}{\leq} t_j.
\end{aligned}$$

$$\text{Fall (ii): } \beta_j \stackrel{\text{(Prop. 13)}}{\leq} \min(B - d_{kj}, \beta_k + \alpha(t_j - t_k - d_{kj}) - d_{kj}) \leq B - d_{kj} < d_j.$$

Beide Fälle führen zu einem Widerspruch. Daher kann die Lösung π nicht zulässig sein. $\square$

Es bleibt zu zeigen, dass die Laufzeit von Algorithmus 1 polynomiell beschränkt ist.

Lemma 6. *Sei s eine Sequenz mit $|s|$ Kunden und sei $|D|$ die Anzahl der Depots. Die Laufzeit von Algorithmus 1 lässt sich asymptotisch abschätzen durch $O(|s| \cdot |D|^2)$.*

Beweis. Bevor die Laufzeit bewiesen wird, wird die maximale Anzahl an Abschnitten betrachtet, die eine gebrochene Gerade besitzen kann. Die gebrochene Gerade am ersten Kunden ist ein Punkt (siehe Zeile 2 von Algorithmus 1). Für alle anderen Kunden ist die Anzahl der Abschnitte durch die Anzahl der Depots in D' begrenzt (siehe Zeile 22). Dabei wird $|D'|$ durch $|D|$ begrenzt. Jedes Depot aus D' kann höchstens einen Abschnitt zur gebrochenen Geraden hinzufügen (siehe Zeilen 34, 38 und 39). Daher ist die Anzahl der Abschnitte einer gebrochenen Geraden durch $|D|$ begrenzt.

Algorithmus 1 ruft Algorithmus 2 $|s| - 1$ mal auf, wobei $|s|$ die Anzahl der Kunden in s angibt. Im Folgenden wird die Laufzeit von Algorithmus 2 abgeschätzt. Die Zeilen 6 bis 8, 26 bis 30 sowie 36 bis 43 haben eine Laufzeit von $O(|D|)$. Die Zeilen 10 bis 21 lassen sich mit $O(|D|^2)$ abschätzen, während die Zeile 31 mit $O(|D| \log |D|)$ abgeschätzt werden kann. Insgesamt lässt sich die Laufzeit von Algorithmus 1 als $O(|s| \cdot |D|^2)$ abschätzen. $\square$

Damit ist der Beweis abgeschlossen. Es folgt die Vorstellung des Algorithmus zur rückwärtsgerichteten Betrachtung des Entscheidungsproblems.

B Rückwärtsgerichtetes dynamisches Programm

Im Folgenden werden die erforderlichen Anpassungen zur Lösung des Entscheidungsproblems 1 mit einer rückwärtsgerichteten Betrachtung der Kundensequenz vorgestellt. Dabei müssen die Ressourcen in umgekehrter Weise berücksichtigt werden. Das bedeutet, dass bei einer Fahrt zwischen zwei Kunden die Zeit rückwärts läuft und der Ladestand steigt. Ladevorgänge am Depot führen zu einer Reduktion der Zeit und des Ladestandes. Es wird die Definition eines rückwärtsgerichteten Zeit-Batterie-Paars benötigt.

Definition 21. Sei s' eine Sequenz und sei π eine zulässige Lösung für s' . Angenommen, π startet am Knoten u_i zum Zeitpunkt t mit Ladestand β . Das Paar (t, β) wird als rückwärtsgerichtetes Zeit-Batterie-Paar für u_i in Bezug auf s' bezeichnet.

Zur Veranschaulichung der Ressourcen werden eine Sequenz $s = (u_1, \dots, u_i, \dots, u_h)$ und ein rückwärtsgerichtetes Zeit-Batterie-Paar (t, b) betrachtet und diese aus der vorwärtsgerichteten Perspektive beschrieben. Die Zeit entspricht dem spätesten Zeitpunkt, zu dem die Teilsequenz $(u_i, \dots, u_h)$ zulässig ausgeführt werden kann. Der Ladestand entspricht dem minimalen Wert, der für diese Ausführung erforderlich ist. Im Folgenden werden die nötigen Anpassungen von Algorithmus 1 beschrieben.

Der Pseudocode für die rückwärtsgerichtete Betrachtung einer Sequenz ist in Algorithmus 7 dargestellt. Bei der Initialisierung des letzten Kunden u_h (Zeile 1) wird das Zeit-Batterie-Paar auf $(\min(b_h, T - d_h), d_h)$ gesetzt, wobei T den Zeithorizont darstellt. Die Überprüfung auf Zulässigkeit in Zeile 2 muss ebenfalls angepasst werden. Es muss weiterhin getestet werden, ob der Kunde zu weit vom Depot entfernt liegt, und zusätzlich wird überprüft, ob der Roboter früher als die untere Zeitfenstergrenze ankommt. In der Schleife werden die Kunden in umgekehrter Reihenfolge betrachtet, und eine rückwärtsgerichtete Erweiterung wird mit Algorithmus 8 durchgeführt.

Algorithmus 7 Rückwärtsgerichteter Ansatz zur Lösung des Entscheidungsproblems

Eingabe: Kundensequenz $s = (u_1, \dots, u_h)$, Depots D

Ausgabe: s ist zulässig oder unzulässig

```

1:  $(\underline{t}_h^1, \underline{\beta}_h^1), (\bar{t}_h^1, \bar{\beta}_h^1) \leftarrow (\min(b_h, T - d_h), d_h)$ 
2: falls  $\underline{t}_h^1 < a_h$  oder  $\underline{\beta}_h^1 > \frac{B}{2}$  dann
3:   Rückgabe unzulässig
4: ende falls
5: für alle  $i = h$  bis 2 mache
6:   rufe Algorithmus 8 auf und erweitere von  $u_i$  zu  $u_{i-1}$ 
7:   falls Erweiterung unzulässig dann
8:     Rückgabe unzulässig
9:   ende falls
10: ende für alle
```

Die rückwärtsgerichtete Erweiterung zwischen zwei Kunden ist als Pseudocode in Algorithmus 8 dargestellt und ist weitestgehend gleich zu Algorithmus 2. Durch die Umkehrung der Betrachtung ändern sich jedoch die Berechnungen der Zeiten und Ladestände. Daher werden im Folgenden die Unterschiede aufgezeigt.

Algorithmus 8 Rückwärtsgerichtete Erweiterung von Kunde u_i zu Kunde u_j

Eingabe: Kunden $u_i, u_j \in s$, Depots D , gebrochene Gerade $(\underline{t}_i^\rho, \underline{\beta}_i^\rho), (\bar{t}_i^\rho, \bar{\beta}_i^\rho)$, mit $\rho = 1, \dots, \sigma$

```

1:  $t_k \leftarrow 0, \beta_k \leftarrow \infty, \forall u_k \in D$ 
2: # Berechne Zeit und Ladestand an Depots direkt von Kunde  $u_i$ 
3: für alle  $u_k \in D$  mit  $d_{ki} \leq B - \beta_i^1$  mache
4:    $\tilde{\rho} \leftarrow \max\{\rho : B - \beta_i^\rho \geq d_{ki}, \rho \in \{1, \dots, \sigma\}\}$ 
5:    $\lambda \leftarrow \frac{1}{\alpha} \min\{0, d_{ki} - (B - \bar{\beta}_i^{\tilde{\rho}})\}$ 
6:    $t_k \leftarrow \bar{t}_i^{\tilde{\rho}} - \lambda - d_{ki}$ 
7:    $\beta_k \leftarrow \bar{\beta}_i^{\tilde{\rho}} - \alpha\lambda + d_{ki}$ 
8: ende für alle
9: # Depot-zu-Depot-Evaluierung
10:  $\tilde{D} \leftarrow \emptyset$ 
11: während  $\tilde{D} \neq D$  mache
12:    $k \leftarrow \arg \max\{t_l : u_l \in D \setminus \tilde{D}\}$ 
13:    $\tilde{D} \leftarrow \tilde{D} \cup \{u_k\}$ 
14:   für alle  $u_l \in D \setminus \tilde{D}$  mit  $d_{lk} \leq B$  mache
15:      $\lambda \leftarrow \frac{1}{\alpha} \max(0, d_{lk} - (B - \beta_k))$ 
16:     falls  $t_k - \lambda - d_{lk} > t_l$  dann
17:        $t_l \leftarrow t_k - \lambda - d_{lk}$ 
18:        $\beta_l \leftarrow \beta_k - \alpha\lambda + d_{lk}$ 
19:     ende falls
20:   ende für alle
21: ende während
22:  $D' \leftarrow \{u_k \in D : t_k - \frac{1}{\alpha} \max(0, d_{kj} + d_j - (B - \beta_k)) - d_{kj} - \tau_j \geq a_j, d_{kj} + d_j \leq B\}$ 
23: falls  $D'$  leer ist dann
24:   Rückgabe Erweiterung unzulässig
25: ende falls
26: # Berechnung unterer und oberer rückwärtsgerichteter Zeit-Batterie-Paare für Kunde  $u_j$ 
27:  $\bar{t}_k \leftarrow \min(b_j, t_k - \frac{1}{\alpha} \max(0, d_{kj} + d_j - (B - \beta_k)) - d_{kj} - \tau_j)$  für alle  $u_k \in D'$ 
28:  $\underline{t}_k \leftarrow \min(t_k, \max(a_j, t_k - \frac{1}{\alpha} \beta_k - d_{kj} - \tau_j))$  für alle  $u_k \in D'$ 
29:  $\bar{\beta}_k \leftarrow \max(d_{kj}, \beta_k - \alpha(t_k - d_{kj} - \tau_j - \bar{t}_k) + d_{kj})$  für alle  $u_k \in D'$ 
30:  $\underline{\beta}_k \leftarrow \max(d_{kj}, \beta_k - \alpha(t_k - d_{kj} - \tau_j - a_j) + d_{kj})$  für alle  $u_k \in D'$ 
31: Sortiere Depots absteigend nach  $\bar{t}_k$ , bei Gleichheit aufsteigend nach  $\bar{\beta}_k$ , bei Gleichstand
    aufsteigend nach  $\underline{\beta}_k$ 
32: Ersetze Indizes der Depots in  $D'$  basierend auf Sortierung von 1 bis  $|D'|$ 
33: # Aufbau der Zeit-Batterie-Kurve für Kunde  $u_j$ 
34:  $(\bar{t}_j^1, \bar{\beta}_j^1) \leftarrow (\bar{t}_1, \bar{\beta}_1), (\underline{t}_j^1, \underline{\beta}_j^1) \leftarrow (\underline{t}_1, \underline{\beta}_1)$ 
35:  $\beta \leftarrow \underline{\beta}_1, \rho \leftarrow 2$ 
36: für alle  $k = 2$  bis  $|D'|$  mache
37:   falls  $\underline{\beta}_k < \beta$  dann
38:      $(\bar{t}_j^\rho, \bar{\beta}_j^\rho) \leftarrow (\bar{t}_k - \frac{1}{\alpha}(\bar{\beta}_k - \beta), \beta)$ 
39:      $(\underline{t}_j^\rho, \underline{\beta}_j^\rho) \leftarrow (\underline{t}_k, \underline{\beta}_k)$ 
40:      $\beta \leftarrow \underline{\beta}_k$ 
41:      $\rho \leftarrow \rho + 1$ 
42:   ende falls
43: ende für alle
44: Rückgabe Gebrochene Gerade für Kunde  $u_j$ 

```

Der Algorithmus erweitert die Lösung vom Kunden u_i zum Kunden u_j . Auf der gebrochenen Geraden wird der späteste Zeitpunkt gewählt, sodass das Depot erreicht werden kann (siehe Zeile 4). Zur Berechnung der Belieferungszeit werden von der aktuellen Zeit die Ladezeit und die Fahrzeit abgezogen (siehe Zeile 6). Zur Berechnung des rückwärtsgerichteten Zeit-Batterie-Paares am Depot wird das späteste zulässige Paar auf der gebrochenen Geraden gewählt. Von der Zeit wird die Fahrzeit zum Depot abgezogen und für den Ladestand muss die Fahrzeit addiert werden (siehe Zeilen 6 und 7).

Bei der Berechnung der Depot-Pfade wird immer das Depot gewählt, das noch nicht berücksichtigt wurde und aktuell die späteste Ankunftszeit aufweist (siehe Zeile 12).

Für die Berechnung der reduzierten Depot-Menge D' gilt, dass ein Depot enthalten ist, wenn der späteste Belieferungszeitpunkt bei u_j vor seiner unteren Zeitfenstergrenze liegt und der Fahrweg zum Kunden sowie die Fahrt vom Kunden zu seinem nächstgelegenen Depot möglich sind (siehe Zeile 22).

Der obere Ankunftszeitpunkt ergibt sich dabei als Minimum aus der oberen Zeitfenstergrenze von u_j und dem spätesten Ankunftszeitpunkt bei u_j (siehe Zeile 27). Der untere Ankunftszeitpunkt ergibt sich als Minimum aus dem oberen Ankunftszeitpunkt und dem Maximum aus der unteren Zeitfenstergrenze von u_j und dem Ankunftszeitpunkt nach einer vollständigen Entladung am Depot (siehe Zeile 28). Für die Ladestände gilt, dass diese sich als Maximum aus einer kompletten Entladung am Depot und dem Ladestand entsprechend des jeweiligen Ankunftszeitpunkts berechnen (siehe Zeile 29 und 30).

