Behandlung des Dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern und Ladebedingungen mittels spezieller Statusgraphen

Dissertation

Dipl. Wirt.-Math. Anke Fabri

09.01.2008

Technische Universität Dortmund
Wirtschafts- und Sozialwissenschaftliche Fakultät

Vorwort

Das Vehicle Routing Problem(VRP) gehört zu den wichtigsten und am meisten erforschten kombinatorischen Optimierungsproblemen. Dank fortschreitender Kommunikations- und Informationstechnologie können Informationen inzwischen zu vertretbaren Kosten in Echtzeit übertragen und in angemessener Zeit verarbeitet werden. Lösungsverfahren für dynamische Entscheidungsprobleme werden damit nicht zuletzt auch wegen der enormen ökonomischen Bedeutung des Transports in der Praxis nachgefragt.

In der vorliegenden Arbeit wird eine Heuristik zur Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern entwickelt, wobei für jeden Auftrag zwei strikte Zeitfenster zu erfüllen sind. Zusätzlich werden auch die Problemvarianten mit den Ladebedingungen Backhauls, LIFO und LIFO-q untersucht und mit der Heuristik gelöst. Die Heuristik besteht aus einem sehr schnellen Algorithmus zur Lösung des Single Vehicle Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern und einer heuristischen Zuordnung der Aufträge zu den Fahrzeugen. Das Single Vehicle Routing Problem wird angegangen, indem der A*-Algorithmus mit besonderen Schätzfunktionen auf einen speziellen Zustandsgraphen angewandt wird. Bei den entwickelten Benchmark-Datensätzen wurden praxistaugliche Rechenzeiten von wenigen Sekunden für interessante Problemstellungen mit mehreren hundert anzufahrenden Orten erreicht. Die Arbeit wird abgerundet durch eine genaue Analyse der entwickelten Graphenstruktur, auch im Hinblick auf die sich ergebenden Änderungen bei Einführung von Ladebedingungen.

Ich möchte mich an dieser Stelle bei Prof. Peter Recht und Prof. Bernhard Fleischmann für die Betreuung während der Promotion bedanken. Weiterhin danke ich Dennis Müller und Reinhard und Ursula Fabri für die Unterstützung der Promotion, sowie Sonja Fabri und Werner und Sina Feilhauer für das fleißige Korrekturlesen.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Motivation, Ziel und Thema

Das *Vehicle Routing Problem* (VRP) gehört zu den wichtigsten und am meisten erforschten kombinatorischen Optimierungsproblemen. Es beinhaltet folgendes Entscheidungsproblem: Gegeben sind ein Fuhrpark mit m Fahrzeugen mit bestimmter Kapazität und einem Depot a sowie n anzufahrende Orte mit definierten Nachfragemengen, die von den Fahrzeugen beliefert werden sollen. Gesucht werden kostenminimale Touren für die Fahrzeuge, so dass jede Tour im Depot beginnt und endet und die Nachfrage an jedem Ort von genau einem Fahrzeug bedient wird, ohne die Kapazitäten der Fahrzeuge zu überschreiten. Abbildung 1.1 zeigt ein Beispiel mit drei Fahrzeugen und acht zu bedienenden Orten.

Vor fast 50 Jahren haben Dantzig und Ramser mit[19] die erste Studie zu einem Vehicle Routing Problem veröffentlicht - es ging um den Transport von Benzin zu Tankstellen. Sie publizierten ein mathematisches Modell des Vehicle Routing Problems und eine Heuristik zur Lösung, die ein paar Jahre später von Clarke und Wright mit der Veröffentlichung des Savings-Verfahrens verbessert wurde [16].

Danach wurden hunderte von Modellen und Verfahren zur Behandlung des Vehicle Routing Problems entwickelt und vorgeschlagen, Dutzende Anwenderprogramme bedienen sich der OR-Verfahren zur Lösung von Vehicle Routing Problemen. Das große Interesse an diesem Problem basiert einerseits auf der praktischen Relevanz der Ergebnisse, andererseits liegt der Reiz in der enormen Komplexität des Problems. Wegen dieser Komplexität wurden zunächst Heuristiken entwickelt, danach erst beschäftigte man sich mit mathematischer Programmierung und der optimalen Lösung der Probleme. Bis heute ist jedoch eine optimale Lösung nur für kleine Probleme in akzeptabler Rechenzeit möglich - für praktische Anwendungen wird daher weiterhin in den meisten Fällen auf Heuristiken zurückgegriffen, die dank des großen Forschungsinteresses an Vehicle Routing Problemen sehr schnell gute Lösungen finden können.

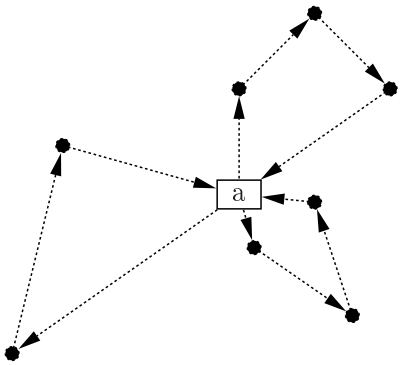

Abbildung 1.1: Beispiel für ein VRP mit drei Fahrzeugen und acht Orten.

Mit der fortschreitenden technischen Entwicklung sind weitere Entscheidungsprobleme in den Vordergrund gerückt. Wegen der rasanten Entwicklung der Hardware werden auch kompliziertere Varianten des Vehicle Routing Problems berechenbar. Dazu zählen zum Beispiel das *Pickup and Delivery Vehicle Routing Problem* (PDVRP), bei dem jeder Auftrag zunächst von einem Ort abgeholt und dann zu seinem Bestimmungsort transportiert werden muss, oder das *Vehicle Routing Problem mit Zeitfenstern* (VRPTW), wo zu jedem Auftrag ein Zeitfenster zur Abholung bzw. zur Lieferung vorgegeben ist. Dank fortschreitender Kommunikations- und Informationstechnologie können Informationen inzwischen zu vertretbaren Kosten in Echtzeit übertragen und in angemessener Zeit verarbeitet werden. Lösungsverfahren für dynamische Entscheidungsprobleme werden damit nicht zuletzt auch wegen der enormen ökonomischen Bedeutung des Transports in der Praxis nachgefragt - leistungsfähige Algorithmen für das dynamische Vehicle Routing müssen entwickelt werden.

Die vorliegende Arbeit beschäftigt sich daher mit dem *dynamischen Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern* (dPDVRPTW). Ziel ist es, einen in der Praxis anwendbaren Algorithmus zur Verfügung zu stellen, der eine Lösung zu folgendem Entscheidungsproblem liefert:
Gegeben sei ein Fuhrpark von m Fahrzeugen. Jedes Fahrzeug wird charakterisiert durch seine Kapazität, sein Depot und das Arbeitszeitfenster, in dem es zur Bearbeitung von Aufträgen zur Verfügung steht. Mit diesem Fuhrpark sollen während einer bestimmten Planungsperiode (zum Beispiel einem Arbeitstag) Aufträge abgearbeitet werden, wobei für jeden Auftrag Pickup-Ort und -Zeitfenster, Delivery-Ort und -Zeitfenster, Be- und Entladezeiten sowie die im Fahrzeug benötigte Kapazität gegeben sind. Die Aufträge sind nicht zu Beginn der Planungsperiode bekannt, sondern gehen erst während der Planungsperiode ein und sollen „online“ in die Planung integriert werden. Ruft ein Kunde an und möchte einen Auftrag erteilen, so soll dem Kunden noch während des Telefonats mitgeteilt werden, ob der Auftrag angenommen wird. Eine zulässige Lösung für das Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern mit allen bisher angenommenen Aufträgen zuzüglich des neuen Auftrags muss daher innerhalb von wenigen Sekunden gefunden werden. Ziel des Unternehmens soll es primär sein, möglichst

viele Aufträge anzunehmen, da jeder angenommene Auftrag einen positiven Deckungsbeitrag leistet. Als sekundäres Ziel wird die Gesamtfahr- und -wartezeit der Fahrzeuge zur Abarbeitung aller akzeptierten Aufträge minimiert und damit die Höhe der Deckungsbeiträge aller Aufträge maximiert.

Dieses dynamische Entscheidungsproblem lässt sich leicht auf ähnliche Entscheidungsprobleme übertragen, die dementsprechend ebenfalls mit dem zu entwickelnden Algorithmus gelöst werden können:

- die Steuerung eines Fuhrparks von Anruf-Sammel-Taxen: die Kunden fahren nicht wie üblich allein in einem Taxi, sondern erlauben die Mitnahme weiterer Fahrgäste. Ein Kunde ruft beim Unternehmen an und bestellt eine Fahrt von Ort A nach Ort B, wobei er für Ankunft und Abfahrt ein Zeitfenster nennen kann, so dass er eventuelle Termine garantiert pünktlich wahrnehmen kann. Während des Telefonats werden die Kundenwünsche in den Computer eingegeben, so dass der Algorithmus sofort eine zulässige Lösung ausgeben und dem Kunden damit zugesagt werden kann. Wird keine zulässige Lösung gefunden, muss der Auftrag abgelehnt werden.

- die Organisation eines Behinderten- oder Seniorenfahrdienstes: die meisten dieser Fahrdienste können im Moment nur mindestens einen Tag im Voraus gebucht werden. Der Unterschied zu normalen Anruf-Sammel-Taxen liegt in der spezielleren Zuordnung der Kunden auf die Fahrzeuge, da zum Beispiel die Mitnahme von Rollstühlen nur begrenzt oder in einigen Fahrzeugen auch gar nicht möglich ist.

- die Steuerung von Fahrzeugen in Lagerhäusern: betrachtet man ein Hochregallager, aus dem Ware ausgeliefert werden soll, so können die einzelnen Teile einer zusammenzustellenden Lieferung als Aufträge angesehen werden, die jeweils von einem bestimmten, bekannten Platz im Hochregalsystem (Pickup) an eine vorgegebene Packstation (Delivery) geliefert werden sollen. Über die Zeitfenster zur Ablieferung an der Packstation kann sichergestellt werden, dass immer nur genau ein Kundenauftrag (möglicherweise bestehend aus mehreren Teilen aus dem Lager) an einer Packstation angeliefert und eingepackt wird.

Zusätzlich soll der zur Verfügung gestellte Algorithmus im Gegensatz zu vielen Metaheuristiken ohne die Einstellung von problemspezifischen Parametern auskommen und leicht auf ähnliche Entscheidungsprobleme übertragbar sein. Unter ähnliche Entscheidungsprobleme fallen dabei Entscheidungsprobleme mit

- einer oder mehreren zusätzlichen Nebenbedingungen, z. B. Ladebedingungen oder Verbot des Transports bestimmter Güter-Kombinationen in einem Fahrzeug
- mehrdimensionalen Kapazitätsbeschränkungen der Fahrzeuge, z. B. Gewicht und Volumen

- anders strukturierten Kosten, z. B. bei einer Trennung von Fahrzeit und Fahrtkosten
- einer anderen Zielfunktion, z. B. Ablehnung von Kunden mit ungenügendem Deckungsbeitrag.

Der zu entwickelnde Algorithmus soll also für bedeutende praxisrelevante Entscheidungsprobleme einsetzbar und gut auf ähnliche real auftretende Entscheidungsprobleme zu übertragen sein.

In der Praxis sind häufig Ladebedingungen zu beachten, die für die Routenplanung bedeutende Restriktionen implizieren. Die Auswirkungen verschiedener Ladebedingungen auf das Laufzeitverhalten des Algorithmus und die Lösungen sollen daher ebenfalls untersucht und analysiert werden.

1.2 Einordnung in die Literatur

Dynamische Vehicle Routing Probleme wurden wegen ihrer hohen Komplexität erst in den letzten 20 Jahren intensiv untersucht. Die rasante technologischen Entwicklung sowohl der Hardware als auch der kabellosen Datenübertragung machte die Forschung auf diesem Gebiet attraktiv, die enorme wirtschaftliche Bedeutung der Transportkosten sorgte innerhalb der verschiedensten Logistik-Konzepte für eine große Nachfrage entsprechender Lösungsansätze.

Psaraftis [60] verfasste 1988 einen ersten Überblick über dynamische Vehicle Routing Probleme. Er definierte ein Vehicle Routing Problem als dynamisch, wenn sich „die Eingabedaten während der Laufzeit des Algorithmus oder der Abarbeitung der Route ändern oder aktualisiert werden“[1]. Als Unterschied zum statischen Vehicle Routing stellt er heraus, dass die Zeit eine wichtige Rolle spielt, der Planungshorizont damit - zumindest theoretisch - unendlich sein kann, Informationen über die Zukunft unpräzise oder unbekannt sind und daher Ereignisse in der nahen Zukunft höheren Stellenwert bekommen. Ferner werden Mechanismen zur Aktualisierung der Informationen benötigt, schnellere Rechenzeiten sind unerlässlich. Das wiederholte Ändern der Zuordnung oder der Reihenfolge kann erwünscht sein, wobei eine unendliche Verschiebung von unattraktiven Aufträgen verhindert werden muss. Die Zielfunktion sowie die zeitlichen Restriktionen unterscheiden sich von ihren Äquivalenzen des Vehicle Routing Problems, auch die Flexibilität zur Anpassung der Flotte ist geringer und eine Übersättigung des Systems muss vermieden werden.[2]

Dynamische Vehicle Routing Probleme gibt es in mehreren Ausprägungen. Sie werden anhand folgender Fragestellungen unterschieden:

- Welche der Eingabedaten unterliegen der Dynamik? Die Dynamik kann z. B. bei den Fahrzeiten, der zu ladenden Menge, dem Ort des Auftrags oder sogar den kompletten Auftragseingängen eine Rolle spielen.

[1] [60], Seite 224f.
[2] [60], Seite 225ff.

- Welche Qualität hat die Dynamik? Liegen schon Informationen in Form von stochastischen Vorhersagen vor oder gibt es keine Informationen im Voraus?
- Welches Entscheidungsproblem liegt zu Grunde? Hier wird unterschieden zwischen dem Standard-Vehicle Routing Problem, dem Pickup and Delivery Vehicle Routing Problem, dem Vehicle Routing Problem mit Zeitfenstern etc.
- Welche Entscheidungsmöglichkeiten sind gegeben? Dürfen Anfragen abgelehnt werden oder müssen alle Aufträge mit möglichst hoher Zufriedenheit erledigt werden?

Bertsimas und Van Ryzin [8] publizierten 1991 die erste Analyse eines dynamischen Vehicle Routing Problems. Larsen [49] veröffentlicht 2001 ein Maß für den Grad der Dynamik, anhand dessen eine Einschätzung der zum Entscheidungsproblem passenden Algorithmen vorgenommen werden soll. Luo gibt in seiner Dissertation 2006 einen ausführlichen Überblick über Heuristiken für das dynamische Dial-A-Ride Problem. Das Dial-A-Ride Problem beschreibt den Transport von Personen mit einer Fahrzeugflotte, wobei mehrere Personen mit unterschiedlichen Abfahrt- und Zielorten in einem Fahrzeug transportiert werden dürfen. Die Auftragseingänge erfolgen dynamisch während der Planungsperiode. Für das Entscheidungsproblem mit strikt einzuhaltenden Zeitfenstern wird nur eine einzige Heuristik aufgeführt: Attanasio, Cordeau, Ghiani und Laporte [4] veröffentlichten 2004 eine parallele Tabu-Suche, die nach jedem neuen Auftragseingang irgendeine zulässige Lösung sucht und anschließend mit der von Gilbert und Laporte für das statische Dial-A-Ride-Problem entwickelten Tabu-Suche verbessert. Caramia, Italiano, Oriolo, Pacifici und Perugia [11] veröffentlichten 2001 eine Heuristik für das Dial-A-Ride-Problem mit strikten Zeitfenstern, wobei sie eine heuristische Zuordnung mit einer optimalen Routenplanung kombinieren. Die optimale Routenplanung wird mit Hilfe der von Psaraftis [60] eingeführten Statusvektoren und dem A*-Algorithmus gelöst. In beiden Problemen dürfen die Kunden nur ein Zeitfenster entweder für Pickup oder für Delivery angeben, außerdem wird die Fahrzeit beschränkt.

Die für das dynamische Vehicle Routing Problem veröffentlichten Analysen und Algorithmen behandeln in den meisten Fällen kein Pickup und Delivery. Das am besten untersuchte Problem in diesem Bereich ist das dynamische Single Vehicle Routing Problem, das zwar in der Realität selten benötigt wird, aber als Subproblem im multi-Vehicle Routing Problem auftaucht.

In der Literatur ist nur ein Algorithmus für das dynamische Pickup and Delivery Vehicle Routing mit zwei strikten Zeitfenstern pro Auftrag vorhanden: Mitrovic-Minic, Krishnamurti und Laporte [52] veröffentlichten 2004 eine Heuristik für das Entscheidungsproblem eines Kurierdienstleisters, wobei sie mit zwei Planungshorizonten arbeiten: im zeitnahen Bereich wird eine kostenminimierende Routenplanung durchgeführt, während die Planungen für die weitere Zukunft auf das bestmögliche Einfügen von zusätzlich ankommenden Aufträgen zielen. Die Besonderheit an dem betrachteten Entscheidungsproblem liegt darin, dass die Fahrzeugkapazitäten nicht bindend sind.

Für das dynamische Pickup and Delivery Vehicle Routing mit zwei strikt einzuhaltenden Zeitfenstern pro Auftrag ist noch keine zufriedenstellende Lösung gefunden worden. In [26] wird der Ansatz von Caramia et al. zwar auf dieses Entscheidungsproblem übertragen, für einen Einsatz in einer dynamischen Umgebung sind die Rechenzeiten jedoch viel zu hoch. Die vorliegende Arbeit wird diese Lücke schließen.

Angaben zur Auswirkung von Ladebedingungen auf die Rechenzeiten wurden bisher nicht in der Literatur gefunden.

1.3 Überblick über die Arbeit

In der vorliegenden Arbeit wird eine Heuristik zur Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern entwickelt, wobei für jeden Auftrag zwei strikte Zeitfenster zu erfüllen sind.

Dazu wird im folgenden Kapitel zunächst ein Überblick über die Entwicklung der Vehicle Routing Probleme gegeben. Anschließend folgt die exakte Formulierung des zu lösenden dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern, indem das zu Grunde liegende Entscheidungsproblem zunächst verbal beschrieben und dann ein zugehöriges mathematisches Modell vorgestellt wird. In Abschnitt 2.4 wird der von Caramia et al. entwickelte Algorithmus zur Lösung des dynamischen Dial-A-Ride Problems mit einem Zeitfenster und einer Beschränkung der Fahrzeit vorgestellt und kritisch hinterfragt. Dieser Algorithmus besteht aus einer Heuristik für die Zuordnung der während des Planungszeitraums neu eingehenden Aufträge zu den Fahrzeugen sowie einer optimalen Lösung des Routenplanungsproblems für jedes Fahrzeug. Das *Routenplanungsproblem* besteht darin, für ein Fahrzeug eine kostenminimale Route zur Bedienung aller dem Fahrzeug zugeordneten Aufträge zu finden, wobei die Route im Ort der aktuellen Fahrzeugposition beginnt und in einem beliebigen Delivery-Ort der dem Fahrzeug zugeordneten Aufträge endet und alle Kapazitäts- und Zeitfensterrestriktionen erfüllen muss. Anhand dieses Algorithmus werden Ansätze zur Entwicklung eines eigenen Algorithmus für das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern abgeleitet. In Abschnitt 2.5 werden eine Testumgebung präsentiert und die Herleitung der Testdatensätzen erläutert, bevor mit der Entwicklung eines eigenen Algorithmus begonnen wird.

Kapitel 3 behandelt die Thematik der Routenplanung für ein einzelnes Fahrzeug bei gegebenen Aufträgen. Dazu wird zunächst das von Caramia et al. eingesetzte Verfahren zur Lösung des Routingproblems analysiert, das aus der Anwendung des A*-Algorithmus zur Bestimmung von kürzesten Wegen in einem speziellen Statusvektorbaum basiert. Im Abschnitt 3.3 wird dann eine alternative Graphenstruktur entwickelt, auf der das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern schneller lösbar ist. Diese Verbesserung wird mit theoretischen Untersuchungen unterlegt und anschließend anhand der Ergebnisse von Testläufen untermauert. Abschließend wird im Abschnitt 3.4 der Ablauf des A*-Algorithmus

beschleunigt, indem spezielle Schätzfunktionen eingeführt und an die Gegebenheiten des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern angepasst werden. Auch die algorithmischen Konsequenzen aus diesen Veränderungen werden anhand umfangreicher Testläufe analysiert.

Im vierten Kapitel liegt der Schwerpunkt auf der Zuordnung der Aufträge zu den Fahrzeugen, wobei zur Bewertung der Güte einer Zuordnung auf die optimale Routenplanung zurückgegriffen wird. Zuerst wird das entsprechende Verfahren von Caramia et al. erläutert, bevor zwei neue Ansätze zur Beschleunigung der Zuordnung vorgestellt werden. Der erste Ansatz basiert darauf, möglichst schnell eine zulässige Lösung zu finden, um dem Auftraggeber im Falle der Existenz einer solchen Lösung schnell zusagen zu können. Im zweiten Ansatz wird versucht, schon vor der Zuordnung eines Auftrags zu einem Fahrzeug zu erkennen, ob eine zulässige Route überhaupt existieren kann. Beide Ansätze werden anhand von Testläufen analysiert und bewertet. Da es sich bei der Zuordnung um eine „einfache“ Heuristik handelt, sind im Allgemeinen keine besonders guten Lösungen zu erwarten. Im Abschnitt 4.2 wird daher ein Verfahren zur Verbesserung der Zuordnung beschrieben, das immer dann zum Einsatz kommt, wenn aktuell keine neuen Auftragseingänge vorliegen und die Rechnerkapazität daher ungenutzt bleibt. Dazu wird ein Verfahren der Tabu Suche eingesetzt. Mit Hilfe umfangreicher Testläufe werden die Ergebnisse des Algorithmus unter Verwendung dieses Verfahrens analysiert.

Im fünften Kapitel werden die Auswirkungen von in der Praxis üblicherweise geltenden Ladebedingungen auf die Lösbarkeit des Problems untersucht. Dazu wird die Veränderung der benötigten Rechnerkapazität im Vergleich zum dynamischen Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern ohne Ladebedingungen sowohl theoretisch analysiert als auch die Auswirkungen für die Testdatensätze dokumentiert.

Abschließend wird ein Fazit gezogen.

Kapitel 2

Das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern

In dieser Arbeit wird das dynamische Pickup and Delivery Vehicle Routing Problem mit zwei Zeitfenstern behandelt. Dieses Kapitel beschreibt das zu Grunde liegende Entscheidungsproblem und gibt grundlegende Informationen für die weitere Arbeit. Dazu werden hier zunächst das Entscheidungsproblem des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern skizziert und die benötigten Begriffe eingeführt. Anschließend wird ein Überblick über die historische Entwicklung der Forschung zu Vehicle Routing Problemen gegeben, wobei sowohl die Entwicklung der Lösungsverfahren zum Vehicle Routing Problem als auch die Entwicklung der bearbeiteten allgemeineren Entscheidungsprobleme mit den dazugehörigen Lösungsverfahren skizziert werden. Im Abschnitt 2.3 wird das dynamische Pickup and Delivery Vehicle Routing Problem mit zwei strikten Zeitfenstern exakt formuliert und ein zugehöriges mathematisches Modell angegeben. Anschließend wird in Abschnitt 2.4 der in [11] entwickelte Algorithmus zur Lösung des Dial-A-Ride Problems mit einem Zeitfenster und einer Beschränkung der maximalen Fahrzeit skizziert, um in den folgenden Kapiteln einen effizienten Algorithmus auf seiner Basis zu entwickeln. Der letzte Abschnitt behandelt die für den Test des Algorithmus benötigte „Peripherie“: die erstellten Testdatensätze werden erläutert und die Testumgebung vorgestellt.

2.1 Grundlegende Begriffe

In diesem Abschnitt werden die in dieser Arbeit verwendeten grundlegenden Begriffe des Vehicle Routing definiert. Dazu wird zunächst das Entscheidungsproblem „dynamisches Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern“ skizziert:
Ein Fuhrunternehmen mit einem Fuhrpark von m Fahrzeugen bekommt alle Aufträge „online“ während der Planungsperiode. Das Unternehmen möchte seinen Deckungsbeitrag maximieren, indem es zunächst möglichst viele Aufträge annimmt und im zweiten Schritt die Gesamtfahrzeit aller Fahrzeuge zur Bedienung der Aufträge minimiert. Jedes Fahrzeug wird charakterisiert durch seine Kapazität, sein Depot und das Arbeitszeitfenster, in dem es zur Bearbeitung von Aufträgen zur Verfügung steht. Ein Auftragseingang geschieht telefonisch, per Fax oder per Internet, und muss sofort angenommen oder abgelehnt werden. Es liegen keine Informationen über zu erwartende Auftragseingänge vor. Ein Auftrag wird charakterisiert durch seinen Pickup-Ort und sein Pickup-Zeitfenster, seinen Delivery-Ort und sein Delivery-Zeitfenster, die benötigten Servicezeiten für das Be- und Entladen sowie die Kapazität, die beim Transport in einem Fahrzeug in Anspruch genommen wird. Alle angenommenen Aufträge müssen den Fahrzeugen zugeordnet und für jedes Fahrzeug eine Route gefunden werden, wobei die Kapazitäten der Fahrzeuge nicht überschritten, die Aufträge jeweils zuerst abgeholt und dann ausgeliefert und die Zeitfenster der Aufträge und Fahrzeuge eingehalten werden.

Die benötigten Begriffe werden definiert als:

Definition 2.1 (Begriffe des Vehicle Routing)

Planungsperiode: *Die Planungsperiode ist der Zeitraum, für den die Planungen durchgeführt werden sollen, d. h. der Zeitraum, in dem optimiert wird.*
Sie umfasst typischerweise die Arbeitszeitfenster aller Fahrzeuge.

Arbeitszeitfenster: *Das Arbeitszeitfenster eines Fahrzeugs umfasst den Zeitraum, in dem dieses Fahrzeug zur Bearbeitung von Aufträgen genutzt werden kann.*

Kapazität des Fahrzeugs: *Die Kapazität eines Fahrzeugs beschreibt die maximal gleichzeitig in dem Fahrzeug transportierbare Menge.*
Sie ist entweder als Gewicht oder als Volumen angegeben.

Depot: *Das Depot eines Fahrzeugs ist der Ort, in dem es sich zu Beginn seines Arbeitszeitfensters befindet.*

Tour: *Eine Tour ist eine Rundreise durch eine bestimmte Anzahl von Orten, die im Depot des Fahrzeugs beginnt und endet und alle Kapazitäts- und Zeitfensterrestriktionen sowohl des Fahrzeugs als auch der zu den Orten gehörigen Aufträge einhält.*

Route: *Eine Route ist eine Sequenz von bestimmten Orten, die zu einer bestimmten Zeit in einem bestimmten Ort mit bestimmter bereits belegter Fahrzeugkapazität beginnt und*

die Kapazitäts- und Zeitfensterrestriktionen sowohl des Fahrzeugs als auch der zu den Orten gehörigen Aufträge einhält. Der Endort der Route ist nicht festgelegt.

Auftrag: *Ein Auftrag besteht aus einem Pickup- und einem Delivery-Ort, einem Pickup- und einem Delivery-Zeitfenster, je einer Servicezeit für das Be- und Entladen und der Auftragskapazität.*

Pickup-Ort, Delivery-Ort: *Der Pickup-Ort ist der Ort, an dem der Auftrag eingeladen werden soll. Der Delivery-Ort bezeichnet den Ort, an dem der Auftrag abgeliefert werden soll.*

Reihenfolgebedingung: *Die Reihenfolgebedingung besagt, dass der Pickup-Ort eines Auftrags vor dessen Delivery-Ort angefahren werden muss.*

Auftragskapazität: *Die Auftragskapazität gibt an, wie viel Kapazität des Fahrzeugs der Auftrag beim Transport beansprucht.*
Sie ist in der gleichen Maßeinheit angegeben wie die Fahrzeugkapazität.

Pickup-Zeitfenster, Delivery-Zeitfenster: *Das Pickup-Zeitfenster beinhaltet den Zeitraum, in dem das Fahrzeug beim Pickup-Ort ankommen muss. Das Delivery-Zeitfenster beschreibt den Zeitraum, in dem der Delivery-Ort erreicht werden muss.*

Servicezeit für Pickup bzw. Delivery: *Die Servicezeit für Pickup (Delivery) gibt an, wie viel Zeit am Pickup-Ort (Delivery-Ort) für das Beladen (Entladen) des Fahrzeugs und alle zu erledigenden Formalitäten einzukalkulieren ist.*

Auftragseingang: *Der Auftragseingang ist der Prozess, bei dem ein Kunde Kontakt zum Unternehmen aufnimmt, um einen neuen Auftrag an das Unternehmen zu vergeben. Dabei werden alle Charakteristika des Auftrags angegeben.*

online: *Der Begriff „online“ beschreibt den Umstand, dass die Auftragseingänge während der Planungsperiode zu beliebigen, vorher nicht bestimmbaren Zeitpunkten eingehen und die Aufträge sofort (möglichst innerhalb weniger Sekunden) angenommen oder abgelehnt werden müssen.*

Entfernung zwischen zwei Orten: *Die Entfernung zwischen zwei Orten μ und ν gibt die Fahrzeit für den kürzesten Weg von μ nach ν an.*
Der kürzeste Weg zwischen diesen Orten wird als bekannt vorausgesetzt und muss nicht berechnet werden.

Mit Hilfe dieser Begriffe kann im Folgenden das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern exakt formuliert werden. Zuvor wird jedoch noch die historische Entwicklung in der Beschäftigung mit Vehicle Routing Problemen skizziert.

2.2 Historische Entwicklung in der Beschäftigung mit Vehicle Routing Problemen

In diesem Abschnitt wird die historische Entwicklung der Lösungsverfahren zum Vehicle Routing Problem aufgezeigt. Im Abschnitt 2.2.1 werden einige spezielle Entscheidungsprobleme aus dem Bereich des Vehicle Routing eingeführt und die historische Entwicklung in ihrer Bearbeitung ebenfalls skizziert.

Dantzig und Ramser veröffentlichten 1959 in [19] die erste Studie zu einem Vehicle Routing Problem, in der sie den kostenminimalen Transport von Benzin zu Tankstellen betrachteten. Sie publizierten ein mathematisches Modell des Vehicle Routing Problems und eine Heuristik zur Lösung, die 1964 von Clarke und Wright durch das Savings-Verfahrens weiterentwickelt wurde [16].

Es folgten weitere Entwicklungen von Verfahren zur Konstruktion von Routen, zum Beispiel von Gaskell 1967 [31] und Yellow 1970 [84], die mit Hilfe eines Skalars die Entfernung zum Depot gewichten, oder 1991 von Altinkemer und Gavish [3], die in einer Iteration mehrere Routenpaare mit Hilfe eines Maximal-Matchings kombinieren und so die Rechenzeit verbessern. Zu den Konstruktionsverfahren wurden Verfahren zur Verbesserung von bereits gefundenen zulässigen Lösungen entwickelt: Nachdem Lin [51] 1965 den k-Austausch und die k-Optimalität für das Traveling Salesman Problem (TSP) eingeführt hatte, übertrugen Christofides und Eilon [13] 1969 die 3-Austausch-Methode auf das Vehicle Routing Problem. Russell [66] verbesserte 1977 die Methode, indem er einige Austauschschritte mit mehr als 3 Kanten hinzunahm.

Parallel zu den Konstruktions- und Verbesserungsverfahren wurden zweistufige Verfahren entwickelt. Das erste dieser Verfahren stammt von Tyagi [78] aus dem Jahr 1968 und ordnet die Aufträge den Routen zu, indem sukzessive der Ort mit der kürzesten Entfernung zum zuletzt eingefügten Ort noch mit in die Route aufgenommen wird. Die Routenplanung erfolgt dann durch Lösung eines Traveling Salesman Problems für jede der zuvor gebildeten Routen. Gillett und Miller veröffentlichten 1974 den sweep-Algorithmus, eingeteilt in den forward- und den backward-sweep. Beim forward-sweep werden die Orte den Touren zugeordnet, indem ausgehend von einem festgelegten Startort immer der noch nicht zugeordnete Ort mit dem - bei geographischer Vorstellung - kleinsten Winkel zum Depot ausgewählt wird. Ist die Kapazität des Fahrzeugs für diese Tour erschöpft, wird zusätzlich durch ein Austauschverfahren die Zuordnung der Orte zu den Touren verbessert. Anschließend werden die Touren mit Hilfe eines Algorithmus zur Lösung des Traveling Salesman Problems geplant. Dieses Verfahren wird für jeden Ort als Startort einmal ausgeführt. Beim anschließenden backward-sweep wird diese Prozedur wiederholt, jedoch wird der geographische Raum aller Orte nun gegen den Uhrzeigersinn durchsucht. Von allen so entstandenen zulässigen Lösungen wird die beste ausgewählt. Der sweep-Algorithmus liefert im Allgemeinen bessere Lösungen als das Verfahren von Clarke und Wright [16], benötigt dafür aber auch längere Rechenzeiten. Christofides, Mingozzi und Toth [15] entwickelten 1979 ebenfalls ein zweistufiges Verfahren: in der ersten Stufe werden Touren

erzeugt, indem so lange wie möglich Orte hinzugefügt werden, während in der zweiten Stufe im ersten Schritt möglichst alle noch freien Orte zugeordnet werden und anschließend eine Tour ausgewählt wird, deren Orte mit den anderen Touren ausgetauscht werden. Golden, Magnanti und Nguyen [39] modifizierten 1977 die Implementierung der Algorithmen von Clarke und Wright [16], Tyagi [78] und Gillett und Miller [35], indem sie effizientere Datenstrukturen verwendeten, und konnten so die Rechenzeiten weiter verringern.

Parallel zu den rein heuristischen Verfahren wurden auch Verfahren auf der Basis der mathematischen Programmierung entwickelt. Balinski und Quandt [6] führten 1964 ein ganzzahliges mathematisches Modell sowie ein Lösungsverfahren ein, das über die Enumeration aller zulässigen Touren und die Lösung eines Mengenüberdeckungsproblems den optimalen Tourenplan bestimmt. Fisher und Jaikumar [28] veröffentlichten 1981 eine Heuristik, die auf der optimalen Lösung des Zuordnungsproblems basiert, wobei die Kosten für die Zuordnung eines Ortes zu einem Fahrzeug eine Abschätzung für die Kosten des Umwegs dieses Fahrzeugs darstellen. Anschließend wird für jedes Fahrzeug das Traveling Salesman Problem optimal gelöst. Haimovich und Rinnooy Kan [40] entwickelten 1985 aus oberen und unteren Grenzen für die optimale Lösung ϵ-optimale Heuristiken. Christofides, Mingozzi und Toth führten 1979 in [15] eine auf einem Suchbaum basierende Modellierung sowie das Durchsuchen des unvollständigen Suchbaums als Lösungsverfahren ein. Zwei Jahre später veröffentlichten sie in [14] mehrere Branch and Bound Verfahren, die mit oberen und unteren Grenzen basierend auf minimalen spannenden Bäumen bei fixiertem Knotengrad eines Knotens und minimalen Touren bei exakt zu erfüllender Kapazitätsbeschränkung arbeiten. Bramel und Simchi-Levi transformierten 1995 in [9] das Problem in ein Capacitated Concentrator Location Problem, um die Zuordnung der Orte zu den Fahrzeugen besser lösen zu können. Als Kosten für die Zuordnung eines Ortes zu einem Fahrzeug setzen sie die Veränderungen in den Kosten der optimalen Touren. Altinkemer und Gavish zeigen 1987 in [2], wie ausgehend von einer Traveling Salesman Tour eine Lösung für das Vehicle Routing Problem mit vorher festgelegter maximaler Abweichung von der Optimallösung gefunden werden kann.

Christofides, Mingozzi und Toth gaben 1979 in [15] einen detaillierten Überblick über die bis dahin veröffentlichten Modelle und Heuristiken für das Vehicle Routing Problem, Christofides erweiterte diesen Überblick 1985 in [12] um exakte Verfahren und neuere Ergebnisse.

Wegen der Komplexität des Vehicle Routing Problems können die exakten Verfahren nur kleine Instanzen und relaxierte Probleme lösen. Die realen Entscheidungsprobleme werden mit Hilfe von Metaheuristiken angegangen. Metaheuristiken sind Algorithmen, die mittels spezieller Strategien ein Verfahren der lokalen Suche lenken, wobei sie zum Überwinden lokaler Optima auch schlechtere oder sogar unzulässige Lösungen akzeptieren. Im Allgemeinen werden bessere Lösungen gefunden als mit einfachen Heuristiken, die Rechenzeit ist allerdings auch höher. Zu diesen Metaheuristiken zählen Tabu Search, Simulated und Deterministic Annealing, Genetische Algorithmen und Ameisen-Algorithmen.

Das Verfahren des Simulated Annealing wurde erstmalig 1983 von Kirkpatrick, Gelatt und Vecchi [45] vorgestellt. Im Simulated Annealing werden ausgehend von einer zulässigen Startlösung andere zulässige Lösungen in einer Nachbarschaft der gegebenen Lösung gesucht. Aus der Nachbarschaft wird zufällig eine Lösung als neue beste Lösung gewählt, wobei schlechtere Lösungen mit bestimmter, im Verlauf der Suche abnehmender Wahrscheinlichkeit akzeptiert werden. Simulated Annealing wurde erstmalig 1991 von Robusté, Daganzo und Souleyrette [63] sowie von Alfa, Heragu und Chen [1] auf das Vehicle Routing Problem übertragen. Zur Definition der Suchumgebung wurden bekannte Verbesserungsheuristiken kombiniert: Robusté, Daganzo und Souleyrette tauschen Orte zwischen zwei Touren aus, drehen einen Teil der Tour um oder verschieben ihn innerhalb der Tour. Alfa, Heragu und Chen nutzen eine 3-opt Umgebung zur Suche nach besseren Lösungen. Osman [54] konnte die Ergebnisse 1993 weiter verbessern, indem er den Austausch von bis zu zwei Orten einer Tour mit bis zu zwei Orten einer anderen Tour als Suchumgebung festlegt und die zuerst gefundene bessere Lösung übernimmt.

Das Deterministic Annealing unterscheidet sich vom Simulated Annealing in der Akzeptanz von schlechteren Lösungen: die Akzeptanz hängt nur von der Qualität der Lösung, nicht vom Fortschreiten des Algorithmus ab. Dazu wurden 1990 von Dueck und Scheurer [23] bzw. 1993 von Dueck [22] zwei Strategien für die Akzeptanz einer Lösung entwickelt: die Akzeptanz bei Erreichen des besten bekannten Zielfunktionswerts abzüglich einer Konstante und die Akzeptanz bei Erreichen eines bestimmten Anteils des besten bekannten Zielfunktionswerts. Golden et al [38] wandten 1998 die letztere Version auf das Vehicle Routing Problem an.

Die Tabu Suche wurde 1989 von Glover [36], [37] veröffentlicht. Dabei handelt es sich um einen Verfahrenstyp, der aufbauend auf eine Startlösung nach besseren Lösungen in einer Umgebung sucht, wobei auf ein „Gedächtnis“ zurückgegriffen wird, so dass bestimmte Lösungen „tabu“ sind. Es werden allerdings zusätzliche Kriterien festgelegt, um eine gute Tabu-Lösung doch akzeptieren zu können. Ferner können Strategien zur Veränderung des Gedächtnis hinterlegt werden, um die Suche diversifizieren und intensivieren zu können. Willard [80] übertrug 1989 erstmalig die Tabu Suche auf das Vehicle Routing Problem. Osman [54] erreichte 1993 gute Ergebnisse, indem er ein oder zwei Orte entweder innerhalb einer Tour an andere Positionen verschiebt oder zwischen zwei Touren austauscht. Er unterscheidet dabei zwei Varianten: die zuerst gefundene bessere Lösung zu akzeptieren oder die ganze Umgebung nach der besten Lösung zu durchsuchen. Viele weitere vielversprechende Varianten wurden in den folgenden Jahren entwickelt. Eine Übersicht bis zum Jahr 2002 geben Gendreau, Laporte und Potvin [34].