Die Sortierung der Depots erfolgt in Zeile 31. Zuerst wird in absteigender Reihenfolge nach dem oberen Belieferungszeitpunkt bei u_j sortiert und bei Gleichheit erfolgt die Sortierung in aufsteigender Reihenfolge nach dem oberen Ladestand. Sollte weiterhin Gleichheit bestehen, wird in aufsteigender Reihenfolge nach dem Ladestand sortiert. Besteht erneut Gleichheit, erfolgt die Sortierung nach IDs.

Der Aufbau der gebrochenen Kurve für u_j ähnelt der vorwärtsgerichteten Variante. Allerdings werden zuerst die spätesten Abschnitte hinzugefügt, und ein neuer Abschnitt wird nur dann aufgenommen, wenn dieser einen geringeren Ladestand aufweist als der aktuell tiefste. Zusätzlich wird das obere temporäre Paar angepasst. Damit ist die Beschreibung der rückwärtsgerichteten Variante des dynamischen Programms abgeschlossen. Es folgen die Korrektheitsbeweise für die Algorithmen aus dem Kapitel 5.

C Korrektheitsbeweise der Algorithmen 5 und 6

Im Folgenden wird die Korrektheit der Algorithmen 5 und 6 bewiesen. Zunächst werden die Lemmata 9 und 10 bewiesen. Diese stellen einen Zusammenhang zwischen der Geschwindigkeitsänderung von zwei aufeinanderfolgenden Kunden und dem Belieferungszeitpunkt des ersten Kunden her.

Lemma 9. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $\mathbf{v} = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt beliefert und dabei bezüglich f kostenminimal ist. Enthält die Sequenz zwei aufeinanderfolgende Kunden u_i und u_{i+1} , für die $v_i > v_{i+1}$ gilt, dann gilt $t_i = b_i$.*

Beweis. Es wird ein Widerspruchsbeweis geführt mit der Annahme, dass $t_i < b_i$ gilt. Wähle $\tilde{d}_i, \tilde{d}_{i+1}, \bar{d}_i, \bar{d}_{i+1}$ wie in Lemma 8, sowie ein δ mit $0 < \delta < \mathbf{v}_i - \mathbf{v}_{i+1}$ und $t_{i-1} + \frac{\bar{d}_i}{\mathbf{v}_i} + \frac{\tilde{d}_i}{\mathbf{v}_i - \delta} \leq b_i$. Die Kosten für die beiden Kunden können reduziert werden, indem die Geschwindigkeit auf $\tilde{d}_i$ um δ gesenkt wird und die Geschwindigkeit auf $\tilde{d}_{i+1}$ um δ erhöht wird (siehe Lemma 8). Es ist zu beachten, dass durch die Veränderung der Geschwindigkeiten die gesamte Fahrzeit reduziert wird (siehe Beweis von Lemma 8). Daher bleibt die restliche Sequenz unverändert. Die Reduzierung der Kosten widerspricht jedoch der Annahme, dass der Vektor $\mathbf{v}$ kostenminimal ist. $\square$

Lemma 10. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $\mathbf{v} = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt beliefert und dabei bezüglich f kostenminimal ist. Enthält die Sequenz zwei aufeinanderfolgende Kunden u_i und u_{i+1} , für die $v_i < v_{i+1}$ gilt, dann gilt $t_i = a_i$.*

Beweis. Auch für dieses Lemma wird ein Widerspruchsbeweis geführt. Dazu wird angenommen, dass $t_i > a_i$ gilt. Wähle $\tilde{d}_i, \tilde{d}_{i+1}, \bar{d}_i, \bar{d}_{i+1}$, wie in Lemma 8, sowie ein δ mit $0 < \delta < \mathbf{v}_{i+1} - \mathbf{v}_i$ und $t_{i-1} + \frac{\bar{d}_i}{\mathbf{v}_i} + \frac{\tilde{d}_i}{\mathbf{v}_i + \delta} \geq a_i$. Gemäß Lemma 8 können die Kosten von s reduziert werden, indem die Geschwindigkeit auf $\tilde{d}_i$ um δ erhöht wird und auf $\tilde{d}_{i+1}$ um δ gesenkt wird. Durch die Verschiebung sinkt die Gesamtfahrzeit, sodass die restliche Sequenz unverändert bleibt. Die Reduzierung der Kosten widerspricht jedoch der Annahme, dass der Vektor $\mathbf{v}$ kostenminimal ist. $\square$

Das folgende Lemma zeigt, dass die Geschwindigkeitsvektoren von Algorithmus 5 zulässig sind und die Belieferungszeitpunkte gleich zu dem Geschwindigkeitsvektor von Algorithmus 4 gewählt werden können. Es wird angenommen, dass $t_i, t_i^*, \tilde{t}_i$ die Belieferungszeitpunkte des Kunden $u_i \in s$ darstellen, basierend auf den Vektoren $\mathbf{v}, \mathbf{v}^*$ und $\tilde{\mathbf{v}}$. Hierbei ist $\mathbf{v}$ der optimale Geschwindigkeitsvektor bezüglich der Funktion f auf $[v_{\min}, v_{\max}]$, $\mathbf{v}^*$ entspricht der Rückgabe von Algorithmus 5 und $\tilde{\mathbf{v}}$ entspricht der Rückgabe von Algorithmus 4.

Lemma 11. *Sei $s = (u_1, \dots, u_h)$ eine Sequenz. Algorithmus 5 berechnet zulässige Geschwindigkeitsvektoren und die Belieferungszeitpunkte von Algorithmus 4 sind auch für den Vektor von Algorithmus 5 zulässig.*

Beweis. Zuerst wird gezeigt, dass die Geschwindigkeiten des Vektors $\mathbf{v}^*$ zulässig sind. Gemäß der Definition der Einträge von $\mathbf{v}^*$ gilt $v_{\min} \leq \max(v_{\text{opt}}, \tilde{\mathbf{v}}_i) \leq v_{\max}$. Für die Zulässigkeit von $\mathbf{v}^*$ muss noch gezeigt werden, dass Belieferungszeitpunkte für s gewählt werden können, sodass alle Kunden innerhalb ihrer Zeitfenster beliefert werden. Da Algorithmus 4 korrekt ist, ist $\tilde{\mathbf{v}}$ zulässig. Wähle die Belieferungszeitpunkte t_i^* für den Geschwindigkeitsvektor $\mathbf{v}^*$ als $t_i^* = \tilde{t}_i$ für $u_i \in s$. Diese Wahl ist zulässig, da $\tilde{\mathbf{v}}_i \leq \mathbf{v}_i^*$ gilt. $\square$

Das folgende Lemma zeigt, dass kein zulässiger Geschwindigkeitsvektor existiert, falls Algorithmus 5 unzulässig zurückgibt.

Lemma 12. *Sei $s = (u_1, \dots, u_h)$ eine Sequenz. Gibt Algorithmus 5 unzulässig zurück, dann kann kein Geschwindigkeitsvektor zu s , der die Geschwindigkeitsgrenzen $[v_{\min}, v_{\max}]$ erfüllt, existieren.*

Beweis. Algorithmus 5 gibt *unzulässig* zurück, falls Algorithmus 4 die Sequenz als *unzulässig* bewertet. Hvattum et al. (2013) haben gezeigt, dass Algorithmus 4 korrekt ist. Es folgt daraus, dass, wenn Algorithmus 4 *unzulässig* zurückgibt, kein zulässiger Geschwindigkeitsvektor zu s mit Geschwindigkeiten aus dem Intervall $[0, v_{\max}]$ existiert. In diesem Fall kann es jedoch auch keinen zulässigen Geschwindigkeitsvektor für s im Intervall $[v_{\min}, v_{\max}]$ geben, da dieses eine Teilmenge von $[0, v_{\max}]$ ist. $\square$

Abgeschlossen wird der Korrektheitsbeweis von Algorithmus 5 mit dem folgenden Theorem. Dieses zeigt, dass Algorithmus 5 optimale Geschwindigkeitsvektoren bezogen auf die Funktion f berechnet. Es wird ohne Beschränkung der Allgemeinheit angenommen, dass die Belieferungszeitpunkte $t_i, t_i^*, \tilde{t}_i, u_i \in s$, frühestmöglich sind.

Theorem 2. *Sei $s = (u_1, \dots, u_h)$ eine Sequenz. Algorithmus 5 berechnet einen optimalen Geschwindigkeitsvektor für s bezüglich der Funktion f für die Geschwindigkeitsgrenzen $[v_{\min}, v_{\max}]$, sofern ein solcher existiert.*

Beweis. Der Beweis erfolgt durch einen Widerspruch. Angenommen, die Kosten der Sequenz s unter $\mathbf{v}^*$ sind größer als die eines optimalen Vektors $\mathbf{v}$. Die Geschwindigkeitsvektoren müssen sich in mindestens einem Eintrag unterscheiden. Gemäß Lemma 11 muss gelten, dass $t_j^* \leq \tilde{t}_j$ für $j = 1, \dots, h$, da die Belieferungszeiten minimal gewählt wurden und $\tilde{v}_j \leq v_j^*$ für $j = 1, \dots, k$ gilt. Es muss außerdem $v_{\max} - v_{\min} > 0$ gelten, da andernfalls $\mathbf{v}^* = \mathbf{v}$ wäre. Sei i der erste Index, für den $\mathbf{v}_i^* \neq \mathbf{v}_i$ gilt. Gemäß Algorithmus 5 gilt $\mathbf{v}_i^* = \max\{v_{\text{opt}}, \tilde{\mathbf{v}}_i\}$. Im Folgenden wird zwischen den Fällen (1) $\mathbf{v}_i^* = v_{\text{opt}}$ und (2) $\mathbf{v}_i^* = \tilde{\mathbf{v}}_i$ unterschieden.

(1) $v_i^* = v_{\text{opt}}$. Es wird zwischen den beiden Fällen (1.1) $\mathbf{v}_i < v_{\text{opt}}$ und (1.2) $\mathbf{v}_i > v_{\text{opt}}$ unterschieden.

(1.1) Dieser Fall ist ein Widerspruch zur Optimalität von $\mathbf{v}$, da $f(\mathbf{v}_i) > f(\mathbf{v}_i^*) = f(v_{\text{opt}})$ gilt, wobei v_{opt} eindeutig ist und $t_{i-1}^* = t_{i-1}$ sowie $t_i^* \leq t_i$ gilt.

(1.2) Es folgt, dass $\mathbf{v}_i > v_{opt} = \mathbf{v}_i^*$. Gemäß der Definition von $\mathbf{v}^*$ gilt $v_{opt} \geq \tilde{v}_i$. Es gilt $v_{opt} > \tilde{\mathbf{v}}_i$ und der Fall, dass $v_{opt} = \tilde{\mathbf{v}}_i$ gilt, wird in Fall (2) behandelt. Es muss $t_i < t_i^* \leq \tilde{t}_i$ gelten, da andernfalls $\mathbf{v}_i$ verringert werden könnte, was einen Widerspruch zur Optimalität von $\mathbf{v}$ darstellen würde. Es werden zwei Fälle unterschieden: $i = h$ oder $i < h$. Der erste Fall ist ein Widerspruch zur Optimalität, da $f(\mathbf{v}_h) > f(\mathbf{v}_h^*)$ und $t_h < t_h^* \leq \tilde{t}_h$ gelten. Im zweiten Fall wird angenommen, dass es einen kleinsten Index j gibt, bei dem $\tilde{\mathbf{v}}_j < \tilde{\mathbf{v}}_{j+1}$ gilt und $i \leq j$. Sei l der kleinste Index mit $i \leq l \leq j$, für den $\mathbf{v}_l > \mathbf{v}_{l+1}$ gilt. Angenommen, es existiert kein Index l , dann gilt $\tilde{\mathbf{v}}_{\tilde{l}} < \mathbf{v}_{\tilde{l}}$, für alle $\tilde{l}$ mit $\tilde{l} = i, \dots, j$. Allerdings gilt $\tilde{t}_j = a_j$. Zusammen mit $t_i < t_i^* \leq \tilde{t}_i$ folgt, dass $\mathbf{v}$ zu Wartezeiten führt. Daher könnte die Geschwindigkeit von $\mathbf{v}_i$ reduziert werden, was einen Widerspruch zur Optimalität von $\mathbf{v}$ darstellen würde. Somit muss es einen Index l geben, für den $i \leq l \leq j$ und $\mathbf{v}_l > \mathbf{v}_{l+1}$ gilt. In diesem Fall folgt jedoch $t_l = b_l$. Da $\mathbf{v}_{\tilde{l}} > \tilde{\mathbf{v}}_{\tilde{l}}$, für $\tilde{l} = i, \dots, l$, und $t_i < t_i^* \leq \tilde{t}_i$ gilt, ergibt sich daraus, dass entweder $\tilde{\mathbf{v}}$ unzulässig ist oder $\mathbf{v}$ zu Wartezeiten führt. Beide Argumentationen führen zu einem Widerspruch. Daraus folgt, dass es keinen Index j geben kann, und es gilt $\tilde{\mathbf{v}}_i \geq \tilde{\mathbf{v}}_{i+1} \geq \dots \geq \tilde{\mathbf{v}}_h$. Gemäß der Definition von $\mathbf{v}^*$ folgt $\mathbf{v}_{\tilde{l}}^* = v_{opt}$ für $\tilde{l} = i, \dots, h$. Dies ist ein Widerspruch zur Optimalität von $\mathbf{v}$, da $\mathbf{v}^*$ zulässig ist und somit $\mathbf{v}_i$ gesenkt werden könnte.

Damit ist der Fall (1) abgeschlossen.

(2) $\mathbf{v}_i^* = \tilde{\mathbf{v}}_i$. Zusammen mit der Definition von $\mathbf{v}^*$ folgt $\tilde{\mathbf{v}}_i \geq v_{opt}$. Es wird zwischen den Fällen (2.1) $t_i \geq \tilde{t}_i$ und (2.2) $t_i < \tilde{t}_i$ unterschieden.

(2.1) Es gilt $t_{i-1} = t_{i-1}^* \leq \tilde{t}_{i-1}$. Daraus folgt notwendigerweise, dass $\mathbf{v}_i < \mathbf{v}_i^* = \tilde{\mathbf{v}}_i$, da andernfalls $\mathbf{v}_i$ gesenkt werden könnte, was einen Widerspruch zur Optimalität von $\mathbf{v}$ darstellen würde. Es wird eine weitere Fallunterscheidung getroffen:

(2.1.1) $t_i = \tilde{t}_i$. Es folgt $t_{i-1} < \tilde{t}_{i-1}$. Es ist anzumerken, dass der Index i größer als 1 sein muss, da $\mathbf{v}$ sonst unzulässig wäre, weil eine Lösung basierend auf $\tilde{\mathbf{v}}$ zum Zeitpunkt 0 vom Depot startet. Falls $\tilde{\mathbf{v}}_{i-1} < \tilde{\mathbf{v}}_i$ gilt, folgt gemäß Lemma 10, dass $\tilde{t}_{i-1} = a_{i-1}$. Dies widerspricht der Zulässigkeit von $\mathbf{v}$, da $t_{i-1} < \tilde{t}_{i-1}$ gilt. Folglich muss $\tilde{\mathbf{v}}_{i-1} \geq \tilde{\mathbf{v}}_i$ gelten. Nach der Definition von $\mathbf{v}^*$ gilt $\mathbf{v}_{i-1}^* = \tilde{\mathbf{v}}_{i-1}$. Zudem wurde angenommen, dass für $l < i$ $\mathbf{v}_l = \mathbf{v}_l^*$ gilt. Somit folgt $\mathbf{v}_{i-1} = \mathbf{v}_{i-1}^* = \tilde{\mathbf{v}}_{i-1}$. Dies impliziert jedoch, dass $\mathbf{v}_{i-1} > \mathbf{v}_i$ und gemäß Lemma 9 $t_{i-1} = b_{i-1}$ gilt. Dies widerspricht der Zulässigkeit des Vektors $\tilde{\mathbf{v}}$, womit dieser Fall abgeschlossen ist.

(2.1.2) $t_i > \tilde{t}_i$. In diesem Fall gilt $t_{i-1} \leq \tilde{t}_{i-1}$, $\mathbf{v}_i < \tilde{\mathbf{v}}_i$ und $t_i > \tilde{t}_i$. Angenommen, es existiert ein kleinster Index j , für den $\tilde{\mathbf{v}}_j > \tilde{\mathbf{v}}_{j+1}$ und $i < j$ gilt. Gemäß Lemma 9 folgt, dass $\tilde{t}_j = b_j$ gilt. Es muss $i \neq j$ gelten, da $t_i > \tilde{t}_i$ ist. Es folgt $\tilde{\mathbf{v}}_i \leq \tilde{\mathbf{v}}_{i+1} \leq \dots \leq \tilde{\mathbf{v}}_j > \tilde{\mathbf{v}}_{j+1}$. Sei l der kleinste Index, für den $i \leq l \leq j$ und $\mathbf{v}_l < \mathbf{v}_{l+1}$ gilt. Ein solcher Index muss existieren, da $\mathbf{v}$ ansonsten unzulässig wäre. Gemäß Lemma 10 gilt $t_l = a_l$. Dies widerspricht jedoch der Zulässigkeit von $\tilde{\mathbf{v}}$, da der Vektor $\tilde{\mathbf{v}}$, angewendet auf s , keine Wartezeiten enthält und $t_i > \tilde{t}_i$ gilt. Somit kann es keinen Index j geben, und es folgt $\tilde{\mathbf{v}}_i \leq \tilde{\mathbf{v}}_{i+1} \leq \dots \leq \tilde{\mathbf{v}}_h$. Jedoch gilt

$\tilde{t}_h = b_h$. Folglich muss es einen Index $\bar{j}$ geben, für den $i \leq \bar{j} \leq h$ und $\mathbf{v}_{\bar{j}} > \mathbf{v}_{\bar{j}+1}$ gilt. Dies führt jedoch, wie bereits gezeigt, zu einem Widerspruch.

(2.2) $t_i < \tilde{t}_i$. Es wird eine weitere Fallunterscheidung getroffen:

(2.2.1) $\mathbf{v}_i < \tilde{\mathbf{v}}_i$. Es gilt $t_{i-1} < \tilde{t}_{i-1}$, da sowohl $t_i < \tilde{t}_i$ sowie $\mathbf{v}_i < \tilde{\mathbf{v}}_i$ gilt. Angenommen, es existiert ein größter Index j , für den $\tilde{\mathbf{v}}_j < \tilde{\mathbf{v}}_{j+1}$ und $j \leq i$ gelten. Gemäß Lemma 10 folgt, dass $\tilde{t}_j = a_j$ gilt. Die Indizes i und j müssen ungleich sein, weil $t_{i-1} < \tilde{t}_{i-1}$ gilt. Daraus folgt $\tilde{\mathbf{v}}_j < \tilde{\mathbf{v}}_{j+1} \geq \dots \geq \tilde{\mathbf{v}}_{i-1} \geq \tilde{\mathbf{v}}_i$. Sei l der größte Index mit $j \leq l \leq i$ und $\mathbf{v}_l > \mathbf{v}_{l+1}$. Ein solcher Index muss existieren, da $\mathbf{v}$ sonst unzulässig wäre, weil $\tilde{\mathbf{v}}$, angewendet auf s , nicht zu Wartezeiten führt, $t_i < \tilde{t}_i$, $\tilde{t}_j = a_j$ und $\mathbf{v}_{\bar{j}} < \tilde{\mathbf{v}}_{\bar{j}}$ für $\bar{j} = j+1, \dots, i$ gilt. Gemäß Lemma 9 folgt $t_l = b_l$. Dies ist aber ein Widerspruch zur Zulässigkeit von $\tilde{\mathbf{v}}$. Folglich kann kein Index j existieren und es gilt $\tilde{\mathbf{v}}_1 \geq \tilde{\mathbf{v}}_2 \geq \dots \geq \tilde{\mathbf{v}}_i$. Damit $t_{i-1} < \tilde{t}_{i-1}$ gelten kann, muss ein größter Index l mit $l \leq i$ und $\mathbf{v}_l > \mathbf{v}_{l+1}$ existieren. Dies führt jedoch, wie bereits gezeigt, zu einem Widerspruch zur Zulässigkeit von $\tilde{\mathbf{v}}$.

(2.2.2) $\mathbf{v}_i > \tilde{\mathbf{v}}_i$. Es gilt $f(\mathbf{v}_i) > f(\tilde{\mathbf{v}}_i) \geq f(v_{opt})$. Dieser Fall ist gleich zu Fall (1.2) mit dem Unterschied, dass $\mathbf{v}_i^* = \tilde{\mathbf{v}}_i$ und $f(\mathbf{v}_i^*) > f(\tilde{\mathbf{v}}_i) \geq f(v_{opt})$ gilt. Gleichmaßen kann gefolgert werden, dass $\tilde{\mathbf{v}}_i \geq \tilde{\mathbf{v}}_{i+1} \geq \dots \geq \tilde{\mathbf{v}}_h$ gilt. Es gilt, dass $t_h < \tilde{t}_h$ oder $\mathbf{v}_{\bar{j}} > \mathbf{v}_{\bar{j}+1}$ für einen Index $\bar{j}$, mit $i \leq \bar{j} \leq h$, gilt. Im ersten Fall kann die Geschwindigkeit $\mathbf{v}_i$ gesenkt werden, weil $\tilde{\mathbf{v}}$ zulässig ist. Dies ist ein Widerspruch zur Optimalität von $\mathbf{v}$. Im zweiten Fall wird $\bar{j}$ minimal gewählt. Es folgt $\mathbf{v}_i \leq \dots \leq \mathbf{v}_{\bar{j}} > \mathbf{v}_{\bar{j}+1}$. Gemäß Lemma 9 folgt $t_{\bar{j}} = b_{\bar{j}}$. Allerdings muss somit $\mathbf{v}$ zu Wartezeiten führen oder $\tilde{\mathbf{v}}$ ist unzulässig. Dies stellt erneut einen Widerspruch zur Optimalität von $\mathbf{v}$ dar.

Damit ist Fall (2.2) abgeschlossen und damit auch der Beweis des Theorems.

□

Damit ist der Korrektheitsbeweis von Algorithmus 5 abgeschlossen. Im Weiteren wird die Korrektheit von Algorithmus 6 bewiesen. Dazu werden zunächst die Lemmata 13 und 14 bewiesen. Diese stellen den Belieferungszeitpunkt und die Änderung der Geschwindigkeit von zwei aufeinanderfolgenden Kunden in Bezug.

Lemma 13. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und $\mathbf{v} = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt beliefert und bezüglich f kostenminimal ist. Falls für einen Kunden $u_i \in s \setminus u_h$ gilt $t_i = b_i$ und $a_i \neq b_i$, dann gilt $v_i \geq v_{i+1}$.*

Beweis. Der Beweis erfolgt durch Widerspruch. Angenommen, es gilt $v_i < v_{i+1}$. Es wird eine Distanz $\tilde{d}$ gewählt, wobei $\tilde{d} = \min\{d_i, d_{i+1}\}$ gilt. Gemäß Lemma 8 lässt sich die Geschwindigkeit von u_i auf $\tilde{d}$ um δ erhöhen und die Geschwindigkeit von u_{i+1} auf $\tilde{d}$ um δ reduzieren. Dabei ist δ so zu wählen, dass der Belieferungszeitpunkt von u_i zulässig bleibt und $v_i + \delta \leq v_{i+1} - \delta$ gilt. Diese Anpassung ist möglich, da $a_i \neq b_i$ gilt. Durch Wartezeiten nach u_i kann der Belieferungszeitpunkt von u_{i+1} unverändert bleiben. Daraus folgt jedoch, dass $\mathbf{v}$ nicht kostenminimal ist. □

Lemma 14. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und $\mathbf{v} = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt beliefert und bezüglich f kostenminimal ist. Falls für einen Kunden $u_i \in s \setminus u_h$ gilt $t_i = a_i$ und $a_i \neq b_i$, dann gilt $v_i \leq v_{i+1}$.*

Beweis. Der Beweis erfolgt durch Widerspruch. Angenommen, es gilt $v_i > v_{i+1}$. Es wird eine Distanz $\tilde{d}$ gewählt, wobei $\tilde{d} = \min\{d_i, d_{i+1}\}$ gilt. Gemäß Lemma 8 lässt sich die Geschwindigkeit von u_i auf $\tilde{d}$ um δ reduzieren und die Geschwindigkeit von u_{i+1} auf $\tilde{d}$ um δ erhöhen. Dabei ist δ so zu wählen, dass der Belieferungszeitpunkt von u_i zulässig bleibt und $v_i - \delta \geq v_{i+1} + \delta$ gilt. Diese Anpassung ist möglich, da $a_i \neq b_i$ gilt. Durch Wartezeiten nach u_i kann der Belieferungszeitpunkt von u_{i+1} unverändert bleiben. Damit folgt allerdings, dass $\mathbf{v}$ nicht kostenminimal ist. $\square$

Im Folgenden wird gezeigt, dass ein kostenminimaler Geschwindigkeitsvektor mit einer Sequenz ein Match bildet, wie in Definition 17 definiert.

Lemma 15. *Sei s eine Sequenz und $\mathbf{v}$ ein Geschwindigkeitsvektor für s . Falls $\mathbf{v}$ kostenminimal ist, dann bilden s zusammen mit $\mathbf{v}$ ein Match und es existiert eine Partitionierung.*

Beweis. Sei $\mathbf{v} = (v_1, \dots, v_h)$ ein kostenminimaler Geschwindigkeitsvektor. Es werden zwei Fälle unterschieden: (1) Die Sequenz s besitzt einen Kunden u_i , für den $t_i = a_i$ gilt oder (2) ein solcher Kunde existiert nicht.