Das Prinzip der genetischen Algorithmen wurde 1970 von Fraser [30] vorgestellt. Ausgehend von einer oder mehreren zulässigen Lösungen werden andere Lösungen erzeugt, indem die genetische Entwicklung nachgeahmt wird: aus einer Population von Lösungen wird durch Kombination und Mutation eine neue Menge von Lösungen erzeugt, die nach einem Selektionsschema andere Lösungen der Population ersetzen oder wieder verworfen werden. Van

Breedam [79] verglich 1996 den Effekt von Operatoren zum Erzeugen neuer zulässiger Lösungen für genetische Algorithmen und Simulated Annealing.

Colorni, Dorigo und Maniezzo [17] entwickelten 1991 den Typ des Ameisen-Algorithmus für das Traveling Salesman Problem mit dem Versuch, die Effizienz von natürlichen Ameisen zu kopieren. Ameisen hinterlassen bei der Futtersuche einen Botenstoff. Je kürzer der Weg zur Nahrungsquelle, desto intensiver ist der Geruch. Andere Ameisen orientieren sich bei der Nahrungssuche an diesen Gerüchen und finden so schnell den kürzesten Weg zur Nahrungsquelle, wobei sie wiederum den Botenstoff hinterlassen. Zur Lösung des Traveling Salesman Problems werden daher heuristisch Touren erzeugt, deren Kanten in Abhängigkeit von ihrer Weglänge „mit Botenstoff gewichtet“ werden. In den folgenden Iterationen werden wiederum heuristisch Touren erzeugt, wobei Kanten mit hoher „Botenstoff-Gewichtung“ mit höherer Wahrscheinlichkeit in die Touren eingehen. Kawamura [44] übertrug 1998 als erster den Ameisen-Algorithmus auf das Vehicle Routing Problem.

2.2.1 Spezielle Entscheidungsprobleme des Vehicle Routing Problems

Mit der Entwicklung der Lösungsmöglichkeiten für das Vehicle Routing Problem, basierend auf der fortschreitenden technologischen Entwicklung und der Forschung in diesem Gebiet, konnten auch komplexere, realitätsnahe Entscheidungsprobleme angegangen werden. Dazu gehören unter anderem das Vehicle Routing Problem mit Zeitfenstern (VRPTW), das Pickup and Delivery Vehicle Routing Problem (PDVRP), das dynamische Vehicle Routing Problem sowie spezielle Facetten dieser Probleme.

Das Vehicle Routing Problem mit Zeitfenstern wurde 1967 erstmalig von Pullen und Webb [62] beschrieben. Im Unterschied zum Vehicle Routing Problem wird jeder Ort mit einem Zeitfenster $[\underline{t}, \overline{t}]$ versehen, während dessen die Bedienung stattfinden muss. Es wird zwischen weichen und harten Zeitfenstern unterschieden: während bei harten Zeitfenstern eine Verletzung des Zeitfensters nicht zulässig ist, wird bei weichen Zeitfenstern eine Verletzung des Zeitfensters erlaubt, aber im Allgemeinen durch Kosten bestraft, indem die Zielfunktion aus einer Linearkombination aus Fahrtkosten und Strafkosten für verletzte Zeitfenster besteht. Der Planungshorizont entspricht im Allgemeinen dem Zeitfenster des Depots. Pullen und Webb schlugen 1967 einfache Heuristiken zur Lösung von Vehicle Routing Problemen mit harten Zeitfenstern vor. Knight und Hofer [46] veröffentlichten 1968 Fallstudien und eine von Hand zu lösende Heuristik, mit der schon effektive Einsparungen erreicht werden. Heuristiken zur Verbesserung von Touren, die auf der k-interchange-Heuristik basieren, werden 1977 von Russell [66] und 1986 von Baker und Schaffer [5] entwickelt. Solomon [71] übertrug 1987 einige Konstruktionsverfahren auf das Vehicle Routing Problem mit Zeitfenstern. 1988 veröffentlichten Solomon, Baker und Schaffer [70] die bereits 1976 von OR [53] entwickelte OR-opt-Methode zur Verbesserung von Touren: nur direkt hintereinander liegende Orte können verschoben und in dieser Reihenfolge wieder eingefügt werden. Basierend auf der OR-opt-Methode entwickel-

ten Thompson und Psaraftis [75] 1989 eine effiziente Heuristik mit zyklischen k-Austauschen und Potvin und Rousseau [57] eine effiziente Heuristik mit 2-opt und OR-opt-Austauschen. Kontoravdis und Bard [48] wandten 1995 eine GRASP-Heuristik (Greedy Randomized Adaptive Search Procedure) auf das Vehicle Routing Problem mit Zeitfenstern an. 1996 führten Potvin, Kervahut, Garcia und Rousseau [56] eine Tabu-Suche für das Vehicle Routing Problem mit Zeitfenstern ein, die für die von Solomon erzeugten Startlösungen gute Ergebnisse liefert. Solomon und Desrosiers gaben 1988 in [73] einen Überblick über Routing Probleme mit Zeitfenstern.

Die Entwicklung exakter Algorithmen basierte auf der Anwendung von Branch and Cut Verfahren auf das Vehicle Routing Problem mit Zeitfenstern. Dazu wurden effektive untere und obere Schranken sowie Strategien zum Verzweigen entwickelt. Als obere Schranken dienen die Ergebnisse der Heuristiken, während die unteren Schranken aus Dekompositionsansätzen oder aus Relaxierungen der Fahrzeugkapazitäten, der Zeitfenster oder der binären Variablen hervorgehen. Cordeau, Deaulniers, Desrosiers, Solomon und Soumis geben in [18] einen detaillierten Überblick über diese Lösungsverfahren.

Zur weiteren Annäherung an Tourenplanungsprobleme in der Realität wurden Pickup and Delivery Vehicle Routing Probleme betrachtet. Während im Vehicle Routing Problem nur die Belieferung von Orten aus einem Depot betrachtet wird, wird im Entscheidungsproblem des Pickup and Delivery Vehicle Routing die Abarbeitung von gegebenen Transport-Aufträgen behandelt, wobei für jeden Auftrag eine bestimmte Menge eines Gutes von einem bestimmten Ort abgeholt und zu einem anderen bestimmten Ort transportiert werden soll. Als Ziel werden im Allgemeinen sowohl die Minimierung der Anzahl der benutzten Fahrzeuge als auch die Minimierung der Fahrtkosten angegeben.

In der Literatur werden in den meisten Fällen Dial-A-Ride Probleme betrachtet. Dial-A-Ride Probleme beinhalten das Entscheidungsproblem der Personentransporte, wobei mehrere Personen mit unterschiedlichen Pickup- und Delivery-Orten zusammen in einem Fahrzeug transportiert werden dürfen. Ein Kunde meldet sich für einen Transport von einem bestimmten Pickup-Ort zu einem bestimmten Delivery-Ort an und kann eventuell weitere Vorgaben machen: für die Abfahrt des Kunden werden zum Teil Zeitfenster oder gewünschten Abfahrtszeiten mit maximalen Abweichungen gegeben, für die Ankunft am Zielort gelten gegebenenfalls maximale Dauern für die Fahrt oder gewünschte Ankunftszeiten als Restriktion. Ziel ist im Allgemeinen die Minimierung der Anzahl Fahrzeuge und die Minimierung der Fahrtkosten, aber auch die Kundenzufriedenheit kann - in Abhängigkeit von der Wartezeit des Kunden beim Abholen oder der Dauer der Fahrt - in die Zielfunktion eingehen.

Roy, Rousseau, Lapalme und Ferland [65], [64] veröffentlichten 1984 eine Heuristik, die für alle Fahrzeuge gleichzeitig Touren konstruiert und auch für dynamische Pickup and Delivery Vehicle Routing Probleme erweitert werden kann. Sexton und Bodin [68], [69] publizierten 1985 einen Algorithmus, der zuerst die Aufträge den Fahrzeugen zuordnet und anschließend die Tourenplanung mit Hilfe von Benders Decomposition durchführt, wobei noch Aufträge ge-

tauscht werden können. Dieser Ansatz wurde 1989 von Dumas, Desrosiers und Soumis [24] verbessert, indem sie einen Teil des Zuordnungsproblems mit in das Routing-Problem verlagern. Dazu werden kleine Cluster von gut zusammen zu bedienenden Aufträgen zusammengefasst und als untrennbar angesehen. Das Subproblem ist damit wiederum ein Pickup and Delivery Vehicle Routing Problem, wegen der Clusterbildung jedoch ein deutlich kleineres. Jaw, Odoni, Psaraftis und Wilson [43] veröffentlichten 1986 eine Heuristik für das statische Dial-A-Ride-Problem mit entweder einem festgelegten Pickup- oder einem festgelegten Delivery-Zeitpunkt pro Kunde, wobei zeitliche Beschränkungen für Verspätungen beim Pickup bzw. Delivery und für die Fahrtdauer gelten. Das Ziel ist die Minimierung einer gewichteten Summe aus Kundenunzufriedenheit und Fahrtkosten. Ihr Algorithmus basiert auf Konstruktions- und Einfüge-Verfahren. Psaraftis [59] entwickelte 1983 das erste Verfahren der lokalen Suche für das Pickup and Delivery Vehicle Routing Problem, indem er die k-Austausche auf dieses Entscheidungsproblem überträgt. Toth und Vigo [76] entwickelten 1997 eine Heuristik, bestehend aus einem Konstruktionsverfahren und einer Tabu Suche. Ein exakter Algorithmus für das Pickup and Delivery Vehicle Routing Problem wurde 1991 von Dumas, Desrosiers und Soumis [25] veröffentlicht. Sie benutzen eine Dantzig-Wolfe-Decomposition, eingebettet in ein Branch and Bound Verfahren. Das Master Problem besteht aus einer linearen Relaxation des Mengenüberdeckungsproblems, während im Subproblem zulässige Routen gesucht werden. Derigs und Metz [20] entwickelten ein exaktes Verfahren für den Spezialfall, dass alle Orte zunächst aus dem Depot beliefert werden und anschließend Ladung für den Transport zum Depot aufgeben, wobei einseitige Zeitrestriktionen eingehalten werden müssen. Ihr Verfahren basiert auf einem Matching-Algorithmus. Savelsbergh und Sol gaben 1995 in [67] einen Überblick über Charakteristika und Lösungsansätze von Pickup and Delivery Problemen.

Mit der fortschreitenden Entwicklung in der Optimierung der Vehicle Routing Probleme zusammen mit der technologischen Entwicklung sowohl im Bereich der Hardware als auch im Bereich der kabellosen Datenübermittlung rücken auch dynamische Entscheidungsprobleme in den Vordergrund. Wegen der immensen ökonomischen Bedeutung der Transportkosten und den gleichzeitig gestiegenen Anforderungen an die Flexibilität der Transportunternehmen ist die Nachfrage nach Lösungsmöglichkeiten der dynamischen Vehicle Routing Probleme rasant gestiegen.

Trotz nur weniger grundlegender Analysen der dynamischen Vehicle Routing Probleme wurden einige Verfahren zur Lösung bestimmter Entscheidungsprobleme entwickelt. Wegen der Komplexität der Probleme und der zeitlich beschränkten Rechenkapazität handelt es sich um Heuristiken, wobei sich einige Autoren auf den Fall des Single Vehicle Routing Problems beschränken. In diesem Entscheidungsproblem wird nur ein Fahrzeug betrachtet, die Zielfunktion minimiert daher nur die Fahrtkosten. Das Single Vehicle Routing Problem wird häufig als Subproblem bei der Lösung von dynamischen Vehicle Routing Problemen benutzt. Da viele der Heuristiken zur Lösung dynamischer Vehicle Routing Probleme praxisorientiert entwickelt wurden, behandeln sie eine Vielzahl von unterschiedlichen Entscheidungsproblemen

des dynamischen Vehicle Routings. Ein Großteil der Algorithmen behandelt dabei das dynamische Dial-A-Ride Problem für den Personentransport in einer Art Anruf-Sammel-Taxi, wobei die Dynamik in den meisten Fällen aus dem Eingang der Transportaufträge während der Planungsperiode besteht.

Wilson [82] veröffentlichte 1971 den ersten Artikel über Algorithmen für ein dynamisches Dial-A-Ride Problem, die er 1976 [83] um weitere Algorithmen und 1977 [81] um einen Bericht über eine reale Anwendung erweiterte. Psaraftis veröffentlichte 1980 einen Ansatz der dynamischen Programmierung, um ein Single Vehicle Dial-A-Ride-Problem ohne weitere Zeitbeschränkungen zu lösen. In der dynamischen Programmierung wird ein „System“ definiert, das durch bestimmte steuerbare Aktionen von einem Zustand in einen Folgezustand übergeht. Das Ziel besteht in der Bestimmung einer besten Sequenz dieser Aktionen zur Erreichung eines optimalen Endzustands. Psaraftis charakterisiert den Systemzustand mit Hilfe der Fahrzeugposition und der Bearbeitungsstatus der Aufträge, die Aktionen bestehen aus der Wahl des als nächstes anzufahrenden Ortes. Die Lösung besteht somit aus einer besten Sequenz der Pickup- und Delivery-Orte der Aufträge. Mit Hilfe eines „maximum position shift“ wird verhindert, dass ein Auftrag „unendlich“ oft nach hinten verschoben wird. Swihart und Papastavrou [74] veröffentlichten 1999 eine Analyse des dynamischen Single Vehicle Pickup and Delivery Problems. Im Jahr 2004 behandelten Attanasio, Cordeau, Ghiani und Laporte [4] eine parallele Tabu-Suche, die bei Eintreffen eines neuen Auftrags unterbrochen wird, irgendeine zulässige Lösung mit dem neuen Auftrag konstruiert und anschließend mit der Tabu-Suche fortfährt.

Für das dynamische Vehicle Routing Problem wurden unterschiedliche Facetten des Entscheidungsproblems untersucht. Brown und Graves [10] gehörten 1981 zu den der ersten, die ein Optimierungstool für die Echtzeit-Steuerung von Tanklastwagen veröffentlichten. 1988 veröffentlichte Psaraftis [60] einen Algorithmus zum Routing von Frachtschiffen, wobei sich die Planungsperiode mit Fortschreiten der Zeit verschiebt: es werden nur die Aufträge betrachtet, deren früheste Anfahrtszeit innerhalb der aktuellen Planungsperiode liegt. Fest zugeordnet werden sogar nur diejenigen Aufträge mit Anfahrtszeiten zu Beginn der aktuellen Planungsperiode. Gendreau, Guertin, Potvin und Taillard [32] veröffentlichten 1999 einen Algorithmus zur Behandlung des Entscheidungsproblems eines Kurier-Dienstleisters, dessen Aufträge online erscheinen und unter Einhaltung von weichen Zeitfenstern bedient werden müssen, d. h. die Verletzung der Zeitfenster ist gestattet, wird aber mit zusätzlichen Kosten bestraft. Zur Lösung wird das Verfahren der Tabu-Suche auf den dynamischen Fall angepasst. Mitrovic-Minic, Krishnamurti und Laporte [52] entwickelten 2004 für das Entscheidungsproblem mit harten Zeitfenstern aufbauend auf [60] eine Heuristik mit zwei verschiedenen Planungsperioden: in der kürzeren Planungsperiode werden die Fahrtkosten der Fahrzeuge minimiert, während die Optimierung in der längeren Planungsperiode darauf zielt, möglichst viele neu eingehende Aufträge annehmen zu können. Fleischmann, Gnutzmann und Sandvoß [29] veröffentlichten 2004 ein Verfahren zur ereignisorientierten Behandlung von Aufträgen aus einem elektronischen Marktplatz, wobei sowohl die Fahrzeiten als auch die Auftragseingänge dynamisch sind.

Gendreau, Guertin, Potvin und Séguin [33] publizierten 2006 eine Tabu-Suche zur Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit weichen Zeitfenstern. Die Nachbarschaft basiert auf zyklischem Tausch eines Auftrags.

Im Bereich des dynamischen Vehicle Routing sind noch einige Fragestellungen nicht bearbeitet. Es gibt zum Beispiel noch keinen Algorithmus zur Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit zwei Zeitfenstern pro Auftrag oder Untersuchungen zu den Auswirkungen von zusätzlichen in der Praxis auftretenden Nebenbedingungen wie z. B. Ladebedingungen. Auch grundlegende Analysen fehlen.

Die vorliegende Arbeit möchte hier einen Beitrag leisten, indem ein entsprechender Algorithmus entwickelt und der zu Grunde liegende Graph analysiert wird. Ebenfalls werden die Auswirkungen von Ladebedingungen untersucht.

2.3 Das Entscheidungsproblem

In der Literatur werden verschiedene Facetten des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern genannt und untersucht, es gibt keine einheitliche Definition eines zugehörigen Entscheidungsproblems. Der folgende Abschnitt (2.3.1) befasst sich daher zunächst mit der genauen Beschreibung des dieser Arbeit zu Grunde liegenden Entscheidungsproblems. Anschließend wird zur formalen Definition des Entscheidungsproblems ein zugehöriges mathematisches Modell formuliert: in Abschnitt 2.3.2 wird das dynamische Pickup and Delivery Vehicle Routing Problem als endliches, zeitdiskretes dynamisches System modelliert, wobei jeder Auftragseingang eine Entscheidungssituation darstellt. Zur Entscheidungsfindung muss eine Lösung für das statische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern gefunden werden. Dieses Teilproblem wird im Abschnitt 2.3.3 als lineares Modell formuliert.

2.3.1 Das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern als spezielles Entscheidungsproblem

Das in dieser Arbeit betrachtete Entscheidungsproblem beinhaltet die Situation eines Fuhrunternehmens, das während der laufenden Planungsperiode online zusätzliche Aufträge bekommt:

Dem Fuhrunternehmen steht zur Erledigung der Aufträge eine Fahrzeugflotte zur Verfügung, die aus mehreren nicht notwendigerweise gleichartigen Fahrzeugen besteht. Die Fahrzeuge werden charakterisiert durch die Kapazität, ein zugehöriges Depot und ein Arbeitszeitfenster, in dem sie für die Bedienung von Aufträgen zur Verfügung stehen. Es wird angenommen, dass alle Fahrzeuge insofern gleichartig sind, als sie für die gleiche Strecke die gleiche Fahrzeit benötigen. Die Kosten der Fahrzeuge werden nur durch die sogenannte Lenkzeit beeinflusst, d. h. die Zeit, in der das Fahrzeug unterwegs ist. Ein Umladen von bereits abgeholten Aufträ-

gen von einem Fahrzeug in ein anderes Fahrzeug ist nicht erlaubt.

Die Aufträge bestehen aus einer bestimmten Menge eines zu transportierenden Guts. Alle Güter dürfen prinzipiell zusammen in einem Fahrzeug transportiert werden. Die Menge des zu transportierenden Guts darf weder die Kapazität eines Fahrzeugs überschreiten, noch kann ein Auftrag gesplittet und auf mehrere Fahrzeuge verteilt werden. Der Kunde gibt die Charakteristika des Auftrags an: in welchem Zeitraum das Gut an welchem Ort abzuholen (Pickup) und in welchem Zeitraum es an welchem anderen Ort wieder auszuladen ist (Delivery) und wie viel Fahrzeugkapazität der Auftrag beim Transport benötigt. Die Pickup- und Delivery-Zeitfenster für den Beginn des Be- oder Entladevorgangs sind strikt und dürfen nicht verletzt werden, ebenso ist die Reihenfolgebedingung einzuhalten. Sowohl für den Be- als auch den Entladevorgang sind die benötigten Servicezeiten bekannt. Weiterhin sind die Fahrzeiten zwischen allen Pickup- und Delivery-Orten und dem Depot bekannt und erfüllen die Dreiecksungleichung, d. h. die kürzeste Verbindung zwischen je zwei Orten ist immer der direkte Weg.

Der Auftragseingang erfolgt online während der Planungsperiode - das können telefonische Anfragen sein, Anfragen per Fax oder über das Internet. Das Unternehmen muss dann sofort entscheiden, ob es diesen Auftrag annimmt oder ablehnt. Diese Entscheidung sollte so schnell möglich sein, dass sie z. B. während einer telefonischen Anfrage direkt mitgeteilt werden kann, also im Bereich von wenigen Sekunden nach Eingabe des Auftrags liegen. Zur Annahme eines Auftrags muss ein zulässiger Routenplan für die Fahrzeuge des Unternehmens vorliegen, wobei alle bisher angenommenen Aufträge zuzüglich des neu eingegangenen Auftrags unter Berücksichtigung der Kapazitäten und Arbeitszeitfenster der Fahrzeuge und der Pickup- und Delivery-Zeitfenster der Aufträge zu bedienen sind und die in der Vergangenheit bereits erfolgten Pickups und Deliverys nicht mehr verändert werden dürfen. Hat das Unternehmen den Auftrag einmal angenommen, so kann es ihn nicht wieder stornieren. Ebensowenig kann ein einmal abgelehnter Auftrag später doch noch angenommen werden - es sei denn, der Kunde erteilhi t den Auftrag noch einmal neu. Die Zuordnung eines angenommenen Auftrags zu den Fahrzeugen kann jedoch im Rahmen der Optimierung der Routen geändert werden, solange der Auftrag nicht bereits abgeholt wurde. Ebenso kann ein Auftrag innerhalb einer Route beliebig oft verschoben werden, solange die Zeitfenster- und Kapazitätsrestriktionen eingehalten werden. Weder die Anzahl, noch die Charakteristika oder die Zeitpunkte der eingehenden Aufträge sind im Voraus bekannt.

Es wird angenommen, dass jeder Auftrag einen Gewinn für das Unternehmen erzeugt. Zum Ziel der Gewinnmaximierung verfolgt das Unternehmen die Strategie, zunächst möglichst viele Aufträge anzunehmen und dann die Kosten für die Fahrzeuge durch eine „gute Routenplanung“ zu minimieren.

Zur genauen Beschreibung des Entscheidungsproblems wird im folgenden Abschnitt das mathematische Modell formuliert.

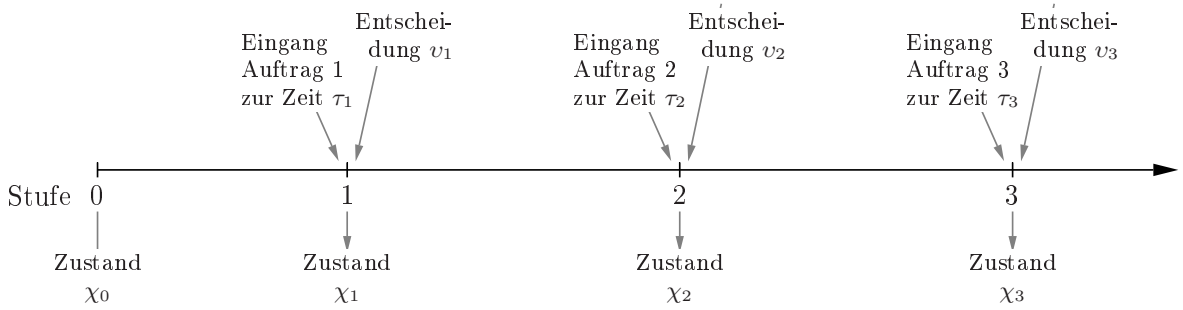

Abbildung 2.1: Schematische Darstellung der Elemente des dynamischen Modells

2.3.2 Ein zugehöriges dynamisches Modell

Ein dynamisches Modell dient der Beschreibung eines mehrstufigen Entscheidungsprozesses. Dabei wird ein System betrachtet, dessen Zustand zu jedem Zeitpunkt durch eine Reihe von Parametern beschrieben werden kann. Diese Zustände werden mit Hilfe sogenannter Zustandsvariablen beschrieben. Zu bestimmten Zeitpunkten, den so genannten Entscheidungsstufen, muss jeweils eine Entscheidung getroffen werden, die den Zustand des Systems und damit die Zustandsvariablen verändern. Die Änderung des Zustands soll dabei nur vom aktuellen Zustand und der getroffenen Entscheidung abhängen und damit unabhängig von der restlichen Vorgeschichte sein. Das Ziel des Entscheidungsprozesses besteht in der Optimierung einer Funktion, die von der Belegung der einzelnen Entscheidungsvariablen abhängt.[1]

Die Entscheidungsstufen werden anhand der Anzahl eingegangener Aufträge gesetzt: der Eingang von Auftrag Nr. κ entspricht der Entscheidungsstufe κ. Der Zeitpunkt des Auftragseingangs wird mit τ_κ bezeichnet. Auf Stufe κ muss die Entscheidung getroffen werden, ob der Auftrag Nr. κ angenommen oder abgelehnt wird. Dazu wird die Entscheidungsvariable $v_\kappa \in \{0;1\}$ eingeführt: $v_\kappa = 1$ bedeutet, dass Auftrag Nr. κ angenommen wird, im Fall $v_\kappa = 0$ wird er abgelehnt.

Der Zustand des Systems in den Entscheidungsstufen wird in der Variablen χ_κ wiedergegeben. χ_κ enthält für alle Fahrzeuge die auf Stufe κ nach der Annahme oder Ablehnung von Auftrag Nr. κ aktuellen Routen und beschreibt damit eindeutig den Zustand des Systems. Die Menge aller zulässigen Zustände des Systems auf der Entscheidungsstufe κ wird in der Menge $\mathcal{X}_\kappa$ zusammengefasst. Ein Zustand χ_κ ist genau dann zulässig, wenn alle angenommenen Aufträge bedient werden, der Routenplan der Fahrzeuge weder Zeitfenster- noch Kapazitäts- oder Reihenfolgebedingungen verletzt, die Gesamtfahr- und -wartezeit der Fahrzeuge minimiert wird und der Routenplan sich frühestens ab dem Zeitpunkt τ_κ vom Routenplan des vorhergehenden Zustands $\chi_{\kappa-1}$ unterscheidet.

Der Übergang von einem Zustand $\chi_{\kappa-1}$ in einen neuen Zustand χ_κ geschieht in Abhängigkeit vom Zeitpunkt τ_κ und der Belegung der Entscheidungsvariable v_κ: falls der Auftrag angenommen wird ($v_\kappa = 1$), muss der Routenplan (ab Zeitpunkt τ_κ) geändert und in den

[1] In Anlehnung an [7], Seite 81f.

Zustand $\chi_{\kappa+1}$ übernommen werden. Für $\upsilon_\kappa = 0$ kann der bisherige Routenplan beibehalten werden. Dieser Übergang von Zustand $\chi_{\kappa-1}$ in den Zustand χ_κ wird durch die sogenannte Übergangsfunktion $\phi_\kappa(\chi_{\kappa-1}, \upsilon_\kappa, \tau_\kappa)$ beschrieben.

Mit jeder Entscheidung wird auch die Zielfunktion des Systems beeinflusst. Die Zielfunktion wird durch die Funktion $\theta_\kappa(\upsilon_\kappa) = \upsilon_\kappa$ dargestellt. Mit ihrer Hilfe wird die Anzahl der angenommenen Aufträge maximiert.

Mit Hilfe dieser Variablen kann nun das dynamische Modell formuliert werden:

$$\max \sum_{\kappa=0}^{\infty} \upsilon_\kappa \tag{2.1}$$

unter den Nebenbedingungen

$$\chi_\kappa \in \mathcal{X}_\kappa \qquad \forall \quad \kappa \in \mathbb{N} \tag{2.2}$$

$$\upsilon_\kappa \in \{0;1\} \qquad \forall \quad \kappa \in \mathbb{N} \tag{2.3}$$

$$\chi_\kappa = \phi_\kappa(\chi_{\kappa-1}, \upsilon_\kappa, \tau_\kappa) \qquad \forall \quad \kappa \in \mathbb{N} \tag{2.4}$$

In der Zielfunktion 2.1 wird die Anzahl der angenommenen Aufträge maximiert. Nebenbedingung 2.2 garantiert, dass Zustand χ_κ in der Menge der zulässigen Zustände der Stufe κ enthalten ist und sowohl die Reihenfolgebedingung als auch alle Zeitfenster- und Kapazitätsrestriktionen eingehalten werden. In Nebenbedingung 2.3 werden nur binäre Entscheidungsvariablen υ_κ zugelassen, und Bedingung 2.4 gewährleistet, dass Zustand χ_κ korrekt aus Zustand $\chi_{\kappa-1}$ hervorgeht, wobei alle Aufträge der Routen von $\chi_{\kappa-1}$ auch in den Routenplan von χ_κ eingehen, der neu eingegangene Auftrag Nr. κ in Abhängigkeit von der Entscheidungsvariable υ_κ aufgenommen wird oder nicht und die Routen der Fahrzeuge sich erst ab Zeitpunkt τ_κ ändern dürfen.

Dieses Modell beschreibt zwar generell das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern, bleibt aber sehr allgemein. Die eigentlichen Schwierigkeiten und genauen Formulierungen des Entscheidungsproblems sind sehr grob in der Menge der zulässigen Zustände und der Übergangsfunktion zusammengefasst, die auch die Menge der möglichen Entscheidungen begrenzen. Die Entscheidung, einen Auftrag abzulehnen, ist immer zulässig. Die Entscheidung, einen Auftrag anzunehmen, kann jedoch nur dann getroffen werden, wenn es eine zulässige Lösung für die Routenplanung nach Zeitpunkt τ_κ unter Einbeziehung des anzunehmenden Auftrags gibt. In der Übergangsfunktion muss dann ein geänderter fahrtkostenminimaler Routenplan unter Berücksichtigung von Auftrag Nr. κ und dem Zeitpunkt τ_κ des Auftragseingangs übernommen werden. Die Berechnung dieses Routenplans unter Einbeziehung des anzunehmenden Auftrags erfordert die Lösung eines statischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern: gesucht wird ein zulässiger Routenplan für fest vorgegebene Mengen von Aufträgen und Fahrzeugen, wobei die Arbeits-, Pickup- und Delivery-Zeitfenster, die Reihenfolgebedingung und die Kapazitätsrestriktionen eingehalten, alle Aufträge von je einem Fahrzeug bedient und die Routen der Fahrzeuge bis zum Zeitpunkt

τ_κ beibehalten werden müssen. Die fest vorgegebene Menge von Aufträgen besteht aus allen bereits angenommenen Aufträgen und dem neu eingegangenen Auftrag. Das Ziel dieses statischen Teilproblems entspricht dem im Entscheidungsproblem formulierten Sekundärziel: der Minimierung der Gesamtfahr- und wartezeit aller Fahrzeuge. Da das Modell für das statische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern allein bereits sehr komplex ist, wird es zur besseren Übersichtlichkeit im folgenden Abschnitt separat als statisches lineares Modell formuliert.

2.3.3 Ein lineares Modell des statischen Teilproblems auf Stufe κ

In diesem Abschnitt wird das Teilproblem der Routenplanung beschrieben, das für die Entscheidung, ob Auftrag Nr. κ angenommen werden kann, gelöst werden muss. Dieses Teilproblem besteht in der Lösung eines (statischen) Pickup and Delivery Problems mit Zeitfenstern. Das Ziel besteht in der Minimierung der von allen Fahrzeugen gemeinsam zurückgelegten Strecke, wobei die Routen der Fahrzeuge erst ab dem Planungszeitpunkt τ_κ veränderbar sind, jeder Auftrag von genau einem Fahrzeug bedient und die Reihenfolge-, Kapazitäts- und Zeitfensterrestriktionen eingehalten werden müssen. In diesem Abschnitt werden daher zunächst die benötigten Begriffe eingeführt und das Entscheidungsproblem exakt beschrieben. Anschließend wird ein zugehöriges lineares Modell formuliert.

Bemerkung: Alle in diesem Abschnitt vorkommenden Variablen müssten mit dem Index κ gekennzeichnet werden, um die Abhängigkeit von der Stufe κ des dynamischen Modells anzuzeigen. Aus Gründen der besseren Lesbarkeit wird diese Indizierung jedoch unterlassen.

2.3.3.1 Das Teilproblem als spezielles Entscheidungsproblem

Dieser Abschnitt enthält die Beschreibung des Entscheidungsproblems „Routenplanung“, das dem Teilproblem des (statischen) Pickup and Delivery Problems mit Zeitfenstern zu Grunde liegt. Dazu werden zunächst folgende Begriffe eingeführt:

Definition 2.2

Wartezeit: *Die Wartezeit an einem Pickup- oder Delivery-Ort bezeichnet die Länge der Zeitspanne zwischen der Ankunftszeit eines Fahrzeugs an diesem Ort und der für diesen Ort gegebenen unteren Zeitfenstergrenze. Sie kann nicht negativ sein.*

Planungszeitpunkt: *Der Planungszeitpunkt ist der Zeitpunkt τ_κ, in dem eine neue Routenplanung angestoßen wird.*

Startzeitpunkt: *Der Startzeitpunkt eines Fahrzeugs ist für ein noch im Depot stehendes Fahrzeug das Maximum aus Planungszeitpunkt und dem Beginn seines Arbeitszeitfensters; für ein beschäftigtes Fahrzeug ist es der Zeitpunkt, an dem das Fahrzeug den nächsten Be- oder Entladevorgang abgeschlossen haben wird.*

Fahrzeugposition: *Die Fahrzeugposition eines stehenden Fahrzeugs ist der Ort, an dem sich das Fahrzeug befindet, für ein fahrendes Fahrzeug enthält die Fahrzeugposition den gerade angesteuerten Ort.*

Startposition: *Die Startposition eines Fahrzeugs ist der Ort, an dem sich ein Fahrzeug zu seinem Startzeitpunkt befindet.*

Bisherige Route: *Die bisherige Route ist die Teilsequenz von Orten aus der Route des Fahrzeugs, die es bis zum Startzeitpunkt bereits angefahren hat.*

Zukünftige Route: *Die zukünftige Route ist die Teilsequenz von Orten aus der Route des Fahrzeugs, die es bis zu seinem Startzeitpunkt noch nicht angefahren hat.*

Noch zu bedienende Aufträge: *Die Menge der noch zu bedienenden Aufträge ist die Teilmenge der angenommenen Aufträge, deren Delivery-Orte noch nicht angefahren wurden, zuzüglich des neuen Auftrags Nr. κ.*

Noch anzufahrende Orte: *Die Menge der noch anzufahrenden Orte ist die Teilmenge aller Depots, Pickup- und Delivery-Orte, die in den zukünftigen Routen der Fahrzeuge enthalten sind, zuzüglich des Pickup- und des Delivery-Orts des neuen Auftrags Nr. κ.*

Die zum Planungszeitpunkt durchzuführende Routenplanung darf nur die zukünftigen Routen der Fahrzeuge verändern, während die bisherigen Routen als fix betrachtet werden müssen. Es genügt daher, die Routen der Fahrzeuge ab den jeweiligen Startzeitpunkten der Fahrzeuge zu berechnen. Dabei muss beachtet werden, dass nur die noch zu bedienenden Aufträge in die Routenplanung eingehen und ein Teil dieser Aufträge bereits in einem Fahrzeug geladen sein kann, wodurch die Zuordnung dieses Teils der Aufträge bereits festgelegt ist. Für einen noch zu bedienenden Auftrag ist das genau dann der Fall, wenn ein Fahrzeug den Pickup-Ort des Auftrags in seiner bisherigen Route bereits vor dem Zeitpunkt τ_κ besucht hat, also der Pickup-Ort eines noch zu bedienenden Auftrags nicht zur Menge der noch anzufahrenden Orte gehört. Es liegt daher folgendes Entscheidungsproblem zu Grunde:

Gegeben ist eine Menge $F = \{1, \ldots, m\}$ von Fahrzeugen und eine Menge $N = \{1, \ldots, n\}$ von noch zu bedienenden Aufträgen. Für jedes Fahrzeug $f \in F$ sind der Startzeitpunkt $\underline{T}_f$, das Ende seines Arbeitszeitfensters $\overline{T}_f$, seine Startposition a_f, die insgesamt zur Verfügung stehende Kapazität $l_f > 0$ sowie die Menge $\tilde{N}_f$ der bereits in diesem Fahrzeug geladenen Aufträge bekannt. Jeder noch zu bedienende Auftrag $i \in \{1, \ldots, n\}$ wird charakterisiert durch seinen Pickup-Ort p_i und seinen Delivery-Ort d_i, sein Pickup-Zeitfenster $[\underline{t}_{p_i}, \overline{t}_{p_i}]$ mit $\underline{t}_{p_i} < \overline{t}_{p_i}$ und sein Delivery-Zeitfenster $[\underline{t}_{d_i}, \overline{t}_{d_i}]$ mit $\underline{t}_{d_i} < \overline{t}_{d_i}$, sowie die für das Be- und Entladen benötigten Servicezeiten s_{p_i} und s_{d_i} und die für den Transport des Auftrags benötigte Fahrzeugkapazität g_i.

Es wird vorausgesetzt, dass die kürzesten Wege zwischen allen noch anzufahrenden Orten gegeben sind. Ferner wird angenommen, dass die Orte nicht geographisch interpretiert werden,

sondern eineindeutig den Aufträgen zugeordnet sind. Falls z. B. ein geographischer Ort in zwei Aufträgen i und i^* als Pickup-Ort angefahren werden soll, so werden im Modell zwei unterschiedliche Orte p_i und p_i* betrachtet, deren Entfernung 0 beträgt.

Gesucht werden zukünftige Routen für alle Fahrzeuge $f \in F$ derart, dass

- die zukünftige Route des Fahrzeugs $f \in F$ im Ort der Startposition a_f und frühestens zum Startzeitpunkt $\underline{T}_f$ beginnt,
- die maximale Arbeitszeitgrenze $\overline{T}_f$ des Fahrzeugs $f \in F$ nicht überschritten wird,
- die Ladung des Fahrzeugs $f \in F$ zum Startzeitpunkt der Summe der von den bereits geladenen Aufträgen benötigten Kapazitäten $\sum_{i \in \tilde{N}_f} g_i$ entspricht und zu keinem Zeitpunkt die Fahrzeugkapazität l_f übersteigt,
- jeder Auftrag i von genau einem Fahrzeug f bedient wird,
- die Reihenfolgebedingung für alle noch nicht eingeladenen Aufträge eingehalten wird,
- jedes Fahrzeug $f \in F$ die bereits eingeladenen Aufträge der Menge $\tilde{N}_f$ selbst ausliefert,
- Pickup- und Delivery-Zeitfenster jedes Auftrags i eingehalten werden: das Fahrzeug muss den Ort innerhalb des Zeitfensters erreichen, wobei eine (positive) Wartezeit des Fahrzeugs bis zum Beginn des Zeitfensters erlaubt ist und der Be- oder Entladevorgang an diesem Ort über die obere Zeitfenstergrenze hinaus andauern darf,
- die Summe der Fahr- und Wartezeiten aller Fahrzeuge $f \in F$ minimiert wird.

Für dieses Entscheidungsproblem wird im folgenden Abschnitt ein lineares Modell formuliert.

2.3.3.2 Ein zugehöriges lineares Modell

Um ein zugehöriges lineares Modell formulieren zu können, werden zunächst weitere Notationen eingeführt und der Graph G^{Orte} definiert. Anschließend werden die Entscheidungs- und Hilfsvariablen definiert und dann ein lineares Modell angegeben.