- (1) Sei u_l der letzte Kunde in s , für den $t_l = a_l$ gilt. Falls $a_l \neq b_l$ gilt, folgt gemäß Proposition 14, dass $v_l \leq v_{l+1}$. Angenommen, es gilt $l = h$, dann folgt $\tilde{s}(\mathbf{v}) = (u_1, \dots, u_h)$. Andernfalls folgt $\tilde{s}(\mathbf{v}) = (u_1, \dots, u_l)$ und $\bar{s}(\mathbf{v}) = (u_{l+1}, \dots, u_k)$. Angenommen, es gilt $v_i < v_{i+1}$, für $l < i < k$. Nach Lemma 10 gilt $t_i = a_i$. Dies ist ein Widerspruch zur Annahme, dass Index l größtmöglich ist. Somit folgt $v_i \geq v_{i+1}$ für $l < i < k$. Angenommen, $t_i = a_i$ gilt. Auch dies ist ein Widerspruch zur Annahme, dass l maximal gewählt wurde.
- (2) Wähle $\bar{s} = (u_1, \dots, u_h)$. Gleichmaßen zu Fall (1) gilt, dass Sequenz $\bar{s}$ keine zwei Kunden mit $v_i < v_{i+1}$ enthalten kann und somit gilt $v_i \geq v_{i+1}$, für $0 < i < k$.

Die Anzahl der Teilsequenzen $\bar{s}_i$, mit $i = 1, \dots, p$, soll minimal gewählt werden, sodass alle Kunden innerhalb einer Teilsequenz die gleichen Geschwindigkeiten haben. $\square$

Mit dem folgenden Lemma wird gezeigt, dass alle Vektoren von Algorithmus 6 ein Match bilden.

Lemma 16. *Sei $s = (u_1, u_2, \dots, u_k)$ eine Sequenz und $\bar{\mathcal{V}}$ die Menge der von Algorithmus 6 berechneten Geschwindigkeitsvektoren. Es gilt, dass alle Vektoren aus $\bar{\mathcal{V}}$ ein Match mit s bilden.*

Beweis. Sei $\mathbf{v}^* = (v_1^*, v_2^*, \dots, v_h^*)$ der optimale Geschwindigkeitsvektor aus der Eingabe von Algorithmus 6. Nach Proposition 15 bildet $\mathbf{v}^*$ ein Match zusammen mit s .

Im Folgenden wird eine Iteration von Algorithmus 6 betrachtet und es wird gezeigt, dass alle Vektoren, die in dieser Iteration zu $\bar{\mathcal{V}}$ hinzugefügt werden, ein Match bilden. Sei $\mathbf{v} = (v_1, v_2, \dots, v_h)$ der Geschwindigkeitsvektor zu Beginn der Iteration. Seien $\tilde{s}(\mathbf{v})$ und $\bar{s}_i(\mathbf{v})$ mit $i = 1, \dots, p$ die Partitionierung von s auf Basis von $\mathbf{v}$. Sei $\mathbf{v}' = (v'_1, v'_2, \dots, v'_h)$ der Geschwindigkeitsvektor zum Ende der Iteration. Sei $\hat{\mathbf{v}} = (\hat{v}_1, \dots, \hat{v}_j, \hat{v}_{j+1}, \dots, \hat{v}_h)$ ein beliebiger Geschwindigkeitsvektor, der in der Iteration berechnet wird. Es gilt $\hat{\mathbf{v}} \neq \mathbf{v}$ und $\hat{\mathbf{v}} \neq \mathbf{v}'$.

Zu Beginn der Iteration wird die längste Teilsequenz s^* für $\mathbf{v}$ berechnet. Es gilt $s^* = \bar{s}_p$. In der Iteration werden die Geschwindigkeiten der Kunden in s^* gleichmäßig erhöht, sodass diese jeweils in $\hat{\mathbf{v}}$ und $\mathbf{v}'$ identisch sind.

Algorithmus 6 verändert die Geschwindigkeit der Kunden in s^* bis einer der folgenden Fälle eintritt:

- (1) $t'_{l'} = a_{l'}, u_{l'} \in s^*$,
- (2) $\mathbf{v}'_h = \mathbf{v}'_j, u_j \in \bar{s}_{p-1}(\mathbf{v})$, oder
- (3) $\mathbf{v}'_h = v_{max}$.

Es folgt, dass $\hat{\mathbf{v}}$ das folgende Match besitzt: $\tilde{s}(\hat{\mathbf{v}}) = \tilde{s}(\mathbf{v})$ und $\bar{s}_i(\hat{\mathbf{v}}) = \bar{s}_i(\mathbf{v})$, mit $i = 1, \dots, p$. Dies gilt aufgrund der folgenden Eigenschaften:

- $v'_j = v_j, u_j \in s \setminus s^*$,
- $\hat{v}_j > \hat{v}_k, u_j \in \hat{\bar{s}}_{p-1}$ und $u_k \in \bar{s}_p(\hat{\mathbf{v}})$, und
- $t_j \neq a_j, u_j \in \bar{s}_p(\hat{\mathbf{v}})$.

Es bleibt zu zeigen, dass der Vektor $\mathbf{v}'$ ein Match besitzt. Es werden drei Fälle unterschieden:

- (1) Es gilt $t'_{l'} = a_{l'}, u_{l'} \in s^*$. In diesem Fall folgt, dass sich nach $\mathbf{v}'$ folgende Partitionierung ergibt: $\tilde{s}(\mathbf{v}')$ und $\bar{s}(\mathbf{v}') = \bar{s}_1(\mathbf{v}')$, wobei $u_j \in \tilde{s}(\mathbf{v}')$, für $j \leq l'$, und $u_j \in \bar{s}(\mathbf{v}')$, für $j > l'$.
- (2) Es gilt $v'_h = v'_j$, wobei $u_j \in \bar{s}_{p-1}(\mathbf{v})$, und $t'_j \neq a_j, u_j \in \bar{s}_{p-1}(\mathbf{v}) \cup \bar{s}_p(\mathbf{v})$. In diesem Fall folgt, dass sich nach $\mathbf{v}'$ folgende Partitionierung ergibt: $\tilde{s}(\mathbf{v}'), \bar{s}_i(\mathbf{v}'), i = 1, \dots, p-1$, wobei gilt
 - $\tilde{s}(\mathbf{v}') = \tilde{s}(\mathbf{v})$,
 - $\bar{s}_i(\mathbf{v}') = \bar{s}_i(\mathbf{v}), i = 1, \dots, p-2$, und
 - $\bar{s}_{p-1}(\mathbf{v}') = \bar{s}_{p-1}(\mathbf{v}) \cup \bar{s}_p(\mathbf{v})$.
- (3) Es gilt $v'_h = v_{max}, t'_j \neq a_j, u_j \in s^*$ und $v'_h > v'_j$, wobei $u_j \in \bar{s}_{p-1}(\mathbf{v})$. Damit folgt $p = 1$ und aus $\mathbf{v}'$ ergibt sich die folgende Partitionierung: $\tilde{s}(\hat{\mathbf{v}}) = \tilde{s}(\mathbf{v})$ und $\bar{s}_1(\hat{\mathbf{v}}) = \bar{s}_1(\mathbf{v}) \cup \bar{s}_2(\mathbf{v})$.

□

Proposition 2. Sei $s = (u_1, u_2, \dots, u_k)$ eine Sequenz und $\mathbf{v}$ ein Geschwindigkeitsvektor, der zusammen mit s ein Match bildet. Es gilt, dass die Teilsequenzen $\bar{s}_i(\mathbf{v})$, für $i = 1, \dots, p$, keine Wartezeiten enthalten und es gilt

$$t_i = t_{i-1} + \tau_i + d_i/v_i, \quad u_i \in \bar{s}(\mathbf{v}).$$

Beweis. Gemäß der Partitionierung von s gilt $t_i > a_i$, für alle $u_i \in \bar{s}$. Der Beweis erfolgt durch Widerspruch. Angenommen, es existiert ein Kunde $u_i \in \bar{s}$, für den $t_i > t_{i-1} + \tau_i + d_i/v_i$ gilt. Dann lässt sich t_i reduzieren, da $t_i > a_i$ gilt. Dies ist ein Widerspruch zur Annahme, dass die Belieferungszeitpunkte frühestmöglich gewählt sind. $\square$

Mit dem folgenden Lemma wird gezeigt, dass die Geschwindigkeitsvektoren von Algorithmus 6 zulässig sind.

Lemma 17. Alle Geschwindigkeitsvektoren aus $\bar{\mathcal{V}}$, berechnet von Algorithmus 6 für die Sequenz $s = (u_1, u_2, \dots, u_h)$, sind zulässig.

Beweis. Der Beweis erfolgt durch Widerspruch. Sei $\tilde{\mathbf{v}} = (\tilde{v}_1, \tilde{v}_2, \dots, \tilde{v}_h)$ der erste, unzulässige Geschwindigkeitsvektor, der der Menge $\bar{\mathcal{V}}$ hinzugefügt wird. Damit existiert ein kleinster Index j , für den $\tilde{v}_j \notin [v_{\min}, v_{\max}]$ oder $\tilde{t}_j \notin [a_j, b_j]$ gilt.

Es gilt $\tilde{\mathbf{v}} \neq \mathbf{v}^*$, wobei $\mathbf{v}^*$ der optimale und somit zulässige Geschwindigkeitsvektor der Eingabe ist (siehe Definition 16). Es wird die Iteration betrachtet, in der $\tilde{\mathbf{v}}$ berechnet wird. Sei $\mathbf{v} = (v_1, v_2, \dots, v_h)$ der Geschwindigkeitsvektor zu Beginn dieser Iteration. Es gilt, dass $\mathbf{v}$ zulässig ist. Gemäß Proposition 16 gilt, dass $\mathbf{v}$ ein Match bildet. Sei $\tilde{s}(\mathbf{v}), \bar{s}_i(\mathbf{v}), i = 1, \dots, p$, die dazugehörige Partition. Sei $\mathbf{v}' = (v'_1, v'_2, \dots, v'_h)$ der Geschwindigkeitsvektor zum Ende der Iteration.

Es gilt $u_j \in \bar{s}_p$, da nur $\bar{s}_p$ verändert wird und diese Änderungen keine Auswirkungen auf die restliche Sequenz haben. Es werden vier Fälle unterschieden: (1) $\tilde{v}_j > v_{\max}$, (2) $\tilde{v}_j < v_{\min}$, (3) $\tilde{t}_j > b_j$ und (4) $\tilde{t}_j < a_j$.

(1) Es gilt $v_j < v_{\max}$ und $v'_j \leq v_{\max}$ (siehe Zeile 13), sowie $\tilde{v}_j \in (v_j, v'_j]$ (siehe Zeile 20). Daraus folgt $\tilde{v}_j \leq v_{\max}$.

(2) Es wird gezeigt, dass $v'_j > v_j$ ist. Dabei gilt

- $v_j < v_{\max}$,
- $v_i > v_j$, $u_i \in \bar{s}_{p-1}$, $u_j \in \bar{s}_p$, falls $p > 1$, und
- $t_i > a_i$, für $u_i \in \bar{s}_p$.

Es werden die folgenden Fälle unterschieden (2.1) $v'_j = v_{\max}$, (2.2) $v'_j = v_i$, für $u_i \in \bar{s}_{p-1}$, oder (2.3) basierend auf $\mathbf{v}'$ gilt $t'_i = a_i$, für $u_i \in \bar{s}_p$. In den Fällen (2.1) und (2.2) folgt $v'_j > v_j$. Im Fall (2.3) gilt $a_i = t'_i < t_i$. Da die Geschwindigkeiten in $\mathbf{v}$ und $\mathbf{v}'$ jeweils für $\bar{s}_p(\mathbf{v})$ einheitlich sind, folgt zwangsläufig $v'_j > v_j$. Damit gilt $v_{\min} < v_j < \tilde{v}_j \leq v'_j$.

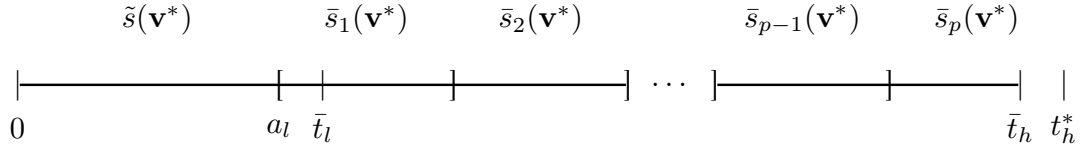


Abbildung C.1: Ankunftszeitpunkte mit Geschwindigkeitsvektor $\bar{\mathbf{v}}$

- (3) Aus Fall (2) folgt $v_i < \tilde{v}_i \leq v'_i$, für $u_i \in \bar{s}_p(\mathbf{v})$. Damit gilt $t'_j \leq \tilde{t}_j < t_j \leq b_j$.
- (4) Sei $u_{l'}$ der letzte Kunde in $\bar{s}_{p-1}(\mathbf{v})$ oder in $\tilde{s}(\mathbf{v})$. Sei $t'_{l'}$, der dazugehörige Belieferungszeitpunkt. Sollte kein solcher Kunde existieren, dann wird $t'_{l'} = 0$ angenommen. Es gilt $\tilde{v}_j \leq \sum_{i=l'+1}^j d_i / (a_j - t'_{l'})$ (siehe Zeile 13). Es folgt

$$a_j > t'_{l'} + \frac{\sum_{i=l'+1}^j d_i}{\tilde{v}_i} \geq t'_{l'} + \frac{\sum_{i=l'+1}^j d_i}{\sum_{i=l'+1}^j d_i} (a_j - t'_{l'}) = a_j.$$

Dies ist ein Widerspruch.

Damit folgt, dass $\tilde{\mathbf{v}}$ zulässig ist. □

Mit dem folgenden Lemma wird gezeigt, dass die Geschwindigkeiten für Kunden in $\tilde{s}^*$, bei der Verringerung des Ankunftszeitpunkts am letzten Kunden einer Sequenz, nicht geändert werden müssen.

Lemma 18. Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $\mathbf{v}^* = (v_1, v_2, \dots, v_h)$ ein Geschwindigkeitsvektor, der u_h zu einem festen Zeitpunkt t_h^* beliefert und bezüglich f kostenminimal ist. Sei $\tilde{s}(\mathbf{v}^*)$ und $\bar{s}_i(\mathbf{v}^*)$, $i = 1, \dots, p$, die Partitionierung von s basierend auf $\mathbf{v}^*$. Sei t'_h ein zulässiger Belieferungszeitpunkt von u_h , wobei $t'_h < t_h^*$ gilt. Dann existiert ein kostenminimaler Geschwindigkeitsvektor $\mathbf{v}$, für den $t_h = t'_h$ und $v_i = v_i^*$, für $u_i \in \tilde{s}(\mathbf{v}^*)$ gilt.

Beweis. Es wird ein Widerspruchsbeweis geführt mit der Annahme, dass kein kostenminimaler Geschwindigkeitsvektor $\bar{\mathbf{v}}$ für Funktion f , der die Bedingung $\bar{t}_h = t'_h$ und $\bar{v}_i = v_i^*$, für alle Kunden $u_i \in \tilde{s}(\mathbf{v}^*)$, erfüllt, existiert. Sei u_l der letzte Kunde aus $\tilde{s}(\mathbf{v}^*)$. Es gilt $t_l^* = a_l$. Damit folgt $\bar{t}_l > a_l$, da ansonsten $\mathbf{v}^*$ bereits ein kostenminimaler Geschwindigkeitsvektor für die Teilsequenz $\tilde{s}(\mathbf{v}^*)$ ist. Eine Visualisierung dieses Falls ist in Abbildung C.1 dargestellt. Alle Geschwindigkeiten der Kunden in $\bar{s}_1^*$ sind für den Vektor $\mathbf{v}^*$ gleich. Es werden zwei Fälle unterschieden:

- (1) $\bar{\mathbf{v}}_l \leq \mathbf{v}_l^*$ oder
- (2) $\bar{\mathbf{v}}_l > \mathbf{v}_l^*$.

(1) Gemäß Proposition 14 gilt $\mathbf{v}_l^* \leq \mathbf{v}_{l+1}^*$, da $t_l^* = a_l$ und $a_l \neq b_l$. Daher muss ein Kunde mit dem kleinsten Index i in $\bar{s}_1(\mathbf{v}^*)$ existieren, wobei $i > l$ und $\bar{\mathbf{v}}_i < \bar{\mathbf{v}}_{i+1}$ gelten. Existiert kein solcher Index, so ist $\bar{\mathbf{v}}$ unzulässig, da $\bar{t}_l > t_l^* = a_l$ und $\bar{\mathbf{v}}_l \leq \mathbf{v}_l^*$. Entweder wird der letzte

Kunde aus $\bar{s}_1(\mathbf{v}^*)$ auf seiner oberen Zeitfenstergrenze beliefert oder es gilt $p = 1$ und $\bar{t}_k < t_k^*$. Gemäß Lemma 10 folgt $\bar{t}_i = a_i$. Dies führt jedoch zu einem Widerspruch zur Zulässigkeit von $\mathbf{v}^*$, da gemäß Proposition 2 $\mathbf{v}^*$ nicht in Wartezeiten für $\bar{s}(\mathbf{v}^*)$ mündet, $t_l^* < \bar{t}_l$ und $\mathbf{v}_j^* \geq \bar{\mathbf{v}}_j$, für $j = l, \dots, i$, gilt.

(2) Sei i der größte Index für den $i < l$ und $\bar{\mathbf{v}}_i \leq \mathbf{v}_i^*$ gilt. Ein solcher Index muss existieren, da $\bar{\mathbf{v}}_l > \mathbf{v}_l^*$ und $\bar{t}_l > t_l^*$ gilt. Es gilt $\bar{t}_j > t_j^*$, für $i \leq j \leq l$. Es werden zwei Fälle unterschieden: (2.1) $\mathbf{v}_i^* > \mathbf{v}_{i+1}^*$ oder (2.2) $\bar{\mathbf{v}}_i < \bar{\mathbf{v}}_{i+1}$. Im Fall (2.1) gilt $t_i^* = b_i$. Daher kann $\bar{\mathbf{v}}$ nicht kostenminimal sein. Im Fall (2.2) gilt $\bar{t}_i = a_i$. Dies ist ein Widerspruch zur Zulässigkeit von $\mathbf{v}^*$. Somit gilt, dass der Vektor $\bar{\mathbf{v}}$ nicht kostenminimal ist. $\square$

Es folgt der Beweis, dass alle Geschwindigkeitsvektoren von Algorithmus 6 für den jeweiligen Ankunftszeitpunkt kostenminimal sind.

Lemma 19. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $\bar{\mathcal{V}}$ die Menge der von Algorithmus 6 berechneten Geschwindigkeitsvektoren. Alle Vektoren in $\bar{\mathcal{V}}$ sind bezüglich f und dem jeweiligen Belieferungszeitpunkt von u_h kostenminimal.*

Beweis. Der Beweis des Satzes wird durch Widerspruch geführt. Sei $\mathbf{v} = (v_1, v_2, \dots, v_h)$ der erste nicht kostenminimale Geschwindigkeitsvektor aus $\bar{\mathcal{V}}$. Sei t_h der Belieferungszeitpunkt von u_h mit $\mathbf{v}$. Es muss also einen Geschwindigkeitsvektor $\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, \dots, \hat{v}_h)$ geben, sodass $t_k = \hat{t}_k$ und die Kosten von $\mathbf{v}$ unter f größer sind als die Kosten des Vektors $\hat{\mathbf{v}}$.

Sei $\mathbf{v}^*$ der Geschwindigkeitsvektor zu Beginn der Iteration in der $\mathbf{v}$ betrachtet wird. Es gilt, dass $\mathbf{v}^*$ kostenminimal ist. Nach Proposition 16 partitioniert Geschwindigkeitsvektor $\mathbf{v}^*$ die Sequenz s in $\tilde{s}(\mathbf{v}^*)$ und $\bar{s}_i(\mathbf{v}^*)$, für $i = 1, \dots, p$. Es gilt, dass sich $\mathbf{v}$ und $\mathbf{v}^*$ nur bei den Kunden in $\bar{s}_p(\mathbf{v}^*)$ unterscheiden.

Es werden zwei Fälle unterschieden:

- (1) Mindestens zwei aufeinanderfolgende Kunden aus $\bar{s}_p(\mathbf{v}^*)$ haben unterschiedliche Geschwindigkeiten in $\hat{\mathbf{v}}$ und es gilt $\mathbf{v}_i = \hat{\mathbf{v}}_i$, für alle $u_i \in s \setminus \bar{s}_p(\mathbf{v}^*)$,
- (2) Der letzte Kunde einer Teilsequenz $\bar{s}_i(\mathbf{v}^*)$ mit $i = 1, \dots, p-1$ wird nicht auf seiner oberen Zeitfenstergrenze beliefert.

Es ist zu beachten, dass nach Lemma 18 die Kunden aus $\tilde{s}(\mathbf{v}^*)$ nicht betrachtet werden müssen.

(1) Sei $u_{l'}$ der letzte Kunde der Teilsequenz $\bar{s}_{p-1}(\mathbf{v}^*)$. Falls $p = 1$, ist $u_{l'}$ der letzte Kunde aus $\tilde{s}(\mathbf{v}^*)$ oder es existiert kein solcher Kunde. Es gilt $t_{l'} = \hat{t}_{l'}$, $t_h = \hat{t}_h$ und $\mathbf{v}_{l'+1} = \dots = \mathbf{v}_h$. Nach Lemma 17 ist $\mathbf{v}$ zulässig. Lemma 8 zeigt, dass das Angleichen der Geschwindigkeiten die Kosten einer Sequenz minimiert. Folglich kann $\hat{\mathbf{v}}$ nicht kostenminimal sein, da ein Vektor $\mathbf{v}$ existiert, der für die Kunden aus $\bar{s}_p(\mathbf{v}^*)$ dieselben Geschwindigkeiten aufweist und zulässig ist.

(2) Wähle i minimal. Sei u_l der letzte Kunde aus Teilsequenz $\bar{s}_{i-1}(\mathbf{v}^*)$ oder aus $\tilde{s}(\mathbf{v}^*)$, oder es existiert kein Kunde u_l . Sei $u_{l'}$ der letzte Kunde aus $\bar{s}_i(\mathbf{v}^*)$. Es gilt $\hat{t}_l = t_l$ oder beide Werte sind 0. Es werden drei Fälle unterschieden:

$$(2.1) \quad \hat{\mathbf{v}}_{l+1} = \dots = \hat{\mathbf{v}}_{l'} > \mathbf{v}_{l+1} = \dots = \mathbf{v}_{l'},$$

(2.2) es gibt zwei Kunden u_j, u_{j+1} in $\bar{s}_i(\mathbf{v}^*)$, für die $\hat{\mathbf{v}}_j > \hat{\mathbf{v}}_{j+1}$ gilt, und es gilt $\hat{\mathbf{v}}_{j+1} = \dots = \hat{\mathbf{v}}_{l'}$ oder

(2.3) es gibt zwei Kunden u_j, u_{j+1} in $\bar{s}_i(\mathbf{v}^*)$, für die $\hat{\mathbf{v}}_j < \hat{\mathbf{v}}_{j+1}$ gilt, und es gilt $\hat{\mathbf{v}}_{j+1} = \dots = \hat{\mathbf{v}}_{l'}$.

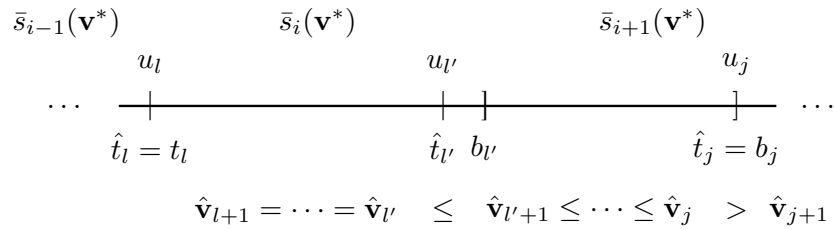


Abbildung C.2: Visualisierung von Fall (2.1)

Eine Visualisierung des Falls (2.1) ist in Abbildung C.2 zu sehen. Es gilt $\hat{t}_l = t_l$. Daraus folgt $\hat{\mathbf{v}}_{l'} \leq \hat{\mathbf{v}}_{l'+1}$, andernfalls ergibt sich ein Widerspruch zu Lemma 9, da $\hat{t}_{l'} \neq b_{l'}$. Sei u_j der erste Kunde, für den $\hat{\mathbf{v}}_j > \hat{\mathbf{v}}_{j+1}$ gilt, wobei $l' < j$. Dies muss gelten, da andernfalls $\hat{t}_h < t_h$ gilt. Gemäß Lemma 9 gilt $\hat{t}_{j'} = b_{j'}$. Dies steht jedoch im Widerspruch zur Zulässigkeit von $\mathbf{v}$, da $\mathbf{v}_{l+1} = \dots = \mathbf{v}_{l'} > \mathbf{v}_{l'+1} \geq \dots \geq \mathbf{v}_h$ gilt, $\bar{s}^*$ keine Wartezeiten mit $\mathbf{v}$ besitzt (Proposition 2) und $\mathbf{v}$ nach Lemma 17 zulässig ist.