Die Menge der noch anzufahrenden Pickup-Orte wird im Folgenden mit P bezeichnet, $P \subset \{p_1, \ldots, p_n\}$, die Menge der noch anzufahrenden Delivery-Orte wird D genannt: $D = \{d_1, \ldots, d_n\}$. Die Startpositionen der Fahrzeuge werden in der Menge $A = \{a_1, \ldots, a_m\}$ zusammengefasst. Zur Modellierung wird ein vollständiger gerichteter Graph $G^{Orte} = (V^{Orte}, E^{Orte}, c^{Orte})$ definiert, der das für die Routenplanung relevante Wegenetz abbildet. Die Knotenmenge umfasst die Menge aller anzufahrenden Orte, die Menge aller Startpositionen der Fahrzeuge sowie einen künstlichen Ort z: $V^{Orte} = P \cup D \cup A \cup \{z\}$ und beinhaltet somit alle für die Routenplanung nötigen Orte. Der künstliche Ort z wird als imaginärer Zielort der Fahrzeuge benötigt. Der Graph ist vollständig, d. h. $E^{Orte} = V^{Orte} \times V^{Orte}$. Die Gewichtsfunktion $c^{Orte} : E \to \mathbb{R}_0^+$ ordnet jeder Kante $(\mu, \nu) \in E^{Orte}$ die Länge $c_{\mu,\nu}$ des kürzesten Wegs vom Anfangsort μ zum Endort ν der Kante zu; allen Kanten (μ, z) und (z, μ) zum bzw. vom

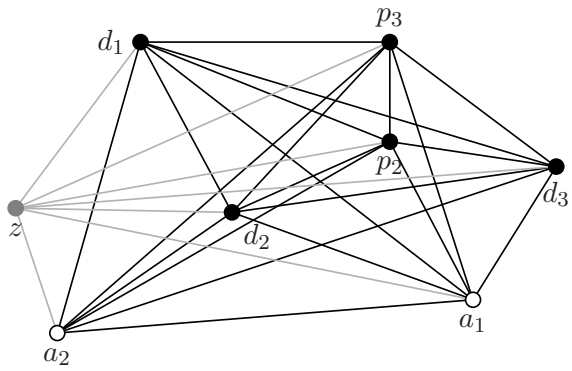

Abbildung 2.2: Beispiel für einen Graphen G^{Orte} für zwei Fahrzeuge und drei noch zu bedienende Aufträge, von denen Auftrag 1 bereits geladen ist.

künstlichen Ort z wird die Länge 0 zugeordnet.[2]

Abbildung 2.2 zeigt beispielhaft einen Graphen G^{Orte} für drei noch zu bedienende Aufträge. Mit Hilfe dieses Graphen kann ein lineares Modell des Entscheidungsproblems formuliert werden.

Für die Lösung des Entscheidungsproblems stellt sich die Frage, ob ein Fahrzeug f die Kante $(\mu, \nu) \in E^{Orte}$ befahren und somit die Orte μ und ν besuchen und - bei Pickup- und Delivery-Orten - die dazugehörigen Be- oder Entladevorgänge vornehmen soll oder nicht. Daher werden binäre Entscheidungsvariablen $x^f_{\mu,\nu} \in \{0,1\}$ eingeführt, die für jedes Tupel $(f,(\mu,\nu)) \in F \times E^{Orte}$ angeben, ob Fahrzeug f die Strecke (μ,ν) benutzt ($x^f_{\mu,\nu} = 1$) oder nicht ($x^f_{\mu,\nu} = 0$).

Als Hilfsvariablen müssen zusätzlich für jeden Ort $\mu \in V_{Orte}$ die geplante Servicestartzeit α_μ sowie die dort anfallende Wartezeit ω_μ berechnet werden. Die bereits geladene Menge (load) von Fahrzeug f nach dem Be- oder Entladen im Ort μ wird in der Variablen λ^f_μ abgelegt.

Zur einfacheren Notation werden auch für die Startpositionen der Fahrzeuge Servicezeiten von 0 Zeiteinheiten definiert: $s_{a_f} = 0 \quad \forall\ f \in F$. Die Zahl M stellt eine Konstante dar und steht für eine sehr große Zahl zur Annäherung des Werts ∞.

Damit kann das Modell folgendermaßen aufgestellt werden:

$$\min \sum_{\mu \in V^{Orte}} \omega_\mu + \sum_{(\mu,\nu) \in E^{Orte}} \sum_{f \in F} c_{\mu,\nu} x^f_{\mu,\nu} \tag{2.5}$$

[2]Bemerkung: Der Graph G^{Orte} bildet kein vollständiges Straßennetz ab, sondern enthält als Knoten nur die Startpositionen der Fahrzeuge und die anzufahrenden Orte. Die Kanten repräsentieren bekannte kürzeste Wege zwischen den Orten, die fest vorgegeben und nicht Teil der Optimierung sind.

unter den Nebenbedingungen:

$$\sum_{\mu \in V^{Orte} \setminus A} x^f_{a_f,\mu} = 1 \qquad \forall\ f \in F \tag{2.6}$$

$$\sum_{\mu \in V^{Orte} \setminus \{z\}} x^f_{\mu,z} = 1 \qquad \forall\ f \in F \tag{2.7}$$

$$\sum_{\mu \in V^{Orte} \setminus \{z\}} x^f_{\mu,\nu} - \sum_{\mu \in V^{Orte} \setminus A} x^f_{\nu,\mu} = 0 \qquad \forall\ \nu \in P \cup D,\ f \in F \tag{2.8}$$

$$\sum_{f \in F} \sum_{\mu \in V^{Orte} \setminus \{z\}} x^f_{\mu,\nu} = 1 \qquad \forall\ \nu \in P \tag{2.9}$$

$$\sum_{\mu \in P \cup D} x^f_{p_i,\mu} - \sum_{\mu \in P \cup D} x^f_{\mu,d_i} = 0 \qquad \forall\ f \in F,\ i \in N \setminus \bigcup_{f \in F} \tilde{N}_f \tag{2.10}$$

$$\sum_{\mu \in V^{Orte} \setminus \{z\}} x^f_{\mu,d_i} = 1 \qquad \forall\ f \in F,\ i \in \tilde{N}_f \tag{2.11}$$

$$\lambda^f_{a_f} - \sum_{i \in \tilde{N}_f} g_i = 0 \qquad \forall f \in F \tag{2.12}$$

$$\lambda^f_{p_i} - \lambda^f_{\mu} - g_i + (1 - x^f_{\mu,p_i})M \geq 0 \qquad \forall\ f \in F,\ i \in N \setminus \bigcup_{f \in F} \tilde{N}_f,\ \mu \in V^{Orte} \tag{2.13}$$

$$\lambda^f_{d_i} - \lambda^f_{\mu} + g_i + (1 - x^f_{\mu,d_i})M \geq 0 \qquad \forall\ f \in F,\ i \in N,\ \mu \in V^{Orte} \tag{2.14}$$

$$0 \leq \lambda^f_{\mu} \leq l_f \qquad \forall\ f \in F,\ \mu \in V^{Orte} \tag{2.15}$$

$$\alpha_\nu - \alpha_\mu - s_\mu - c_{\mu,\nu} + (1 - \sum_{f \in F} x^f_{\mu,\nu})M \geq 0 \qquad \forall\ \mu \in V^{Orte} \setminus \{z\},\ \nu \in V^{Orte} \setminus A \tag{2.16}$$

$$\alpha_\nu - \alpha_\mu - s_\mu - c_{\mu,\nu} - \omega_\nu - (1 - \sum_{f \in F} x^f_{\mu,\nu})M \leq 0 \qquad \forall\ \mu \in V^{Orte} \setminus \{z\},\ \nu \in V^{Orte} \setminus A \tag{2.17}$$

$$\alpha_{a_f} \geq \underline{T}_f \qquad \forall\ f \in F \tag{2.18}$$

$$\alpha_\mu + (x^f_{\mu,z} - 1)M \leq \overline{T}_f \qquad \forall\ f \in F,\ \mu \in D \tag{2.19}$$

$$\alpha_{p_i} \leq \alpha_{d_i} \qquad \forall\ i \in N \setminus \bigcup_{f \in F} \tilde{N}_f \tag{2.20}$$

$$\underline{t}_\mu \leq \alpha_\mu \leq \overline{t}_\mu \qquad \forall\ \mu \in P \cup D \tag{2.21}$$

$$\omega_\mu \geq 0 \qquad \forall\ \mu \in P \cup D \tag{2.22}$$

$$x^f_{\mu,\nu} \in \{0,1\} \qquad \forall\ \mu,\nu \in V^{Orte},\ f \in F \tag{2.23}$$

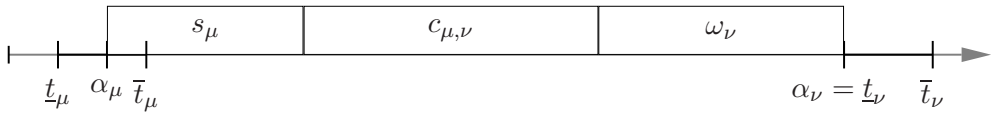

Abbildung 2.3: Schematische Darstellung der zeitlichen Komponenten Service-, Fahr- und Wartezeit zwischen der Ankunft in einem Ort μ und der Ankunft im nächsten Ort ν bei anfallender Wartezeit

In der Zielfunktion 2.5 wird die Gesamtfahr- und -wartezeit aller Fahrzeuge minimiert. Die Bedingungen 2.6 und 2.7 erzwingen, dass jedes Fahrzeug seine zukünftige Route in seiner Startposition beginnt und im künstlichen Ort z beendet.

Nebenbedingung 2.8 erhält den Fahrzeugfluss aufrecht, indem sie dafür sorgt, dass jeder angefahrene Pickup- oder Delivery-Ort auch wieder verlassen wird. In Bedingung 2.9 wird erzwungen, dass jeder noch anzufahrende Pickup-Ort von genau einem Fahrzeug angefahren wird. Das Anfahren der Delivery-Orte wird in den Bedingungen 2.10 und 2.11 garantiert. In 2.10 wird dafür gesorgt, dass Pickup- und Delivery-Ort eines zum Planungszeitpunkt noch nicht eingeladenen Auftrags von ein und demselben Fahrzeug besucht werden, während 2.11 erzwingt, dass die zum Planungszeitpunkt bereits in einem Fahrzeug geladenen Aufträge von genau diesem Fahrzeug zu ihren Delivery-Orten transportiert werden.

In Bedingung 2.12 wird die Hilfsvariable zur Speicherung der bereits belegten Fahrzeugkapazität im Ort der Startposition auf die Summe der von den geladenen Aufträgen benötigten Kapazitäten gesetzt. Die Nebenbedingungen 2.13 und 2.14 sorgen dafür, dass die Belegung dieser Hilfsvariable immer mindestens der tatsächlichen Ladung entspricht. Die Zulässigkeit der geladenen Menge wird für jeden Ort in Bedingung 2.15 überprüft.

Nebenbedingung 2.16 stellt sicher, dass die Hilfsvariable zur Speicherung der Servicestartzeit in den Orten mindestens alle Fahrt- und Servicezeiten der bisherigen Route beinhaltet (vgl. Abbildung 2.3). Sie sorgt gleichzeitig für die Unterbindung von Subtouren. In Nebenbedingung 2.17 wird die mindestens erforderliche Wartezeit gesetzt. Mit Hilfe der Servicestartzeiten stellen 2.18 und 2.19 sicher, dass die Arbeitszeitfenster der Fahrzeuge eingehalten werden. Nebenbedingung 2.20 erzwingt, dass jeder zum Planungszeitpunkt noch nicht geladene Auftrag erst abgeholt wird, bevor er ausgeliefert werden kann. Die Einhaltung der Pickup- und Delivery-Zeitfenster der Aufträge wird in Bedingung 2.21 erzwungen, Nebenbedingung 2.22 stellt sicher, dass nur positive Wartezeiten eingebracht werden können. Abschließend erzwingt Nebenbedingung 2.23 binäre Entscheidungsvariablen.

Damit ist nun die Formulierung des Teilproblems „Routenplanung“ abgeschlossen. Die Routenplanung ist als statisches Teilproblem im dynamischen Modell zum Entscheidungsproblem des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern enthalten. Dieses Entscheidungsproblem soll durch eine Weiterentwicklung des Algorithmus von Caramia et al. gelöst werden [11]. Die Idee dieses Algorithmus wird im folgenden Abschnitt vorgestellt.

2.4 Der bestehende Algorithmus

Caramia et al. stellten 2001 einen Algorithmus zur heuristischen Lösung des dynamischen Pickup and Delivery Problems vor [11]. Das ihrem Modell zu Grunde liegende Entscheidungsproblem beschreibt die Situation eines Unternehmens für Anruf-Sammel-Taxen: die Kunden rufen während eines Tages an und bestellen die Taxen zu einem bestimmten Zeitraum an einen bestimmten Ort, geben den Zielort an und legen fest, wie viel Umweg sie bis zum Erreichen ihres Ziels in Kauf nehmen. Das Taxi-Unternehmen muss dann sofort am Telefon entscheiden, ob dieser Kunde angenommen wird. Dazu wird der Auftrag in das System eingegeben und eine zulässige Lösung gesucht, d. h. eine Zuordnung dieses Kunden zu einer der Taxen sowie zulässige Routen für alle Taxen, wobei keiner der bisher schon angenommenen Aufträge wieder storniert und kein bereits eingestiegener Kunde in ein anderes Taxi „versetzt“ werden darf. Zur Vereinfachung wird vorausgesetzt, dass immer nur ein Kunde pro Auftrag transportiert werden soll. Für die Routenplanung werden außerdem nur eine geringe Anzahl von Kunden pro Taxi zugelassen.

Im Jahr 2002 wurde der Algorithmus soweit angepasst, dass die Kunden sowohl ein Zeitfenster für den Beginn der Fahrt als auch ein Zeitfenster für die Ankunft am Zielort angeben [26]. Zur Anpassung an den Warentransport werden auch Wartezeiten für die Fahrzeuge zugelassen. Ferner wurde die heuristische Lösung verbessert, indem ein Algorithmus zur Tabu Suche eingesetzt wird.

Der auf diese Weise entstandene Algorithmus zur Lösung des dynamischen Vehicle Routing Problems besteht aus zwei Komponenten: einem Verfahren der Tabu Suche zur Verbesserung des aktuellen Routenplans und einer Heuristik zur Berechnung eines neuen zulässigen Routenplans unter Einbeziehung eines neu eingegangenen Auftrags, die bei jedem Auftragseingang genau einmal durchgeführt wird. Abbildung 2.4 zeigt eine schematische Darstellung des Algorithmus.

Das Verfahren der Tabu Suche wird eingesetzt, sobald kein neuer Auftragseingang vorliegt. Dieses Verfahren bestimmt in jeder Iteration aus der Menge aller angenommenen Aufträge einen zu tauschenden Auftrag und versucht, den Routenplan durch Zuordnung dieses Auftrags zu einem anderen Fahrzeug zu verbessern. Dazu werden optimale Routen für die Fahrzeuge mit den geänderten Mengen von zugeordneten Aufträgen bestimmt. Nach jeder Iteration wird überprüft, ob ein neuer Auftrag eingegangen ist: liegt ein neuer Auftrag vor, so wird mit Hilfe der Heuristik ein neuer zulässiger Routenplan berechnet und anschließend die Tabu Suche fortgesetzt, andernfalls wird sofort mit der nächsten Iteration der Tabu Suche fortgefahren.

Die Heuristik zur Berechnung eines zulässigen Routenplans wird bei Eingang eines neuen Auftrags durchgeführt. Sie besteht aus einer heuristischen Zuordnung des neuen Auftrags zu einem der Fahrzeuge unter Beibehaltung der Zuordnung aller anderen Aufträge entsprechend der aktuell besten Lösung und der Berechnung einer optimalen Route für dieses Fahrzeug. Der Ablauf lässt sich folgendermaßen skizzieren: Bei Eingang des Auftrags Nr. κ zum Zeitpunkt

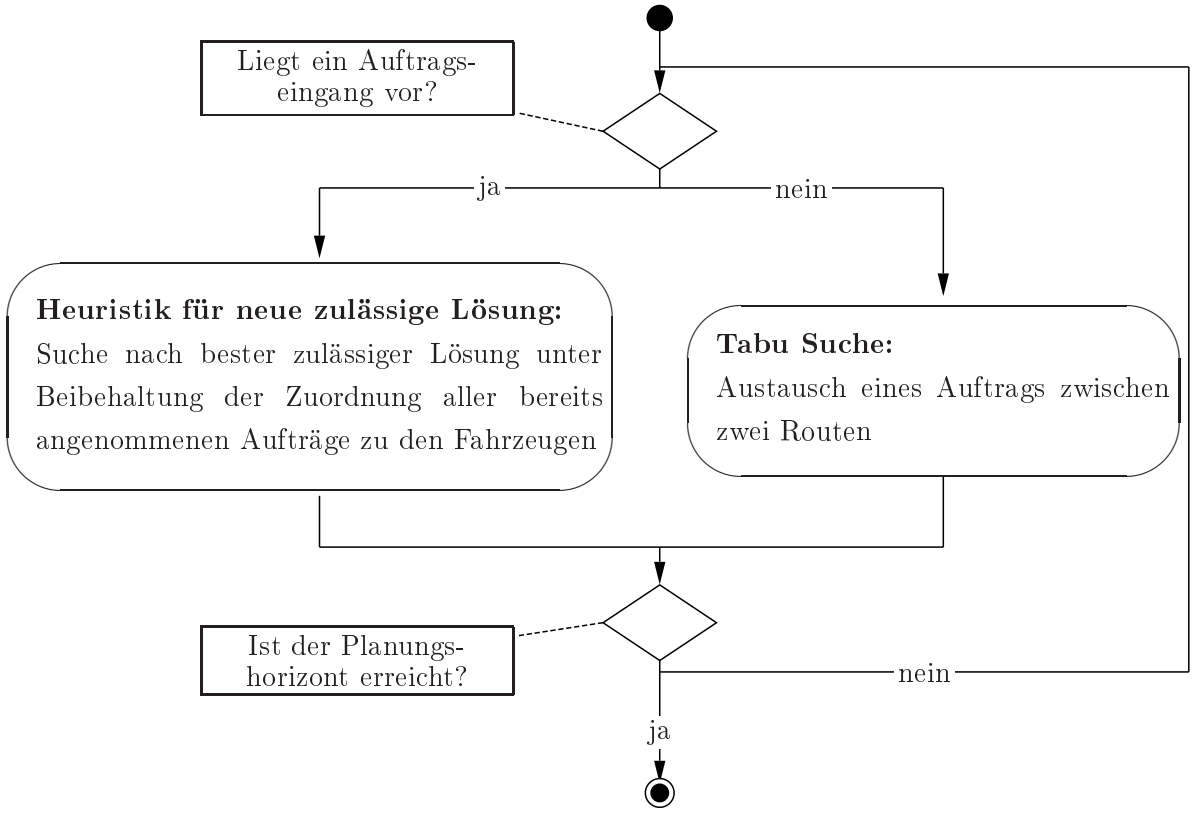

Abbildung 2.4: Schematische Darstellung des Ablaufs des Algorithmus aus [26].

τ_κ wird die aktuell beste Lösung des Verfahrens der Tabu Suche übernommen, d. h. alle bisher angenommenen Aufträge sind den Fahrzeugen bereits zugeordnet. Ebenso ist für jedes Fahrzeug die optimale Route zur Abarbeitung der ihm zugeordneten Aufträge bekannt. Zum Zeitpunkt τ_κ wird nun der neu eingegangene Auftrag Nr. κ „testweise“ jedem Fahrzeug einmal zugeordnet und für dieses Fahrzeug eine neue optimale zukünftige Route bestimmt, falls sie existiert. Gibt es mindestens ein Fahrzeug mit zulässiger Route bei Hinzunahme von Auftrag Nr. κ, so wird er angenommen. Sind für mehrere Fahrzeuge zulässige Routen bei Aufnahme des Auftrags Nr. κ gefunden worden, so wird der Auftrag dem Fahrzeug zugeordnet, dessen Route durch die Aufnahme des Auftrags Nr. κ absolut am wenigsten verlängert wurde. Der Ablauf der Heuristik wird in Abbildung 2.5 schematisch dargestellt.

Die in dieser Heuristik enthaltene Fahrzeug-Routenplanung wird exakt gelöst, d. h. es wird eine optimale Route für das Fahrzeug berechnet. Dazu wird der A*-Algorithmus zur Bestimmung eines kürzesten Wegs in einem Graphen benutzt. Caramia et al. haben mit dem Statusvektorbaum einen speziellen Graphen zur effizienten Routenplanung eingeführt, in dem die Reihenfolgebedingung „erst Pickup, dann Delivery“ für jeden Auftrag automatisch erfüllt wird. Die genaue Vorgehensweise wird in Abschnitt 3.2 beschrieben und analysiert.

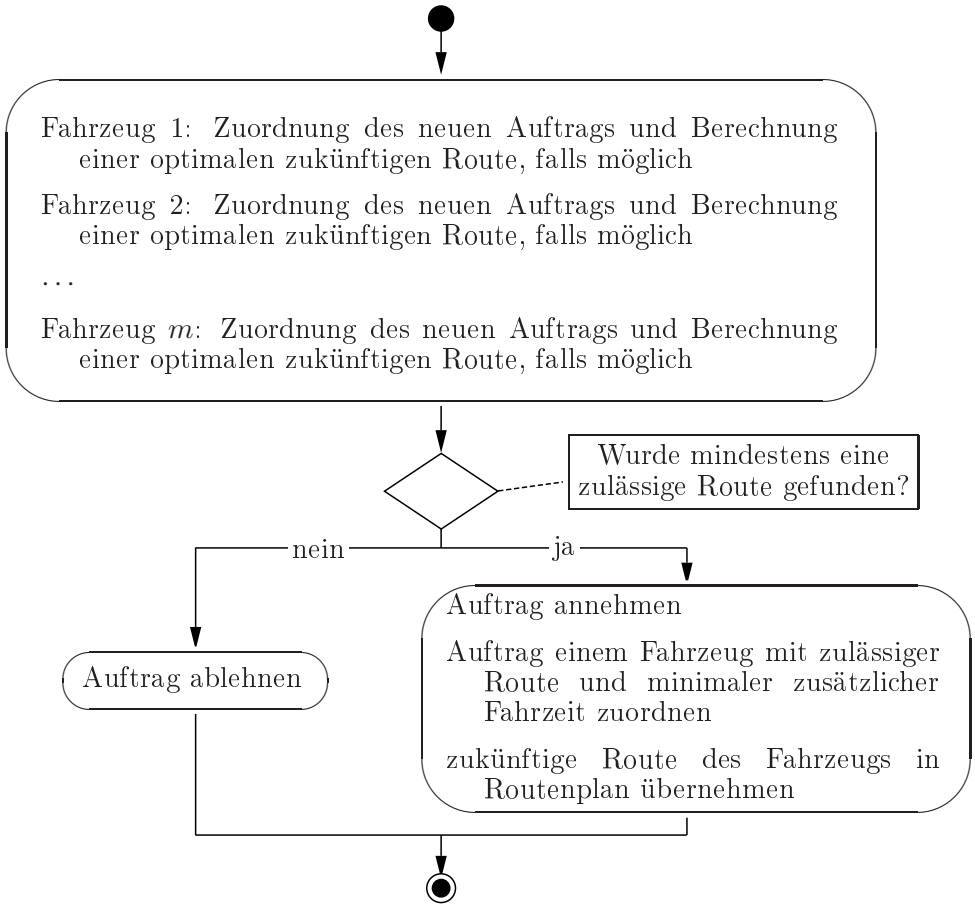

Abbildung 2.5: Schematische Darstellung der Heuristik zur Berechnung eines neuen zulässigen Routenplans aus [11].

Damit wurde in [11] und [26] ein Algorithmus geschaffen, mit dem man das Dynamische Pickup and Delivery Problem angehen kann. Aufgrund der Definition des Entscheidungsproblems konnten jedoch vereinfachende Annahmen getroffen werden, die eine schnelle Lösung ermöglichen: die Anzahl der noch nicht vollständig abgearbeiteten Aufträge, die jedem Fahrzeug zugeordnet werden können, ist beschränkt und die Zeitfenster sind relativ klein. In der mit [55] zur Verfügung gestellten Implementierung beläuft sich die Anzahl der Aufträge pro Fahrzeug auf acht. Dieser Wert und die Begrenzung auf kleine Zeitfenster stellen für ein Anruf-Sammel-Taxi eine angemessene Größe dar, für ein Fuhrunternehmen sind sie jedoch nicht akzeptabel. Lässt man allerdings diese Beschränkungen fallen, so steigen die Rechenzeiten wegen der enormen Komplexität der Routenplanung deutlich an, so dass der Algorithmus in dieser Form nicht mehr zur Lösung eines realen dynamischen Entscheidungsproblems verwendet werden kann.

Die vorliegende Arbeit beschäftigt sich daher mit der Weiterentwicklung dieser Ideen, um einen in der Realität anwendbaren Algorithmus zur Bearbeitung des dynamischen Vehicle Routing Problems mit Zeitfenstern zu erhalten. Dazu werden alle Komponenten des bestehen-

den Algorithmus analysiert und hinterfragt, um schnell eine zulässige Lösung zu erreichen und sie anschließend weiter zu verbessern. Den Kern der Bemühungen bildet dabei der exakte Algorithmus zur Routenplanung, da die Routenplanung wegen ihrer Komplexität sehr aufwändig ist und häufig aufgerufen wird, so dass an dieser Stelle der Großteil der Rechenzeit verbraucht wird. Weiterhin werden Strategien für eine schnellere Zuordnung des neuen Auftrags zu einem Fahrzeug entwickelt.

Der von Caramia et al. entwickelte Algorithmus kann also als Basis für die Entwicklung eines in der Praxis anwendbaren Algorithmus genutzt werden. Im folgenden Abschnitt werden nun die Rahmenbedingungen für die Entwicklung dieses Algorithmus beschrieben, indem auf die erforderlichen Testdatensätze und die Testumgebung eingegangen wird.

2.5 Testdatensätze und Testumgebung

Zur Entwicklung eines schnellen Algorithmus ist es zwingend erforderlich, dass Testläufe mit reproduzierbaren Ergebnissen durchgeführt werden können. Dazu müssen sowohl Testdatensätze als auch eine passende Testumgebung vorhanden sein. Die Testdatensätze sollten den Algorithmus in unterschiedlicher Art und Weise „fordern“ und ihn „an seine Grenzen bringen“, um die Praxistauglichkeit überprüfen und auch die Grenzen der Einsetzbarkeit aufzeigen zu können. Um Testdatensätze für ein dynamisches Problem durchzuführen, wird eine Testumgebung benötigt. Sie muss sicherstellen, dass für alle Testläufe die gleichen Rahmenbedingungen z. B. in Form von Störfaktoren, Rechenzeiten oder Zeitrestriktionen gelten, so dass die Vergleichbarkeit der Testläufe gewährleistet ist. Weiterhin kann eine Testumgebung verschiedene Testzwecke unterstützen, indem sie entsprechende Konfigurationsmöglichkeiten bereitstellt. Im Folgenden werden zunächst der Aufbau der Testdatensätze und anschließend die Testumgebung näher erläutert.

2.5.1 Testdatensätze

Für das Entscheidungsproblem des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern wurden bisher keine Benchmarkprobleme veröffentlicht. Zum Entscheidungsproblem des statischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern gibt es jedoch Benchmarkinstanzen, die auf die Benchmarkprobleme von Solomon [72] zurückgehen. Die Solomon-Benchmarkprobleme sind entsprechend der Verteilung der anzufahrenden Orte in Problemklassen eingeteilt: die Klasse „r“ steht für zufällig verteilte Orte (engl. random), die Klasse „c“ beinhaltet „Cluster“, d. h. lokale Häufungen von anzufahrenden Orten, und die Klasse „rc“ steht für eine Mischung aus zufällig verteilten und gehäuften anzufahrenden Orten. Jede dieser Klassen besteht aus 100 anzufahrenden Orten und dem Depot. Diese drei Testdatensätze werden dann mit verschiedenen Ausprägungen des Planungshorizonts, der Anzahl Zeitfenster und der Größe der Zeitfenster versehen. Als nächste Unterklasse wird die Länge des

Planungshorizonts angegeben: eine „1“ steht für einen kurzen Planungshorizont, der zu 5-10 anzufahrenden Orten pro Fahrzeug führt, während eine „2“ für einen langen Planungshorizont mit mehr als 30 anzufahrenden Orten pro Fahrzeug steht. Die unterschiedlichen Ausprägungen bezüglich Anzahl und Enge der Zeitfenster sind nicht explizit in der Nomenklatur der Instanzen wieder zu finden.

Homberger erweiterte diese Testdatensätze in [42] um Instanzen mit 200, 400, 600, 800 und 1000 Orten, wobei er die Struktur von Solomon aufrecht erhält und die Nomenklatur mit der Angabe der Anzahl Orte verfeinert.

Li und Lim passten in [50] die Benchmarkprobleme von Solomon und Homberger an das statische Pickup- and Delivery Vehicle Routing Problem mit Zeitfenstern (PDVRPTW) an, indem sie jeweils zwei Nachfragen zu einem Auftrag zusammenfassen und entsprechende Zeitfenster für Pickup und Delivery generieren. Sie behalten die Nomenklatur von Solomon bei, fügen jedoch ein „l“ zur Kenntlichmachung ihrer Änderungen hinzu. Die Testinstanz „lr_1_2_2“ beinhaltet beispielsweise einen kurzen Planungshorizont, 100 Aufträge mit insgesamt 200 zufällig verteilten Pickup- und Delivery-Orten.

Zur Erzeugung von Testdatensätzen für das dynamische Pickup- and Delivery Vehicle Routing Problem mit Zeitfenstern auf Basis der Benchmarkprobleme von Li und Lim für das entsprechende statische Problem wurde folgendermaßen verfahren:

1. Für jeden Auftrag i einer Testinstanz von Li und Lim wurde mit Hilfe eines Zufallszahlengenerators eine zufällige Anrufzeit τ_i erzeugt, die zwischen 0 und dem Minimum aus der gegebenen unteren Pickup-Zeitfenstergrenze $\underline{t}_{p_i}$ und der oberen Pickup-Zeitfenstergrenze $\overline{t}_{p_i}$ abzüglich der Fahrtdauer c_{a_f,p_i} vom Anfangsort des Fahrzeugs zum Pickup-Ort des Auftrags liegt. Anschließend wurden die Aufträge in der Reihenfolge ihrer Anrufzeit sortiert.

2. Die Anzahl der Fahrzeuge wurde „verknappt“: es stehen nur so viele Fahrzeuge zur Verfügung, wie in den sogenannten „best known solutions“ für das Pickup- and Delivery Vehicle Routing Problem mit Zeitfenstern von Li und Lim (Stand September 2004) benötigt werden.

Diese Testinstanzen wurden im Herbst 2005 erstellt und im Internet zur Verfügung gestellt [27]. Sie befinden sich auf der beiliegenden CD.

2.5.2 Testumgebung

Um den Algorithmus mit Hilfe dieser Testdatensätze testen zu können, wird eine Testumgebung benötigt, die jeweils eine Planungsperiode simuliert. In der Testumgebung werden die Testdatensätze eingelesen und dem Algorithmus zu der im Datensatz angegebenen „Anrufzeit“

übergeben. Gleichzeitig werden dem Algorithmus die aktuellen Standorte der Fahrzeuge zur Verfügung gestellt.

Dabei wird auf die Anforderungen der verschiedenen Testläufe eingegangen. Bei den Tests zur Beschleunigung der Routenplanung (Kapitel 3) stehen die Rechenzeiten des A*-Algorithmus im Vordergrund. Da es sich um ein exaktes Verfahren handelt, sind Änderungen im Zielfunktionswert nicht zu erwarten. Es geht nur um die Reduzierung der Rechenzeit, um den A*-Algorithmus auf das dynamische Entscheidungsproblem anwenden zu können. Die Testumgebung wird daher ereignisdiskret eingestellt, so dass die Wartezeit bis zum Eintreffen einer neuen Anfrage entfällt: sobald ein Auftrag eingeht und die Zuordnung des zuletzt eingegangenen Auftrags abgeschlossen ist, wird die Zeit der Testumgebung auf die Anrufzeit dieses Auftrags gesetzt. Das minimiert nicht nur die Zeit eines Testlaufs wegen der entfallenden Wartezeiten, sondern sorgt auch dafür, dass die Ergebnisse vergleichbar bleiben. Einer langsameren Version des Algorithmus könnte sonst wegen zu langer Rechenzeiten die Anfrage zu einer späteren Simulations-Zeit übergeben werden als der schnelleren Version. Falls bei den Berechnungen der langsameren Version die Anrufzeit des nächsten Auftrags überschritten wird, beginnt die langsamere Version die Berechnungen für den nächsten Auftrag zu einem späteren Simulationszeitpunkt und damit mit eventuell bereits veränderten Fahrzeugpositionen. Die ereignisdiskrete Simulation sorgt dafür, dass alle Versionen mit den gleichen Bearbeitungsstatus der Fahrzeuge rechnen und somit vergleichbar bleiben.

Bei den Testläufen zur Verbesserung der Zuordnung durch ein Verfahren der Tabu Suche im Kapitel 4.2 hingegen kommt es gerade auf die Länge der Wartezeit zwischen zwei Anrufen an, da diese Zeit zur Verbesserung der Zuordnung genutzt werden soll. Die Testumgebung kann daher auch auf „Echtzeit“ eingestellt werden, so dass eine kontinuierliche Simulation durchgeführt wird.

Da mit der Version des Algorithmus von Caramia et al. zum Teil Rechenzeiten von mehreren Tagen für einen Testlauf festgestellt wurden, musste eine weitere Anpassung vorgenommen werden: es wird die Möglichkeit gegeben, die Rechenzeit künstlich „klein“ zu halten, indem die Anzahl der Rechenoperationen entweder für die Berechnung einer optimalen Route oder für den gesamten Testlauf nach oben beschränkt wird. Bei Erreichen der Obergrenze wird die aktuelle Berechnung abgebrochen. Diese Beschränkung führt zu akzeptablen Rechenzeiten pro Testlauf und schränkt die Aussagekraft der Testläufe nicht ein: zusätzlich zur benötigten Anzahl der erzeugten Knoten erlaubt die Anzahl der bearbeiteten Aufträge Aufschlüsse über das Laufzeitverhalten. Außerdem wird so gewährleistet, dass die Testläufe von Anfang an mit allen Testdatensätzen durchgeführt werden können, so dass die Aussagen zum Laufzeitverhalten empirisch belegt werden können. Der Ablauf innerhalb der Testumgebung wird in der Abbildung 2.6 schematisch dargestellt: ein Auftrag wird eingelesen, dann wird auf den Übergabezeitpunkt gewartet. Ist dieser erreicht, werden der Auftrag und die aktuellen Fahrzeugpositionen an den Algorithmus übergeben und, falls vorhanden, der nächste Auftrag eingelesen etc. Die flexible Anpassung der Testumgebung an die Anforderungen des Testlaufs sind dabei im Warten

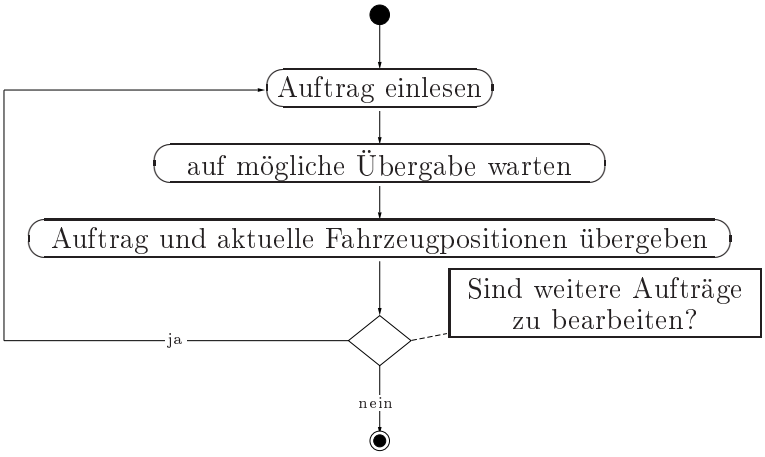

Abbildung 2.6: Schematische Darstellung der Funktion der Testumgebung.

auf den möglichen Übergabezeitpunkt versteckt: bei den Testläufen zur Routenplanung ist der mögliche Übergabezeitpunkt erreicht, sobald der Algorithmus den vorhergehenden Auftrag abgelehnt oder akzeptiert hat. Die Uhrzeit wird dabei auf den Anrufzeitpunkt des neu übergebenen Auftrags vor- oder zurück gestellt. Bei den Testläufen zur lokalen Suche werden diese zeitlichen Verschiebungen nicht in Anspruch genommen - bei diesen Testläufen geht es schließlich darum, die Wartezeit bis zum Eintreffen des nächsten Auftrags sinnvoll für die Verbesserung der Lösung zu nutzen.

Alle Testläufe wurden auf einem PC mit AMD Athlon 64 3500+ Prozessor mit 1 GB Arbeitsspeicher und dem Betriebssystem Windows 2000 durchgeführt. Zur Entwicklung des Algorithmus wurde der Borland C++ Builder 6.0 verwendet.

In diesem Kapitel wurde zunächst die Entwicklung der Vehicle Routing Probleme erläutert und dann das dieser Arbeit zu Grunde liegende Entscheidungsproblem definiert. Anschließend wurden der bisher existierende Algorithmus skizziert und Anpassungsmöglichkeiten an das Entscheidungsproblem des dynamischen Pickup- and Delivery Vehicle Routing Problems mit Zeitfenstern aufgezeigt. Abschließend wurde die Erstellung der Testdatensätze erläutert und die entwickelte Testumgebung vorgestellt. Im folgenden Kapitel kann nun mit der Anpassung des Algorithmus begonnen werden. Dazu wird die Routenplanung analysiert, angepasst und getestet.

Kapitel 3

Die Beschleunigung des Single Vehicle Routings

Das Verfahren zur Lösung des Single Vehicle Pickup and Delivery Routing Problems mit Zeitfenstern bildet das Kernstück des gesamten Algorithmus. Wegen der hohen Komplexität der Berechnungen und der häufigen Aufrufe des Verfahrens ist die Entwicklung eines sehr schnellen Algorithmus nötig, um eine exakte Lösung in einem dynamischen System mit mehreren Aufträgen pro Fahrzeug zu ermöglichen.

Unter *Single Vehicle Routing* wird hier die Berechnung einer optimalen Route für ein einzelnes Fahrzeug zu einem bestimmten Zeitpunkt τ verstanden. Das zugrunde liegende Entscheidungsproblem ist ein Single Vehicle Pickup and Delivery Routing Problem mit Zeitfenstern, das folgendermaßen definiert wird: Einem Fahrzeug f mit gegebener Fahrzeugkapazität l_f, gegebenem Arbeitszeitfenster und gegebener Route ist zu einem bestimmten Zeitpunkt τ_κ eine bestimmte Menge von Aufträgen $i = 1, \ldots, n_f$ mit den zugehörigen Mengen der Pickup- und Delivery-Orte P_f bzw. D_f zugeordnet, die von dem Fahrzeug bedient werden müssen und zum Teil schon im Fahrzeug geladen sein können. Das Ziel besteht in der Minimierung der Fahrzeit der zukünftigen Route, wobei das Fahrzeug alle noch zu bedienenden Aufträge unter Einhaltung der zugehörigen Reihenfolge-, Kapazitäts- und Zeitfensterrestriktionen bedienen muss. Dieses Entscheidungsproblem ist der Spezialfall für $|F| = 1$ des in Abschnitt 2.3.3 beschriebenen statischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern.

Bemerkung: Alle in diesem Kapitel vorkommenden Variablen müssten mit dem Index f gekennzeichnet werden, da sie sich jeweils auf ein bestimmtes Fahrzeug beziehen. Diese Indizierung wird jedoch aus Gründen der besseren Lesbarkeit unterlassen, da im gesamten Kapitel immer nur genau ein Fahrzeug betrachtet wird. Im Folgenden werden also beispielsweise die Kapazität des Fahrzeugs l, die zugeordneten Aufträge $1, \ldots, n$ und die Mengen der zugehörigen Pickup- und Delivery-Orte des Fahrzeugs P bzw. D genannt.

Das Single Vehicle Routing wird immer dann benötigt, wenn für ein Fahrzeug eine neue zukünftige Route berechnet werden muss. Dies geschieht sowohl in der Heuristik zur Berechnung einer neuen zulässigen Lösung nach Eingang eines neuen Auftrags als auch während des Verfahrens der Tabu Suche zur Verbesserung der Zuordnung.