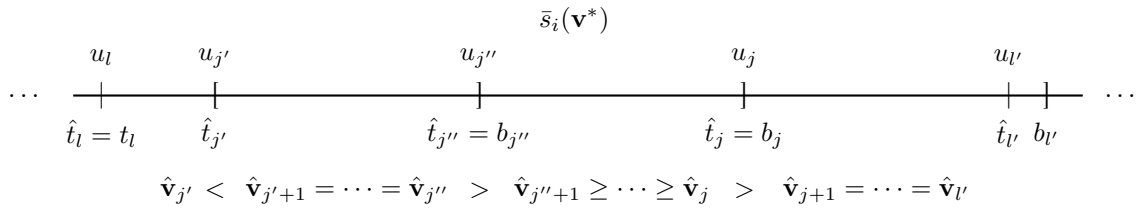


Abbildung C.3: Visualisierung von Fall (2.2)

(2.2) Eine Visualisierung des Falls ist in Abbildung C.3 zu sehen. Da j mit $l < j < l'$ maximal gewählt ist, folgt $\hat{\mathbf{v}}_{j+1} = \hat{\mathbf{v}}_{j+2} = \dots = \hat{\mathbf{v}}_{l'}$. Gemäß Lemma 9 gilt $\hat{t}_j = b_j$. Dies würde einen Widerspruch zur Zulässigkeit von $\mathbf{v}$ darstellen, da $\mathbf{v}$ für alle Kunden aus $\bar{s}_i(\mathbf{v}^*)$ gleiche Geschwindigkeiten besitzt und somit die obere Zeitfenstergrenze von u_j verletzen würde. Somit gibt es einen Kunden $u_{j'}$ in $\bar{s}_i(\mathbf{v}^*)$, für den $\hat{\mathbf{v}}_{j'} < \hat{\mathbf{v}}_{j'+1}$ gilt. Wähle j' mit dem größten Index, für den $j' < j$ gilt. Sei $u_{j''}$ der letzte Kunde, für den $\hat{\mathbf{v}}_{j'+1} = \hat{\mathbf{v}}_{j'+2} = \dots = \hat{\mathbf{v}}_{j''}$, mit $j' < j'' \leq j$, gilt. Gemäß Lemma 9 folgt, dass $\hat{t}_{j''} = b_{j''}$ gilt. Für die Zulässigkeit der Kunden $u_{j'+1}, \dots, u_{j''}$ folgt, dass mindestens eine durchschnittliche Geschwindigkeit von $\hat{\mathbf{v}}_{j''}$ benötigt wird. Diese Geschwindigkeit stellt somit auch eine untere Grenze für alle Kunden aus $\bar{s}_i(\mathbf{v}^*)$ dar, falls alle Geschwindigkeiten

gleich sein sollen. Mit dieser Geschwindigkeit ergibt sich jedoch, dass $t_{l'} < b_{l'}$ für den Kunden $u_{l'}$ gilt, was einen Widerspruch zur Zulässigkeit von $\mathbf{v}$ darstellt.

(2.3) Da der Index j maximal gewählt wurde mit $l < j < l'$, folgt, dass $\hat{\mathbf{v}}_{j+1} = \dots = \hat{\mathbf{v}}_{l'}$. Gemäß Lemma 10 muss gelten, dass $\hat{t}_j = a_j$. Es lassen sich die folgenden zwei Fälle unterscheiden:

(2.3.1) Es gilt $\hat{\mathbf{v}}_{l+1} \leq \dots \leq \hat{\mathbf{v}}_j$, oder

(2.3.2) es existiert ein Kunde $u_{j'}$, für den $\hat{\mathbf{v}}_{j'} > \hat{\mathbf{v}}_{j'+1}$ und $l < j' < j$ gilt.

(2.3.1) Sei $u_{j'}$ der erste Kunde, für den $l+1 \leq j' < j$ und $\hat{v}_{j'} < \hat{v}_{j'+1}$ gilt. Dabei gilt

$$v_{l+1} = \dots = v_{l'} < \hat{v}_{l+1} = \dots = \hat{v}_{j'} < \hat{v}_{j+1} = \dots = \hat{v}_{l'}.$$

Dies folgt aus der Aufteilung von s , nach der $t_j > a_j$ gilt. Dieser Fall ist in Abbildung C.4 dargestellt und ähnelt dem Fall (2.1). Sei $u_{j''}$ der erste Kunde, für den $\hat{v}_{j''} > \hat{v}_{j''+1}$ und $l' < j''$ zutrifft. Ein solcher Kunde muss existieren, da andernfalls $\hat{t}_h < t_h$ gelten würde. Gemäß Lemma 9 gilt $\hat{t}_{j''} = b_{j''}$. Allerdings ist dies ein Widerspruch zur Zulässigkeit von $\mathbf{v}$, da $v_{l+1} = \dots = v_{l'} > v_{l'+1} \geq \dots \geq v_h$ gilt.

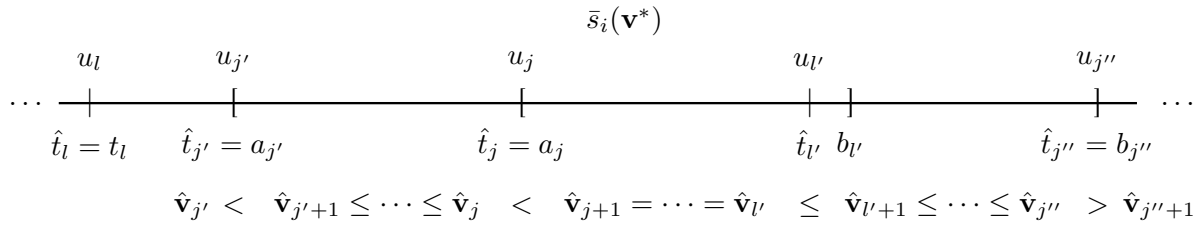


Abbildung C.4: Visualisierung von Fall (2.3.1)

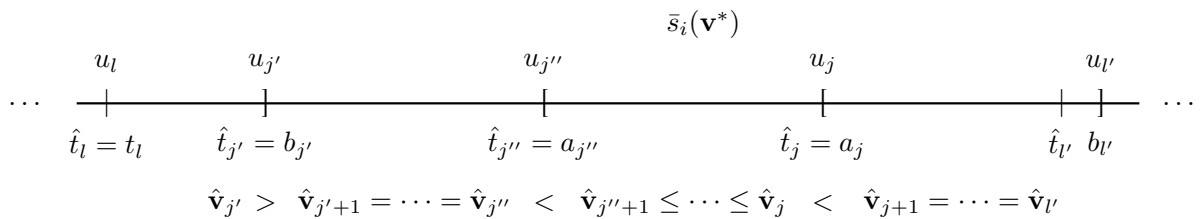


Abbildung C.5: Visualisierung von Fall (2.3.2)

(2.3.2) Sei $u_{j'}$ der Kunde mit dem größten Index, für den $\hat{\mathbf{v}}_{j'} > \hat{\mathbf{v}}_{j'+1}$ und $j' < j$ gilt. Eine Visualisierung des Falls ist in Abbildung C.5 dargestellt. Gemäß Lemma 9 gilt $\hat{t}_{j'} = b_{j'}$. Sei $u_{j''}$ der Kunde mit dem kleinsten Index, für den $j' < j'' \leq j$ und $\hat{\mathbf{v}}_{j''} < \hat{\mathbf{v}}_{j''+1}$ gilt. Gemäß Lemma 10 gilt $\hat{t}_{j''} = a_{j''}$. Es gilt, dass $\hat{\mathbf{v}}_{j'+1} = \dots = \hat{\mathbf{v}}_{j''}$. Außerdem gilt $\hat{\mathbf{v}}_{j'+1} = \dots = \hat{\mathbf{v}}_{j''} < \hat{\mathbf{v}}_{j+1} = \dots = \hat{\mathbf{v}}_{l'}$. Allerdings ist es dann nicht möglich, eine einheitliche Geschwindigkeit für Kunden aus $\bar{s}_i(\mathbf{v}^*)$ ohne Wartezeiten zu verwenden, da die Kunden $u_{j'+1}$ bis $u_{j''}$ eine kleinere Geschwindigkeit als

$\hat{\mathbf{v}}_{j'+1}$ benötigen, während die Kunden u_{j+1} bis $u_{l'}$ eine Geschwindigkeit von mindestens $\hat{\mathbf{v}}_{j+1}$ haben. Dies ist ein Widerspruch zur Zulässigkeit von $\mathbf{v}$. Damit ist der Beweis abgeschlossen. $\square$

Mit dem folgenden Lemma wird gezeigt, dass Algorithmus 6 die Funktion arr , die alle relevanten Ankunftszeitpunkte am letzten Kunden beschreibt, korrekt berechnet.

Lemma 20. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz und sei $\mathbf{v}^*$ der optimale Geschwindigkeitsvektor der Eingabe von Algorithmus 6. Sei arr die Funktion, die die Belieferungszeitpunkte von u_h beschreibt. Die Funktion arr ist korrekt und beschreibt alle zulässigen Belieferungszeitpunkte von u_h , die kleiner oder gleich t_h^* sind.*

Beweis. Die Funktion arr ordnet die Geschwindigkeit des Kunden u_h dem Zeitpunkt seiner Belieferung zu. Diese Abbildung ist stetig, wie in Zeile 18 zu sehen ist. Es wird ein Widerspruchsbeweis geführt und angenommen, dass es einen Geschwindigkeitsvektor $\mathbf{v} = (v_1, \dots, v_h)$ gibt, bei dem Kunde u_h zum Zeitpunkt t'_h beliefert wird, jedoch $t'_h \neq arr(v_h)$ gilt. Sei $\tilde{v}_h$ die maximale Geschwindigkeit von Kunde u_h , die durch Algorithmus 6 berechnet wird. Da die Funktion arr stetig ist, gilt $t'_h < arr(\tilde{v}_h)$. Es werden zwei Fälle unterschieden:

(1) $\tilde{v}_h < v_{max}$. Es folgt $arr(\tilde{v}_h) = a_h$, andernfalls erhöht Algorithmus 6 die Geschwindigkeit von u_h weiter. Dies führt zu einem Widerspruch.

(2) $\tilde{v}_h = v_{max}$. Es wird die letzte Iteration von Algorithmus 6 betrachtet, in der v_h auf v_{max} erhöht wird. Sei $\mathbf{v}$ der Geschwindigkeitsvektor zu Beginn der letzten Iteration. Nach Proposition 16 lässt sich s mit dem Vektor $\mathbf{v}$ in $\tilde{s}(\mathbf{v}), \bar{s}_i(\mathbf{v})$, für $i = 1, \dots, p$, partitionieren. Am Ende der letzten Iteration besitzen alle Kunden aus $\bar{s}_p(\mathbf{v})$ die Geschwindigkeit v_{max} . Daher kann $\bar{s}_p(\mathbf{v})$ nicht in kürzerer Zeit erledigt werden. Daher muss die Bearbeitungszeit einer vorhergehenden Teilsequenz $\bar{s}_i(\mathbf{v})$, für $i = 1, \dots, p - 1$, weiter reduziert werden können. Es ist zu beachten, dass Geschwindigkeitsanpassungen der Kunden in $\tilde{s}(\mathbf{v})$ den Belieferungszeitpunkt bei u_h nicht verändern können (siehe Lemma 18). Basierend auf der Partitionierung von s muss gelten, dass die Geschwindigkeiten von zwei aufeinanderfolgenden Teilsequenzen abnehmend sind. Daher kann $\bar{s}_p(\mathbf{v})$ nur einen einzigen Vorgänger $\bar{s}_1(\mathbf{v})$ haben, da der Algorithmus andernfalls zusätzliche Iterationen erfordern würde. Die Kunden in $\bar{s}_1(\mathbf{v})$ werden bereits mit v_{max} beliefert und eine weitere Reduzierung der benötigten Zeit ist nicht möglich. Damit folgt die Aussage. $\square$

Mit dem folgenden Lemma wird gezeigt, dass Algorithmus 6 eine polynomielle Laufzeit in der Eingabe besitzt.

Lemma 21. *Die Laufzeit von Algorithmus 6 ist polynomiell beschränkt durch $O(|s|^2)$.*

Beweis. Im schlechtesten Fall hat jeder Kunde aus s eine unterschiedliche Geschwindigkeit. Dabei sind die Geschwindigkeiten so verteilt, dass sie von vorn nach hinten streng monoton ansteigend sind. In der ersten Iteration wird die Geschwindigkeit des letzten Kunden erhöht, bis sie der des vorletzten Kunden entspricht. Dieses Vorgehen wird iterativ fortgesetzt, bis alle Kunden aus s die gleiche Geschwindigkeit haben. Anschließend wird die Geschwindigkeit aller Kunden aus s

erhöht, bis der erste Kunde in der Sequenz auf seiner unteren Zeitfenstergrenze beliefert wird. Danach wird s ohne den ersten Kunden beschleunigt, bis der zweite Kunde in der Sequenz auf seiner unteren Zeitfenstergrenze beliefert wird. Dieser Prozess wird fortgesetzt, bis nur noch die Geschwindigkeit des letzten Kunden erhöht wird und er auf seiner unteren Zeitfenstergrenze beliefert wird. Insgesamt ergeben sich somit $2|s| - 1$ Iterationen. In jeder Iteration müssen die Teilsequenz $\bar{s}^*$ und die erhöhte Geschwindigkeit berechnet werden. Dies benötigt maximal $2|s|$ Rechenschritte.

Es ist zu beachten, dass jeder Kunde in die Teilsequenz $\bar{s}^*$, die in einer Iteration beschleunigt wird, genau einmal aufgenommen werden kann. Daher ist die Laufzeit durch $O(|s|^2)$ beschränkt. $\square$

Damit kann gezeigt werden, dass Algorithmus 6 korrekt ist.