Die Heuristik zur Berechnung einer neuen zulässigen Lösung wird bei Eingang eines neuen Auftrags Nr. κ durchgeführt, das Ergebnis entscheidet über Annahme oder Ablehnen dieses Auftrags. Zum Zeitpunkt τ_κ des Auftragseingangs hat jedes Fahrzeug eine bestimmte Menge an Aufträgen und eine optimale Route zur Bedienung der Aufträge unter Einhaltung der Nebenbedingungen für die Zeitfenster, die Fahrzeugkapazität und die Reihenfolge. In der Heuristik wird der neue Auftrag Nr. κ „testweise“ jedem Fahrzeug zusätzlich zugeordnet und mit dem Single Vehicle Routing eine neue optimale zukünftige Route bestimmt (vgl. Abbildung 2.5 in Abschnitt 2.4). Wird für mindestens ein Fahrzeug eine optimale Route gefunden, wird der Auftrag angenommen und dem Fahrzeug mit optimaler Route und minimaler Erhöhung der Fahrzeit durch Annahme des neuen Auftrags Nr. κ zugeordnet. Andernfalls wird der Auftrag abgelehnt.

Das Verfahren der lokalen Suche wird immer dann aufgerufen, wenn gerade kein neuer Auftragseingang vorliegt. Es verändert die Zuordnung der bereits angenommenen Aufträge zu den Fahrzeugen, um die von der Heuristik gefundene Lösung zu verbessern. Zum Zeitpunkt τ des Beginns der Berechnungen verfügt jedes Fahrzeug über eine bestimmte Menge an Aufträgen und eine optimale Route zur Bedienung der Aufträge unter Einhaltung der dazugehörigen Nebenbedingungen. Gesucht wird eine andere Zuordnung der Aufträge zu den Fahrzeugen, wobei nur die zukünftigen Routen der Fahrzeuge verändert werden dürfen und alle noch zu bedienenden Aufträge unter Einhaltung der dazugehörigen Nebenbedingungen bedient werden müssen. Zur Bewertung jeder geänderten Zuordnung wird die Gesamtfahr- und -wartezeit des dazugehörigen neuen Routenplans mit der Gesamtfahr- und -wartezeit des bisherigen Routenplans verglichen. Zur Erstellung dieses neuen Routenplans wird für jedes von der geänderten Zuordnung betroffene Fahrzeug mit Hilfe des Single Vehicle Routings eine neue optimale Route bestimmt.

Caramia et al. [11] haben für ihr Verfahren zum Single Vehicle Routing den sogenannten Statusvektorbaum entwickelt, der die spezielle Struktur des Single Vehicle Pickup and Delivery Routing Problems mit Zeitfenstern abbildet, indem die Reihenfolgebedingung „erst Pickup, dann Delivery“ in der Struktur des Graphen verankert wird. Auf diesem Graphen suchen sie mit Hilfe des A*-Algorithmus einen kürzesten Weg von der Quelle zu einer der Senken, wobei die Quelle des Graphen die Situation des Fahrzeugs zum Start der Berechnungen darstellt und die Senken mögliche Situationen nach Bedienung aller Fahrzeuge beschreiben.

Sowohl die von Caramia et al. entwickelte Graphenstruktur als auch die von ihnen gewählte Schätzfunktion für den A*-Algorithmus, eine Funktion zur Abschätzung der Länge des Weges von einem Knoten des Graphs zur Senke, beinhalten Potential für eine zum Teil erhebliche

Reduktion des Rechenaufwands im Single Vehicle Routing. In diesem Kapitel wird daher nach der Angabe eines dynamischen Modells für das Single Vehicle Routing in Abschnitt 3.1 im Abschnitt 3.2 zunächst das Vorgehen von Caramia et al. detailliert beschrieben und analysiert. Im Abschnitt 3.3 wird dann eine spezielle Änderung an der Struktur des Graphen vorgestellt, analysiert und ihre rechentechnische Auswirkung in Testläufen dokumentiert. Abschnitt 3.4 befasst sich mit den sich anbietenden Anpassungen des A*-Algorithmus an die besonderen Gegebenheiten des Single Vehicle Routings. Verschiedene Ansätze zur Beschleunigung des Algorithmus durch eine geschickte Wahl der Schätzfunktion werden vorgestellt und mit Hilfe von Testläufen hinsichtlich ihrer rechentechnischen Auswirkungen ausgewertet. Abschließend wird in Abschnitt 3.5 ein Fazit gezogen.

3.1 Ein dynamisches Modell unter Verwendung von Statusvektoren

In diesem Abschnitt wird ein dynamisches Modell des Entscheidungsproblems formuliert. Dazu werden die auch von Caramia et al. benutzten Statusvektoren eingeführt.

Der Begriff der dynamischen Programmierung wurde in den 1950er Jahren von Bellman geprägt [7]. Es handelt sich hierbei um einen algorithmischen Ansatz zur Lösung von mehrstufigen, Vorgeschichte-unabhängigen Entscheidungsprozessen (vgl. Abschnitt 2.3.2). Als Voraussetzung für die Anwendung der dynamischen Programmierung muss sich das Problem in Entscheidungsstufen unterteilen lassen, wobei in jeder Entscheidungsstufe eine Entscheidung getroffen werden muss und das System entsprechend dieser Entscheidung seinen Zustand verändert.

Im vorliegenden Problem muss eine Route mit minimaler Fahr- und Wartezeit gefunden werden, die zum Startzeitpunkt τ im Ort der Fahrzeugstartposition μ_0 beginnt und alle noch anzufahrenden Orte beinhaltet, für einen Teil der Orte die Reihenfolgebedingung und für alle Orte die Zeitfenster- und Kapazitätsrestriktionen einhält. Dazu wird für jeden noch zu bedienenden Auftrag $i = 1, \ldots, n$ ein zugehöriger *Status* $\varphi(i)$ eingeführt, der den Fortschritt der Bearbeitung dieses Auftrags beschreibt.[1] In dieser Arbeit haben die Status $\varphi(i)$ folgende Bedeutung:

$$\varphi(i) = \begin{cases} 0\text{: Auftrag } i \text{ wurde noch nicht abgeholt} \\ 1\text{: Auftrag } i \text{ wurde abgeholt, aber noch nicht ausgeliefert} \\ 2\text{: Auftrag } i \text{ wurde bereits ausgeliefert} \end{cases}$$

Für einen Auftrag i mit $\varphi(i) = 0$ kommt wegen der Reihenfolgebedingung als nächste Aktion nur das Pickup in Frage, während für einen Auftrag i' mit $\varphi(i') = 1$ nur das Delivery als nächste Aktion möglich ist. Der Übergang von einem Zustand der Bearbeitung eines Auftrags zum

[1] Die Beschreibung des Fortschritts der Bearbeitung eines Auftrags durch einen Status-Wert geht auf Psaraftis [58] zurück und wurde von Caramia et al. übernommen.

nächstmöglichen Zustand seiner Bearbeitung kann daher durch Addition von 1 zum aktuellen Status abgebildet werden.

Die Zustände aller Aufträge $i = 1, \ldots, n$ werden im *Statusvektor* $\Phi = (\varphi(1), \varphi(2), \ldots, \varphi(n))^T$ zusammengefasst. Der Statusvektor beinhaltet damit alle nötigen Informationen zu den Bearbeitungsstatus der Aufträge, die das Fahrzeug bedienen muss. Eine Änderung des Statusvektors in genau einem Eintrag $\varphi(i)$ bildet das Anfahren eines Orts des zugehörigen Auftrags i ab: ändert sich z. B. der Status $\varphi(i)$ von 1 auf 2, so hat das Fahrzeug den Delivery-Ort des Auftrags i angefahren. Das Anfahren eines nächsten Orts durch das Fahrzeug unter Beachtung der Reihenfolgebedingung kann also durch Addition eines Einheitsvektors der Dimension n zum Statusvektor abgebildet werden.

Diese einfache Abbildung erlaubt die Einbettung der Reihenfolgebedingung in ein dynamisches Modell. Die Entscheidungsstufen $k \in \{0, \ldots, 2n\}$ werden anhand der Anzahl bereits angefahrener Orte gesetzt, wobei die Pickup-Orte der zum Startzeitpunkt bereits im Fahrzeug geladenen Aufträge mitgezählt werden. Sind also bereits k^* Aufträge im Fahrzeug geladen, so beginnt das dynamische Modell auf Stufe $k^*,\ 0 \leq k^* \leq n$.

Die Zustandsvariablen $x_k = (\Phi_k, \mu_k, \alpha_k)^T$ bilden mit Hilfe eines Statusvektors $\Phi_k = (\varphi_k(1), \varphi_k(2), \ldots, \varphi_k(n))^T$ sowie einer zugehörigen Fahrzeugposition $\mu_k \in V^{Orte} \setminus \{z\}$ und einer Ankunftszeit α_k im Ort der Fahrzeugposition den Zustand eines Fahrzeugs auf Stufe k ab. Die Angabe der Fahrzeugposition μ_k ist wegen der von ihr abhängigen Kosten für den Weg zum nächsten Ort entscheidend, die Ankunftszeit muss wegen der zu überprüfenden Zeitfenster mitgeführt werden. Für die Zulässigkeit einer Zustandsvariablen x_k der Stufe k muss gelten, dass bereits k Orte angefahren wurden: $\sum_{i=1}^{n} \varphi_k(i) = k$, die Fahrzeugposition ein möglicherweise zuletzt angefahrener Pickup- oder Delivery-Ort ist: $\mu \in \{p_i \in P \,|\, \varphi_k(i) = 1 \neq \varphi_{k^*}(i)\} \cup \{d_i \in D \,|\, \varphi_k(i) = 2\}$, und die Ankunftszeit innerhalb der Zeitfenster des Orts μ_k liegen: $\alpha_k \in [\underline{t}_{\mu_k}, \overline{t}_{\mu_k}]$. Die Menge X_k der zulässigen Zustände auf Stufe k lässt sich daher schreiben als

$$X_k = \Big\{(\Phi_k, \mu_k, \alpha_k)^T |\ \Phi_k \in \{0,1,2\}^n,\ \sum_{i=1}^{n} \varphi_k(i) = k,$$
$$\mu_k \in \{p_i \in P \,|\, \varphi_k(i) = 1 \neq \varphi_{k^*}(i)\} \cup \{d_i \in D \,|\, \varphi_k(i) = 2\},\ \alpha_k \in [\underline{t}_{\mu_k}, \overline{t}_{\mu_k}]\Big\}$$

Die Entscheidungsvariablen $y_k \in \{e_i |\ i \in \{1, \ldots, n\}\}$ sind Einheitsvektoren der Dimension n. Sie bilden die Auswahl eines Auftrags i ab, dessen Ort als nächstes angefahren werden soll. Eine Entscheidung y_k ist nur dann zulässig, wenn die Kapazität des Fahrzeugs nicht überschritten wird, also die Summe der benötigten Kapazitäten g_i aller im Fahrzeug geladenen Aufträge i die Fahrzeugkapazität l nicht übersteigt: $\sum_{i:(\phi_{k-1}+y_k)(i)=1} g_i \leq l$. Die Einhaltung der Zeitfensterrestriktion am Ort der neuen Fahrzeugposition wird bereits in der Menge der zulässigen Zustände überprüft. Für die Menge der zulässigen Entscheidungen gilt daher:

$$Y_k = \{e_i |\ i \in \{1, \ldots, n\},\ \sum_{i:(\Phi_{k-1}+y_k)(i)=1} g_i \leq l\}$$

Der Übergang von einem Zustand des Fahrzeugs x_{k-1} zum nächsten Zustand x_k wird in der Übergangsfunktion $h(x_{k-1}, y_k)$ abgebildet. Der neue Zustand $(\Phi_k, \mu_k, \alpha_k)^T$ ergibt sich durch Addition der Entscheidungsvariablen y_k zum Statusvektor Φ_k sowie der Änderung der Fahrzeugposition auf den entsprechenden Ort μ_k und der Berechnung der Ankunftszeit in μ_k als Minimum aus der Ankunftszeit des Zustands x_{k-1} zuzüglich der anfallenden Fahr- und Servicezeit und der unteren Zeitfenstergrenze des Ortes μ_k:

$$h(x_{k-1}, y_k) = \begin{pmatrix} \Phi_{k-1} + y_k \\ \mu_k = \begin{cases} p_j, & y_k = e_j \wedge \varphi_{k-1}(j) = 0 \\ d_j, & y_k = e_j \wedge \varphi_{k-1}(j) = 1 \end{cases} \\ \alpha_k = \max(\alpha_{k-1} + s_{\mu_{k-1}} + c_{\mu_{k-1},\mu_k}, \underline{t}_{\mu_k}) \end{pmatrix}$$

Die Kostenfunktion $C(x_{k-1}, y_k)$ gibt die für den Übergang von Zustand x_{k-1} in den Zustand x_k anfallende Fahr- und Wartezeit wieder:

$$C(x_{k-1}, y_k) = c_{\mu_{k-1},\mu_k} + \max\{\underline{t}_{\mu_k} - (\alpha_{k-1} + s_{\mu_{k-1}} + c_{\mu_{k-1},\mu_k}),\ 0\}$$

Als Startzustand dient der Zustandsvektor $(\Phi_{k^*}, \mu_{k^*}, \tau)^T$, der den Zustand des Fahrzeugs zum Startzeitpunkt wiedergibt: im Statusvektor Φ_{k^*} werden alle noch zu bedienenden Aufträge $i = 1, \ldots, n$ durch ihren Zustand $\varphi_{k^*}(i) \in \{0, 1\}$ zum Startzeitpunkt charakterisiert, zusätzlich sind die Fahrzeugstartposition $\mu_{k^*} \in V^{Orte} \setminus \{z\}$ und der Startzeitpunkt τ bekannt.

Das Modell lässt sich nun folgendermaßen formulieren:

$$\min \sum_{k=k^*}^{2n} C(x_{k-1}, y_k) \tag{3.1}$$

u.d.N.

$$x_{k^*} = (\Phi_{k^*}, \mu_{k^*}, \tau)^T \tag{3.2}$$

$$x_k \in X_k \qquad \forall\ k = k^*, \ldots, 2n \tag{3.3}$$

$$y_k \in Y_k \qquad \forall\ k = k^* + 1, \ldots, 2n \tag{3.4}$$

$$x_k = h(x_{k-1}, y_k) \qquad \forall\ k = k^* + 1, \ldots, 2n \tag{3.5}$$

Die Zielfunktion 3.1 minimiert die Gesamtfahr- und -wartezeit. In Bedingung 3.2 wird der Startzustand x_{k^*} festgelegt. Die Bedingung 3.3 lässt nur zulässige Kombinationen aus Statusvektoren, Fahrzeugpositionen und Ankunftszeiten als Zustandsvariablen zu, während Bedingung 3.4 die möglichen Entscheidungsvariablen auf die Einheitsvektoren des $\mathbb{R}^n$ beschränkt, die keine Kapazitätsüberschreitung zur Folge haben. In der Bedingung 3.5 wird schließlich der Übergang von einem Zustand zum nächsten abgebildet.

Im folgenden Abschnitt wird nun das Vorgehen von Caramia et al. zur Berechnung einer optimalen Lösung dieses Entscheidungsproblems beschrieben.

3.2 Die algorithmische Behandlung des Single Vehicle Routings von Caramia et al.

Dieser Abschnitt beschäftigt sich mit dem Single Vehicle Routing, wie es Caramia et al. in [11] vorstellten. Hier besteht die Problemstellung darin, zu einem bestimmten Zeitpunkt τ für ein bestimmtes Fahrzeug eine optimale zukünftige Route durch alle von diesem Fahrzeug noch anzufahrenden Orte unter Einhaltung der entsprechenden Zeitfenster-, Kapazitäts- und Reihenfolgebedingungen zu bestimmen.

Zur Bestimmung einer optimalen Lösung für das Single Vehicle Routing benutzen Caramia et al. sogenannte Statusvektoren, mit deren Hilfe sie einen Graphen definieren, den Statusvektorbaum. Die Besonderheit dieses Graphen besteht darin, dass die für die Aufträge geltende Reihenfolgebedingung in der Struktur des Graphen enthalten ist. Das Finden eines kürzesten Wegs über alle noch zu bedienenden Orte kann daher – ähnlich zu dem in 3.1 vorgestellten dynamischen Modell – mit den Methoden der dynamischen Programmierung realisiert werden.

Im folgenden Abschnitt wird der Statusvektorbaum eingeführt und anschließend im Abschnitt 3.2.2 analysiert. Die algorithmische Lösung des Problems mit dem A*-Algorithmus wird im Abschnitt 3.2.3 detailliert beschrieben, bevor im Abschnitt 3.2.4 die Ergebnisse der Testläufe präsentiert werden. Zum Abschluss wird der Algorithmus im Abschnitt 3.2.5 kritisch gewürdigt.

3.2.1 Der Statusvektorbaum

Caramia et al. benutzen für ihren Algorithmus eine Baumstruktur, mit deren Hilfe sie alle möglichen Sequenzen von Statusvektoren für die Abarbeitung aller noch zu bedienenden Aufträge abbilden können. Sie nutzen aus, dass zur Beschreibung des Zustands eines Fahrzeugs Statusvektoren zur Angabe der Bearbeitungsstatus aller Aufträge herangezogen werden können und eine Änderung des Zustands durch die Addition eines Einheitsvektors dargestellt werden kann, wobei die Reihenfolgebedingung automatisch erfüllt ist. Dabei beziehen sie – im Unterschied zum dynamischen Modell aus Abschnitt 3.1 – weder die Fahrzeugposition noch die Ankunftszeit in die Zustandsbeschreibung des Fahrzeugs ein, da sie wegen der genutzten Baumstruktur über die Historie der vorhergehenden Zustände des Fahrzeugs die Fahrzeugposition und die Ankunftszeit bestimmen können.

Im Folgenden wird der Statusvektorbaum eingeführt. Der Aufbau des Statusvektorbaums entspricht der von Caramia et al. genutzten Baumstruktur, beinhaltet jedoch keine Beachtung der Nebenbedingungen wie Zeitfensterrestriktionen oder einzuhaltende Kapazitäten. Da die Nebenbedingungen abhängig von den speziellen, laut Aufgabenstellung nicht vorhersehbaren Ausprägungen der Aufträge sind, kann darüber keine allgemeine Aussage gemacht werden. Zur Analyse der Baumstruktur werden sie daher zunächst nicht betrachtet.

Zur nachfolgenden Beschreibung des Statusvektorbaums werden Multimengen benötigt, die folgendermaßen definiert werden:

Definition 3.1 (Multimenge)
Eine Multimenge Γ auf einer Menge B wird definiert durch eine Zugehörigkeitsfunktion $\gamma : B \to \mathbb{N}_0$. Für ein Element $b \in B$ gibt $\gamma(b)$ die Anzahl der Wiederholungen von b in der Multimenge Γ an. Die Kardinalität der Multimenge wird definiert als

$$|\Gamma| = \sum_{b \in B} \gamma(b)$$

Für die Vereinigung von zwei Multimengen Γ_1 auf der Menge B_1 und Γ_2 auf der Menge B_2 gilt:

$$\Gamma^* = \Gamma_1 \uplus \Gamma_2 \quad \Leftrightarrow \quad \forall\, b \in B = B_1 \cup B_2 : \gamma^*(b) = \begin{cases} \gamma_1(b) + \gamma_2(b), & b \in B_1 \cap B_2 \\ \gamma_1(b), & b \in B \setminus B_2 \\ \gamma_2(b), & b \in B \setminus B_1 \end{cases}$$

Mit Hilfe dieser Multimengen kann nun der Statusvektorbaum (S) als Graph G^S_{n,k^*} formuliert werden, $G^S_{n,k^*} = (V^S_{n,k^*}, E^S_{n,k^*}, c^S_{n,k^*})$. Die Knoten des Statusvektorbaums sind Statusvektoren, die Kanten bilden mögliche Übergänge von einem Statusvektor zum nächsten Statusvektor ab. Die Wurzel des Baums stellt den Zustand des Fahrzeugs zum Startzeitpunkt τ dar und entspricht dem Statusvektor Φ_{k^*}, der die Bearbeitungsstatus zum Zeitpunkt τ der noch zu bedienenden Aufträge wiedergibt. Entsprechend den Stufen im dynamischen Modell in 3.1 kann auch der Statusvektorbaum anhand der Anzahl der bereits erfolgten Statusänderungen in Stufen unterteilt werden. Die Wurzel liegt auf Stufe k^*, da bereits k^* Einträge im Statusvektor den Wert 1 haben.

Für die Definition der Knotenmenge V^S_{n,k^*} werden zunächst die Knotenmengen $-\, {}_kV^S_{n,k^*}$ der Stufen $k = k^*, \ldots, 2n$ definiert. Auf Stufe k^* gibt es nur die Wurzel:

$${}_{k^*}V^S_{n,k^*} = \{\Phi_{k^*}\}$$

Die Menge der Knoten auf einer Stufe $k+1$ besteht aus allen, nicht notwendigerweise verschiedenen Statusvektoren, die aus den Statusvektoren der Stufe k durch Addition eines Einheitsvektors hervorgehen. Auf Stufe k gilt für alle Statusvektoren, dass die Summe aller Status genau die Anzahl k der bereits erfolgten Statusänderungen ergeben muss und für jeden Auftrag i der Status mindestens seinem Status zum Startzeitpunkt entsprechen muss. Die Menge ${}_kW_{n,k^*}$ der möglichen Statusvektoren der Stufe k lässt sich daher schreiben als

$${}_kW_{n,k^*} = \{\Phi \in \{0,1,2\}^n \mid \sum_{i=1}^{n} \varphi(i) = k,\ \varphi(i) \geq \varphi_{k^*}(i)\ \forall\ i = 1, \ldots, n\}$$

Für die Definition der Knotenmengen auf den folgenden Stufen $k = k^* + 1, \ldots, 2n$ wird für einen Statusvektor Φ die *Menge aller Nachfolger* $Q(\Phi)$ definiert als

$$Q(\Phi) = \{\Phi' \in \{0,1,2\}^n \mid \exists\, e_i : \Phi' = \Phi + e_i\}$$

Die Menge der Nachfolger $Q(\Phi)$ eines Knotens Φ der Stufe $k-1$ kann als Multimenge $\tilde{Q}(\Phi)$ auf der Menge W_k mit der Zugehörigkeitsfunktion $\tilde{q}_\Phi : W_k \to \{0,1\}$ geschrieben werden, die jedem Statusvektor $\Phi' \in W_k$ genau dann den Wert 1 zuordnet, falls er in der Menge $Q(\Phi)$ enthalten ist:

$$\tilde{q}_\Phi(\Phi') = \begin{cases} 1, & \Phi' \in Q(\Phi) \\ 0, & \text{sonst} \end{cases}$$

Die Menge ${}_kV^S_{n,k^*}$ der Knoten auf einer Stufe k ist die Multimenge auf der Menge W_k, die aus den Mengen der Nachfolger aller Knoten der vorhergehenden Stufe $k-1$ besteht. Mit Hilfe der Multimengen $\tilde{Q}(\Phi)$ kann man nun die Multimenge ${}_kV^S_{n,k^*}$ der Knoten auf einer Stufe $k > k^*$ definieren als:

$${}_kV^S_{n,k^*} = \biguplus_{\Phi \in {}_{k-1}V^S_{n,k^*}} \tilde{Q}(\Phi)$$

Die Multimenge aller Knoten des Graphen ist die Vereinigung der Multimengen der Knoten auf den Stufen $k = k^*, \ldots, 2n$:

$$V^S_{n,k^*} = \biguplus_{k=k^*,\ldots,2n} {}_kV^S_{n,k^*}$$

Die Kanten des Graphen bilden den Übergang von einem Statusvektor zu seinem Nachfolger ab. Die Menge aller Kanten stellt ebenfalls eine Multimenge dar. Sie wird analog zur Knotenmenge V^S_{n,k^*} mit Hilfe der Kantenmengen auf den einzelnen Stufen k definiert. Dazu wird zu jedem Knoten Φ einer Stufe $k, k^* \le k < 2n$, die Menge $E(\Phi)$ der von diesem Knoten ausgehenden Kanten definiert:

$$E(\Phi) = \{(\Phi, \Phi') \,|\, \Phi' \in \{0,1,2\}^n, \exists\ e_i : \Phi' = \Phi + e_i\}$$

Die Menge der ausgehenden Kanten $E(\Phi)$ kann man als Multimenge $\tilde{E}(\Phi)$ auf der Menge $W_k \times W_{k+1}$ schreiben, indem man die Zugehörigkeitsfunktion $\tilde{e} : W_k \times W_{k+1} \to \{0,\ 1\}$ definiert als:

$$\tilde{e}_\Phi(\Phi, \Phi') = \begin{cases} 1, & (\Phi, \Phi') \in E(\Phi) \\ 0, & \textit{sonst} \end{cases}$$

Alle von einer Stufe k ausgehenden Kanten können nun als die Vereinigung der Multimengen $\tilde{E}(\Phi)$ aller Knoten $\Phi \in {}_kV^S_{n,k^*}$ dargestellt werden:

$${}_kE^S_{n,k^*} = \biguplus_{\Phi \in {}_kV^S_{n,k^*}} \tilde{E}(\Phi)$$

Die Multimenge aller Kanten des Graphen ist die Vereinigung der Multimengen der ausgehenden Kanten der Stufen $k = k^*, \ldots, 2n-1$:

$$E^S_{n,k^*} = \biguplus_{k=k^*,\ldots,2n-1} {}_kE^S_{n,k^*}$$

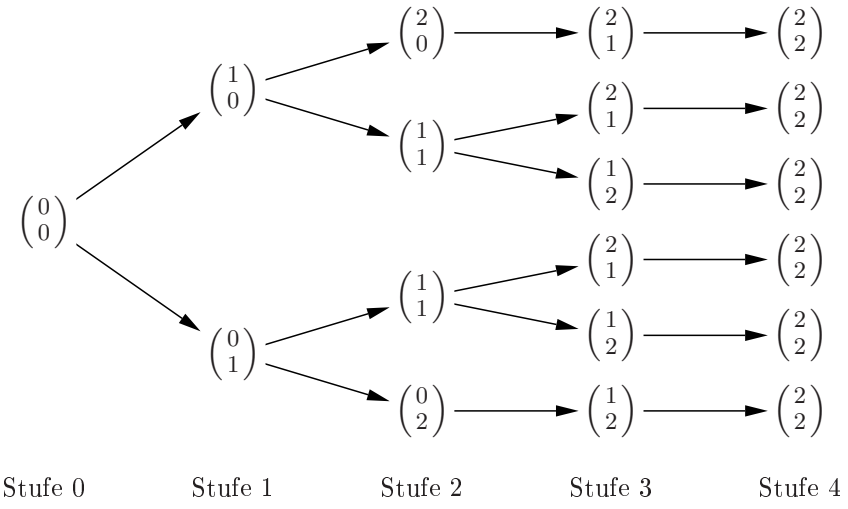

Abbildung 3.1: Der Statusvektorbaum $G^S_{2,0}$ für zwei bisher nicht bearbeitete Aufträge.

Die Kostenfunktion $c^S_{n,k^*} : E^S_{n,k^*} \to \mathbb{R}^+_0$ ordnet jeder Kante $(\Phi, \Phi') \in E^S_{n,k^*}$ die zugehörige Fahrzeit für den Übergang von Statusvektor Φ zu Statusvektor Φ' zu. Für die Wurzel des Statusvektorbaums ist die Fahrzeugstartposition bekannt, für alle anderen Knoten Φ muss über den Vorgänger Φ'' die eindeutig zugehörige Fahrzeugposition μ ermittelt werden: Es gibt genau einen Einheitsvektor e_{i^*} mit $\Phi = \Phi'' + e_{i^*}$, so dass darüber der dazugehörige bediente Auftrag i^* bestimmt werden kann. Der Wert $\varphi(i^*)$ im Vektor $\Phi = (\varphi(1), \ldots, \varphi(n))$ gibt an, ob der Pickup- oder Delivery-Ort des Auftrags i^* angefahren wurde (vgl. Abschnitt 3.1). Analog kann die zum Statusvektor Φ' zugehörige Fahrzeugposition μ' ermittelt werden. Die Kostenfunktion c^S_{n,k^*} gibt die Fahrzeit zwischen den Orten μ und μ' an, d. h. $c^S_{n,k^*}(\Phi, \Phi') = c_{\mu,\mu'}$. Damit ist der Statusvektorbaum $G^S_{n,k^*} = (V^S_{n,k^*}, E^S_{n,k^*}, c^S_{n,k^*})$ definiert.

Abbildung 3.1 zeigt beispielhaft einen Statusvektorbaum für zwei Aufträge, die noch nicht bedient wurden. Die Wurzel des Baums ist der Statusvektor $(0,0)^T$, da beide Aufträge noch auf den Pickup warten. Danach kann entweder Auftrag 1 oder Auftrag 2 eingeladen werden, d. h. auf Stufe 1 ist entweder der Status des ersten oder der Status des zweiten Auftrags 1: ${}_1V^S_{2,0} = \{(1,0)^T, (0,1)^T\}$. In der Multimenge ${}_2V^S_{2,0}$ der Knoten auf Stufe 2 ist der Statusvektor $(1,1)^T$ mit Kardinalität 2 vorhanden, da er in den Mengen der Nachfolger von Knoten $(1,0)^T$ und $(0,1)^T$ der Stufe 1 enthalten ist. Die Kanten des Graphen bilden gerade diese Nachfolgebeziehungen ab. Entsprechend sind die Blätter des Statusvektorbaums immer solche Statusvektoren, die keine Nachfolger besitzen, deren Einträge also sämtlich den Wert $\varphi(i) = 2$ haben. Ein solcher Statusvektor bildet ab, dass alle Aufträge vollständig bedient wurden.

Zur Verdeutlichung der Bedeutung der Stufe k^* zeigt Abbildung 3.2 einen Statusvektorbaum für drei Aufträge, von denen zwei bereits abgeholt wurden, d. h. $k^* = 2$. Die Wurzel des Statusvektorbaums ist daher ein Statusvektor mit genau $k^* = 2$ Einträgen $\varphi(i) = 1$. Die Abbildung zeigt den Fall, dass der erste und zweite Auftrag bereits geladen sind.

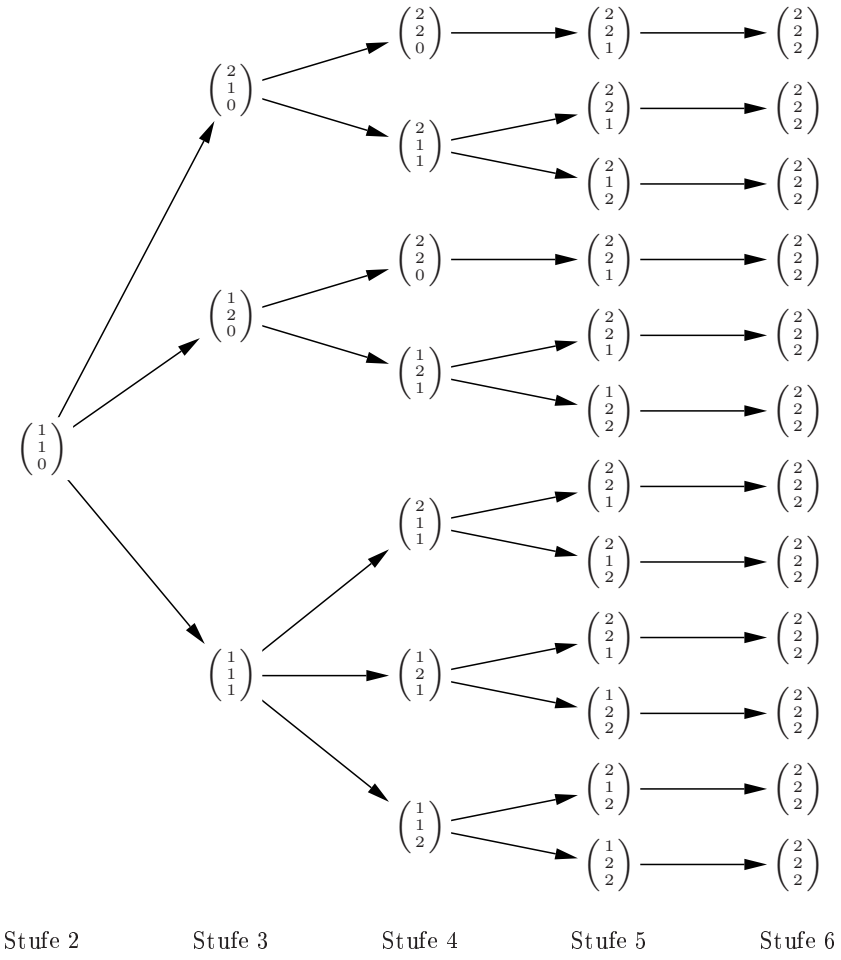

Abbildung 3.2: Der Statusvektorbaum $G^S_{3,2}$ für drei Aufträge, von denen zwei bereits abgeholt wurden.

Jeder Weg von der Wurzel des Statusvektorbaums zu einem der Blätter bildet eine mögliche Route des Fahrzeugs ab: über die Änderungen der Statusvektoren kann eindeutig die Sequenz der angefahrenen Orte, beginnend im Startort des Fahrzeugs, rekonstruiert werden. Die Suche nach einer fahrtkostenminimalen Route des Fahrzeugs entspricht daher der Suche nach einem kürzesten Weg von der Wurzel des Statusvektorbaums zu einem der Blätter.

Die Komplexität der Berechnung eines kürzesten Wegs in einem Graphen hängt von der Anzahl der Knoten und Kanten im Graphen ab. Für die angestrebte kurze Rechenzeit ist es daher von besonderem Interesse, die Berechnungen auf einem Graphen mit möglichst wenigen Knoten und Kanten durchzuführen, ohne jedoch mögliche Lösungen zu übergehen. Der Statusvektorbaum erfüllt diese Anforderung nicht: Durch die gewählte Baumstruktur muss mit Multimengen gearbeitet werden, weil einige Statusvektoren mehrfach vorkommen. Allerdings kann die Knotenmenge einer Stufe k auch bei einer anderen Graphenstruktur nicht unbedingt

auf die Menge ${}_kW_{n,k^*}$ der möglichen Statusvektoren reduziert werden, da die Position des Fahrzeugs mitentscheidend für die eindeutige Bestimmung des Zustands des Fahrzeugs ist (vgl. Abschnitt 3.1). In Abbildung 3.1 ist zum Beispiel auf Stufe 2 der Statusvektor $(1,1)^T$ doppelt vorhanden. Dabei werden aber zwei verschiedene Situationen beschrieben, wie sich am Aufbau des Baums erkennen lässt: im oberen Fall befindet sich das Fahrzeug im Pickup-Ort des zweiten Auftrags, während es im unteren Fall im Pickup-Ort des ersten Auftrags steht. Auf der dritten Stufe sind hingegen schon Zustände doppelt beschrieben: es gibt jeweils drei Zustandsvektoren $(2,1)^T$ und $(1,2)^T$, die Fahrzeuge können sich jedoch nur in zwei verschiedenen Orten befinden: für den Vektor $(2,1)^T$ z. B. im Delivery-Ort des ersten oder im Pickup-Ort des zweiten Auftrags. Jeweils einer dieser Statusvektoren könnte eingespart werden, wenn man auf die Baumstruktur verzichtet und statt dessen die Fahrzeugposition als zusätzliche Charakteristik einführt. Zur Überprüfung, ob sich der ungleich höhere Implementierungsaufwand dieser anderen Graphenstruktur lohnen könnte, und zum besseren Verständnis der Komplexität der Berechnung eines kürzesten Wegs wird im Folgenden der Statusvektorbaum genauer analysiert.

3.2.2 Analyse des Statusvektorbaums

Um den Rechenaufwand zur Bestimmung eines kürzesten Wegs von der Wurzel zu einem der Blätter des Statusvektorbaums abschätzen zu können, sind die Anzahl der Knoten und die Anzahl der Kanten im Graphen entscheidend: die Anzahl der Knoten entspricht der Anzahl der zu speichernden Strukturen, während die Anzahl der Kanten für die verschiedenen Möglichkeiten zum Erreichen eines Knotens und damit für die Rechenoperationen steht. Für den Statusvektorbaum wird in diesem Abschnitt zunächst die Anzahl der Knoten angegeben. Dazu wird zunächst eine intuitive rekursive Formel und anschließend eine Formel zur direkten Berechnung hergeleitet. Die Anzahl der Kanten muss nicht gesondert berechnet werden, da jeder Knoten des Statusvektorbaums außer der Wurzel genau eine eingehende Kante hat und die Anzahl der Kanten daher der Anzahl der Knoten minus 1 entspricht. Zum Abschluss dieses Abschnitts werden zur Verdeutlichung die Ergebnisse für Statusvektorbäume mit bis zu 10 Aufträgen präsentiert.

Für die folgende Analyse wird der Begriff des *Teilbaums* eingeführt: Gegeben ist ein Statusvektorbaum $G^S_{n,k^*} = (V^S_{n,k^*}, E^S_{n,k^*}, c^S_{n,k^*})$ mit Wurzel Φ_{k^*}. Ein Teilbaum mit Wurzel $\Phi \in V^S_{n,k^*} \setminus \{\Phi_{k^*}\}$ ist ein Baum mit Wurzel Φ, dessen Knoten- und Kantenmengen jeweils Teilmengen der Knoten- und Kantenmengen des Statusvektorbaums G^S_{n,k^*} darstellen und der sämtliche von Φ erreichbaren Blätter des Statusvektorbaums G^S_{n,k^*} enthält. Im Unterschied zu einem Statusvektorbaum kann die Wurzel des Teilbaums auch Einträge $\varphi(i) = 2$ enthalten. Abbildung 3.3 zeigt beispielhaft einen Teilbaum in einem Statusvektorbaum für 3 Aufträge, beginnend auf Stufe $k* = 2$.

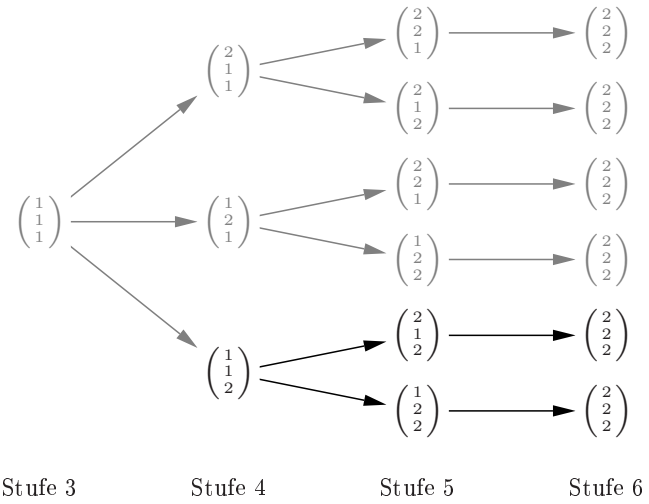

Abbildung 3.3: Ein Teilbaum mit Wurzel $(1,1,2)^T$ in einem Statusvektorbaum $G^S_{3,3}$ für drei Aufträge mit Wurzel auf Stufe 3.

Der Statusvektorbaum für n Aufträge mit Wurzel $\Phi_{k^*} \in \{0,1\}^n$ hat offensichtlich die folgenden Eigenschaften (vgl. zur Plausibilisierung auch Abbildungen 3.1 und 3.2):

- Für einen Statusvektor $\Phi = (\varphi(1), \ldots, \varphi(n))^T \in {}_kW_{n,k^*}$ auf Stufe k gilt für die Summe aller Elemente: $\sum_{i=1}^{n} \varphi(i) = k$.
- Für die Stufen k mit $k^* \leq k \leq n$ gilt: Jeder Vektor $\Phi \in {}_kW_{n,k^*}$ mit $\varphi(i) \leq 1$ für alle $i = 1, \ldots, n$ hat genau k Einträge 1 und $n-k$ Einträge 0. Die Menge seiner Nachfolger $Q(\Phi)$ enthält genau k Vektoren Φ' mit jeweils einem neuen Eintrag $\varphi'(i) = 2$ sowie $n-k$ Vektoren Φ' mit $\varphi'(i) \leq 1$ für alle $i = 1, \ldots, n$.
- Für jeden Statusvektor Φ einer Stufe k mit genau einem Element $\varphi(i^*) = 2$ gilt: Die Anzahl der Knoten des Teilbaums mit Wurzel Φ, entspricht der Anzahl der Knoten des Statusvektorbaums für die Aufträge $\{1, \ldots, n\} \setminus \{i^*\}$ mit Wurzel auf Stufe $k-1$, dessen Wurzel sich durch Streichen des Elements $\varphi(i^*)$ aus dem Statusvektor Φ ergibt.
- Auf Stufe $n+1$ hat jeder Statusvektor Φ mindestens einen Eintrag $\varphi(i) = 2$. Die Anzahl der Knoten jedes Teilbaums mit Wurzel Φ auf Stufe $n+1$ kann daher rekursiv aus kleineren Statusvektorbäumen übertragen werden.

Mit Hilfe dieser Eigenschaften lässt sich die Anzahl der Knoten im Statusvektorbaum rekursiv bestimmen, indem man die Anzahl der Knoten jedes Teilbaums, dessen Wurzel ein Statusvektor Φ mit einem Element $\varphi(i^*) = 2$ bildet, rekursiv aus der Anzahl der Knoten des „kleineren“ Statusvektorbaums für $n-1$ Aufträge bestimmt, dessen Wurzel aus dem Statusvektor Φ durch Streichen des Elements $\varphi(i^*)$ entsteht.

Satz 3.1 (Rekursive Formulierung)
Es bezeichne $|V^S_{n,k}|$ die Anzahl der Knoten eines Teilbaums mit Wurzel $\Phi = (\varphi(1), \ldots, \varphi(n))$ mit $\varphi(i) < 2 \; \forall \; i = 1, \ldots, n$ auf Stufe k des Statusvektorbaums für n Aufträge mit

$$|V^S_{n,k}| = 0 \quad \forall \; n < 0 \vee k \notin \{0, \ldots, n\}$$

Dann gilt für $n \in \mathbb{N}_0, \; 0 \leq k \leq n$ rekursiv:

$$|V^S_{n,k}| = \sum_{j=0}^{n-k} \frac{(n-k)!}{(n-k-j)!}(1 + (k+j)|V^S_{n-1,k+j-1}|)$$

Beweis: Der Beweis erfolgt konstruktiv. Um die Anzahl der Knoten des Teilbaums mit Wurzel Φ zu zählen, braucht man:

- die Wurzel Φ mit Anzahl 1
- die Anzahl der Folgeknoten $\Phi' \in Q(\Phi)$, in denen genau ein Element $\varphi'(i) = 2$ ist: Die Anzahl der Knoten eines Teilbaums, der einen solchen Knoten Φ' als Wurzel besitzt, kann rekursiv aus kleineren Statusvektorbäumen berechnet werden. Da die Wurzel eines Teilbaums ein Statusvektor Φ mit Status $\varphi(i) < 2$ für alle $i = 1, \ldots, n$ ist und daher genau k Elemente von Φ den Wert $\varphi(i) = 1$ haben, werden k Folgeknoten $\Phi' \in Q(\Phi)$ mit genau einem Element $\varphi'(i) = 2$ erzeugt. Die Anzahl der Knoten eines Teilbaums mit solch einer Wurzel Φ' entspricht der Anzahl der Knoten des Statusvektorbaums für $n-1$ Aufträge, dessen Wurzel durch Streichen des Elements $\varphi'(i)$ aus dem Statusvektor Φ' entsteht, also $|V^S_{n-1,k-1}|$.
- die Anzahl der Folgeknoten $\Phi'' \in Q(\Phi)$, deren Elemente $\varphi''(i)$ alle kleiner oder gleich 1 sind. Diese Statusvektoren entstehen, indem zum Statusvektor Φ mit einem Element $\varphi(i^*) = 0$ der Einheitsvektor e_{i^*} addiert wird. Da im Ausgangsvektor Φ $n-k$ Elemente $\varphi(i) = 0$ waren, werden auf diese Weise $n-k$ Statusvektoren Φ'' erzeugt. Diese Statusvektoren haben jeweils $|V^S_{n,k+1}|$ Folgeknoten.

Man erhält damit folgende rekursive Formel:

$$|V^S_{n,k}| = 1 + k \; |V^S_{n-1,k-1}| + (n-k)|V^S_{n,k+1}|$$

Durch Auflösen der zweiten Rekursion ergibt sich:

$$\begin{aligned} |V^S_{n,k}| &= 1 + k|V^S_{n-1,k-1}| + (n-k)|V^S_{n,k+1}| \\ &= 1 + k|V^S_{n-1,k-1}| + \\ &\quad + (n-k)\Big(1 + (k+1)|V^S_{n-1,k}| + (n-k-1)|V^S_{n,k+2}|\Big) \end{aligned}$$

$$
\begin{aligned}
&= 1 + k|V^S_{n-1,k-1}| + \\
&\quad + (n-k) + (n-k)(k+1)|V^S_{n-1,k}| + (n-k)(n-k-1)|V^S_{n,k+2}| \\
&= 1 + k|V^S_{n-1,k-1}| + \\
&\quad + (n-k) + (n-k)(k+1)|V^S_{n-1,k}| + \\
&\quad + (n-k)(n-k-1)\Big(1 + (k+2)|V^S_{n-1,k+1}| + (n-k-2)|V^S_{n,k+3}|\Big) \\
&= 1 + k|V^S_{n-1,k-}| + \\
&\quad + (n-k) + (n-k)(k+1)|V^S_{n-1,k}| + \\
&\quad + (n-k)(n-k-1) + (n-k)(n-k-1)(k+2)|V^S_{n-1,k+1}| \\
&\quad + (n-k)(n-k-1)(n-k-2)|V^S_{n,k+3}| \\
&= \dots \\
&= 1 + k|V^S_{n-1,k-1}| + \\
&\quad + (n-k) + (n-k)(k+1)|V^S_{n-1,k}| + \\
&\quad + (n-k)(n-k-1) + (n-k)(n-k-1)(k+2)|V^S_{n-1,k+1}| + \\
&\quad \dots \\
&\quad + (n-k)(n-k-1)\cdot\dots\cdot 1 + (n-k)(n-k-1)\cdot\dots\cdot 1\ n|V^S_{n-1,n-1}| \\
&= \sum_{j=0}^{n-k} \frac{(n-k)!}{(n-k-j)!}(1 + (k+j)|V^S_{n-1,k+j-1}|)
\end{aligned}
$$

□

Die rekursive Formulierung ist sehr aufwändig, wenn man tatsächlich die Anzahl der Knoten berechnen möchte. Um eine direkte Formulierung für die Anzahl der Knoten angeben zu können, werden die beiden folgenden Lemmata benötigt:

Lemma 3.1
Für alle $q, k \in \mathbb{N}$ mit $q < k$ gilt:

$$\sum_{j=1}^{k-q} \frac{(j+2q)!}{(j-1)!} = \frac{(k+q+1)!}{2(q+1)(k-q-1)!}$$

BEWEIS: Für den Beweis wird zunächst eine Substitution vorgenommen: $r := k - q$. Zu beweisen ist demnach:

$$\sum_{j=1}^{r} \frac{(j+2q)!}{(j-1)!} = \frac{(r+2q+1)!}{2(q+1)(r-1)!}$$

Der Beweis erfolgt per vollständiger Induktion über r. Der Induktionsanfang sei $r = 1$. Einsetzen von $r = 1$ liefert die Behauptung für alle $q \in \mathbb{N}$:

$$\sum_{j=1}^{1} \frac{(j+2q)!}{(j-1)!} = \frac{(2q+1)!}{0!} = \frac{(2q+2)!}{2(q+1)0!}$$

Der Induktionsschritt geht von r zu $r+1$. Für alle $q \in \mathbb{N}$ gilt:

$$\begin{aligned}
\sum_{j=1}^{r+1} \frac{(j+2q)!}{(j-1)!} &= \sum_{j=1}^{r} \frac{(j+2q)!}{(j-1)!} + \frac{(r+2q+1)!}{(r)!} \\
&= \frac{(r+2q+1)!}{2(q+1)(r)!} + \frac{(r+2q+1)!}{(r)!} \\
&= \frac{(r+2q+1)!(r+2(q+1))}{2(q+1)r!} \\
&= \frac{(r+2q+2)!}{2(q+1)r!}
\end{aligned}$$

□

LEMMA 3.2

Faktorisierung der q-fach-Summe

Für die q-fach-Summe mit $q, k \in \mathbb{N}$ gilt:

$$\sum_{j_1=1}^{k} \sum_{j_2=j_1-1}^{k-1} \sum_{j_3=j_2-1}^{k-2} \cdots \sum_{j_q=j_{q-1}-1}^{k-q+1} j_1 j_2 j_3 \cdots j_q = \begin{cases} 0 & \text{für} \quad k < q \\ \frac{(k+q)!}{2^q (k-q)! q!} & \text{für} \quad k \geq q \end{cases}$$

BEWEIS: Der Beweis beginnt mit einer Indextransformation der q-fach-Summe und wird anschließend entsprechend der Fallunterscheidung für $k < q$ bzw. $k \geq q$ geführt.

Zunächst wird die Indextransformation vorgenommen. Für die q-fach Summe mit einer beliebigen Funktion $\delta : \mathbb{N} \to \mathbb{R}$ gilt:

$$\sum_{j_1=1}^{k} \sum_{j_2=j_1-1}^{k-1} \cdots \sum_{j_q=j_{q-1}-1}^{k-q+1} j_1 j_2 \cdots j_q \cdot \delta(j_q) = \sum_{j_q=1}^{k-q+1} \sum_{j_{q-1}=1}^{j_q+1} \cdots \sum_{j_1=1}^{j_2+1} j_1 j_2 \cdots j_q \cdot \delta(j_q)$$

Die Gleichheit wird mit Hilfe der vollständigen Induktion über die Anzahl q der Summen gezeigt. Für den Induktionsanfang wird $q = 2$ gesetzt:

$$\begin{aligned}\sum_{j_1=1}^{k}\sum_{j_2=j_1-1}^{k-1} j_1 j_2 \cdot \delta(j_2) =& 1\cdot 0\cdot\delta(0) + 1\cdot 1\cdot\delta(1) + 1\cdot 2\cdot\delta(2) + \ldots + 1\cdot(k-1)\cdot\delta(k-1)\\ &+ 2\cdot 1\cdot\delta(1) + 2\cdot 2\cdot\delta(2) + \ldots + 2\cdot(k-1)\cdot\delta(k-1)\\ &+ 3\cdot 2\cdot\delta(2) + \ldots + 3\cdot(k-1)\cdot\delta(k-1)\\ &\cdots\\ &+ k\cdot(k-1)\cdot\delta(k-1)\\ =& \sum_{j_2=1}^{k-1}\sum_{j_1=1}^{j_2+1} j_1 j_2\cdot\delta(j_2) \qquad (3.6)\end{aligned}$$

Insbesondere ist nicht nur der Wert der Summe gleich, auch die positiven Summanden sind identisch. Die für $q \geq 3$ auftretenden Summanden mit negativen Faktoren haben den Wert 0 und können daher ignoriert werden. Die Induktionsvoraussetzung lautet somit:

$$\sum_{j_1=1}^{k}\sum_{j_2=j_1-1}^{k-1}\cdots\sum_{j_q=j_{q-1}-1}^{k-q+1} j_1 j_2\cdots j_q\cdot\delta(j_q) = \sum_{j_q=1}^{k-q+1}\sum_{j_{q-1}=1}^{j_q+1}\sum_{j_{q-2}=1}^{j_{q-1}+1}\cdots\sum_{j_1=1}^{j_2+1} j_1 j_2\cdots j_q\cdot\delta(j_q) \qquad (3.7)$$

Der Induktionsschluss läuft von q zu $q+1$ und benutzt, dass $\delta : \mathbb{N} \to \mathbb{R}$ eine beliebige Funktion sein darf:

$$\begin{aligned}&\sum_{j_1=1}^{k}\sum_{j_2=j_1-1}^{k-1}\cdots\sum_{j_q=j_{q-1}-1}^{k-q+1}\sum_{j_{q+1}=j_q-1}^{k-q} j_1 j_2\cdots j_q j_{q+1}\cdot\delta(j_{q+1})\\ &= \sum_{j_1=1}^{k}\sum_{j_2=j_1-1}^{k-1}\cdots\sum_{j_q=j_{q-1}-1}^{k-q+1} j_1 j_2\cdots j_q\cdot\Big(\sum_{j_{q+1}=j_q-1}^{k-q} j_{q+1}\cdot\delta(j_{q+1})\Big)\\ &= \sum_{j_1=1}^{k}\sum_{j_2=j_1-1}^{k-1}\cdots\sum_{j_q=j_{q-1}-1}^{k-q+1} j_1 j_2\cdots j_q\cdot\delta'(j_q)\\ &\overset{3.7}{=} \sum_{j_q=1}^{k-q+1}\sum_{j_{q-1}=1}^{j_q+1}\sum_{j_{q-2}=1}^{j_{q-1}+1}\cdots\sum_{j_1=1}^{j_2+1} j_1 j_2\cdots j_q\cdot\delta'(j_q)\\ &= \sum_{j_q=1}^{k-q+1}\sum_{j_{q-1}=1}^{j_q+1}\sum_{j_{q-2}=1}^{j_{q-1}+1}\cdots\sum_{j_1=1}^{j_2+1} j_1 j_2\cdots j_q\cdot\Big(\sum_{j_{q+1}=j_q-1}^{k-q} j_{q+1}\cdot\delta(j_{q+1})\Big)\\ &= \sum_{j_q=1}^{k-q+1} j_q\cdot\Big(\sum_{j_{q+1}=j_q-1}^{k-q} j_{q+1}\cdot\delta(j_{q+1})\Big)\Big(\sum_{j_{q-1}=1}^{j_q+1}\sum_{j_{q-2}=1}^{j_{q-1}+1}\cdots\sum_{j_1=1}^{j_2+1} j_1 j_2\cdots j_{q-1}\Big)\\ &\overset{3.6}{=} \sum_{j_q=1}^{k-q+1}\sum_{j_{q+1}=j_q-1}^{k-q}\big(j_q j_{q+1}\cdot\delta(j_{q+1})\big)\Big(\sum_{j_{q-1}=1}^{j_q+1}\sum_{j_{q-2}=1}^{j_{q-1}+1}\cdots\sum_{j_1=1}^{j_2+1} j_1 j_2\cdots j_{q-1}\Big)\\ &= \sum_{j_{q+1}=1}^{k-q}\sum_{j_q=1}^{j_{q+1}+1}\sum_{j_{q-1}=1}^{j_q+1}\cdots\sum_{j_1=1}^{j_2+1} j_1 j_2\cdots j_q j_{q+1}\cdot\delta(j_{q+1})\end{aligned}$$

Die Induktionsvoraussetzung 3.7 kann angewandt werden, da die Funktion $\delta' : \mathbb{N} \to \mathbb{R}$ mit

$$\delta'(j_q) = \sum_{j_{q+1}=j_q-1}^{k-q} j_{q+1} \cdot \delta(j_{q+1})$$

nur von der Variablen $j_q \in \mathbb{N}$ abhängt und somit die Anforderungen der Induktionsvoraussetzung erfüllt. Die vorletzte Umformung kann analog zum Induktionsanfang 3.6 durch Ausschreiben der ersten beiden Summen erhalten werden.

Für die Funktion $\delta(j_q) = 1$ kann die Indextransformation auf die gegebene q-fach-Summe angewandt werden. Man erhält

$$\sum_{j_1=1}^{k} \sum_{j_2=j_1-1}^{k-1} \cdots \sum_{j_q=j_{q-1}-1}^{k-q+1} j_1 j_2 \cdots j_q = \sum_{j_q=1}^{k-q+1} \sum_{j_{q-1}=1}^{j_q+1} \cdots \sum_{j_1=1}^{j_2+1} j_1 j_2 \cdots j_q$$

Mit Hilfe der umindizierten Summen kann nun die eigentliche Behauptung bewiesen werden. Zu zeigen bleibt noch, dass

$$\sum_{j_q=1}^{k-q+1} \sum_{j_{q-1}=1}^{j_q+1} \cdots \sum_{j_1=1}^{j_2+1} j_1 j_2 \cdots j_q \cdot \delta(j_q) = \begin{cases} 0 & \text{für} \quad k < q \\ \frac{(k+q)!}{2^q (k-q)! q!} & \text{für} \quad k \geq q \end{cases}$$

Im ersten Fall sei $k < q$. Damit hat die q-fach-Summe den Wert 0, da die Menge der Laufindizes $j_q = 1, \ldots, k-q+1$ leer ist. Der Beweis für den Fall $k \geq q$ wird mit Hilfe einer vollständigen Induktion über die Anzahl der Summen q geführt. Der Induktionsanfang ist $q = 1$:

$$\sum_{j_1=1}^{k} j_1 = \frac{k(k+1)}{2}$$

Die Induktionsvoraussetzung lautet daher:

$$\sum_{j_q=1}^{k-q+1} \cdots \sum_{j_2=1}^{j_3+1} \sum_{j_1=1}^{j_2+1} j_1 j_2 \cdots j_q = \frac{(k+q)!}{2^q (k-q)! q!} \tag{3.8}$$

Der Induktionsschluss geht von q zu $q+1$. Im zweiten Schritt wird die Induktionsvoraussetzung 3.8 eingesetzt. Lemma 3.1 ergibt die letzte Umformung:

$$\begin{aligned} \sum_{j_{q+1}=1}^{k-q} \sum_{j_q=1}^{j_{q+1}+1} \cdots \sum_{j_2=1}^{j_3+1} \sum_{j_1=1}^{j_2+1} j_1 j_2 \cdots j_{q+1} &= \sum_{j_{q+1}=1}^{k-q} j_{q+1} \Big(\sum_{j_q=1}^{(j_{q+1}+q)-q+1} \cdots \sum_{j_2=1}^{j_3+1} \sum_{j_1=1}^{j_2+1} j_1 j_2 \cdots j_q \Big) \\ &\overset{3.8}{=} \sum_{j_{q+1}=1}^{k-q} j_{q+1} \Big(\frac{((j_{q+1}+q)+q)!}{2^q ((j_{q+1}+q)-q)! q!} \Big) \\ &= \frac{1}{2^q q!} \sum_{j_{q+1}=1}^{k-q} \frac{(j_{q+1}+2q)!}{(j_{q+1}-1)!} \end{aligned}$$

$$\overset{\text{Lemma 3.1}}{=} \frac{1}{2^q q!}\frac{(k+q+1)!}{2(q+1)!(k-q-1)!}$$
$$= \frac{(k+q+1)!}{2^{q+1}(k-q-1)!(q+1)!}$$

□

Mit Hilfe dieser beiden Lemmata kann nun eine direkte Formulierung für die Anzahl der Knoten im Statusvektorbaum für n Aufträge und $k^* = 0$ (also mit Wurzel $\Phi_{k^*} = (0,\ldots,0)^T$) angegeben werden:

Satz 3.2 (direkte Formulierung)
Die Anzahl der Knoten im Statusvektorbaum für n Aufträge und $k^* = 0$ beträgt

$$|V^S_{n,0}| = \sum_{l=0}^{n} \frac{n!}{(n-l)!}\left(1 + \sum_{q=1}^{l} \frac{(l+q)!}{2^q(l-q)!q!}\right)$$

Beweis: Die rekursive Formulierung aus Satz 3.1 für $k = 0$ ergibt bei Auflösung der Rekursion:

$$|V^S_{n,0}| \overset{\text{Satz 3.1}}{=} \sum_{j_1=0}^{n} \frac{n!}{(n-j_1)!}(1 + j_1|V^S_{n-1,j_1-1}|)$$
$$\overset{\text{Satz 3.1}}{=} \sum_{j_1=0}^{n} \frac{n!}{(n-j_1)!}\Big(1 + j_1\ (\sum_{j_2=0}^{n-j_1} \frac{(n-j_1)!}{(n-j_1-j_2)!}(1 + (j_1+j_2-1)\ |V^S_{n-2,j_1+j_2-2}|))\Big)$$
$$= \sum_{j_1=0}^{n} \frac{n!}{(n-j_1)!} + \sum_{j_1=0}^{n}\sum_{j_2=0}^{n-j_1} \frac{j_1\ n!}{(n-j_1-j_2)!}(1 + (j_1+j_2-1)\ |V^S_{n-2,j_1+j_2-2}|)$$
$$\overset{\text{Satz 3.1}}{=} \sum_{j_1=0}^{n} \frac{n!}{(n-j_1)!} + \sum_{j_1=0}^{n}\sum_{j_2=0}^{n-j_1} \frac{j_1\ n!}{(n-j_1-j_2)!}(1 +$$
$$+(j_1+j_2-1)(\sum_{j_3=0}^{n-j_1-j_2} \frac{(n-j_1-j_2)!}{(n-j_1-j_2-j_3)!}(1 +$$
$$+(j_1+j_2+j_3-2)\ |V^S_{n-3,j_1+j_2+j_3-3}|)))$$
$$= \sum_{j_1=0}^{n} \frac{n!}{(n-j_1)!} + \sum_{j_1=0}^{n}\sum_{j_2=0}^{n-j_1} \frac{j_1\ n!}{(n-j_1-j_2)!} +$$
$$+ \sum_{j_1=0}^{n}\sum_{j_2=0}^{n-j_1}\sum_{j_3=0}^{n-j_1-j_2} \frac{j_1(j_1+j_2-1)n!}{(n-j_1-j_2-j_3)!}(1 + (j_1+j_2+j_3-2)\ |V^S_{n-3,j_1+j_2+j_3-3}|) +$$
$$\vdots$$

$$
\begin{aligned}
= & \sum_{j_1=0}^{n} \frac{n!}{(n-j_1)!} + \sum_{j_1=0}^{n} \sum_{j_2=0}^{n-j_1} \frac{j_1\ n!}{(n-j_1-j_2)!} + \sum_{j_1=0}^{n} \sum_{j_2=0}^{n-j_1} \sum_{j_3=0}^{n-j_1-j_2} \frac{j_1(j_1+j_2-1)n!}{(n-j_1-j_2-j_3)!} + \\
& + \sum_{j_1=0}^{n} \sum_{j_2=0}^{n-j_1} \sum_{j_3=0}^{n-j_1-j_2} \sum_{j_4=0}^{n-j_1-j_2-j_3} \frac{j_1(j_1+j_2-1)(j_1+j_2-j_3-2)n!}{(n-j_1-j_2-j_3-j_4)!} + \\
& \quad \vdots \\
& + \sum_{j_1=0}^{n} \cdots \sum_{j_n=0}^{n-\sum_{q=0}^{n-1} j_q} \frac{j_1(j_1+j_2-1)\cdots(j_1+\ldots+j_{n-1}-n+2)\ n!}{(n-j_1-j_2-\cdots-j_n)!} \big(1 + \\
& + (j_1+\cdots+j_n-n+1)\ \mathrm{anz}(0, j_1+\cdots+j_n-n)\big) \\
= & \sum_{j_1=0}^{n} \frac{n!}{(n-j_1)!} + \sum_{j_1=0}^{n} \sum_{j_2=0}^{n-j_1} \frac{j_1\ n!}{(n-j_1-j_2)!} + \sum_{j_1=0}^{n} \sum_{j_2=0}^{n-j_1} \sum_{j_3=0}^{n-j_1-j_2} \frac{j_1(j_1+j_2-1)n!}{(n-j_1-j_2-j_3)!} + \\
& + \sum_{j_1=0}^{n} \sum_{j_2=0}^{n-j_1} \sum_{j_3=0}^{n-j_1-j_2} \sum_{j_4=0}^{n-j_1-j_2-j_3} \frac{j_1(j_1+j_2-1)(j_1+j_2-j_3-2)n!}{(n-j_1-j_2-j_3-j_4)!} \\
& \quad \vdots \\
& + \sum_{j_1=0}^{n} \cdots \sum_{j_n=0}^{n-\sum_{q=0}^{n-1} j_q} \frac{j_1(j_1+j_2-1)\cdot\ldots\cdot(j_1+\ldots+j_{n-1}-n+2)\ n!}{(n-j_1-j_2-\cdots-j_n)!} \\
& + \sum_{j_1=0}^{n} \cdots \sum_{j_n=0}^{n-\sum_{q=0}^{n-1} j_q} \frac{j_1(j_1+j_2-1)\cdot\ldots\cdot(j_1+\ldots+j_n-n+1)\ n!}{(n-j_1-j_2-\cdots-j_n)!}
\end{aligned}
$$

Beim Auflösen der Klammern kann jeweils der Nenner des ersten Bruchs mit dem Zähler des zweiten gekürzt werden. Wegen $\mathrm{anz}(0,0) = 1$ bricht die Rekursion in der n. Summe ab.

Die Summanden der mehrfach-Summen werden aus reziproken Fakultäten gebildet, die jeweils mit mindestens einem Vorfaktor multipliziert werden. Schreibt man eine Mehrfachsumme aus, kann man ihre Summanden entsprechend ihrer Nenner von 0! bis $n!$ sortieren und die Summanden mit gleichem Nenner zusammenfassen. Dazu wird eine neue Indizierung $j_1^*, \ldots, j_n^*$ eingeführt und folgendermaßen vorgegangen:

Da der Laufindex j_q einer Summe von der Summe der Laufindizes der vorhergehenden Summen abhängt, haben die Summanden eine spezielle Struktur. Der Nenner $n!$ kann nur erreicht werden, indem für alle Laufindizes $j_q = 0$ gilt. Der Nenner $(n-1)!$ wird erreicht, wenn genau ein Laufindex 1 ist. Für den Zähler bedeutet dies, dass der erste Faktor $j_1^* := j_1$ nur 0 oder 1 sein kann, während für den zweiten Faktor $j_2^* := (j_1 + j_2 - 1)$ nur die Werte -1 oder 0 möglich sind. Für den Nenner $(n-l)!$ mit $0 \leq l \leq n$ gilt analog, dass die Summe der Indizes genau l ist. Daraus folgt für den Zähler, dass der erste Faktor j_1^* zwischen 0 und l liegt, während der zweite Faktor j_2^* zwischen $j_1^* - 1$ und $l - 1$ liegen muss, $j_3^* := (j_1 + j_2 + j_3 - 2)$

n	$V^S_{n,0}$
1	3
2	19
3	271
4	7 365
5	326 011
6	21 295 783
7	1 924 223 799
8	229 714 292 041
9	35 007 742 568 755
10	6 630 796 801 779 771

Tabelle 3.1: Anzahl Knoten in Statusvektorbäumen mit n noch nicht bearbeiteten Aufträgen.

entsprechend zwischen $j_2^* - 1$ und $l - 2$ usw. Schreibt man alle Summen entsprechend um und klammert $n!$ aus, so erhält man

$$
\begin{aligned}
|V^S_{n,0}| &= n! \left(\sum_{l=0}^{n} \frac{1}{(n-l)!} + \sum_{l=0}^{n} \frac{\sum_{j_1^*=1}^{l} j_1^*}{(n-l)!} + \sum_{l=0}^{n} \frac{(\sum_{j_1^*=1}^{l} \sum_{j_2^*=j_1^*-1}^{l-1} j_1^* j_2^*)}{(n-l)!} + \ldots + \right. \\
&\qquad \left. + \sum_{l=0}^{n} \frac{\sum_{j_1^*=1}^{l} \cdots \sum_{j_l^*=j_{l-1}^*-1}^{1} j_1^* \cdots j_{l-1}^*}{(n-l)!} \right) \\
&= \sum_{l=0}^{n} \frac{n!}{(n-l)!} \left(1 + \sum_{j_1^*=1}^{l} j_1^* + \sum_{j_1^*=1}^{l} \sum_{j_2^*=j_1^*-1}^{l-1} j_1^* j_2^* + \ldots + \sum_{j_1^*=1}^{l} \cdots \sum_{j_l^*=j_{l-1}^*-1}^{1} j_1^* \ldots j_{l-1}^* \right) \\
&\overset{\text{Lemma 3.2}}{=} \sum_{l=0}^{n} \frac{n!}{(n-l)!} \left(1 + \sum_{q=1}^{l} \frac{(l+q)!}{2^q (l-q)! q!} \right)
\end{aligned}
$$

□

Um sich eine Vorstellung vom schnellen Wachstum der Knotenmenge machen zu können, stellt Tabelle 3.2.2 die Anzahl der Knoten im Graphen mit bis zu 10 noch nicht bedienten Aufträgen exemplarisch dar. Auf eine explizite Darstellung der Anzahl Kanten wird hier verzichtet, da im Statusvektorbaum außer der Wurzel jeder Knoten genau eine eingehende Kante hat und die Anzahl der Kanten daher der Anzahl der Knoten weniger 1 entspricht.

3.2.3 Der A*-Algorithmus

Im Abschnitt 3.1 wurde gezeigt, dass die Problemstellung des Single Vehicle Routings, das Finden eines kürzesten Wegs durch alle noch zu bedienenden Orte, mit Hilfe von Statusvektoren zur Beschreibung des Fortschritts in der Auftragsbearbeitung mit den Mitteln der dynamischen Programmierung möglich ist. In den Abschnitten 3.2.1 und 3.2.2 wurde beschrieben, wie der von Caramia et al. benutzte Statusvektorbaum aufgebaut ist und wie groß er bei gegebener Anzahl Aufträge werden kann. Dieser Abschnitt behandelt nun die Suche nach einem kürzesten Weg von der Wurzel des Statusvektorbaums zu einem der Blätter und damit die Suche nach einer kürzesten Route für das Fahrzeug, wobei alle noch zu bedienenden Aufträge abgearbeitet und die Reihenfolgebedingungen eingehalten werden müssen.

Wegen der Größe des zu Grunde liegenden Statusvektorbaums ist Wert auf einen schnellen Algorithmus zu legen, wegen der hohen Bedeutung der Fahrzeiten soll jedoch auch eine möglichst gute Lösung gefunden werden. Da in der Problemstellung von Caramia et al. die Anzahl der noch zu bedienenden Aufträge pro Fahrzeug relativ gering bleibt[2], kann trotz der Komplexität des Problems eine optimale Lösung in kurzer Zeit berechnet werden. Dazu wird der A*-Algorithmus [41] benutzt.

Der A*-Algorithmus wurde 1968 von Hart, Nilsson und Raphael veröffentlicht. Ihr Ziel bestand darin, während der Berechnung eines kürzesten Wegs mit einem exakten Verfahren zusätzliche heuristische Informationen an das exakte Verfahren zu übergeben, um möglichst viele problemspezifische Informationen zur Verkürzung der Rechenzeit nutzen zu können. Der A*-Algorithmus wird im Folgenden beschrieben:

Gegeben ist ein beliebiger Graph $G = (V, E, c)$ mit einer Knotenmenge V, einer Kantenmenge E und einer Kostenfunktion $c : E \to \mathbb{R}^+$. Gesucht wird der kürzeste Weg von einem bestimmten Knoten $v_0 \in V$ zu einem *Zielknoten* v_s aus einer gegebenen Menge der Zielknoten $V_s \subset V$.

Als exaktes Verfahren zur Berechnung eines kürzesten Wegs im Graphen G wird der Dijkstra-Algorithmus benutzt. Der Dijkstra-Algorithmus berechnet iterativ, in einem Knoten v_0 beginnend, die Länge der kürzesten Wege von v_0 zu allen erreichbaren Knoten, indem immer die Mengen der Nachfolger eines erreichbaren und noch nicht benutzten Knotens v mit kürzestem Weg von v_0 zur Menge der erreichbaren Knoten hinzugefügt und ihre Entfernung von v_0 über Knoten v als Länge des kürzesten bisher bekannten Weg gesetzt werden, falls kein kürzerer Weg bekannt ist. Dazu wird für jeden bereits erreichten Knoten $v \in V$ die Länge des bisher bekannten kürzesten Wegs gespeichert und im Folgenden mit $\overleftarrow{C}(v)$ bezeichnet.

Die von Hart, Nilsson und Raphael hinzugefügte heuristische Information besteht aus einer Abschätzung der Länge des noch verbleibenden Wegs von einem Knoten $v \in V$ zu einem der Zielknoten $v_s \in V_s$. Diese Abschätzung wird mit der *Schätzfunktion* $\overrightarrow{C}(v, V_s)$ berechnet. Die Eigenschaften der Schätzfunktion sind für den Verlauf des Algorithmus von elementarer

[2] Caramia et al. beschränken die Anzahl der noch zu bearbeitenden Aufträge pro Fahrzeug auf acht [55]

Bedeutung. Sie gehen in die geschätzte Gesamtlänge des Weges von v_0 über den Knoten v zu einem der Zielknoten $v_s \in V_s$ ein:

$$C(v) = \overleftarrow{C}(v) + \overrightarrow{C}(v, V_s)$$

und können somit den Ablauf des Algorithmus grundlegend verändern.

Um die Funktionsweise des Algorithmus angeben zu können, werden zunächst zwei Begriffe eingeführt: Für die Knoten $v \in V$ des Graphen werden die Marken „*offen*“ und „*geschlossen*“ verwendet. Ein Knoten v wird als „offen“ markiert, wenn bereits ein Weg vom Knoten v_0 zum Knoten v bekannt ist und somit die Länge der Wege zu seinen Nachfolgern über diesen Knoten v berechnet werden kann. Der Knoten v wird als „geschlossen“ markiert, wenn bereits die Länge der kürzesten Wege seiner Nachfolger über diesen Knoten v berechnet wurde. Damit kann nun die Funktionsweise des Algorithmus skizziert werden:[3]

1. Markiere den Startknoten v_0 als offen und berechne $C(v_0)$.
2. Wähle aus der Menge der offenen Knoten den Knoten v mit minimalem $C(v)$. Ist diese Wahl nicht eindeutig und sind Zielknoten v_s in dieser Menge enthalten, so wähle einen offenen Zielknoten v_s, ansonsten wähle zufällig einen offenen Knoten aus.
3. Falls der gewählte Knoten v ein Zielknoten $v_s \in V_s$ ist, beende den Algorithmus.
4. Sonst markiere v als geschlossen. Markiere alle Nachfolger v' von v, die nicht bereits als geschlossen markiert wurden, als offen. Berechne $C(v')$ für alle Nachfolger. Falls für einen bereits geschlossenen Knoten v' das neu berechnete $C(v')$ kleiner als die bisherigen Kosten dieses Knotens ist, markiere diesen Knoten wiederum als offen. Gehe zu Schritt 2.

Der A*-Algorithmus liefert ein optimales Ergebnis, sofern die Schätzfunktion $\overrightarrow{C}(v, V_s)$ zur Abschätzung der Länge des kürzesten Wegs vom Knoten v zu einem der Zielknoten $v_s \in V_s$ die tatsächliche Länge dieses Wegs nicht überschätzt ([41]). Hart, Nilsson und Raphael haben weiterhin gezeigt, dass der A*-Algorithmus auch in dem Sinne optimal ist, dass es keinen anderen Algorithmus geben kann, der bei gleicher Information weniger Schritte zum Auffinden des optimalen Wegs benötigt (abgesehen von Zufallsentscheidungen bei mehreren Knoten mit minimalen Kosten). Diese Eigenschaft wird im Folgenden das „*schnellstmögliche Finden einer Lösung*“ genannt. Voraussetzung hierfür ist allerdings, dass die Schätzfunktion $\overrightarrow{C}(v, V_s)$ die Dreiecksungleichung erfüllt (vgl. [41]), d. h. :

$$\overrightarrow{C}(v, \{v'\}) + \overrightarrow{C}(v', V_s) \geq \overrightarrow{C}(v, V_s) \quad \forall\, v, v' \in V$$

Sie sagt aus, dass die geschätzte Länge des kürzesten Wegs von einem Knoten v zu einem der Zielknoten $v_s \in Vs$ kleiner oder gleich der geschätzten Länge des kürzesten Wegs vom Knoten v über einen Knoten v' zu einem der Zielknoten $v'_s \in V_s$ ist.

[3] in Anlehnung an [41], Seite 102

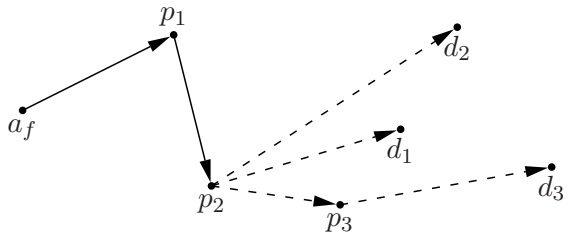

Abbildung 3.4: Beispiel zur Berechnung eines geschätzten Werts für die verbleibende Fahrzeit mit der Schätzfunktion „Maximalauftragszeit“

Ein großer Vorteil des A*-Algorithmus besteht darin, dass der Graph nicht explizit gegeben sein muss, sondern während des Verfahrens konstruiert werden kann. Das kann zu erheblichen Einsparungen im Speicher- und Rechenaufwand führen, wenn nicht alle Knoten des Graphen untersucht werden müssen. Begonnen wird dazu mit einem Graphen, der nur den Startknoten v_0 enthält. Im Schritt 4 des Algorithmus werden dann alle nachfolgenden Knoten des ausgewählten offenen Knotens v erzeugt und der Graph somit schrittweise aufgebaut. Das Erzeugen der nachfolgenden Knoten eines offenen Knotens v wird „*expandieren*“ genannt.

Caramia et al. haben den A*-Algorithmus gewählt, um im Statusvektorbaum einen kürzesten Weg von der Wurzel Φ_{k^*} zu einem der Blätter $\Phi = (2, \ldots, 2) \in {}_{2n}V^S_{n,k^*}$ zu berechnen. Die Kantengewichte im Statusvektorbaum entsprechen den Fahr-, Warte- und Servicezeiten zwischen den zu den Endknoten der Kanten gehörigen Orten. Die Laufzeit des A*-Algorithmus ist bei gegebenem Graphen $G^S_{n,k^*} = (V^S_{n,k^*}, E^S_{n,k^*}, c^S_{n,k^*})$ (vgl. Abschnitt3.2.1) in besonderem Maße von einer geeigneten Wahl der Schätzfunktion $\overrightarrow{C}$ abhängig. Sie darf die tatsächliche Fahrzeit für den kürzesten Weg von einem Knoten Φ zu einem der Blätter $\Phi' \in {}_{2n}V^S_{n,k^*}$ nicht überschätzen, sollte aber natürlich möglichst nah am tatsächlichen Wert liegen, ohne dabei viel Rechenaufwand zu verursachen.