Theorem 3. *Sei $s = (u_1, u_2, \dots, u_h)$ eine Sequenz. Algorithmus 6 berechnet in polynomieller Zeit alle relevanten Belieferungszeitpunkte für u_h und die dazugehörigen kostenminimalen Geschwindigkeitsvektoren.*

Beweis. Zur Korrektheit des Algorithmus muss gezeigt werden, dass alle von Algorithmus 6 berechneten Geschwindigkeitsvektoren kostenminimal in Bezug auf die Belieferungszeitpunkte des letzten Kunden sind und die Funktionen arr und $cost$ korrekt sind. Aus Lemma 17 folgt, dass alle Vektoren aus der Menge $\bar{\mathcal{V}}$ zulässig sind. Mit Lemma 19 folgt, dass alle Geschwindigkeitsvektoren aus der Menge $\bar{\mathcal{V}}$ kostenminimal sind. Die Kostenfunktion $cost$ ist korrekt, da alle Geschwindigkeitsvektoren aus $\bar{\mathcal{V}}$ kostenminimal sind und die Kosten durch die Summation der Distanzen, multipliziert mit den entsprechenden Geschwindigkeiten, berechnet werden (siehe Zeile 19). Die Korrektheit der Funktion arr folgt aus Lemma 20. Damit ist der Korrektheitsbeweis von Algorithmus 6 abgeschlossen. Die polynomielle Laufzeitbeschränkung wurde bereits in Lemma 21 bewiesen. $\square$

D Resultate für das Pollution Routing Problem

Im Folgenden erfolgt eine detaillierte Aufstellung der Resultate für das Pollution-Routing Problem, die in Unterkapitel 6.2.2 vorgestellt wurden. Die Resultate des Branch-and-Price-Algorithmus aus Kapitel 5 sind in den folgenden Tabellen dargestellt. Dabei identifiziert die erste Spalte den Namen der Instanz (Inst), gefolgt von der Anzahl der Fahrzeuge (V), der besten zulässigen Lösung (BUB), der Rechenzeit (T) in Sekunden und der Gap (G(%)) in Prozent. Sollte eine Instanz nicht innerhalb des Zeitlimits gelöst werden können, dann wird die Rechenzeit mit *tl* angegeben, und sollte keine zulässige Lösung gefunden werden, dann werden das Feld und die Gap gestrichen. Zusätzlich sind die Resultate des Vergleichs zwischen dem Branch-and-Price-Algorithmus und dem BCP-Algorithmus aus Fukasawa et al. (2017) und Fukasawa et al. (2018) in Tabellen aufgezeigt. Zu beachten ist, dass sich die Anzahl der Fahrzeuge, Zielfunktionswerte und damit auch die Rechenzeiten unterscheiden können. Dabei sind die Tabellen gleich zum vorherigen Aufbau, außer dass diese um drei Spalten erweitert werden, die die Resultate aus Fukasawa et al. (2017) und Fukasawa et al. (2018) aufzeigen.

Inst	V	Branch-and-Price			Inst	V	Branch-and-Price		
		BUB	T	G(%)			BUB	T	G(%)
A-10-1	2	170,641	1,79	0	A-20-1	3	323,185	825,12	0
A-10-2	2	204,877	0,47	0	A-20-2	3	330,046	tl	-
A-10-3	3	200,34	0,25	0	A-20-3	3	-	tl	-
A-10-4	2	189,884	0,48	0	A-20-4	3	-	tl	-
A-10-5	2	175,59	2,08	0	A-20-5	3	295,538	tl	0,69
A-10-6	2	214,48	0,31	0	A-20-6	3	347,687	tl	-
A-10-7	2	190,144	0,52	0	A-20-7	5	-	tl	-
A-10-8	2	222,168	0,20	0	A-20-8	3	277,232	tl	-
A-10-9	2	174,541	0,58	0	A-20-9	3	321,678	1.599,28	-
A-10-10	2	189,816	0,72	0	A-20-10	3	325,439	836,14	-
A-10-11	2	262,079	0,16	0	A-20-11	3	363,796	tl	-
A-10-12	2	183,189	1,47	0	A-20-12	3	-	tl	-
A-10-13	2	195,97	0,62	0	A-20-13	3	308,944	1.425,06	-
A-10-14	2	163,166	0,75	0	A-20-14	4	416,921	tl	2,23
A-10-15	2	127,102	1,43	0	A-20-15	3	319,718	tl	-
A-10-16	2	186,632	3,74	0	A-20-16	3	335,245	tl	19,61
A-10-17	2	159,034	1,62	0	A-20-17	3	-	tl	-
A-10-18	2	162,086	1,94	0	A-20-18	3	345,277	tl	1,55
A-10-19	2	169,463	0,67	0	A-20-19	5	337,663	tl	2,59
A-10-20	2	168,798	7,58	0	A-20-20	3	325,439	1.168,11	-
A-15-1	2	286,577	249,76	0	A-25-1	3	-	tl	-
A-15-2	2	209,118	566,48	0	A-25-2	4	-	tl	-
A-15-3	3	280,887	7,37	0	A-25-3	3	-	tl	-
A-15-4	3	297,557	tl	2,71	A-25-4	3	-	tl	-
A-15-5	2	284,937	22,54	0	A-25-5	4	-	tl	-
A-15-6	2	233,621	tl	-	A-25-6	4	-	tl	-
A-15-7	3	254,226	20,02	0	A-25-7	3	-	tl	-
A-15-8	2	167,989	tl	0,14	A-25-8	4	-	tl	-
A-15-9	3	263,155	602,64	0	A-25-9	4	-	tl	-
A-15-10	2	215,663	1.718,44	0	A-25-10	4	-	tl	-
A-15-11	2	258,955	94,39	0	A-25-11	4	-	tl	-
A-15-12	3	310,32	98,20	0	A-25-12	4	-	tl	-
A-15-13	3	248,402	343,42	0	A-25-13	4	-	tl	-
A-15-14	3	332,259	12,14	0	A-25-14	4	-	tl	-
A-15-15	2	222,271	1.097,48	0	A-25-15	3	-	tl	-
A-15-16	2	205,735	440,22	0	A-25-16	4	-	tl	-
A-15-17	3	282,639	6,69	0	A-25-17	4	-	tl	-
A-15-18	3	315,064	8,48	0	A-25-18	3	-	tl	-
A-15-19	2	166,067	1.946,50	0	A-25-19	4	-	tl	-
A-15-20	3	201,712	265,50	0	A-25-20	4	-	tl	-

Tabelle D.1: Ergebnisse für 10, 15, 20 und 25 Kunden auf den UK-A Instanzen

Inst	V	Fukasawa et al. (2018)			Branch-and-Price		
		BUB	T	G(%)	BUB	T	G(%)
A-10-1	2	170,64	1,4	0	170,641	1,77	0
A-10-2	2	204,88	1	0	204,877	0,46	0
A-10-3	3	200,34	0,9	0	200,34	0,25	0
A-10-4	2	189,88	1,4	0	189,884	0,46	0
A-10-5	2	175,59	3,4	0	175,59	2,09	0
A-10-6	2	214,48	0,5	0	214,48	0,31	0
A-10-7	2	190,14	1,7	0	190,144	0,50	0
A-10-8	2	222,17	0,1	0	222,168	0,19	0
A-10-9	2	174,54	2,1	0	174,541	0,57	0
A-10-10	2	189,82	0,4	0	189,816	0,70	0
A-20-1	5	351,82	64	0	351,818	362,63	0
A-20-2	5	365,77	16,7	0	365,766	296,38	0
A-20-3	5	230,49	556,7	0	230,485	tl	-
A-20-4	5	347,04	659,1	0	347,043	tl	0,8
A-20-5	5	323,44	134,8	0	323,444	648,69	0
A-20-6	5	364,23	94,6	0	364,231	tl	0,11
A-20-7	5	-	tl	0	-	tl	-
A-20-8	5	301,51	50,1	0	302,29	tl	0,61
A-20-9	5	362,56	98,3	0	362,562	330,23	0
A-20-10	5	313,33	231,4	0	368,033	248,33	0
		Fukasawa et al. (2017)			Branch-and-Price		
A-25-1	6	389,46	tl	-	-	tl	-
A-25-2	6	373,92	tl	-	-	tl	-
A-25-3	6	273,08	tl	-	-	tl	-
A-25-4	6	321,14	tl	-	-	tl	-
A-25-5	6	365	2925	0	-	tl	-
A-25-6	6	313,72	tl	-	-	tl	-
A-25-7	6	3.431,12	tl	-	-	tl	-
A-25-8	6	359,08	1107	0	-	tl	-
A-25-9	6	380,68	tl	-	-	tl	-
A-25-10	6	411,33	tl	-	-	tl	-

Tabelle D.2: Vergleich zwischen dem BCP-Algorithmus von (Fukasawa et al., 2017, 2018) und dem Branch-and-Price-Algorithmus auf den UK-A Instanzen

Inst	V	Branch-and-Price			Inst	V	Branch-and-Price		
		BUB	T	G(%)			BUB	T	G(%)
B-10-1	2	246,445	0,03	0	B-20-1	4	445,854	0,07	0
B-10-2	2	303,732	0,01	0	B-20-2	3	440,444	0,14	0
B-10-3	3	301,891	0,01	0	B-20-3	3	323,991	0,15	0
B-10-4	2	273,904	0,02	0	B-20-4	3	473,398	0,18	0
B-10-5	2	255,074	0,02	0	B-20-5	4	430,502	0,12	0
B-10-6	3	332,34	0,01	0	B-20-6	4	488,408	0,15	0
B-10-7	3	314,641	0,01	0	B-20-7	3	322,556	1,40	0
B-10-8	2	339,316	0,01	0	B-20-8	3	423,221	0,14	0
B-10-9	2	261,1	0,01	0	B-20-9	3	502,65	0,13	0
B-10-10	2	285,202	0,02	0	B-20-10	3	389,45	0,13	0
B-10-11	3	408,545	0,01	0	B-20-11	5	577,228	0,03	0
B-10-12	2	251,645	0,01	0	B-20-12	4	505,88	0,47	0
B-10-13	3	274,074	0,01	0	B-20-13	4	460,918	0,13	0
B-10-14	2	267,916	0,02	0	B-20-14	4	516,486	0,23	0
B-10-15	2	197,503	0,02	0	B-20-15	4	503,011	0,07	0
B-10-16	2	245,755	0,05	0	B-20-16	4	493,533	0,08	0
B-10-17	2	283,834	0,02	0	B-20-17	4	495,739	2,51	0
B-10-18	2	241,526	0,02	0	B-20-18	4	477,612	0,05	0
B-10-19	3	330,377	0,01	0	B-20-19	4	508,121	0,58	0
B-10-20	2	208,059	0,01	0	B-20-20	4	498,288	0,10	0
B-15-1	4	388,91	0,02	0	B-25-1	3	414,424	0,63	0
B-15-2	3	295,537	0,04	0	B-25-2	4	471,313	2,95	0
B-15-3	4	440,984	0,02	0	B-25-3	3	332,777	3,19	0
B-15-4	3	405,852	0,28	0	B-25-4	3	404,858	9,75	0
B-15-5	5	449,46	0,02	0	B-25-5	4	440,384	5,90	0
B-15-6	3	332,221	0,21	0	B-25-6	4	458,459	2,02	0
B-15-7	3	392,438	0,11	0	B-25-7	4	473,772	9,72	0
B-15-8	2	227,632	0,11	0	B-25-8	4	504,551	1,38	0
B-15-9	3	336,785	0,08	0	B-25-9	4	396,934	4,35	0
B-15-10	3	316,918	0,08	0	B-25-10	4	474,518	15,20	0
B-15-11	4	431,525	0,02	0	B-25-11	4	508,682	84,20	0
B-15-12	3	386,913	0,03	0	B-25-12	5	600,618	0,14	0
B-15-13	3	345,128	0,03	0	B-25-13	4	376,791	128,36	0
B-15-14	3	460,849	0,03	0	B-25-14	4	529,525	0,66	0
B-15-15	2	342,711	0,09	0	B-25-15	3	529,222	0,74	0
B-15-16	3	357,16	0,07	0	B-25-16	4	494,667	0,70	0
B-15-17	3	403,568	0,07	0	B-25-17	4	613,078	0,49	0
B-15-18	3	432,229	0,02	0	B-25-18	4	520,221	0,20	0
B-15-19	3	266,899	0,05	0	B-25-19	5	610,108	0,43	0
B-15-20	3	290,131	0,14	0	B-25-20	5	521,012	32,66	0

Tabelle D.3: Ergebnisse für 10, 15, 20 und 25 Kunden auf den UK-B Instanzen

Inst	V	Branch-and-Price			
		BUB	T	G(%)	
B-50-1	7	882,732	234,20	0	
B-50-2	7	865,818	10,06	0	
B-50-3	7	857,256	106,70	0	
B-50-4	7	981,272	658,31	0	
B-50-5	7	877,556	28,69	0	
B-50-6	8	829,741	tl	0,31	
B-50-7	7	747,763	3.045,33	0	
B-50-8	7	806,212	71,20	0	
B-50-9	7	923,475	252,14	0	
B-50-10	7	891,328	tl	1,03	
B-50-11	7	876,708	88,63	0	
B-50-12	6	777,205	1.081,94	0	
B-50-13	7	802,425	82,38	0	
B-50-14	7	943,39	38,02	0	
B-50-15	7	814,382	235,22	0	
B-50-16	7	781,632	849,65	0	
B-50-17	7	706,142	285,93	0	
B-50-18	8	907,733	18,37	0	
B-50-19	7	807,664	2.309,86	0	
B-50-20	7	909,435	5,41	0	