Caramia et al. führen dazu eine einfache Schätzfunktion $\overrightarrow{C}^{\triangleright}$ ein: sie betrachten die Fahrzeit jedes im Statusvektor Φ noch zu bedienenden Auftrags einzeln und wählen die maximale Fahrzeit eines einzelnen Auftrags als Wert der Schätzfunktion. Diese Schätzfunktion $\overrightarrow{C}^{\triangleright}$ wird „*Maximalauftragszeit*“ genannt und im Folgenden näher beschrieben:

Die Maximalauftragszeit $\overrightarrow{C}^{\triangleright} : V^S_{n,k^*} \times V^* \to \mathbb{R}^+$ gibt die geschätzte minimale Fahrzeit von einem Knoten $\Phi \in V^S_{n,k^*}$ zu einem der Zielknoten $\Phi^* \in V^* \subset V^S_{n,k^*}$ zurück. Die Knoten Φ des Statusvektorbaums G^S_{n,k^*} enthalten bekanntlich die Status der noch zu bedienenden Aufträge: $\Phi = (\varphi(1), \ldots, \varphi(n))^T$. Zu jedem Auftrag $i \in \{1, \ldots, n\}$ des Fahrzeugs sind der Pickup-Ort p_i und der Delivery-Ort d_i gegeben, außerdem sind die Fahrzeiten $c_{\nu,\nu'}$ der kürzesten Wege zwischen je zwei zu besuchenden Orten ν, ν' bekannt. Die Position μ des Fahrzeugs kann berechnet werden, da der Vorgänger des Knotens bekannt ist und über die Veränderung des Statusvektors der bediente Auftrag berechnet werden kann. Die für einen einzelnen Auftrag i zum Erreichen eines Statusvektors $\Phi^* \in V^*$ verbleibende Fahrzeit wird berechnet als:

$$\overrightarrow{C_i}^{\triangleright}(\Phi,\Phi^*) = \begin{cases} 0 & \text{falls } \varphi(i) = \varphi^*(i) \\ c_{\mu,d_i} & \text{falls } \varphi(i) = 1 \wedge \varphi^*(i) = 2 \\ c_{\mu,p_i} & \text{falls } \varphi(i) = 0 \wedge \varphi^*(i) = 1 \\ c_{\mu,p_i} + c_{p_i,d_i} & \text{falls } \varphi(i) = 0 \wedge \varphi^*(i) = 2 \end{cases}$$

$\overrightarrow{C_i}^{\triangleright}(\Phi,\Phi^*)$ bildet die noch verbleibende Fahrzeit bei alleiniger Bedienung des Auftrags i ab: falls Auftrag i in beiden Statusvektoren den gleichen Status $\varphi(i) = \varphi^*(i)$ hat, wird keine Fahrzeit benötigt. Ändert sich der Status von $\varphi(i) = 0$ auf $\varphi^*(i) = 1$, wird die Fahrzeit von der Fahrzeugposition μ zum Pickup-Ort p_i zurückgegeben, bei einer Änderung des Status von $\varphi(i) = 1$ auf $\varphi^*(i) = 2$ wird die Fahrzeit von der Fahrzeugposition μ zum Delivery-Ort d_i benötigt. Für einen Auftrag i mit Status $\varphi(i) = 0$ und $\varphi^*(i) = 2$ werden die minimale Fahrzeit von μ zum Pickup-Ort p_i und die Fahrzeit von p_i zum Delivery-Ort d_i des Auftrags addiert. Da jeder Auftrag i bedient werden muss, kann mit

$$\max_{i=1}^{n} \overrightarrow{C_i}^{\triangleright}(\Phi,\Phi^*)$$

eine Schätzung für die Fahrzeit angegeben werden, die die tatsächlich benötigte Fahrzeit auf keinen Fall überschätzt. Damit wird die Maximalauftragszeit als geschätzte Fahrzeit zu einem der Zielknoten $\Phi^* \in V*$ als Minimum der geschätzten Fahrzeiten $\max_{i=1}^{n} \overrightarrow{C_i}^{\triangleright}(\Phi,\Phi^*)$ definiert als:

$$\overrightarrow{C}^{\triangleright}(\Phi,V^*) \quad = \quad \min_{\Phi^* \in V^*} \max_{i=1}^{n} \overrightarrow{C_i}^{\triangleright}(\Phi,\Phi^*)$$

Die Berechnung der Maximalauftragszeit wird in Abbildung 3.4 beispielhaft an einem geographischen Graphen G^{Orte} für ein Fahrzeug mit drei Aufträgen gezeigt, für dessen Routing gerade ein Zustandsvektor $(1,1,0)^T$ untersucht wird: Auftrag 1 und Auftrag 2 wurden bereits eingeladen, dargestellt durch die durchgehenden Linien. Zu Auftrag 1 müssen der Pickup-Ort p_1 und der Delivery-Ort d_1 besucht werden, für Auftrag 2 die Orte p_2 und d_2, usw. Die verbleibende Fahrzeit wird abgeschätzt, indem für jeden Auftrag die Fahrzeit für seine vollständige Abarbeitung, dargestellt durch die gestrichelten Linien, berechnet und dann die längste Fahrzeit als Ergebnis der Schätzfunktion zurückgegeben wird.

Für die Schätzfunktion $\overrightarrow{C}^{\triangleright}$ gilt:

Satz 3.3

Der A-Algorithmus findet mit der Schätzfunktion „Maximalauftragszeit" die optimale Lösung. Das schnellstmögliche Auffinden der Lösung ist nicht garantiert.*

Beweis: Die Optimalität der Lösung des A*-Algorithmus hängt davon ab, ob die Schätzfunktion die tatsächlich noch anfallende Fahrzeit überschätzen kann. Im A*-Algorithmus wird die Fahrzeit von einem Knoten Φ des Statusvektorbaums mit bekannter Fahrzeugposition

μ zu einem Zielknoten $\Phi_{\vec{2}} \in {}_{2n}V^S_{n,k^*}$ betrachtet. Für einen Statusvektor $\Phi_{\vec{2}} \in {}_{2n}V^S_{n,k^*}$ mit $\Phi_{\vec{2}} = (2,\ldots,2)^T$ kann $\overrightarrow{C_i}^{\triangleright}(\Phi, {}_{2n}V^S_{n,k^*})$ geschrieben werden als:

$$\overrightarrow{C_i}^{\triangleright}(\Phi, \Phi_{\vec{2}}) = \begin{cases} 0 & \text{falls } \varphi(i) = 2 \\ c_{\mu,d_i} & \text{falls } \varphi(i) = 1 \\ c_{\mu,p_i} + c_{p_i,d_i} & \text{falls } \varphi(i) = 0 \end{cases}$$

Da alle Statusvektoren $\Phi_{\vec{2}} \in {}_{2n}V^S_{n,k^*}$ identisch sind, kann auch $\overrightarrow{C}^{\triangleright}(\Phi, {}_{2n}V^S_{n,k^*})$ einfacher dargestellt werden:

$$\overrightarrow{C}^{\triangleright}(\Phi, {}_{2n}V^S_{n,k^*}) = \max_{i=1\ldots,n} \overrightarrow{C_i}^{\triangleright}(\Phi, \Phi_{\vec{2}})$$

Man betrachte nun für den Statusvektor Φ mit Fahrzeugposition μ einen gewichteten Untergraphen $G^{Orte}_{\Phi,\mu}$ des Graphen G^{Orte}, dem Graphen zur Darstellung der geographischen Verteilung der Orte. Die Knotenmenge $V^{Orte}_{\Phi,\mu}$ von $G^{Orte}_{\Phi,\mu}$ enthalte den Ort μ der aktuellen Fahrzeugposition sowie alle noch anzufahrenden Orte, die Kantenmenge $E^{Orte}_{\Phi,\mu}$ bilde die möglichen Nachfolge-Relationen der Orte in einer Route durch alle noch anzufahrenden Orte ab:

$$\begin{aligned} V^{Orte}_{\Phi,\mu} &= \{\mu\} \cup \{p_i \in P \mid \varphi(i) = 0\} \cup \{d_i \in D \mid \varphi(i) < 2\} \\ E^{Orte}_{\Phi,\mu} &= V^{Orte}_{\Phi,\mu} \times V^{Orte}_{\Phi,\mu} \setminus \{\mu\} \end{aligned}$$

Das Gewicht $c^{Orte}_{\Phi,\mu}(\nu,\nu')$ einer Kante $(\nu,\nu') \in V^{Orte}_{\Phi,\mu}$ entspreche der Fahrzeit $c_{\nu,\nu'}$ von Ort ν zu Ort ν'. Für den Beweis, dass die Maximalauftragszeit die verbleibende Fahrzeit nicht überschätzt, betrachtet man alle Kanten $(\nu',\nu)^* \in E^{Orte}_{\Phi,\mu}$ im Graphen $G^{Orte}_{\Phi,\mu}$, die in einer fahrzeitminimalen Route $(\mu,\nu_1,\nu_2,\ldots,\nu_l)$ von der Fahrzeugposition μ durch alle noch anzufahrenden Orte $\nu_1,\ldots,\nu_l \in V^{Orte}_{\Phi,\mu}$ benutzt werden. Die Fahrzeit der Kante $(\nu',\nu)^*$ zum Erreichen eines Ortes ν auf dieser Route sei mit c^*_ν bezeichnet. Diese Kanten verbinden offensichtlich alle noch anzufahrenden Orte $\nu',\nu \in V^{Orte}_{\Phi,\mu}$, wobei jeder Ort $\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}$ genau eine eingehende und maximal eine ausgehende Kante besitzt. Die Fahrzeit der Route beträgt somit $\sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} c^*_\nu$.

Die Maximalauftragszeit $\overrightarrow{C}^{\triangleright}(\Phi, {}_{2n}V^S_{n,k^*})$ ist das Maximum der $\overrightarrow{C_i}^{\triangleright}(\Phi, \Phi_{\vec{2}})$ über alle Aufträge i. Es sei i' ein Auftrag mit maximalem $\overrightarrow{C_i}^{\triangleright}(\Phi, \Phi_{\vec{2}})$ mit $\varphi(i') = 2$. Dann gilt für die Fahrzeiten unter Betrachtung der Teil-Routen $(\mu,\nu_1,\nu_2\ldots,p_i)$, $(p_i,\ldots,d_i)$ und $(d_i,\ldots,\nu_l)$:

$$\begin{aligned} \overrightarrow{C}^{\triangleright}(\Phi, {}_{2n}V^S_{n,k^*}) &= c_{\mu,p_{i'}} + c_{p_{i'},d_{i'}} \\ &\leq c^*_{\nu_1} + c^*_{\nu_2} + \cdots + c^*_{p_{i'}} + \cdots + c^*_{d_{i'}} \\ &\leq \sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} c^*_\nu \end{aligned}$$

da die Gültigkeit der Dreiecksungleichung für die Fahrzeiten $c_{\nu,\nu'}$ vorausgesetzt wurde (vgl. Abschnitt 2.3). Diese Eigenschaft ist analog für Aufträge i' mit maximalem $\overrightarrow{C_i}^{\triangleright}(\Phi, \Phi_{\vec{2}})$ und mit $\varphi(i') < 2$ nachzuweisen. Die verbleibende Fahrzeit kann daher von $\overrightarrow{C}^{\triangleright}$ nicht überschätzt werden.

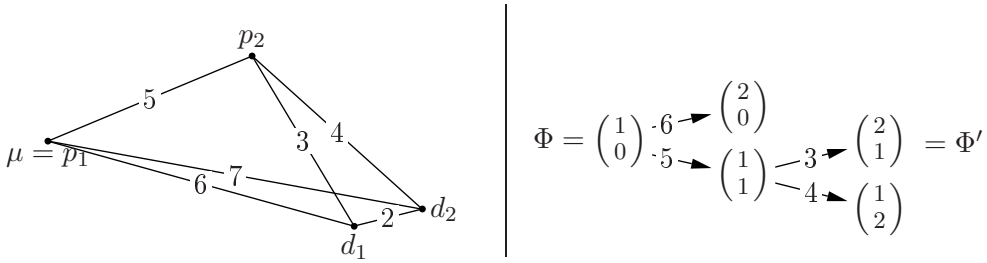

Abbildung 3.5: Gegenbeispiel für die Dreiecksungleichung.

Um das schnellstmögliche Auffinden der optimalen Lösung zu garantieren, muss nachgewiesen werden, dass die geschätzten Kosten $\overrightarrow{C}^{\triangleright}(\Phi, {}_{2n}V^S_{n,k^*})$ eines Knotens Φ die Dreiecksungleichung erfüllen, d. h. dass für jeden Statusvektor Φ und jeden beliebigen von Φ erreichbaren Statusvektor Φ' gilt [41]:

$$\overrightarrow{C}^{\triangleright}(\Phi, \{\Phi'\}) + \overrightarrow{C}^{\triangleright}(\Phi', {}_{2n}V^S_{n,k^*}) \geq \overrightarrow{C}^{\triangleright}(\Phi, {}_{2n}V^S_{n,k^*})$$

Anhand eines Gegenbeispiels wird gezeigt, dass die Maximalauftragszeit diese Dreiecksungleichung nicht erfüllt. Man betrachte dazu die dritte Fallunterscheidung in der Definition von $\overrightarrow{C}_i^{\triangleright}(\Phi, \{\Phi'\})$. Hier wird der Fall behandelt, dass ein Auftrag i auf dem Weg von Zustand Φ zu Zustand Φ' eingeladen wurde: $\varphi(i) = 0 \wedge \varphi'(i) = 1$. Da bei der Berechnung von $\overrightarrow{C}_i^{\triangleright}(\Phi')$ nun auf den neuen Standort μ' des Fahrzeugs zurückgreift, der bezüglich der Fahrzeit näher an d_i liegen kann als p_i, kann in diesem Fall die Dreiecksungleichung verletzt werden:

Man betrachte ein Fahrzeug, das zwei Aufträge bedienen soll und den ersten Auftrag bereits eingeladen hat. Diese Situation wird durch den Statusvektor $\Phi = (1, 0)^T$ beschrieben. Statusvektor $\Phi' = (2, 1)$ entstehe durch das Pickup des zweiten und das Delivery des ersten Auftrags. Abbildung 3.5 zeigt den Graphen $G^{Orte}_{\Phi,\mu}$ der geographischen Verteilung der noch anzufahrenden Pickup- und Delivery-Orte sowie den partiell erzeugten Statusvektorbaum mit den Knoten Φ und Φ'. Mit den in der Abbildung auf den Kanten gegebenen Fahrzeiten gilt, dass

$$\begin{aligned}
\overrightarrow{C}^{\triangleright}(\Phi, {}_4V^S_{2,0}) &= \max(6, 5+4) = 9 \\
\overrightarrow{C}^{\triangleright}(\Phi, \{\Phi'\}) &= \max(5, 6) = 6 \\
\overrightarrow{C}^{\triangleright}(\Phi', {}_4V^S_{2,0}) &= 2
\end{aligned}$$

was die Dreiecksungleichung nicht erfüllt: $6 + 2 \ngeq 9$. Somit ist nicht gewährleistet, dass der A*-Algorithmus mit der von Caramia et al. gewählten Funktion $\overrightarrow{C}^{\triangleright}$ zur Abschätzung der verbleibenden Kosten die optimale Lösung schnellstmöglich findet.

□

Nun ist bekannt, wie die Routenplanung im Algorithmus von Caramia et al. genau abläuft. Der A*-Algorithmus kann den Statusvektorbaum während der Laufzeit konstruieren und beinhaltet zusätzlich heuristische Komponenten zur Beschleunigung der Berechnung des kürzesten

Problem-	Eigenschaften		Aufträge		gelöste Instanzen	
gruppe	Aufträge	Instanzen	absolut	%	absolut	%
100	51,86	56	29,48	56,85%	25	44,64%
200	103,08	60	33,30	32,30%	10	16,67%
400	205,66	59	52,86	25,70%	7	11,86%
600	308,97	60	62,88	20,35%	3	5,00%
800	411,75	59	70,98	17,24%	0	0,00%
1000	514,98	58	78,91	15,32%	0	0,00%

Tabelle 3.2: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung des Statusvektorbaums

Wegs, so dass nicht unbedingt der ganze Statusvektorbaum erzeugt werden muss. Um Testläufe zur Analyse der tatsächlichen Rechenoperationen und -zeiten durchführen zu können, müssen zunächst Testdatensätze erzeugt werden, da keine Benchmarkprobleme zum dynamischen Pickup- and Delivery Vehicle Routing Problem mit Zeitfenstern bekannt sind.

3.2.4 Testläufe

Ziel der Testläufe ist es, Aussagen über das Laufzeitverhalten des Algorithmus treffen zu können. Dabei geht es zum Einen um die Güte des A*-Algorithmus, gemessen an der Anzahl der tatsächlich erzeugten Statusvektoren im Vergleich zur Größe des Statusvektorbaums. Zum Anderen werden die tatsächlich erreichten Laufzeiten für die Annahme bzw. Ablehnung der Aufträge aus den Testinstanzen untersucht.

Für die Testläufe wurde die Testumgebung ereignisdiskret eingestellt, so dass das Ergebnis nicht durch die Länge der Rechenzeiten beeinflusst wird. Außerdem wird der Testlauf nach maximal einer Million erzeugter Statusvektoren abgebrochen, damit alle Testinstanzen in Testläufe einbezogen werden und die Testläufe in angemessener Zeit berechnet werden können. Anhand der Anzahl gelöster Instanzen lässt sich erkennen, inwieweit der Algorithmus durch diese künstliche Grenze gestoppt wurde. Die Testinstanzen entsprechen dem dieser Arbeit zu Grunde liegenden Entscheidungsproblem, d. h. es wird ein Verfahren mit der grundlegenden Idee von Caramia et al. getestet, jedoch auf das neue Entscheidungsproblem angepasst: Wartezeiten sind demnach erlaubt, jeder Auftrag hat ein Pickup- und ein Delivery-Zeitfenster und die Anzahl der Aufträge pro Fahrzeug ist nicht beschränkt. Es werden die in Abschnitt 2.5 beschriebenen Testdatensätze verwendet.

In den Tabellen 3.2 bis 3.4 sind die Ergebnisse der Testläufe entsprechend der Anzahl der anzufahrenden Orte (vgl. Abschnitt 2.5) angegeben, die Problemklassen lc, lr und lrc werden jeweils zu einer Problemgruppe zusammengefasst.

[3]Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Problem-	♯ erzeugter Knoten			♯ Knoten	
gruppe	gesamt	gelöst	abgebrochen	unzulässig	expandiert
100	344.451,45	192.189,80	467.243,10	162.511,14	103.940,43
200	544.077,98	239.862,90	604.921,00	303.286,72	154.186,90
400	662.541,54	566.860,14	675.421,73	377.665,68	190.593,53
600	681.008,57	799.841,67	674.754,19	419.378,43	188.078,82
800	680.914,17	0	680.914,17	410.813,22	185.470,97
1000	754.078,74	0	754.078,74	451.380,81	209.935,43

Tabelle 3.3: Anzahl der im A*-Algorithmus erzeugten Knoten bei Verwendung des Statusvektorbaums

In Tabelle 3.2 werden zunächst die Eigenschaften der Problemgruppen vorgestellt und die Anzahl der durchschnittlich bis zur Grenze von einer Million Knoten sowie die Anzahl der gelösten Testinstanzen je Problemgruppe angegeben: Die Tabelle zeigt in der Spalte „Eigenschaften - Aufträge“ die Anzahl der Aufträge, die durchschnittlich in den Instanzen dieser Problemgruppe zu bearbeiten sind und in der Spalte „Eigenschaften - Instanzen“ die Anzahl der in der Problemgruppe enthaltenen Testinstanzen. Die Spalte „Aufträge“ enthält die absolute und die relative durchschnittliche Anzahl der Aufträge je Testinstanz, die bis zur künstlichen Rechenzeitbegrenzung von einer Million erzeugter Knoten tatsächlich abgearbeitet wurden. In den Spalten „gelöste Instanzen“ ist sowohl absolut als auch relativ zur Menge der Testinstanzen angegeben, wie viele Testinstanzen gelöst wurden, d. h. in wie vielen Testinstanzen alle Aufträge bearbeitet wurden, ohne die Grenze von einer Million Knoten zu überschreiten. Es ist gut zu erkennen, dass die künstliche Rechenzeitbeschränkung sehr restriktiv wird, d. h. nur wenige Testinstanzen überhaupt mit einer Lösung terminieren, während viele Probleme schon nach wenigen Aufträgen abgebrochen werden. Erwartungsgemäß nimmt die Anzahl der vollständig abgearbeiteten Probleminstanzen mit steigender Größe der Testdatensätze ab, jedoch steigt die Anzahl der insgesamt bearbeiteten Aufträge mit der Größe der Testfälle.

Eine Erklärung für dieses Phänomen liefert die Tabelle 3.3, in der die Anzahl der Knoten, die durchschnittlich während einer Testinstanz erzeugt werden, dargestellt wird. Zur Analyse wird sowohl die Anzahl der erzeugten Knoten als auch die Anzahl der als unzulässig erkannten und daher wieder gelöschten Knoten und die Anzahl der expandierten Knoten angegeben. Ein Knoten heißt unzulässig, wenn die Kapazitäts- oder die Zeitfenster-Nebenbedingungen nicht eingehalten werden. Er heißt expandiert, wenn er während des Algorithmus ausgewählt und seine Nachfolger erzeugt wurden (vgl. Abschnitt 3.2.3). Für die erzeugten Knoten werden dabei zusätzlich zum Durchschnitt aller Testinstanzen der Problemgruppe auch der Durchschnitt der gelösten Testinstanzen und der Durchschnitt der wegen der Grenze von einer Million Knoten abgebrochenen Testinstanzen angegeben. Die Tabelle zeigt, dass die Anzahl der erzeugten Knoten mit der Anzahl Aufträge der Instanzen einer Problemgruppe steigt. Das lässt sich dadurch erklären, dass in den Problemgruppen mit wenigen Aufträgen die Grenze von einer

Problem-gruppe	Antwortzeit in Sek.		
	global max.	Ø max.	Ø
100	2.746,36	218,55	7,63
200	3.155,86	301,14	9,69
400	2.797,82	322,04	6,53
600	2.604,75	195,12	3,41
800	1.101,46	136,80	2,25
1000	3.456,06	261,61	3,53

Tabelle 3.4: Antwortzeiten bei Verwendung des Statusvektorbaums

Million Knoten bei weitem nicht erreicht wurde, wenn die Instanzen gelöst wurden, wie in der Spalte „erzeugte Knoten - gelöst“ zu erkennen ist. Aber auch bei den Testinstanzen, deren Berechnung unterbrochen wurde, steigt die Anzahl der erzeugten Knoten, bleibt dabei aber weit unterhalb der Grenze von einer Million. Dies liegt daran, dass die Berechnungen zwar nach einer Million Knoten abgebrochen wurde, die Anzahl der Knoten zur besseren Vergleichbarkeit aber angibt, wie viele Knoten bis zum letzten abgearbeiteten Auftrag erzeugt wurden. Dass die Anzahl der erzeugten Knoten auch in den nicht komplett gelösten Testinstanzen der Problemgruppen mit vielen Aufträgen steigen, mag daran liegen, dass die Anzahl der erzeugten Knoten vor dem Abbruch aufgrund der höheren Komplexität auch schon höher lag.

Anhand der Anzahl der erzeugten Knoten lässt sich nun erklären, warum auch die durchschnittliche Anzahl der bearbeiteten Aufträge mit zunehmender Komplexität der Problemgruppen steigt: In den Problemgruppen mit wenigen Aufträgen wurden mehr Probleminstanzen gelöst, wobei die Knotenrestriktion nicht annähernd erreicht war. In den höheren Problemgruppen können an dieser Stelle mehr Aufträge abgearbeitet werden, bis eine Million Knoten erreicht werden und der Algorithmus damit abbricht.

Die Zahl der unzulässigen Knoten beträgt zwischen 47% in der Problemgruppe „100“ und 62% in der Problemgruppe „600“. Dies lässt darauf schließen, dass die Zeitfenster- und Kapazitätsrestriktionen sehr restriktiv wirken, d. h. viele Routen diese Restriktionen nicht erfüllen. Die Zahl der expandierten Knoten gibt Auskunft darüber, wie viele der erzeugten und nicht wegen Unzulässigkeit wieder gelöschten Knoten für den Algorithmus zur Expansion ausgewählt wurden. Der A*-Algorithmus versucht mit Hilfe der Schätzfunktion, die Anzahl der expandierten Knoten zu senken. Die Tabelle zeigt, dass zwischen 57% (in der Problemgruppe „100“) und fast 72% (in der Problemgruppe „600“) der erzeugten, nicht unzulässigen Knoten expandiert wurden.

Zur Analyse der Laufzeiten werden in Tabelle 3.4 die absolut maximalen, der Durchschnitt der maximalen und die durchschnittlichen Antwortzeiten der Problemgruppe angegeben. Unter *Antwortzeit* wird dabei die Zeit verstanden, die von der Anrufzeit des Auftrags bis zur Antwort, ob der Auftrag angenommen oder angelehnt wird, verstreicht. Die absolut maximale Antwortzeit ist das Maximum aller Antwortzeiten, die während aller Testläufe der Problem-

Aufträge	♯ Aufrufe	Ø ♯ Knoten absolut	Δ	Ø max. ♯ Knoten absolut	Δ	Ø min. ♯ Knoten absolut	Δ
1	333.852	2,99	-0,31%	3,00	0,00%	2,76	-8,10%
2	164.997	2,94	-68,16%	7,64	-16,66%	4,00	-56,12%
3	202.642	9,05	-89,81%	40,40	-54,96%	19,28	-78,50%
4	151.241	30,13	-98,06%	310,30	-82,40%	183,03	-89,86%
5	60.067	156,86	-99,66%	3.760,22	-93,92%	2.620,88	-95,91%
6	16.964	2.734,65	-99,91%	73.593,60	-97,87%	67.732,78	-98,08%
7	4.142	16.886,35	-99,99%	151.001,04	-99,94%	135.485,91	-99,95%
8	773	32.661,33	$<$ -99,99%	126.779,36	$<$ -99,99%	116.559,89	$<$ -99,99%
9	104	45.312,14	$<$ -99,99%	99.283,77	$<$ -99,99%	92.494,41	$<$ -99,99%
10	22	34.815,45	$<$ -99,99%	59.259,82	$<$ -99,99%	46.340,76	$<$ -99,99%

Tabelle 3.5: Vergleich der Anzahl erzeugter Knoten im A*-Algorithmus mit der Anzahl Knoten des Statusvektorbaums für $n = 1, \ldots, 10$ Aufträge.

gruppe erreicht wurden, während die durchschnittlich maximale Antwortzeit den Durchschnitt der maximalen Antwortzeiten der Testinstanzen dieser Problemgruppe darstellt.

Die durchschnittlichen Rechenzeiten in Tabelle 3.4 sind zwar sehr niedrig, die maximalen Rechenzeiten zeigen jedoch, dass erhebliche Rechenzeiten zu erwarten sind: diese maximalen Rechenzeiten sind immerhin vor dem Abbruch wegen Überschreitung der Knotenrestriktion bereits erreicht worden.

Zu untersuchen bleibt jedoch noch, inwiefern die Anzahl der Knoten davon abhängt, dass der Statusvektorbaum während der Laufzeit des A*-Algorithmus evtl. nur teilweise erzeugt wird. Zusätzlich haben natürlich auch die Zeitfenster und die Kapazitätsrestriktion einen erheblichen Anteil an einer Reduktion des Graphen, da ihretwegen viele Knoten unzulässig sind und daher ganze Äste des Statusvektorbaums nicht erzeugt werden müssen. Zur Analyse dieser Auswirkungen wird nicht nur die Anzahl der insgesamt erzeugten Knoten, sondern die detaillierte Anzahl der Knoten bei jeder Lösung des Single Vehicle Routing benötigt, da jedes Single Vehicle Routing auf einem anderen Statusvektorgraphen arbeitet. Tabelle 3.5 zeigt die durchschnittliche Anzahl der erzeugten Knoten über alle Testläufe insgesamt, sortiert nach der Anzahl der noch zu bedienenden Aufträge n. Die Anzahl der Knoten im Statusvektorbaum hängt auch von der Stufe k der Quelle des Statusvektorbaums ab, also von der Anzahl der zu Beginn des Single Vehicle Routings bereits im Fahrzeug geladenen Aufträge. Zur besseren Übersicht wurden die Ergebnisse hier für alle Stufen des Statusvektorbaums mit n Aufträgen aggregiert. In der ersten Spalte wird die Anzahl der zu Beginn des Single Vehicle Routings noch zu bedienenden Aufträge n angegeben, während die zweite Spalte angibt, wie oft das Single Vehicle Routing in allen Testläufen zusammen auf einem Statusvektorbaum für diese Anzahl Aufträge gelöst wurde. In den folgenden Spalten wird die Anzahl der erzeugten Knoten dargestellt, unterteilt in durchschnittlich, maximal und minimal erzeugte Knoten. Bei den maximal

und minimal erzeugten Knoten handelt es sich um den Durchschnitt aller Testinstanzen, bei der durchschnittlichen Anzahl erzeugter Knoten um den Durchschnitt aller Aufrufe des Single Vehicle Routings auf einem Statusvektorbaum mit n Aufträgen. Die Anzahl der Knoten ist jeweils absolut und in Prozent der Abweichung von der berechneten Größe dieses Statusvektorbaums angegeben, gerundet auf zwei Nachkommastellen. Beide Größen werden als gewichtetes Mittel der Ergebnisse für die Statusvektorgraphen der Stufe k angegeben, wobei das Gewicht der relativen Häufigkeit der Aufrufe des Single Vehicle Routings auf Stufe k entspricht.

Die Tabelle zeigt, dass der A*-Algorithmus und die Zeitfenster- und Kapazitätsrestriktionen einen erheblichen Einfluss auf die Anzahl der erzeugten Knoten haben. Die durchschnittliche Anzahl der erzeugten Knoten liegt bedeutend unter der Anzahl Knoten im Statusvektorbaum. Erwartungsgemäß steigen die Einsparungsmöglichkeiten mit zunehmender Anzahl der Aufträge, da bei größeren Bäumen und entsprechend guter Schätzfunktion im A*-Algorithmus entsprechend größere Äste nicht betrachtet werden müssen. Auch die Zeitfenster wirken umso restriktiver, je mehr Aufträge mit zeitlich nah beieinander liegenden Zeitfenstern in einem Fahrzeug zusammengefasst werden, da die Anzahl der möglichen Kombinationen der Pickup- und Delivery-Orte mit jedem neuen Auftrag weiter eingeschränkt werden. Auffällig ist jedoch, dass die durchschnittliche minimale Anzahl Knoten pro Testinstanz über der durchschnittlichen Anzahl der erzeugten Knoten liegt. Dieses Phänomen liegt in der unterschiedlichen Durchschnittsbildung begründet: die durchschnittliche Anzahl der erzeugten Knoten wird als Durchschnitt über alle Aufrufe des Single Vehicle Routing berechnet, während die durchschnittliche minimale Anzahl als Durchschnitt über alle Testinstanzen gebildet wird. Die größte Abweichung findet sich bei den Statusvektorbäumen mit 6 Aufträgen: die durchschnittliche minimale Anzahl Knoten liegt hier um das 24-fache über der Anzahl der durchschnittlich pro Aufruf erzeugten Knoten, weil es eine Testinstanz gibt, die das Single Vehicle Routing auf einem Statusvektorbaum mit 6 Aufträgen zwar nur einmal löst, dabei jedoch 983.958 Knoten erzeugt. In die Durchschnittsbildung über alle Aufrufe gehen diese Werte nur mit Faktor 1/16.694 ein, was einen ungleich kleineren Beitrag ergibt als in der Durchschnittsbildung über die 444 Testinstanzen, in denen ein Single Vehicle Routing auf einem Statusvektorbaum für 6 Aufträge und Quelle auf Stufe $0 \leq k \leq 6$ aufgerufen wurde.
Die Einträge „$< -99,99\%$“, bedeuten, dass sich bei einer Rundung auf zwei Nachkommastellen eine Reduktion um 100% ergeben würde.

3.2.5 Fazit

Caramia et al. haben mit der Idee, das Single Vehicle Routing mit Hilfe der Statusvektoren im Rahmen der dynamischen Programmierung zu lösen, einen entscheidenden Schritt zu einer zeitlich schnellen, optimalen Lösung des Single Vehicle Routings gemacht. Zusätzlich haben sie mit der Wahl des A*-Algorithmus, der einen exakten Algorithmus mit heuristischen Informationen zum schnelleren Auffinden der optimalen Lösung verbindet, einen effizienten

Algorithmus zur Lösung herangezogen, der während des Verfahrens den Graphen erzeugen kann. Durch die mögliche Beschränkung der Anzahl Kunden pro Fahrzeug war damit für sie eine zufriedenstellende Lösung gefunden.
Um jedoch diese Vorgehensweise allgemein auf Vehicle Routing Probleme anwenden zu können, muss diese Beschränkung der Aufträge, die pro Fahrzeug zugelassen werden, fallen gelassen werden. Damit erhöhen sich allerdings auch die Rechenzeiten dramatisch, wie in den Testläufen gezeigt wurde. Daher muss nach Möglichkeiten gesucht werden, wie diese Performance-Nachteile überwunden werden können, um die dargestellte Verfahrensidee trotzdem nutzen zu können. Dazu bieten sich vor Allem zwei Möglichkeiten an:

- Der Graph kann erheblich kleiner werden, wenn nicht der Statusvektorbaum genutzt wird, sondern eine andere Repräsentation aller möglichen Kombinationen gewählt würde, wie sie im dynamischen Modell zum Single Vehicle Routing in Abschnitt 3.1 benutzt wurde.

- Die Schätzfunktion des A*-Algorithmus kann noch spezieller auf das Problem zugeschnitten werden, so dass eventuell weniger Schritte bis zum Finden einer optimalen Lösung nötig sind.

Aufbauend auf die guten Ansätze kann das Single Vehicle Routing also im Hinblick auf das Entscheidungsproblem des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern wahrscheinlich noch deutlich verbessert werden. Die folgenden Abschnitte beschäftigen sich mit den vorgenommenen Änderungen und deren Ergebnissen. Zunächst wird ein anderer Graph zur Repräsentation aller möglichen Routen untersucht: der Zustandsgraph.

3.3 Beschleunigung durch Verwendung des Zustandsgraphen

Caramia et al. setzen zur Lösung des Teilproblems „Single Vehicle Routing“ Statusvektoren ein, um dieses Teilproblem mit den Methoden der dynamischen Programmierung bearbeiten zu können (vgl. Abschnitt 3.2). Sie benutzen Statusvektorbäume zur Repräsentation aller möglichen Routen und nehmen dabei in Kauf, dass identische Systemzustände mehrfach als Knoten vorkommen und damit insbesondere bei Entscheidungsproblemen mit relativ vielen Aufträgen Redundanzen erzeugen, die zu einer erheblichen Steigerung der Rechenzeiten führen können. In diesem Kapitel wird daher erläutert, wie der Graph effizienter aufgebaut und implementiert werden kann. Der so entstehende Graph wird im Folgenden *Zustandsgraph* (Z) genannt, er wird in Abschnitt 3.3.1 definiert. Zur Analyse, ob und wie viel Rechenzeit mit Hilfe dieser veränderten Graphenstruktur eingespart werden kann, folgt in Abschnitt 3.3.2 eine Untersuchung der Anzahl Knoten und Kanten im Zustandsgraphen sowie ein Vergleich zum ursprünglichen Statusvektorbaum. Im Anschluss werden in Abschnitt 3.3.3 die Ergebnisse der Testläufe präsentiert und ein Fazit gezogen.

3.3.1 Der Zustandsgraph

In diesem Abschnitt wird die Entstehung des Zustandsgraphen G^Z_{n,k^*} aus dem Statusvektorbaum G^S_{n,k^*} erläutert und eine detaillierte Beschreibung des Zustandsgraphen gegeben.

Die Entwicklung des Zustandsgraphen beruht auf der Beobachtung, dass im Statusvektorbaum wegen der Baumstruktur viele Knoten mit identischem Statusvektor Φ enthalten sind. Der Statusvektor dient bekanntlich der Charakterisierung des Systemzustands und besteht aus den Status $\varphi(i)$ aller dem Fahrzeug zugeordneten und zum Startzeitpunkt τ noch zu bedienenden Aufträge $i = 1, \ldots, n$: $\Phi = (\varphi(1), \varphi(2), \ldots, \varphi(n))^T$. Er enthält jedoch nur Informationen über den Fortschritt der Bedienung der Aufträge, und reicht daher allein nicht zur eindeutigen Zustandsbeschreibung, da keine Informationen über die Reihenfolge der Bedienung oder die Fahrzeugposition gegeben werden. Die entstehenden Fahrzeiten hängen jedoch essentiell von der Fahrzeugposition ab, so dass diese zusätzlich bekannt sein muss. Wegen der Baumstruktur kann im Statusvektorbaum auf die Angabe der Fahrzeugposition verzichtet werden, da sie über den eindeutig bestimmten Vorgänger-Knoten berechnet werden kann.

Im Zustandsgraphen hingegen soll jeder Knoten nur genau einmal in der Knotenmenge enthalten sein, um die Anzahl der Knoten und Kanten und somit die Rechenzeiten für das Bestimmen eines kürzesten Wegs im Graphen möglichst gering zu halten. Daher wird auf die Baumstruktur verzichtet und stattdessen die Fahrzeugposition als zusätzliches Merkmal des Systemzustands eingeführt. Im Folgenden wird daher der *Zustandsvektor* $(\Phi, \mu)^T$ zur Charakterisierung des Systemzustands benutzt, so dass ein kleinerer Graph zur Repräsentation aller möglichen Routen ausreicht – der Zustandsgraph $G^Z_{n,k^*} = (V^Z_{n,k^*}, E^Z_{n,k^*}, c^Z_{n,k^*})$. Er kann aus dem Statusvektorbaum erzeugt werden, indem

- jeder Knoten einen Zustandsvektor zur Beschreibung des Systemzustands erhält, indem zusätzlich zum Statusvektor Φ die Fahrzeugposition μ angegeben wird
- identische Knoten (mit gleichem Statusvektor Φ und gleicher Fahrzeugposition μ) zu einem Knoten zusammengezogen werden, wobei alle ein- und ausgehenden Kanten erhalten bleiben
- mehrfache Kanten zwischen zwei Knoten zu einer Kante zusammengefasst werden.

Damit erhält man den Zustandsgraphen, der wie folgt beschrieben werden kann:

Wie auch im Statusvektorbaum enthält der Statusvektor $\Phi_{k^*} = (\varphi_{k^*}(1), \ldots, \varphi_{k^*}(n))^T$ die zum Zeitpunkt τ beobachteten Status $\varphi_{k^*}(i) \in \{0, 1\}$ aller zu diesem Zeitpunkt noch zu bedienenden Aufträge $i = 1, \ldots, n$. Zusammen mit der bekannten Fahrzeugposition μ_{k^*} bildet er den Zustandsvektor $(\Phi_{k^*}, \mu_{k^*})^T$, der den Systemzustand zum Startzeitpunkt τ darstellt und die Quelle des Zustandsgraphen bildet. Die Knotenmenge V^Z_{n,k^*} des Zustandsgraphen kann direkt angegeben werden: sie beinhaltet alle möglichen Zustandsvektoren $(\Phi, \mu)^T$, wobei im Statusvektor Φ kein Status kleiner als in der Quelle Φ_{k^*} sein darf: $\varphi(i) \geq \varphi_{k^*}(i) \quad \forall\, i = 1, \ldots, n$, und die Fahrzeugposition $\mu \in P \cup D$ zum Statusvektor passen muss, d. h. für $\mu = d_i$

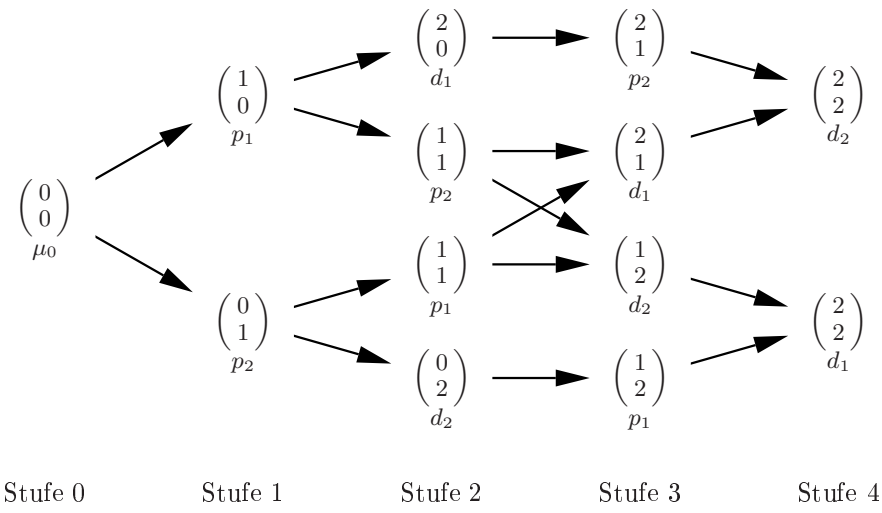

Abbildung 3.6: Der Zustandsgraph $G^Z_{2,0}$ für zwei nicht bearbeitete Aufträge.

muss der entsprechende Status $\varphi(i) = 2$ sein, $\mu = p_i$ ist nur für $\varphi(i) = 1$ erlaubt, falls außerdem im Zustandsvektor der Quelle $\varphi_{k^*}(i) = 0$ ist. Damit ergibt sich die Knotenmenge V^Z_{n,k^*} als:

$$V^Z_{n,k^*} = \{(\Phi_{k^*}, \mu_{k^*})\} \cup \big\{(\Phi,\mu)^T \in \{0,1,2\}^n \times (P \cup D) \mid \varphi(i) \geq \varphi_{k^*}(i), \mu \in \{p_i \in P \mid \varphi(i) = 1 > \varphi_{k^*}(i)\} \cup \{d_i \in D \mid \varphi(i) = 2\}\big\}$$

Die Kantenmenge E^Z_{n,k^*} kann ebenfalls direkt angegeben werden. Analog zum Statusvektorbaum beschreibt eine Kante den Übergang von einem Systemzustand zum nächsten. Im Statusvektorbaum konnte dieser Übergang durch die Addition eines Einheitsvektors $e_i \in \mathbb{R}^n$ zu einem Statusvektor Φ dargestellt werden. Im Zustandsgraphen wird der Übergang von einen Zustandsvektor $(\Phi,\mu)^T$ zu einem anderen Zustandsvektor $(\Phi',\mu')^T$ nun durch eine Addition eines Einheitsvektors $e_i \in \mathbb{R}^n$ zum Statusvektor Φ und der zugehörigen, vom Wert des neuen Status $\varphi'(i)$ bestimmten Änderung der Fahrzeugposition in den Pickup-Ort p_i oder den Delivery-Ort d_i dargestellt. Die Kantenmenge E^Z_{n,k^*} ergibt sich daher als

$$E^Z_{n,k^*} = \left\{ \left(\begin{pmatrix}\Phi\\ \mu\end{pmatrix}, \begin{pmatrix}\Phi'\\ \mu'\end{pmatrix} \right) \in V^Z_{n,k^*} \times V^Z_{n,k^*} \,\middle|\, \exists\, i \in \{1,\ldots,n\} : \Phi' = \Phi + e_i \wedge \mu' = \begin{cases} p_i, & \varphi'(i) = 1 \\ d_i, & \varphi'(i) = 2 \end{cases} \right\}$$

Die Kostenfunktion $c^Z_{n,k^*} : E^Z_{n,k^*} \to \mathbb{R}^+_0$ ordnet jeder Kante $((\Phi,\mu)^T, (\Phi',\mu')^T) \in E^Z_{n,k^*}$ die zugehörige Fahrzeit für den Übergang von der Fahrzeugposition μ zur Fahrzeugposition μ' zu: $c^Z_{n,k^*}((\Phi,\mu)^T, (\Phi',\mu')^T) = c_{\mu,\mu'}$.

Damit ist der Zustandsgraph $G^Z_{n,k^*} = (V^Z_{n,k^*}, E^Z_{n,k^*}, c^Z_{n,k^*})$ eindeutig definiert. Es handelt sich um einen gerichteten Graphen ohne gerichtete Kreise.

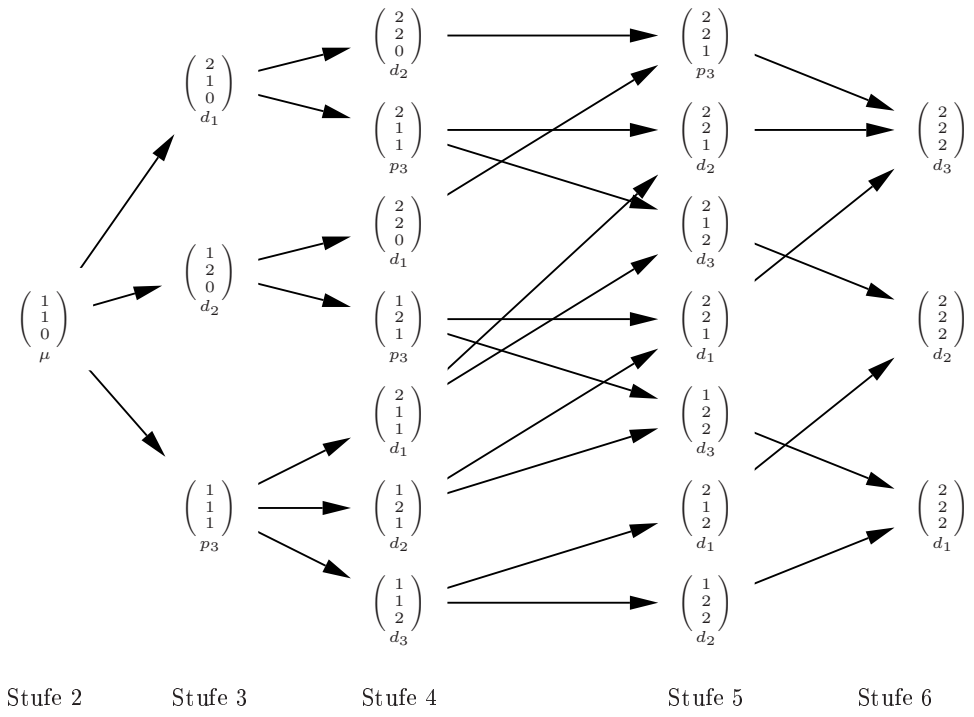

Abbildung 3.7: Der Zustandsgraph $G^Z_{3,2}$ für drei Aufträge, von denen zwei bereits abgeholt wurden.

Bemerkung:
Für den A*-Algorithmus gilt auch bei Verwendung des Zustandsgraphen G^Z_{n,k^*}, dass mit der Schätzfunktion $C^{\triangleright}$ die optimale Lösung gefunden wird, aber das schnellstmögliche Auffinden dieser Lösung nicht garantiert werden kann. Der Beweis verläuft analog zum Beweis von Satz 3.3.

Die Abbildungen 3.6 und 3.7 zeigen beispielhaft Zustandsgraphen für zwei noch nicht bediente Aufträge ($k^* = 0$) bzw. für drei Aufträge, von denen zwei bereits abgeholt wurden ($k^* = 2$). Zusätzlich zum Statusvektor sind die Positionen der Fahrzeuge angegeben: Auftrag $i \in \{1, \ldots, n\}$ wird von Pickup-Ort p_i zu seinem Delivery-Ort d_i gebracht, μ_{k^*} kennzeichnet die Startposition (vgl. dazu die Statusvektorbäume in Abbildungen 3.1 und 3.2).

Die Abbildungen lassen vermuten, dass sich die Anzahl der Knoten und Kanten gegenüber dem Statusvektorbaum deutlich verringert hat. Die Analyse des Zustandsgraphen im folgenden Abschnitt wird diesen Aspekt näher untersuchen.

3.3.2 Analyse des Zustandsgraphen

In diesem Abschnitt wird der Zustandsgraph analysiert, um die möglichen Einsparungen in der Rechenzeit im Vergleich zum Statusvektorbaum (vgl. Abschnitte 3.2.1 und 3.2.2) beziffern zu können. Dazu wird die Anzahl der Knoten und Kanten im Zustandsgraphen berechnet. Die Anzahl der Knoten repräsentiert die Anzahl der zu speichernden Strukturen, während die Anzahl der Kanten die Anzahl der Rechenoperationen widerspiegelt.

Im Gegensatz zur Analyse des Statusvektorbaums ist hier kein rekursives Vorgehen möglich, da der Zustandsgraph nicht wie der Statusvektorbaum in knotendisjunkte Untergraphen von jedem Knoten einer Stufe bis zu allen von diesem Knoten erreichbaren Senken unterteilt werden kann. Diese Eigenschaft kann leicht in Abbildung 3.7 gesehen werden: auf Stufe 3 wird der erste Eintrag „2“ erzeugt. Um eine rekursive Formulierung zu erlauben, müssten die Untergraphen, die den Knoten $((2,1,0)^T, d_1)^T$ und alle von ihm erreichbaren Knoten bzw. den Knoten $((1,2,0)^T, d_2)^T$ und alle von ihm erreichbaren Knoten enthalten, knotendisjunkt sein. Dies ist nicht der Fall, da auf Stufe 5 mehrfach vorkommende Zustandsvektoren zusammengefasst wurden und die Anzahl der Knoten auf dieser Stufe somit geringer ist als die Summe der Folgeknoten der Knoten $((2,1,0)^T, d_1)^T$ und $((1,2,0)^T, d_2)^T$.
Die Anzahl der Knoten und Kanten wird daher nicht rekursiv, sondern direkt angegeben:

Satz 3.4
Es bezeichne $|V^Z_{n,0}|$ die Anzahl der Knoten und $|E^Z_{n,0}|$ die Anzahl der Kanten im Zustandsgraphen $G^Z_{n,0} = (V^Z_{n,0}, E^Z_{n,0}, c^Z_{n,0})$ mit n Aufträgen mit Quelle auf Stufe 0. Dann gilt:

$$|V^Z_{n,0}| = 1 + \sum_{k=1}^{n} \sum_{l=0}^{k/2} \frac{n!\ (k-l)}{l!\ (k-2l)!\ (n-k+l)!} + \sum_{k=n+1}^{2n} \sum_{l=k-n}^{k/2} \frac{n!\ (k-l)}{l!\ (k-2l)!\ (n-k+l)!}$$

$$|E^Z_{n,0}| = n + \sum_{k=1}^{n} \sum_{l=0}^{k/2} \frac{n!\ (k-l)(n-l)}{l!\ (k-2l)!\ (n-k+l)!} + \sum_{k=n+1}^{2n-1} \sum_{l=k-n}^{k/2} \frac{n!\ (k-l)(n-l)}{l!\ (k-2l)!\ (n-k+l)!}$$

Beweis: Der Beweis erfolgt konstruktiv mit Hilfe kombinatorischer Überlegungen. Dazu werden für jede Stufe k des Graphen $G^Z_{n,0}$ zuerst die Anzahl der auf Stufe k auftretenden verschiedenen Statusvektoren $\Phi \in {}_kW_{n,0}$ sowie anschließend die sich daraus ergebende Anzahl der Zustandsvektoren $(\Phi, \mu)^T \in {}_kV^Z_{n,0}$ angegeben. Die Knotenzahl $|V^Z_{n,0}|$ des gesamten Graphen ergibt sich dann aus der Summe der möglichen Zustandsvektoren aller Stufen, die Kantenzahl $|E^Z_{n,0}|$ kann anhand der ausgehenden Kanten der Zustandsvektoren berechnet werden.

Die Menge der Statusvektoren auf Stufe k in einem Graphen mit Quelle auf Stufe 0 lässt sich schreiben als

$${}_kW_{n,0} = \{\Phi \in \{0,1,2\}^n \mid \sum_{i=1}^{n} \varphi(i) = k, \quad \forall\ i = 1, \ldots, n\}$$

Die Anzahl $|_kW_{n,0}|$ der auf Stufe k möglichen Statusvektoren entspricht der Anzahl der Permutationen einer n-elementigen geordneten Menge mit den drei Ausprägungen $0, 1$ und 2. Angenommen, man sucht Statusvektoren mit l_0 Einträgen 0, l_1 Einträgen 1 und l_2 Einträgen 2. Dann gibt es

$$\frac{n!}{l_2!\ l_1!\ l_0!}$$

verschiedene Statusvektoren mit diesen Ausprägungen. Für den Zustandsgraphen mit n Aufträgen gilt für die Anzahlen l_0, l_1 und l_2 der Einträge $0, 1, 2$ im Statusvektor Φ auf Stufe k:

- die Summe aller Einträge $\varphi(i)$ eines Statusvektors auf Stufe k ist gleich k, also
  $l_1 + 2l_2 = k \quad \Leftrightarrow \quad l_1 = k - 2l_2$
- ein Statusvektor Φ besteht aus genau n Elementen, d. h.
  $l_0 + l_1 + l_2 = n \quad \Leftrightarrow \quad l_0 = n - l_1 - l_2 = n - k + l_2$
- der Statusvektor Φ hat höchstens $\frac{k}{2}$ Einträge mit Wert 2, d. h.
  $l_2 \leq \frac{k}{2}$
- für $k > n$ müssen mindestens $k - n$ Einträge von Φ den Wert 2 haben, d.h.
  $l_2 \geq \max(k - n, 0)$

Die Gesamtzahl der unterschiedlichen Statusvektoren auf Stufe k ergibt sich daher als

$$|_kW_{n,0}| = \begin{cases} \displaystyle\sum_{l_2=0}^{k/2} \frac{n!}{l_2!\ (k-2l_2)!\ (n-k+l_2)!} & \text{für } k \leq n \\ \displaystyle\sum_{l_2=k-n}^{k/2} \frac{n!}{l_2!\ (k-2l_2)!\ (n-k+l_2)!} & \text{für } k > n \end{cases}$$

Nun muss die Anzahl der Zustandsvektoren auf Stufe k, $|_kV^Z_{n,0}|$ ermittelt werden, wobei ein Zustandsvektor $(\Phi, \mu)^T$ aus dem Statusvektor Φ und einer zugehörigen Fahrzeugposition μ besteht. Es stellt sich daher die Frage, wie viele Fahrzeugpositionen zu einem bestimmten Statusvektor Φ möglich sind. Prinzipiell kann sich das Fahrzeug in den Pickup-Orten der l_1 Aufträge mit Status 1 oder in den Delivery-Orten der l_2 Aufträge mit Status 2 befinden. Es gibt demnach $l_1 + l_2$ verschiedene mögliche Fahrzeugpositionen, die sich wegen $l_1 + 2l_2 = k$ ausdrücken lassen als $k - l_2$. Auf Stufe k gibt es daher

$$|_kV^Z_{n,0}| = \begin{cases} \displaystyle\sum_{l_2=0}^{k/2} \frac{n!\ (k-l_2)}{l_2!\ (k-2l_2)!\ (n-k+l_2)!} & \text{für } k \leq n \\ \displaystyle\sum_{l_2=k-n}^{k/2} \frac{n!\ (k-l_2)}{l_2!\ (k-2l_2)!\ (n-k+l_2)!} & \text{für } k > n \end{cases}$$

verschiedene Zustandsvektoren, bestehend aus einem Statusvektor Φ und einer zugehörigen Fahrzeugposition μ.

Der Zustandsgraph $G^Z_{n,0}$ besteht aus den Stufen 0 bis $2n$. Auf Stufe 0 gibt es genau eine Quelle, für die restlichen Stufen $k = 1, \ldots, 2n$ ist die Anzahl der verschiedenen Zustandsvektoren und somit die Anzahl der Knoten $|_kV^Z_{n,0}|$ bereits bekannt. Für den gesamten Graphen $G^Z_{n,0}$ ergeben sich somit

$$|V^Z_{n,0}| = 1 + \sum_{k=1}^{n} \sum_{l_2=0}^{k/2} \frac{n!\ (k-l_2)}{l_2!\ (k-2l_2)!\ (n-k+l_2)!} + \sum_{k=n+1}^{2n} \sum_{l_2=k-n}^{k/2} \frac{n!\ (k-l_2)}{l_2!\ (k-2l_2)!\ (n-k+l_2)!}$$

Knoten. Ersetzen von l_2 durch l liefert die erste Behauptung.

Für die Anzahl der Kanten von Stufe k zu Stufe $k+1$ gelten folgende Überlegungen: Man betrachte einen Zustandsvektor $(\Phi, \mu)^T$ der Stufe k mit $\Phi = (\varphi(1), \ldots, \varphi(n))^T$. Für jeden Auftrag i mit Status $\varphi(i) = 0$ oder $\varphi(i) = 1$ kann durch die Addition des Einheitsvektor e_i und der zugehörigen Änderung der Fahrzeugposition μ ein Zustandsvektor der Stufe $k+1$ erzeugt werden. Jeder dieser Übergänge wird durch eine Kante $((\Phi, \mu)^T, (\Phi', \mu')^T)$ repräsentiert. Von jedem Knoten $(\Phi, \mu)^T$ der Stufe k gehen daher $l_0 + l_1$ Kanten zu Knoten der Stufe $k+1$ des Graphen. Wegen $l_0 = n - l_1 - l_2$ und $l_1 = k - 2l_2$ gilt: $l_0 + l_1 = n - l_2$. Unter Zuhilfenahme der bereits bekannten Anzahl $|_kV^Z_{n,0}|$ der Knoten auf Stufe k erhält man somit für $k \geq 1$ die Anzahl $|_kE^Z_{n,0}|$ der Kanten von Stufe k zu Stufe $k+1$:

$$|_kE^Z_{n,0}| = \begin{cases} \displaystyle\sum_{l_2=0}^{k/2} \frac{n!\ (k-l_2)(n-l_2)}{l_2!\ (k-2l_2)!\ (n-k+l_2)!} & \text{für } k \leq n \\ \displaystyle\sum_{l_2=k-n}^{k/2} \frac{n!\ (k-l_2)(n-l_2)}{l_2!\ (k-2l_2)!\ (n-k+l_2)!} & \text{für } k > n \end{cases}$$

Von der Quelle auf Stufe 0 führen n Kanten zu Knoten der Stufe 1, da prinzipiell jeder der Aufträge zuerst bedient werden kann. Somit ergibt sich für die gesamte Anzahl $|E^Z_{n,0}|$ der Kanten im Graphen

$$|E^Z_{n,0}| = n + \sum_{k=1}^{n} \sum_{l_2=0}^{k/2} \frac{n!\ (k-l_2)(n-l_2)}{l_2!\ (k-2l_2)!\ (n-k+l_2)!} + \sum_{k=n+1}^{2n-1} \sum_{l_2=k-n}^{k/2} \frac{n!\ (k-l_2)(n-l_2)}{l_2!\ (k-2l_2)!\ (n-k+l_2)!}$$

Ersetzen von l_2 durch l liefert die zweite Behauptung.

□

In Tabelle 3.6 werden die Anzahl der Knoten und Kanten für die Zustandsgraphen mit $n = 1, \ldots, 10$ Aufträgen und Quelle auf Stufe 0 exemplarisch dargestellt und die relative Veränderung zur Anzahl Knoten und Kanten im entsprechenden Statusvektorbaum $G^S_{n,0}$ angegeben.

Um später die Auswirkungen des A*-Algorithmus untersuchen zu können, wird jedoch nicht nur die Anzahl der Knoten im Graphen $G^Z_{n,0}$ mit n bisher nicht bedienten Aufträgen

♯ Aufträge	Knoten		Kanten	
n	$\lvert V^{\boldsymbol{Z}}_{n,0}\rvert$	Δ	$\lvert E^{\boldsymbol{Z}}_{n,0}\rvert$	Δ
1	3	0,00%	2	0,00%
2	13	-31,58%	16	-11,11%
3	55	-79,70%	102	-62,22%
4	217	-97,05%	544	-92,61%
5	811	-99,75%	2.570	-99,21%
6	2.917	-99,99%	11.184	-99,95%
7	10.207	$<$ -99,99%	45.934	$<$ -99,99%
8	34.993	$<$ -99,99%	180.800	$<$ -99,99%
9	118.099	$<$ -99,99%	688.914	$<$ -99,99%
10	393.661	$<$ -99,99%	2.558.800	$<$ -99,99%

Tabelle 3.6: Die Anzahl der Knoten und Kanten in Zustandsgraphen mit Quelle auf Stufe 0, im Vergleich zur entsprechenden Anzahl Knoten und Kanten im Statusvektorbaum.

benötigt, da die Berechnungen in der Simulation häufig zu einem Zeitpunkt angestoßen werden, an dem ein Teil der Aufträge bereits im Fahrzeug geladen ist (vgl. Abbildung 3.7). Zum Startzeitpunkt besteht der Zustandsvektor $(\Phi, \mu)^T$ in diesem Fall aus einem Statusvektor $\Phi \in \{0,1\}^n$ und einer Fahrzeugposition, die Quelle des Zustandsgraphen liegt somit auf Stufe $k^* = \sum_{i=1}^{n} \varphi(i)$.
Daher müssen nun noch die Anzahl Knoten und Kanten eines Zustandsgraphen G^Z_{n,k^*} mit $0 < k^* \leq n$ ermittelt werden. Da eine rekursive Berechnung nicht möglich ist, werden wiederum kombinatorische Überlegungen zur direkten Berechnung der Anzahl Knoten und Kanten angeführt. Für die Anzahl der Knoten und Kanten im Graphen G^Z_{n,k^*} mit n Aufträgen und einer Quelle auf Stufe $k^* \leq n$ gilt folgender Sachverhalt:

Satz 3.5
Es bezeichne $|V^Z_{n,k^*}|$ die Anzahl der Knoten und $|E^Z_{n,k^*}|$ die Anzahl der Kanten im Zustandsgraphen G^Z_{n,k^*} mit n Aufträgen und Quelle auf Stufe $0 \leq k^* \leq n$. Dann gilt:

$$|V^Z_{n,k^*}| = 1 + \sum_{k=k^*+1}^{2n} \sum_{\substack{l=\max\{0,\\ k-2n+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j=\max\{0,\\ k-n-l\}}}^{\substack{\min\{n-k^*,\\ \frac{1}{2}(k-k^*-l)\}}} \frac{k^*!(n-k^*)!(k-k^*-j)}{(k^*-l)!l!(n-k+l+j)!(k-k^*-l-2j)!j!}$$

$$|E^Z_{n,k^*}| = n + \sum_{k=k^*+1}^{2n-1} \sum_{\substack{l=\max\{0,\\ k-2n+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j=\max\{0,\\ k-n-l\}}}^{\substack{\min\{n-k^*,\\ \frac{1}{2}(k-k^*-l)\}}} \frac{k^*!(n-k^*)!(k-k^*-j)(n-l-j)}{(k^*-l)!l!(n-k+l+j)!(k-k^*-l-2j)!j!}$$

BEWEIS: Auch dieser Beweis erfolgt konstruktiv. Analog zum Beweis von Satz 3.4 wird zunächst die Anzahl $|_kW_{n,k^*}|$ der auf einer Stufe $k \geq k^*$ auftretenden Statusvektoren angegeben, bevor daraus die Anzahl $|_kV^Z_{n,k^*}|$ der verschiedenen Zustandsvektoren der Stufe k hergeleitet wird. Daraus wird dann die Anzahl $|V^Z_{n,k^*}|$ aller Knoten des Graphen als Summe über die Anzahl $|_kV^Z_{n,k^*}|$ der Zustandsvektoren der Stufen $k = k^*, \ldots, 2n$ berechnet. Die Anzahl $|_kE^Z_{n,k^*}|$ der ausgehenden Kanten einer Stufe k kann mit Hilfe der Anzahl $|_kV^Z_{n,k^*}|$ der Knoten der Stufe k angegeben und die gesamte Anzahl $|E^Z_{n,k^*}|$ aller Kanten damit als Summe über die ausgehenden Kanten der Stufen $k = k^*, \ldots, 2n-1$ berechnet werden.

Die Quelle des Zustandsgraphen G^Z_{n,k^*} enthält den Zustandsvektor $(\Phi_{k^*}, \mu_{k^*})^T$ mit $\Phi_{k^*} \in \{0,1\}^n$ und $\sum_{i=1}^n \varphi(i) = k^*$. O. B. d. A. kann davon ausgegangen werden, dass der Statusvektor Φ_{k^*} die Form $\Phi_{k^*} = (1, \ldots, 1, 0, \ldots, 0)^T$ hat.

Für den Beweis ist es von elementarer Bedeutung, ob ein Auftrag i in der Quelle des Zustandsgraphen den Status $\varphi_{k^*}(i) = 0$ oder $\varphi_{k^*}(i) = 1$ hat. Es werden daher folgende Begriffe eingeführt: Die ersten k^* Einträge in einem Statusvektor Φ, $\varphi(1), \ldots, \varphi(k^*)$, werden im Folgenden der „*obere Bereich*“ des Statusvektors genannt, der „*untere Bereich*“ bezieht sich auf die letzten $n - k^*$ Einträge $\varphi(k^*+1), \ldots, \varphi(n)$.

Für die Anzahl der Einträge mit Status $\varphi(i) = 0$, $\varphi(i) = 1$ und $\varphi(i) = 2$ gelten folgende Namenskonventionen: im oberen Bereich wird die Anzahl der Einträge mit Status $\varphi(i) = 1$ mit l_1 und die Anzahl der Einträge $\varphi(i) = 2$ mit l_2 bezeichnet, für den unteren Bereich bedeuten j_0 die Anzahl der Einträge $\varphi(i) = 0$, j_1 die Anzahl der Einträge $\varphi(i) = 1$ und j_2 die Anzahl der Einträge $\varphi(i) = 2$ in diesem Bereich des Statusvektors.

Für den Graphen gelten damit folgende Beobachtungen:

- Der Statusvektor Φ_{k^*} auf Stufe k^* hat genau k^* Einträge $\varphi(i) = 1$ und $n - k^*$ Einträge $\varphi(i) = 0$. Einträge mit $\varphi(i) = 2$ kann es nicht geben, da bereits bediente Aufträge nicht in die zukünftige Route eingehen.

- Ein Statusvektor Φ auf einer Stufe $k \geq k^*$ besteht aus genau n Einträgen, von denen k^* Einträge im oberen Bereich von Φ liegen. Die Summe aller Einträge von Φ ist k. Diese Eigenschaften führen zu folgendem unterbestimmten Gleichungssystem:

$$l_1 + l_2 + j_0 + j_1 + j_2 = n \tag{3.9}$$

$$l_1 + l_2 = k^* \tag{3.10}$$

$$l_1 + 2l_2 + j_1 + 2j_2 = k \tag{3.11}$$

Die Anzahl aller möglichen Statusvektoren auf Stufe k lässt sich mit Hilfe kombinatorischer Überlegungen berechnen. Dazu wird zunächst nur der obere Bereich eines Statusvektors Φ der Stufe k betrachtet, bevor auch der untere Bereich hinzugezogen wird:

Im oberen Bereich eines Statusvektors Φ auf Stufe $k > k^*$ sind alle Einträge $\varphi(i) \geq 1, i = 1, \ldots, k^*$. Für die mögliche Anzahl l_2 der Aufträge i mit Status $\varphi(i) = 2$ auf Stufe k gilt:

- Von Stufe k^* bis Stufe k werden genau $k-k^*$ Einheitsvektoren $e_i \in \mathbb{R}^n$ zum Statusvektor Φ_{k^*} addiert. Somit können im oberen Bereich eines Statusvektors Φ der Stufe k maximal $k-k^*$ Einträge den Wert $\varphi(i)=2$ haben. Gleichzeitig befinden sich nur k^* Aufträge im oberen Bereich des Statusvektors Φ, so dass für die mögliche Anzahl l_2 der Aufträge im oberen Bereich mit Status $\varphi(i)=2$ gilt:

$$l_2 \leq \begin{cases} k-k^*, & k < 2k^* \\ k^*, & k \geq 2k^* \end{cases}$$

- Um die minimale Anzahl l_2 der Aufträge im oberen Bereich mit Status $\varphi(i)=2$ auf einer Stufe k zu erhalten, muss man den unteren Bereich betrachten: Von Stufe k^* bis Stufe k werden genau $k-k^*$ Einheitsvektoren $e_i \in \mathbb{R}^n$ zum Statusvektor Φ_{k^*} addiert. Die $n-k^*$ Einträge im unteren Bereich des Statusvektors Φ_{k^*} der Quelle haben den Wert $\varphi_{k^*}(k^*+1) = \cdots = \varphi_{k^*}(n) = 0$. Es können daher maximal $2(n-k^*)$ Einheitsvektoren e_i mit $i > k^*$ zum Statusvektor Φ_{k^*} addiert werden. Spätestens nach $2(n-k^*)$ Stufen muss daher ein Einheitsvektor e_i mit $i \leq k^*$ zum Statusvektor Φ_{k^*} addiert werden, so dass sich ein Eintrag i im oberen Bereich eines Statusvektors Φ' mit $\varphi'(i)=2$ ergibt.
  Für einen Statusvektor Φ auf Stufe k gilt daher, dass die Anzahl l_2 der Einträge 2 im oberen Bereich nur dann den Wert 0 haben kann, falls die Stufe k^* weniger als $2(n-k^*)$ Stufen entfernt liegt, also falls $k < k^* + 2n - 2k^* = 2n - k^*$. Falls $k \geq 2n-k^*$, so muss für jede Stufe, die über $2n-k^*$ hinausgeht, mindestens ein Eintrag im oberen Bereich den Status 2 haben. Damit erhält man:

$$l_2 \geq \begin{cases} 0, & k < 2n-k^* \\ k-2n+k^*, & k \geq 2n-k^* \end{cases}$$

- Es gibt

$$\frac{k^*!}{l_2!\, l_1!}$$

  verschiedene Möglichkeiten, einen Vektor aus k^* Einträgen zu bilden, von denen l_1 die Ausprägung 1 und l_2 die Ausprägung 2 haben.

- Da im oberen Bereich jedes Statusvektors Φ k^* Einträge existieren, kann man die Anzahl l_1 der Einträge mit Status $\varphi(i)=1$ mit Hilfe von Formel 3.10 darstellen als

$$l_1 = k^* - l_2 \tag{3.12}$$

Damit erhält man insgesamt

$$\sum_{l_2=\max\{0,k-2n+k^*\}}^{\min\{k^*,k-k^*\}} \frac{k^*!}{(k^*-l_2)!l_2!}$$

Möglichkeiten, die Einträge 1 oder 2 im oberen Bereich eines Statusvektors Φ auf Stufe k anzuordnen.

Im unteren Bereich von Φ hängen die möglichen Einträge von den Einträgen im oberen Bereich ab, da die Summe aller Einträge den Wert k haben muss. Für den unteren Bereich gelten daher folgende Überlegungen:

- Im unteren Bereich des Statusvektors Φ auf Stufe $k > k^*$ gibt es j_0 Einträge mit Status $\varphi(i) = 0$, j_1 Einträge mit Status $\varphi(i) = 1$ und j_2 Einträge mit Status $\varphi(i) = 2$. Aus den Gleichungen 3.9 und 3.11 erhält man mit $l_1 + l_2 = k^*$:

$$j_1 = k - k^* - 2j_2 - l_2 \tag{3.13}$$

$$j_0 = n - k^* - j_1 - j_2 = n - k + l_2 + j_2 \tag{3.14}$$

  Damit reicht für den unteren Bereich von Φ die Betrachtung der Anzahl j_2 Einträge mit Status $\varphi(i) = 2$ aus.

- Von Stufe k^* bis Stufe k können maximal $\frac{k-k^*}{2}$ Aufträge, die auf Stufe k^* noch den Status 0 hatten, den Status 2 erreichen. Jeder Eintrag mit Status 2 im oberen Bereich von Φ verringert zusätzlich die Anzahl der möglichen Einträge 2 im unteren Bereich von Φ, da die Summe aller Einträge von Φ genau k betragen muss. Die Anzahl der Statusänderungen im oberen Bereich wird mit l_2 wiedergegeben, so dass im unteren Bereich maximal $\frac{k-k^*-l_2}{2}$ Einträge den Status 2 haben können. Da nur $n - k^*$ Einträge im unteren Bereich existieren, ist j_2 ab einer bestimmten Stufe k' natürlich auch durch diese Zahl begrenzt. Zum ersten Mal kann diese Grenze nach $2(n - k^*)$ Änderungen im unteren Bereich erreicht werden, also auf Stufe $k = k^* + 2(n - k^*) + l_2 = 2n - k^* + l_2$. Es gilt somit:

$$j_2 \leq \begin{cases} \frac{k-k^*-l_2}{2}, & k \leq 2n - k^* + l_2 \\ n - k^*, & k > 2n - k^* + l_2 \end{cases}$$

- Auf einer Stufe k muss erst für $k > n$ ein Eintrag mit Status $\varphi(i) = 2$ im Statusvektor Φ auftauchen. Dann gilt allerdings, dass mindestens $k - n$ Einträge den Status $\varphi(i) = 2$ haben müssen. Da die Anzahl der Einträge mit Status $\varphi(i) = 2$ im oberen Bereich bereits durch l_2 gegeben sind, müssen im unteren Bereich $n-k-l_2$ Einträge den Status $\varphi(i) = 2$ besitzen. Nach unten ist die minimale Anzahl natürlicherweise durch 0 begrenzt, so dass sich für j_2 folgende untere Grenze ergibt:

$$j_2 \geq \begin{cases} 0, & k \leq n + l_2 \\ k - n - l_2, & k > n + l_2 \end{cases}$$

- Es gibt

$$\frac{(n - k^*)!}{j_0!\ j_1!\ j_2!}$$

  Möglichkeiten, einen Vektor aus $n - k^*$ Einträgen mit drei möglichen Ausprägungen mit den Anzahlen j_0, j_1 und j_2 zu bilden.

Somit erhalten wir für den unteren Bereich in Abhängigkeit von l_2 aus dem oberen Bereich mit den Formeln 3.13 und 3.14 insgesamt

$$\sum_{j_2=\max\{0,\ k-n-l_2\}}^{\min\{\frac{1}{2}(k-k^*-l_2),\ n-k^*\}} \frac{(n-k^*)!}{(n-k+l_2+j_2)!(k-k^*-l_2-2j_2)!j_2!}$$

Möglichkeiten für die Einträge auf Stufe k.

Durch die Kombination aller möglichen Permutationen des oberen Bereichs mit allen dazu passenden Permutationen des unteren Bereichs erhält man die Anzahl $|{}_kW_{n,k^*}|$ der verschiedenen Statusvektoren auf Stufe k als

$$|{}_kW_{n,k^*}| = \sum_{\substack{l_2=\max\{0,\\ k-2n+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \frac{k^*!}{(k^*-l_2)!l_2!} \Big(\sum_{\substack{j_2=\max\{0,\\ k-n-l\}}}^{\substack{\min\{n-k^*,\\ \frac{1}{2}(k-k^*-l)\}}} \frac{(n-k^*)!}{(n-k+l_2+j_2)!(k-k^*-l_2-2_2)!j_2!} \Big)$$

Um die Anzahl $|{}_kV^Z_{n,k^*}|$ der Zustandsvektoren einer Stufe k zu erhalten, fehlt nun noch die Information, wie viele Fahrzeugpositionen bei einem Statusvektor Φ der Stufe k möglich sind. Für den oberen Bereich des Statusvektors gilt, dass sich das Fahrzeug in den Delivery-Orten aller Aufträge mit Status $\varphi(i) = 2$ befinden kann, also in l_2 verschiedenen Orten. Die Pickup-Orte der Aufträge mit Status $\varphi(i) = 1$ im oberen Bereich kommen nicht als Fahrzeugposition in Frage, da sich der Status dieser Aufträge nicht geändert hat. Im unteren Bereich des Statusvektors hingegen könnten alle Aufträge mit Status $\varphi(i) = 1$ und $\varphi(i) = 2$ zuletzt ihren Status geändert haben, so dass hier die entsprechenden j_1 Pickup-Orte und j_2 Delivery-Orte als Fahrzeugpositionen möglich sind. Insgesamt erhält man somit $l_2 + j_1 + j_2$ mögliche Fahrzeugpositionen zu einem Statusvektor Φ der Stufe k. Mit Hilfe von Formel 3.13 lässt sich dies umformen zu $(k-k^*-j_2)$, so dass man die Anzahl $|{}_kV^Z_{n,k^*}|$ der Zustandsvektoren auf Stufe k erhält als:

$$|{}_kV^Z_{n,k^*}| = \sum_{\substack{l_2=\max\{0,\\ k-2n+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \frac{k^*!}{(k^*-l_2)!l_2!} \Big(\sum_{\substack{j_2=\max\{0,\\ k-n-l\}}}^{\substack{\min\{n-k^*,\\ \frac{1}{2}(k-k^*-l)\}}} \frac{(n-k^*)!(k-k^*-j_2)}{(n-k+l_2+j_2)!(k-k^*-l_2-2_2)!j_2!} \Big)$$

Ein auf Stufe k^* beginnender Zustandsgraph G^Z_{n,k^*} hat also auf Stufe k^* eine Quelle und die soeben angegebene Anzahl $|{}_kV^Z_{n,k^*}|$ Zustandsvektoren als Knoten auf den folgenden Stufen $k = k^* + 1, \ldots, 2n$. Für den Graphen G^Z_{n,k^*} erhält man somit die Anzahl Knoten $|{}_kV^Z_{n,k^*}|$ als:

$$|V^Z_{n,k^*}| = 1 + \sum_{k=k^*+1}^{2n} \sum_{\substack{l_2=\max\{0,\\ k-2n+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j_2=\max\{0,\\ k-n-l_2\}}}^{\substack{\min\{n-k^*,\\ \frac{1}{2}(k-k^*-l_2)\}}} \frac{k^*!(n-k^*)!(k-k^*-j_2)}{(k^*-l_2)!l_2!(n-k+l_2+j_2)!(k-k^*-l_2-2j_2)!j_2!}$$

Ersetzen von l_2 durch l und j_2 durch j liefert die Behauptung.

Für die Anzahl der Kanten gilt:

- Von der Quelle auf Stufe k^* führen n Kanten zu Zustandsvektoren der Stufe $k^* + 1$.
- Von jedem Knoten $(\Phi, \mu)^T$ einer Stufe k mit $k^* < k < 2n$ führen so viele Kanten zur nächsthöheren Stufe $k + 1$, wie neue Zustandsvektoren $(\Phi', \mu')^T$ aus der Addition eines Einheitsvektors e_i zu Φ und entsprechender Anpassung der Fahrzeugposition hervorgehen können. Da für jeden Auftrag i mit $\varphi(i) < 2$ der Einheitsvektor e_i zu einem neuen Zustand auf der nächsthöheren Stufe führt, entspricht die Anzahl der von einem Knoten $(\Phi, \mu)^T$ der Stufe k ausgehenden Kanten der Anzahl Aufträge mit Status $\varphi(i) = 0$ und $\varphi(i) = 1$, also $l_1 + j_0 + j_1$.
- Sämtliche Knoten $(\Phi, \mu)^T$ der Stufe $2n$ bilden die Senken des Zustandsgraphen G^Z_{n,k^*}. Die Statusvektoren Φ dieser Zustandsvektoren enthalten nur Einträge 2.

Mit Hilfe der Formeln 3.12, 3.13 und 3.14 ersetzt man $l_1 + j_0 + j_1$ durch $n - l_2 - j_2$ und erhält somit die Anzahl $|E^Z_{n,k^*}|$ Kanten eines Zustandsgraphen G^Z_{n,k^*}:

$$|_k E^Z_{n,k^*}| = n + \sum_{k=k^*+1}^{2n-1} \sum_{\substack{l_2=\max\{0,\\ k-2n+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j_2=\max\{0,\\ k-n-l_2\}}}^{\substack{n-k^*,\\ \min\{\frac{1}{2}(k-k^*-l_2)\}}} \frac{k^*!(n-k^*)!(k-k^*-j_2)(n-l_2-j_2)}{(k^*-l_2)!l_2!(n-k+l_2+j_2)!(k-k^*-l_2-2j_2)!j_2!}$$

Ersetzen von l_2 durch l und j_2 durch j liefert die Behauptung.

□

Die Größe des Zustandsgraphen G^Z_{n,k^*} ist somit bekannt. Die Ergebnisse der Tabelle 3.6 lassen deutliche Einsparungen an Knoten und Kanten gegenüber dem Statusvektorbaum erkennen, so dass eine Implementierung dieser Graphenstruktur lohnenswert erscheint. Die Details der Implementierung sind im Anhang dargestellt. Im folgenden Abschnitt 3.3.3 werden die Anzahlen der Knoten und Kanten des Zustandsgraphen und des Statusvektorbaums verglichen sowie die Auswertung der Testläufe vorgestellt.