Inst	V	Branch-and-Price			
		BUB	T	G(%)	
B-75-1	11	1.343,8	tl	1,23	
B-75-2	11	1.157,55	tl	0,81	
B-75-3	10	1.233,11	tl	1,40	
B-75-4	11	1.146,39	tl	1,56	
B-75-5	10	1.214,06	1.082,94	0	
B-75-6	11	1.281,86	440,57	0	
B-75-7	11	1.307,11	760,69	0	
B-75-8	11	1.250,78	tl	0,97	
B-75-9	10	1.236,03	1.084,09	0	
B-75-10	10	1.257,07	tl	0,39	
B-75-11	10	995,816	tl	0,01	
B-75-12	10	1.164,18	tl	0,20	
B-75-13	10	1.311,32	1.159,54	0	
B-75-14	10	1.238,92	tl	1,43	
B-75-15	10	1.347,81	tl	0,31	
B-75-16	10	1.245,64	tl	0,39	
B-75-17	10	1.300,63	tl	0,32	
B-75-18	10	1.170,04	tl	0,48	
B-75-19	10	1.169,13	tl	0,53	
B-75-20	11	1.276,56	324,27	0	

Tabelle D.4: Ergebnisse für 50 und 75 Kunden auf den UK-B Instanzen

Inst	V	Fukasawa et al. (2018)			Branch-and-Price		
		BUB	T	G(%)	BUB	T	G(%)
B-10-1	2	246,44	0	0	246,445	0,01	0
B-10-2	2	303,73	0	0	303,732	0,01	0
B-10-3	3	301,89	0	0	301,891	0,01	0
B-10-4	2	273,9	0	0	273,904	0,02	0
B-10-5	2	255,07	0	0	255,074	0,02	0
B-10-6	3	332,34	0	0	332,34	0,01	0
B-10-7	3	314,64	0,1	0	314,641	0,01	0
B-10-8	2	339,36	0	0	339,316	0,01	0
B-10-9	2	261,1	0	0	261,1	0,01	0
B-10-10	2	285,20	0	0	285,202	0,02	0
B-20-1	5	469,35	0	0	469,355	0,04	0
B-20-2	5	477,05	0,1	0	477,046	0,06	0
B-20-3	5	354,46	0,1	0	354,458	0,08	0
B-20-4	5	523,59	0,1	0	523,589	0,09	0
B-20-5	5	447,33	0,1	0	447,334	0,10	0
B-20-6	5	511,78	0,4	0	511,777	0,22	0
B-20-7	5	379,02	0,6	0	379,018	0,39	0
B-20-8	5	431,31	0,6	0	431,311	0,11	0
B-20-9	5	548,68	0,1	0	548,675	0,09	0
B-20-10	5	410,32	0	0	410,321	0,06	0
		Fukasawa et al. (2017)			Branch-and-Price		
B-25-1	6	475,13	1	0	475,128	0,43	0
B-25-2	6	533,59	10,6	0	533,595	2,49	0
B-25-3	6	390,27	2,6	0	390,27	1,44	0
B-25-4	6	467,49	132	0	467,487	2,83	0
B-25-5	6	491,69	11,2	0	491,687	4,08	0
B-25-6	6	516,71	1,5	0	516,707	0,82	0
B-25-7	6	526,12	2,3	0	526,122	0,84	0
B-25-8	6	537,15	3,5	0	537,155	0,76	0
B-25-9	6	436,84	2,8	0	436,843	3,77	0
B-25-10	6	533,88	5	0	533,877	2,68	0
B-50-1	12	992,15	tl	0,3	991,996	91,03	0
B-50-2	12	1.028,01	tl	0	1.028,01	63,12	0
B-50-3	12	1.022,4	tl	0,5	1.022,4	359,43	0
B-50-4	12	1.149,08	1.985	0	1.149,08	42,93	0
B-50-5	12	964,72	6,8	0	964,718	3,55	0
B-50-6	12	921,95	tl	0,8	921,326	1.384,85	0
B-50-7	12	852,33	1.022	0	852,326	113,27	0
B-50-8	12	924,43	3,6	0	924,43	2,13	0
B-50-9	12	1.034,92	2.252	0	1.034,92	34,72	0
B-50-10	12	1.026,8	tl	0,2	1.027,38	tl	0,45

Tabelle D.5: Vergleich zwischen dem BCP-Algorithmus von (Fukasawa et al., 2017, 2018) und dem Branch-and-Price-Algorithmus auf den UK-B Instanzen

Inst	V	Branch-and-Price			Inst	V	Branch-and-Price		
		BUB	T	G(%)			BUB	T	G(%)
C-10-1	2	210,182	0,08	0	C-20-1	3	375,421	1,62	0
C-10-2	2	271,93	0,08	0	C-20-2	3	385,641	17,85	0
C-10-3	2	229,177	0,07	0	C-20-3	3	263,809	263,38	0
C-10-4	2	230,519	0,05	0	C-20-4	3	380,916	23,25	0
C-10-5	2	205,491	0,07	0	C-20-5	3	344,787	4,68	0
C-10-6	2	255,819	0,07	0	C-20-6	3	403,96	5,10	0
C-10-7	2	217,788	0,07	0	C-20-7	3	278,687	2.896,40	0
C-10-8	2	251,294	0,05	0	C-20-8	3	365,272	4,42	0
C-10-9	2	186,041	0,04	0	C-20-9	3	381,897	2,09	0
C-10-10	2	231,619	0,05	0	C-20-10	3	341,601	9,73	0
C-10-11	2	298,199	0,07	0	C-20-11	3	445,45	1,48	0
C-10-12	2	206,583	0,05	0	C-20-12	3	357,169	7,30	0
C-10-13	2	211,751	0,10	0	C-20-13	3	385,788	7,14	0
C-10-14	2	209,067	0,28	0	C-20-14	4	457,871	1,75	0
C-10-15	2	176,555	0,12	0	C-20-15	3	372,526	9,36	0
C-10-16	2	229,15	0,26	0	C-20-16	3	406,187	26,93	0
C-10-17	2	219,196	0,04	0	C-20-17	3	421,107	tl	3,45
C-10-18	2	195,044	0,25	0	C-20-18	3	405,931	1,00	0
C-10-19	2	218,194	0,43	0	C-20-19	3	387,482	103,26	0
C-10-20	2	189,56	0,62	0	C-20-20	3	416,401	1,05	0
C-15-1	3	351,817	2,27	0	C-25-1	3	366,488	tl	1,55
C-15-2	3	279,633	0,54	0	C-25-2	4	434,472	tl	2,4
C-15-3	3	335,988	0,31	0	C-25-3	3	283,172	tl	1,87
C-15-4	3	358,873	30,10	0	C-25-4	3	332,92	596,37	0
C-15-5	3	367,494	0,21	0	C-25-5	4	-	tl	-
C-15-6	3	267,776	11,16	0	C-25-6	4	366,521	129,83	0
C-15-7	3	299,433	0,58	0	C-25-7	3	384,794	2.606,07	0
C-15-8	3	235,804	7,92	0	C-25-8	3	441,657	93,31	0
C-15-9	3	304,293	18,79	0	C-25-9	4	375,377	tl	2,10
C-15-10	3	273,432	1,09	0	C-25-10	4	456,15	tl	4,22
C-15-11	3	341,874	0,72	0	C-25-11	4	423,319	tl	1,07
C-15-12	3	342,744	0,17	0	C-25-12	4	482,438	3,68	0
C-15-13	3	296,725	0,51	0	C-25-13	4	-	tl	-
C-15-14	3	395,146	0,18	0	C-25-14	4	441,563	103,16	0
C-15-15	3	264,536	0,63	0	C-25-15	3	444,491	132,04	0
C-15-16	3	273,333	3,38	0	C-25-16	4	428,867	tl	0,8
C-15-17	3	322,531	0,15	0	C-25-17	4	555,488	336,12	0
C-15-18	3	356,273	0,25	0	C-25-18	4	472,169	12,33	0
C-15-19	3	241,497	1,90	0	C-25-19	4	474,434	35,73	0
C-15-20	3	267,925	0,53	0	C-25-20	4	432,194	tl	2,94

Tabelle D.6: Ergebnisse für 10, 15, 20 und 25 Kunden auf den UK-C Instanzen

Inst	V	Branch-and-Price		
		BUB	T	G(%)
C-50-1	7	-	tl	-
C-50-2	7	-	tl	-
C-50-3	8	-	tl	-
C-50-4	7	-	tl	-
C-50-5	6	-	tl	-
C-50-6	8	-	tl	-
C-50-7	7	-	tl	-
C-50-8	7	-	tl	-
C-50-9	7	824,031	tl	0,76
C-50-10	7	-	tl	-
C-50-11	7	-	tl	-
C-50-12	6	-	tl	-
C-50-13	7	-	tl	-
C-50-14	7	-	tl	-
C-50-15	6	-	tl	-
C-50-16	6	-	tl	-
C-50-17	7	-	tl	-
C-50-18	8	-	tl	-
C-50-19	7	-	tl	-
C-50-20	7	-	tl	-

Inst	V	Branch-and-Price		
		BUB	T	G(%)
C-75-1	11	-	tl	-
C-75-2	11	-	tl	-
C-75-3	10	-	tl	-
C-75-4	11	-	tl	-
C-75-5	10	-	tl	-
C-75-6	11	-	tl	-
C-75-7	11	-	tl	-
C-75-8	11	-	tl	-
C-75-9	10	-	tl	-
C-75-10	10	-	tl	-
C-75-11	10	-	tl	-
C-75-12	10	-	tl	-
C-75-13	10	-	tl	-
C-75-14	10	-	tl	-
C-75-15	10	-	tl	-
C-75-16	10	-	tl	-
C-75-17	10	-	tl	-
C-75-18	10	-	tl	-
C-75-19	10	-	tl	-
C-75-20	11	-	tl	-

Tabelle D.7: Ergebnisse für 50 und 75 Kunden auf den UK-C Instanzen

Inst	V	Fukasawa et al. (2018)			Branch-and-Price		
		BUB	T	G(%)	BUB	T	G(%)
C-10-1	2	210,18	0,1	0	210,182	0,08	0
C-10-2	2	271,93	0,3	0	271,93	0,08	0
C-10-3	2	229,18	0,1	0	229,177	0,06	0
C-10-4	2	230,52	0,2	0	230,519	0,05	0
C-10-5	2	205,49	0,1	0	205,491	0,07	0
C-10-6	2	255,82	0	0	255,819	0,07	0
C-10-7	2	217,79	0,1	0	217,788	0,06	0
C-10-8	2	251,29	0	0	251,294	0,04	0
C-10-9	2	186,04	0,1	0	186,041	0,04	0
C-10-10	2	231,62	0	0	231,619	0,04	0
C-20-1	5	432,35	90,3	0	432,345	1,21	0
C-20-2	5	448,29	4,4	0	448,289	4,17	0
C-20-3	5	287,48	4,1	0	287,037	5,19	0
C-20-4	5	434,23	4,4	0	434,232	7,52	0
C-20-5	5	381,7	4,5	0	381,701	7,60	0
C-20-6	5	444,35	28,9	0	444,354	2,39	0
C-20-7	5	317,73	63,1	0	317,728	168	0
C-20-8	5	410,35	0,5	0	410,347	0,83	0
C-20-9	5	421,39	0,5	0	421,385	0,92	0
C-20-10	5	384,88	13,7	0	384,878	1,40	0
		Fukasawa et al. (2017)			Branch-and-Price		
C-25-1	6	417,59	163	0	417,761	172,74	0
C-25-2	6	468,95	548	0	468,95	442,81	0
C-25-3	6	328,47	tl	0,1	328,466	2.796,7	0
C-25-4	6	391,34	tl	1,3	389,564	97	0
C-25-5	6	458,55	tl	-	453,863	tl	1,38
C-25-6	6	418,12	135	0	418,123	340,33	0
C-25-7	6	482,31	tl	0,4	481,749	tl	1,97
C-25-8	6	499,18	30,5	0	499,177	34,98	0
C-25-9	6	412,51	520	0	412,511	426,33	0
C-25-10	6	513,37	tl	0,4	513,368	3.089,32	0
C-50-1	12	1.198,26	tl	32,8	-	tl	-
C-50-2	12	1.227,79	tl	38,6	-	tl	-
C-50-3	12	1.015,2	tl	19	-	tl	-
C-50-4	12	1.003,43	tl	0,5	1.003,52	tl	0,63
C-50-5	12	925,85	tl	1	926,784	tl	0,63
C-50-6	12	2.607,89	tl	228,5	-	tl	-
C-50-7	12	852,67	tl	0	857	tl	0,88
C-50-8	12	1.233,17	tl	56	-	tl	-
C-50-9	12	962,69	tl	0,2	962,691	550,38	0
C-50-10	12	8.330,19	tl	-	-	tl	-

Tabelle D.8: Vergleich zwischen dem BCP-Algorithmus von (Fukasawa et al., 2017, 2018) und dem Branch-and-Price-Algorithmus auf den UK-C Instanzen