3.3.3 Ergebnisse und Fazit

Ziel der Testläufe ist es, Aussagen über das Laufzeitverhalten des Algorithmus bei Verwendung des Zustandsgraphen zu treffen. Dabei geht es darum, ob die Anzahl der erzeugten Knoten, gemessen an der Anzahl der erzeugten Knoten bei Verwendung des Statusvektorbaums, reduziert werden kann, um somit auch die Laufzeit des Algorithmus zu verringern. Zusätzlich wird die Güte des A*-Algorithmus gemessen, indem die pro Aufruf eines Single Vehicle Routings auf einem Zustandsgraphen G^Z_{n,k^*} durchschnittlich erzeugten Knoten mit den Knoten eines Zustandsgraphen dieser Größe verglichen werden.

Problem-gruppe	Aufträge	Aufträge absolut	Aufträge Δ	gelöste Instanzen absolut	gelöste Instanzen Δ
100	51,86	33,45	13,45%	28	12,00%
200	103,08	44,07	32,33%	14	40,00%
400	205,66	65,51	23,92%	8	14,29%
600	308,97	73,58	17,02%	4	33,33%
800	411,75	81,86	15,33%	0	-
1000	514,98	92,00	16,58%	0	-

Tabelle 3.7: Die Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung des Zustandsgraphen G^Z_{n,k^*} im Vergleich zur Anzahl bei Verwendung des Statusvektorbaums G^S_{n,k^*}

Für die Testläufe wurde die Testumgebung wiederum ereignisdiskret eingestellt und die künstliche Grenze von einer Million erzeugter Knoten aufrecht erhalten. Es werden die gleichen Testdatensätze wie in Abschnitt 3.2.4 verwendet. Die einzige Änderung besteht darin, dass der Algorithmus den Zustandsgraphen G^Z_{n,k^*} anstelle des Statusvektorbaums G^S_{n,k^*} benutzt.

Die Tabellen 3.7 - 3.11 haben zur besseren Vergleichbarkeit die gleiche Struktur wie die Tabellen im Abschnitt 3.2.4. Neben den absoluten Werten werden nun jedoch in der Spalte „Δ“ die relativen Veränderungen im Vergleich zu den Testläufen bei Verwendung des Statusvektorbaums aus Abschnitt 3.2.4 angegeben.

In Tabelle 3.7 werden die Anzahl der Aufträge, die durchschnittlich bis zur Grenze von einer Million Knoten bearbeitet wurden, und die Anzahl der vollständig gelösten Instanzen angegeben. Die Spalte „Aufträge – Δ“ zeigt, dass die Anzahl der bearbeiteten Aufträge im Vergleich um 13,45% bis maximal 32,33% gestiegen sind. Die Steigerungsraten fallen mit zunehmender Anzahl Aufträge der Testinstanzen, da in den Instanzen mit vielen Aufträgen tendenziell größere Graphen bearbeitet werden müssen. Eine Ausnahme bilden die Testläufe der Problemklasse „100“, die mit 13,45% die kleinste Zuwachsrate an bearbeiteten Aufträgen hat. Dies liegt an der Anzahl der Aufträge: einige Instanzen waren bei Erzeugung von wenigen Knoten vollständig gelöst. Die Tabelle zeigt weiterhin, dass bei Verwendung des Zustandsgraphen einige Instanzen mehr vollständig gelöst werden können. Für die Problemklassen „800“ und „1000“ konnte jedoch noch immer keine Instanz bei Erzeugung von maximal einer Million Knoten gelöst werden.

Die Zunahme der bearbeiteten Aufträge und gelösten Instanzen bei gleichbleibender Beschränkung auf eine Million Knoten im Graphen lässt auf eine Reduktion der erzeugten Knoten pro Testlauf schließen. Die Anzahl der Knoten wird in Tabelle 3.8 dargestellt. Dazu werden zusätzlich zur Anzahl der erzeugten Knoten auch die Anzahl der als unzulässig erkannten und daher wieder gelöschten Knoten, die Anzahl der expandierten Knoten und die Anzahl der Duplikat-Knoten angegeben. Wie schon in den Testläufen aus Abschnitt 3.2.4, steigt die Anzahl der erzeugten Knoten mit zunehmender Anzahl Aufträge der Problemklasse an. Die

Problem-gruppe	♯ Knoten absolut			
	erzeugt	unzulässig	expandiert	Duplikate
100	405.959,57	181.832,27	72.705,41	138.538,88
200	656.222,38	310.588,53	125.398,02	204.313,07
400	729.593,19	381.458,56	151.043,27	176.805,17
600	817.006,63	437.742,37	171.161,40	186.235,35
800	835.726,05	457.479,83	180.075,81	176.916,51
1000	813.649,83	443.302,86	182.193,05	161.348,57

Tabelle 3.8: Die Anzahl der Knoten pro Testlauf bei Verwendung des Zustandsgraphen G^Z_{n,k^*}

Anzahl der Duplikat-Knoten steigt von Problemklasse „100“ zur Problemklasse „200“ merklich an, ist dann aber in den folgenden Problemklassen ähnlich hoch, obwohl sich die Anzahl der erzeugten Knoten von der Problemklasse „100“ bis zu den Problemklassen „600“, „800“ und „1000“ verdoppelt. Dieses Phänomen liegt darin begründet, dass die Anzahl der Duplikat-Knoten in den Graphen frühestens auf Stufe $k^* + 3$ erzeugt werden. In den Problemklassen mit vielen Aufträgen werden die Berechnungen schon wegen erreichter einer Million Knoten unterbrochen, bevor viele Duplikat-Knoten erzeugt werden. In den Problemklassen „100“ und „200“ wird aber schon deutlich, dass viele Knoten redundant erzeugt wurden und mit dem Zustandsgraphen, der diese Knoten im Gegensatz zum Statusvektorbaum nicht weiter expandiert, daher viele Knoten und somit auch Rechenzeit eingespart werden können.

Die absoluten Werte der Tabelle 3.8 können für diesen Vergleich jedoch nicht herangezogen werden, da die Testläufe bei Verwendung des Zustandsgraphen bzw. des Statusvektorbaums unterschiedlich viele Aufträge bearbeitet haben. Zur genauen Analyse werden daher in Tabelle 3.9 die Anzahl der Knoten bei Beschränkung der Aufträge auf die in den Testläufen aus Abschnitt 3.2.4 bearbeiteten Aufträge dargestellt. Die Spalte „♯ Vgl“ gibt jeweils die absolute Anzahl der Knoten wieder, während die Spalte „Δ“ die relative Veränderung im Vergleich zu den Testläufen in Abschnitt 3.2.4 beinhaltet. Die Spalte „Duplikate“ hat nur absolute Werte, da kein Vergleichswert gegeben ist. Es ist gut zu erkennen, dass die Anzahl der Knoten deutlich reduziert werden konnte. Es verwundert zunächst, dass die Einsparungen mit zunehmender Anzahl Aufträge in einer Problemgruppe fallen, obwohl die relative Reduktion der Anzahl Knoten im Zustandsgraphen zur Anzahl Knoten im Statusvektorbaum bei wachsender Anzahl Aufträge n deutlich ansteigt (vgl. Tabelle 3.6). Die Anzahl der erzeugten Knoten während eines Testlaufs hängt jedoch auch von der Güte des A*-Algorithmus ab. Wie in Tabelle 3.5 dargestellt wurde, wurden in den Statusvektorbäumen für mehr als $n = 3$ Aufträge nur ein kleiner Bruchteil der Knoten erzeugt. Daher kann die Verwendung des Zustandsgraphen hier „nur“ 55% der erzeugten Knoten einsparen.

[3] Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.
100	63.274,13	-81,63%	36.005,07	-77,84%	15.870,84	-84,73%	8.937,61
200	113.594,73	-79,12%	73.033,43	-75,92%	27.501,98	-82,16%	10.362,43
400	194.987,54	-70,57%	125.812,00	-66,69%	51.469,24	-73,00%	13.200,61
600	260.171,85	-61,80%	173.092,07	-58,73%	70.639,62	-62,44%	11.851,48
800	297.429,14	-56,32%	197.687,08	-51,88%	81.921,86	-55,83%	11.655,78
1000	334.268,59	-55,67%	219.584,67	-51,35%	91.670,22	-56,33%	13.668,76

Tabelle 3.9: Die Anzahl der Knoten pro Testlauf bei Verwendung des Zustandsgraphen G^Z_{n,k^*} im Vergleich zur Anzahl Knoten bei Verwendung des Statusvektorbaums G^S_{n,k^*}

Problem-gruppe	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	131,44	0,76	-99,97%	10,34	0,04	-99,98%	0,35	0,01	-99,93%
200	58,12	0,80	-99,97%	9,34	0,05	-99,98%	0,26	0,01	-99,92%
400	71,93	0,48	-99,98%	7,04	0,05	-99,98%	0,14	0,01	-99,89%
600	121,11	0,27	-99,99%	10,07	0,05	-99,97%	0,18	0,01	-99,77%
800	126,03	0,32	-99,97%	8,04	0,05	-99,96%	0,14	0,01	-99,65%
1000	48,83	0,81	-99,98%	4,72	0,07	-99,97%	0,09	0,01	-99,76%

Tabelle 3.10: Die Antwortzeiten bei Verwendung des Zustandsgraphen G^Z_{n,k^*} im Vergleich zu den Antwortzeiten bei Verwendung des Statusvektorbaums G^S_{n,k^*}

Die Auswirkungen der Reduktion der Anzahl Knoten auf die Antwortzeiten wird in Tabelle 3.10 dargestellt. Dazu werden wiederum die global maximale Antwortzeit über alle Testinstanzen der Problemklasse, die durchschnittliche maximale Zeit aller Testinstanzen einer Problemklasse und die durchschnittliche Antwortzeit aller Aufträge in einer Problemklasse angegeben. Es werden jeweils in der Spalte „sec“ die absolut erreichte Zeit in Sekunden, in der Spalte „Vgl“ die bei Beschränkung der zu bearbeitenden Aufträge pro Testlauf auf die in den Testläufen aus Abschnitt 3.2.4 bearbeiteten Aufträge erreichten Zeiten und in Spalte „Δ“ die relative Veränderung dieser Zeiten angegeben. Die Ergebnisse sind mit Einsparungen von 99% der global maximalen, durchschnittlichen maximalen und der durchschnittlichen Antwortzeiten in allen Problemklassen sehr gut. Dennoch zeigen die in diesen Testläufen erreichten absoluten Werte, dass die Reduzierungen noch nicht ausreichend sind. Die durchschnittliche Antwortzeit liegt zwar deutlich unter einer Sekunde, die global maximale Antwortzeit liegt aber in der Hälfte der Problemklassen noch bei ca. zwei Minuten. Die Antwortzeiten werden jedoch vermutlich weiter steigen, sobald die Grenze von maximal einer Million zu erzeugenden Knoten entfernt wird.

Tabelle 3.11 zeigt abschließend den Vergleich zwischen der Anzahl Knoten, die während der Testläufe in einem Zustandsgraphen für $n = 1, \ldots, 11$ Aufträge erzeugt wurden, und der

Auf-träge	♯ Aufrufe	Ø ♯ Knoten absolut	Δ	Ø max. ♯ Knoten absolut	Δ	Ø min. ♯ Knoten absolut	Δ
1	376.565,00	3,00	-0,10%[4]	3,00	0,00%	2,84	-5,22%
2	166.252,00	2,89	-64,59%	8,34	-21,95%	5,41	-52,00%
3	212.668,00	8,64	-74,60%	25,62	-38,33%	14,83	-66,97%
4	175.766,00	26,08	-80,41%	81,00	-47,60%	43,87	-74,42%
5	79.841,00	86,16	-82,92%	263,21	-54,37%	148,24	-77,11%
6	24.383,00	325,99	-83,29%	887,09	-59,10%	557,04	-76,53%
7	7.520,00	1.172,40	-83,66%	3.183,87	-61,23%	2.365,21	-72,85%
8	2.104,00	5.655,15	-78,85%	11.686,86	-61,12%	9.296,20	-69,75%
9	510,00	22.964,17	-75,62%	38.433,58	-62,78%	34.167,67	-67,38%
10	233,00	59.835,24	-80,06%	77.187,57	-76,43%	73.898,63	-77,90%
11	43,00	87.018,93	-89,23%	94.983,82	-88,38%	91.906,08	-88,88%
12	7,00	71.131,14	-97,66%	71.131,14	-97,66%	71.131,14	-97,66%

Tabelle 3.11: Die Anzahl der im A*-Algorithmus bei Verwendung des Zustandsgraphen erzeugten Knoten im Vergleich zur theoretischen Anzahl Knoten in den entsprechenden Zustandsgraphen.

Anzahl Knoten V^Z_{n,k^*} dieses Zustandsgraphen. Zur besseren Übersicht wurden die Ergebnisse hier für alle Zustandsgraphen G^Z_{n,k^*} mit n Aufträgen und Quelle auf Stufe $k^* = 0, \ldots, n$ aggregiert, indem jeweils das gewichtetes Mittel der Ergebnisse für die Zustandsgraphen G^Z_{n,k^*} mit Quelle auf Stufe k^* angegeben wird, wobei das Gewicht der relativen Häufigkeit der Aufrufe des Single Vehicle Routings auf diesem Zustandsgraphen G^Z_{n,k^*} entspricht. Die Tabelle zeigt, dass der A*-Algorithmus in den Zustandsgraphen für mehr als drei Aufträge mehr als 80% der Knoten V^Z_{n,k^*} dieses Zustandsgraphen nicht erzeugt. Betrachtet man die Anzahl der maximal in einem Zustandsgraphen für $n > 4$ Aufträge in einem Testlauf erzeugten Knoten, so werden – selbst in diesem für den A*-Algorithmus eher „schwierigen“ Fall – über 50% der Knoten nicht erzeugt. Vergleicht man die Einsparungen des A*-Algorithmus bei Verwendung des Zustandsgraphen mit dessen Einsparungen bei Verwendung des Statusvektorbaums (Tabelle 3.5), so fallen die relativen Verbesserungen bei Verwendung des Zustandsgraphen etwas kleiner aus. Das lässt sich dadurch erklären, dass wegen der geänderten Struktur des Zustandsgraphen weniger Folgeknoten nicht erzeugt werden müssen, wenn ein Teil des Graphen nicht vom Algorithmus expandiert wird.

Die Einführung des Zustandsgraphen G^Z_{n,k^*} hat sehr gute Ergebnisse gebracht. Es konnten zwar noch längst nicht alle Testinstanzen in der begrenzten Rechenzeit, beschränkt durch eine Million Knoten, die erzeugt werden dürfen, gelöst werden, doch der Anteil der gelösten Probleme konnte deutlich gesteigert und die Rechenzeit merklich gesenkt werden.

Alle weiteren Betrachtungen zur Beschleunigung des Single Vehicle Routings werden daher auf Grundlage des Zustandsgraphen weiter entwickelt.

3.4 Beschleunigung durch andere Schätzfunktionen

Die Lösung des Single Vehicle Routings kann nicht nur durch den zugrunde liegenden Graphen, sondern auch durch Anpassung des A*-Algorithmus mit Hilfe einer anderen Schätzfunktion beschleunigt werden (vgl. Abschnitt 3.2.3). Der A*-Algorithmus basiert auf dem Algorithmus von Dijkstra [21], benutzt aber zur Bewertung eines expandierten Knotens zusätzlich zur Fahrzeit des bekannten kürzesten Wegs von der Quelle des Graphen bis zu diesem Knoten zusätzlich eine Abschätzung der verbleibenden Fahrzeit bis zur Senke. In [11] wird dazu die Schätzfunktion „Maximalauftragszeit“ benutzt, wobei die maximale Fahrzeit berechnet wird, die noch bis zur kompletten Abarbeitung eines einzelnen Auftrags zurückgelegt werden muss.

Grundsätzlich stellt sich die Frage, ob die Schätzfunktion nicht dahingehend verbessert werden kann, dass die berechneten Werte die tatsächlich verbleibende Fahrzeit von einem Knoten zu einer Senke möglichst genau wiedergeben, um so die Zahl der expandierten Knoten und damit die Laufzeit des Algorithmus zu reduzieren. Dazu sind zwei Ansätze denkbar:

- die Schätzfunktion kann grundsätzlich geändert werden, so dass nicht mehr die „Maximalauftragszeit“ als längste verbleibende Fahrzeit eines einzelnen Auftrags betrachtet wird,
- Varianten der Schätzfunktion „Maximalauftragszeit“ können genauer an das Entscheidungsproblem angepasst werden, indem zusätzliche Informationen aus dem Entscheidungsproblem in diese Schätzfunktionen eingebracht werden.

Für beide Ansätze gilt, dass weder die Berechnung der Schätzfunktion viel Rechenzeit in Anspruch nehmen noch die verbleibende Fahrzeit überschätzt und somit eine suboptimale Lösung ermöglicht werden darf.

Für die Schätzfunktion „Maximalauftragszeit“ gilt, dass der Unterschied zwischen dem Wert der Schätzfunktion und der tatsächlich verbleibenden Fahrzeit vermutlich umso größer wird, je mehr Aufträge pro Fahrzeug betrachtet werden. Das liegt daran, dass die „Maximalauftragszeit“ genau dann die tatsächliche verbleibende Fahrzeit relativ genau zurückgibt, wenn entweder nur noch ein Auftrag zu bedienen ist oder aber die geographische Verteilung aller noch zu anzufahrenden Orte einem Strahl mit Beginn im Ort der Fahrzeugposition ähnelt.

Ein möglicher Ansatz zur Verwendung einer anderen Schätzfunktion besteht darin, die Schätzfunktion als Summe der minimalen Distanzen aller noch anzufahrenden Orte einschließlich des aktuellen Standorts zu definieren, wie es beispielhaft auf der rechten Seite der Abbildung 3.8 dargestellt wird (vgl. auch Abschnitt 3.2.3). Dieser Ansatz wird im Abschnitt 3.4.1 näher beschrieben. Um „genauere“ Varianten einer bestehenden Schätzfunktion zu erhalten, können sowohl die Zeitfenster als auch die Servicezeiten in die Funktion einbezogen werden. Mit den Servicezeiten kann die Länge der noch verbleibenden Fahrzeit für einen Auftrag genauer berechnet werden. Da alle Servicezeiten exakt einzuhalten sind, dürfen sie für die Abschätzung verwendet werden. Die Ausführungen dazu finden sich im Abschnitt 3.4.2.

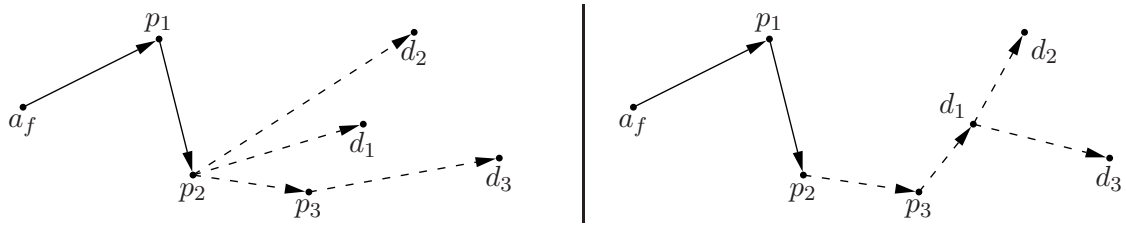

Abbildung 3.8: Beispiel für zwei mögliche Abschätzungen der verbleibenden Fahrtstrecke: „maximale Länge für einen Auftrag“ und „Minimalgerüst der verbleibenden Orte“.

Die Zeitfenster beeinflussen zum einen die Fahrzeit, wenn gewartet werden muss, zum anderen sorgen sie – bei Überschreitung der oberen Zeitfenstergrenze eines Ortes – für unzulässige Knoten, die nicht weiter untersucht werden müssen. Im Abschnitt 3.4.3 wird daher eine Schätzfunktion untersucht, in deren Berechnung die Zeitfenster einfließen.

In jedem Abschnitt werden dabei die zu Grunde liegenden Ideen beschrieben, die Umsetzung erläutert sowie deren Effekt auf die Ergebnisse anhand von Testläufen dokumentiert und ausgewertet. Die Testläufe werden wie in den vorhergehenden Abschnitten 3.2 und 3.3 bei ereignisdiskreter Einstellung der Testumgebung mit einer künstlichen Beschränkung von einer Million Knoten durchgeführt. Zu Grunde gelegt wird der Zustandsgraph G^Z_{n,k^*}. Als Vergleichsdaten dienen die Testläufe zum Zustandsgraphen aus Abschnitt 3.3.3, bei denen als Schätzfunktion die „Maximalauftragszeit“ benutzt wurde.

3.4.1 Die Schätzfunktion „Minimalzeitensumme“

Dieser Abschnitt befasst sich mit der Schätzfunktion „*Minimalzeitensumme*“, $\overrightarrow{C}^+$. Dabei handelt es sich um einen anderen Ansatz, einen geschätzten Wert für die noch verbleibende Fahrzeit zu berechnen. Ziel ist es, eine Schätzfunktion zu bestimmen, deren Wert insbesondere bei mehreren noch zu bedienenden Aufträgen näher an der tatsächlich verbleibenden Fahrzeit liegt als der entsprechende Wert der Schätzfunktion „Maximalauftragszeit“, um so eine schnellere Lösung des Single Vehicle Routing mit Hilfe des A*-Algorithmus unter Verwendung der neuen Schätzfunktion zu ermöglichen. Dazu muss allerdings auch die Zeit für die Berechnung des Werts der Schätzfunktion für einen gegebenen Knoten $(\Phi, \mu)^T$ klein gehalten werden, da der Wert der Schätzfunktion $\overrightarrow{C}^+((\Phi, \mu)^T, {}_{2n}V^Z_{n,k^*})$ für jeden erzeugten Knoten berechnet wird.

Für die Schätzfunktion „Maximalauftragzeit“ wird nur das Erfüllen eines Auftrags zur Berechnung eines geschätzten Werts für die noch verbleibenden Fahrzeit zu Grunde gelegt. Bei wenigen noch zu bedienenden Aufträgen, deren anzufahrende Orte nah beieinander liegen, wird diese Abschätzung relativ gute Resultate bringen, da die Fahrzeit zu den für den berechneten Wert nicht beachteten anzufahrenden Orten gering ist und somit nur eine relativ kleine Abweichung vom Wert der Schätzfunktion gegeben ist. Je mehr Aufträge aber noch zu bearbeiten sind, um so länger wird tendenziell die noch verbleibende Fahrzeit, während sich der Wert der Schätzfunktion nicht notwendigerweise ändert. Diese Eigenschaft führt dazu, dass

prinzipiell Knoten auf niedrigen Stufen des Zustandsgraphen bevorzugt expandiert werden, da immer der Knoten mit minimaler Summe aus bisher tatsächlich zurückgelegter Fahrzeit und geschätzter verbleibender Fahrzeit zur Expandierung ausgewählt wird. Die tatsächlich bisher zurückgelegte Fahrzeit zum Erreichen eines Knotens $(\Phi, \mu)^T$ auf einer niedrigen Stufe k im Zustandsgraphen ist tendenziell kleiner als die tatsächlich bisher zurückgelegte Fahrzeit zum Erreichen eines Knotens $(\Phi', \mu')^T$ auf einer Stufe $k' < k$, da dazu bereits $k - k'$ mehr Orte angefahren wurden, während der Wert der Schätzfunktion für die noch verbleibende Fahrzeit in beiden Fällen nur auf einem noch zu bedienenden Auftrag basiert und daher ähnlich hoch sein kann.

Eine andere Möglichkeit zur Berechnung eines geschätzten Werts für die noch verbleibende Fahrzeit von einem Zustandsvektor $(\Phi, \mu)^T$ besteht in der Berechnung der Minimalzeitensumme: man betrachte den gewichteten Untergraphen $G^{Orte}_{\Phi,\mu}$ von G^{Orte}, dem Graphen zur Darstellung der geographischen Verteilung der Orte, der den Ort μ der aktuellen Fahrzeugposition sowie alle noch anzufahrenden Orte als Knotenmenge $V^{Orte}_{\Phi,\mu}$ enthält und dessen Kanten alle möglichen Nachfolge-Relationen der Orte in einer Route durch alle noch anzufahrenden Orte abbilden:

$$\begin{aligned} V^{Orte}_{\Phi,\mu} &= \{\mu\} \cup \{p_i \in P \mid \varphi(i) = 0\} \cup \{d_i \in D \mid \varphi(i) < 2\} \\ E^{Orte}_{\Phi,\mu} &= V^{Orte}_{\varphi,\mu} \times V^{Orte}_{\varphi,\mu} \setminus \{\mu\} \end{aligned}$$

Das Gewicht $c^{Orte}_{\Phi,\mu}(\nu, \nu')$ einer Kante $(\nu, \nu') \in V^{Orte}_{\Phi,\mu}$ entspricht der Fahrzeit $c_{\nu,\nu'}$ von Ort ν zu Ort ν'. Jeder der noch anzufahrenden Orte $\nu \in V^{Orte}_{\Phi,\mu}$ soll in der Schätzfunktion berücksichtigt werden, um auch bei vielen noch anzufahrenden Orten einen „verlässlichen“ Wert für die noch verbleibende Fahrzeit zu berechnen. Die Minimalzeitensumme verfolgt daher die Idee, die Fahrzeit zu jedem der noch anzufahrenden Orte zu addieren. Da die Reihenfolge der Orte in der Route nicht gegeben ist und die verbleibende Fahrzeit nicht überschätzt werden darf, wird für jeden noch anzufahrenden Ort ν in $V^{Orte}_{\Phi,\mu}$ die minimale Fahrzeit $c_{\nu,\nu'}$ zu einem anderen noch anzufahrenden Ort ν' in $V^{Orte}_{\Phi,\mu}$ addiert. Somit werden alle noch anzufahrenden Orte berücksichtigt. Insbesondere bei geographisch gleichverteilten Orten sind gute Ergebnisse zu warten, während bei einer Häufung von jeweils exakt zwei Orten der Wert der „Minimalzeitensumme“ die tatsächliche Fahrzeit unterschätzen wird. Diese Idee muss nun auf den Zustandsgraphen übertragen werden:

Die Minimalzeitensumme $\overrightarrow{C}^+ \colon V^Z_{n,k^*} \times {}_{2n}V^Z_{n,k^*} \to \mathbb{R}^+$ gibt die geschätzte minimale Fahrzeit von einem Knoten $(\Phi, \mu)^T \in {}_{2n}V^Z_{n,k^*}$ zu einem der Zielknoten $(\Phi^*, \mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ zurück. Ausgehend vom Zustandsvektor $(\Phi, \mu)^T$, bestehend aus dem Statusvektor $\Phi = (\varphi(1), \ldots, \varphi(n))^T$ und dem Ort μ der Fahrzeugposition, der zu jedem Auftrag $i \in \{1, \ldots, n\}$ gegebenen Pickup-Orte p_i und Delivery-Orte d_i und der bekannten Fahrzeiten $c_{\nu,\nu'}$ zwischen je zwei anzufahrenden Orten ν, ν', wird für jeden Auftrag i die minimale Fahrzeit zu einem Zustandsvektor $(\Phi^*, \mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ berechnet als:

$$\overrightarrow{C}_i^+\left(\binom{\Phi}{\mu},\binom{\Phi^*}{\mu^*}\right)=\begin{cases} 0, & \varphi(i)=\varphi^*(i) \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j),\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\}, & \varphi(i)=0 \wedge \varphi^*(i)=1 \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j),\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu d_i}, c_{p_j d_i}, c_{d_l d_i}\}, & \varphi(i)=1 \wedge \varphi^*(i)=2 \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j)\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\} + \min\limits_{\substack{0=\varphi(j)<\varphi^*(j)\\ \varphi(l)<\varphi^*(l)=2}} \{c_{p_j d_i}, c_{d_l d_i}\}, & \varphi(i)=0 \wedge \varphi^*(i)=2 \end{cases}$$

Mit $\overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$ wird somit die mindestens noch benötigte Fahrzeit zu den für Auftrag i noch anzufahrenden Orten definiert: falls Auftrag i in beiden Zustandsvektoren den gleichen Status $\varphi(i)=\varphi^*(i)$ hat, wird keine Fahrzeit benötigt. Falls sich der Status von $\varphi(i)=0$ auf $\varphi^*(i)=1$ ändert, so muss nur die minimale Fahrzeit zum Pickup-Ort p_i berechnet werden, bei einer Änderung des Status von $\varphi(i)=1$ auf $\varphi^*(i)=2$ wird die minimale Fahrzeit zum Delivery-Ort d_i zurückgegeben. Für einen Auftrag i mit Status $\varphi(i)=0$ und $\varphi^*(i)=2$ werden die minimale Fahrzeit zum Pickup-Ort p_i und die minimale Fahrzeit zum Delivery-Ort d_i des Auftrags addiert. Die Summe

$$\sum_{i=1}^{n} \overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

gibt damit den geschätzten Wert der minimalen Fahrzeit von $(\Phi,\mu)^T$ zum Zielknoten $(\Phi^*,\mu^*)^T$ an. Damit kann die Minimalzeitensumme $\overrightarrow{C}^+((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})$ als Minimum der zu jedem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ geschätzten minimalen Fahrzeit definiert werden:

$$\overrightarrow{C}^+((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) \quad = \quad \min_{(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}} \sum_{i=1}^{n} \overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

Mit dieser Definition können nun die Eigenschaften des A*-Algorithmus bei Verwendung der Minimalzeitensumme als Schätzfunktion angegeben werden. Wie in Abschnitt 3.2.3 beschrieben, kann die Wahl einer Schätzfunktion dazu führen, dass die Optimalität der Lösung oder aber das schnellstmögliche Auffinden der Lösung nicht gewährleistet sind. Hier gilt:

Satz 3.6

Der A-Algorithmus findet bei Verwendung der Schätzfunktion „Minimalzeitensumme“ $\overrightarrow{C}^+$ schnellstmöglich einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Zustandsgraphen G^Z_{n,k^*}.*

Beweis: Um zu gewährleisten, dass der A*-Algorithmus mit der Schätzfunktion $\overrightarrow{C}^+$ schnellstmöglich eine optimale Lösung findet, muss sicher gestellt sein, dass die Minimalzeitensumme die tatsächlich verbleibende Fahrzeit nicht überschätzt und die Dreiecksungleichung für die mit der Schätzfunktion berechneten Werte gilt (vgl. [41]).

Im A*-Algorithmus wird jeweils die Fahrzeit $\overrightarrow{C}_i^+((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})$ von einem Knoten $(\Phi,\mu)^T$ zu einem der Zielknoten $(\Phi^*_{2n},\mu^*_{2n})^T \in {}_{2n}V^Z_{n,k^*}$ berechnet. Jeder dieser Zielknoten

$(\Phi^*_{2n}, \mu^*_{2n})^T$ hat den Statusvektor $\Phi^*_{2n} = (2, \ldots, 2)$, die Zustandsvektoren unterscheiden sich nur in der Fahrzeugposition μ_{2n}, die allerdings nicht in die Minimalzeitensumme eingeht. Für die Berechnung der Minimalzeitensumme muss daher nicht dass Minimum über alle $(\Phi^*_{2n}, \mu^*_{2n})^T \in {}_{2n}V^Z_{n,k^*}$ berechnet werden. Auch die minimalen Fahrzeiten für jeden Auftrag i können einfacher geschrieben werden:

$$\overrightarrow{C}_i^+\left(\begin{pmatrix}\Phi\\ \mu\end{pmatrix}, \begin{pmatrix}\Phi^*_{2n}\\ \mu^*_{2n}\end{pmatrix}\right) = \begin{cases} 0, & \varphi(i) = 2 \\ \min\limits_{\varphi(j)=0, \varphi(l)\leq 1} \{c_{\mu d_i}, c_{p_j d_i}, c_{d_l d_i}\}, & \varphi(i) = 1 \\ \min\limits_{\varphi(j)=0, \varphi(l)\leq 1} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\} + \min\limits_{\varphi(j)=0, \varphi(j)\leq 1} \{c_{p_j d_i}, c_{d_l d_i}\}, & \varphi(i) = 0 \end{cases}$$

Für die Minimalzeitensumme $\overrightarrow{C}^+((\Phi, \mu)^T, {}_{2n}V^Z_{n,k^*})$ ergibt sich somit

$$\overrightarrow{C}^+((\Phi, \mu)^T, {}_{2n}V^Z_{n,k^*}) = \sum_{i=1}^{n} \overrightarrow{C}_i^+((\Phi, \mu)^T, (\Phi^*_{2n}, \mu^*_{2n})^T)$$

Für den Beweis, dass die Minimalzeitensumme die verbleibende Fahrzeit nicht überschätzt, betrachtet man alle Kanten $(\nu', \nu)^* \in E^{Orte}_{\Phi,\mu}$ im Untergraphen $G^{Orte}_{\Phi,\mu}$, die in einer fahrzeitminimalen Route von der Fahrzeugposition μ durch alle noch anzufahrenden Orte $\nu \in V^{Orte}_{\Phi,\mu}$ benutzt werden, und die Fahrzeit c^*_ν der Kante $(\nu', \nu)^*$. Diese Kanten verbinden offensichtlich alle noch anzufahrenden Orte $\nu', \nu \in V^{Orte}_{\Phi,\mu}$, wobei jeder Ort $\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}$ genau eine eingehende und maximal eine ausgehende Kante besitzt und der Ort der Fahrzeugposition μ nur eine ausgehende Kante hat. Die Fahrzeit der Route beträgt somit $\sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} c^*_\nu$. Für jede dieser Kanten $(\nu', \nu)^*$ gilt, dass die Fahrzeit c^*_ν nicht kürzer als die minimale Fahrzeit zum Erreichen des Ortes von allen anderen Orten $\tilde{\nu} \in V^{Orte}_{\Phi,\mu}\{\nu\}$ sein kann, wobei zwischen Pickup und Delivery von Aufträgen i mit Status $\varphi(i) = 0$ und $\varphi(i) = 1$ unterschieden werden muss:

$$c^*_\nu \geq \begin{cases} \min\left\{c_{\mu,p_i}, \min\limits_{\varphi(j)=0}\{c_{p_j,p_i}\}, \min\limits_{\varphi(j)\leq 1}\{c_{d_j,p_i}\}\right\} & \forall\ \nu' = p_i \in P \\ \min\left\{\min\limits_{\varphi(j)=0}\{c_{p_j,d_i}\}, \min\limits_{\varphi(j)\leq 1}\{c_{d_j,d_i}\}\right\} & \forall\ \nu' = d_i \in D \wedge \varphi(i) = 0 \\ \min\left\{c_{\mu,d_i}, \min\limits_{\varphi(j)=0}\{c_{p_j,d_i}\}, \min\limits_{\varphi(j)\leq 1}\{c_{d_j,d_i}\}\right\} & \forall\ \nu' = d_i \in D \wedge \varphi(i) = 1 \end{cases}$$

Somit gilt dies auch für die Summe:

$$\begin{aligned} \sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} c^*_\nu &\geq \sum_{i=1}^{n} \overrightarrow{C}_i^+((\Phi, \mu)^T, (\Phi^*_{2n}, \mu^*_{2n})^T) \\ &= \overrightarrow{C}^+((\Phi, \mu)^T, {}_{2n}V^Z_{n,k^*}) \end{aligned}$$

wobei zu beachten ist, dass für einen Auftrag i mit $\varphi(i) = 0$ die Summe der minimalen Fahrzeiten zu zwei Orten in $\overrightarrow{C}_i^+((\Phi, \mu)^T, (\Phi^*_{2n}, \mu^*_{2n})^T)$ enthalten sind, während die Kanten zu diesen Orten einzeln in die Summe $\sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} c^*_\nu$ eingehen. Die Minimalzeitensumme kann also die tatsächlichen Kosten nicht überschätzen.

Um das schnellstmögliche Auffinden der Lösung garantieren zu können, muss die Dreiecksungleichung für die Abschätzung gelten. Es ist zu zeigen, dass für jeden Zustand $(\Phi,\mu)^T$ und jeden von $(\Phi,\mu)^T$ erreichbaren Zustand $(\Phi',\mu')^T$ gilt:

$$\overrightarrow{C}^+((\Phi,\mu)^T,\{(\Phi',\mu')^T\})+\overrightarrow{C}^+((\Phi',\mu')^T,{}_{2n}V^Z_{n,k^*})\geq\overrightarrow{C}^+((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})$$

Laut Definition der Minimalzeitensumme muss also gelten:

$$\sum_{i=1}^{n}\overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi',\mu')^T)+\sum_{i=1}^{n}\overrightarrow{C}_i^+((\Phi',\mu')^T,(\Phi^*_{2n},\mu^*_{2n}))\geq\sum_{i=1}^{n}\overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi^*_{2n},\mu^*_{2n}))$$

Wegen der Struktur der Minimalzeitensumme, in die jeder noch anzufahrende Ort mit minimaler Fahrzeit von allen anderen noch anzufahrenden Orten eingeht, werden sich die Summanden für die Zustände $(\Phi,\mu)^T$ und $(\Phi',\mu')^T$ in der Höhe unterscheiden. Für jeden noch zu bedienenden Auftrag i gilt jedoch:

$$\overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi',\mu')^T)+\overrightarrow{C}_i^+((\Phi',\mu')^T,(\Phi^*_{2n},\mu^*_{2n})^T)\geq\overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi^*_{2n},\mu^*_{2n})^T)$$

da sich die Anzahl der für das Minimum in Frage kommenden Strecken verringert. Dies wird anhand einer Fallunterscheidung aus der Definition von $\overrightarrow{C}_i^+((\Phi,\mu)^T,\{(\Phi^*,\mu^*)^T\})$ gezeigt; hier wird diese Eigenschaft für den zweiten Fall mit Status $\varphi(i)=0$ in Zustand $(\Phi,\mu)^T$ und Status $\varphi'(i)=1$ in Zustand $(\Phi',\mu')^T$ bewiesen. Der Beweis der anderen Fälle verläuft analog. Das Fahrzeug habe im Zustand $(\Phi',\mu')^T$ den Standort μ'. Dann gilt:

$$\begin{aligned}
&\overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi',\mu')^T)+\overrightarrow{C}_i^+((\Phi',\mu')^T,(\Phi^*_{2n},\mu^*_{2n})^T)\\
&\quad=\min_{\substack{0=\varphi(j)<\varphi'(j)\\ \varphi(l)<\varphi'(l)=2}}\{c_{\mu p_i},c_{p_jp_i},c_{d_lp_i}\}+\min_{\substack{\varphi'(j)=0\\ \varphi'(l)<1}}\{c_{\mu' d_i},c_{p_jd_i},c_{d_ld_i}\}\\
&\quad\geq\min_{\substack{\varphi(j)=0\\ \varphi(l)<2}}\{c_{\mu p_i},c_{p_jp_i},c_{d_lp_i}\}+\min_{\substack{\varphi(j)=0\\ \varphi(l)<1}}\{c_{p_jd_i},c_{d_ld_i}\}\\
&\quad=\overrightarrow{C}_i^+((\Phi,\mu)^T,(\Phi^*_{2n},\mu^*_{2n})^T)
\end{aligned}$$

Die beiden oberen Zeilen enthalten jeweils zwei zu minimierende Ausdrücke, die sich nur in der Menge der zu minimierenden Strecken unterscheiden. Dabei gilt, dass

$$\begin{aligned}
\{p_j\in P\mid 0=\varphi(j)<\varphi'(j)\}&\subseteq\{p_j\in P\mid\varphi(j)=0\}\\
\{d_l\in D\mid\varphi(l)<\varphi'(l)=2\}&\subseteq\{d_l\in D\mid\varphi'(j)<2\}\\
\{p_j\in P\mid\varphi'(j)=0\}&\subseteq\{p_j\in P\mid\varphi(j)=0\}\\
\{d_l\in D\mid\varphi'(l)<1\}&\subseteq\{d_l\in D\mid\varphi(j)<1\}
\end{aligned}$$

Die letzten beiden Teilmengen-Beziehungen beruhen darauf, dass der Status $\varphi'(i)$ eines Auftrags i in $(\Phi',\mu')^T$ immer größer oder gleich dem Status $\varphi(i)$ dieses Auftrags in Zustand $(\Phi,\mu)^T$ ist. Weiterhin ist zu beachten, dass die Fahrzeugposition μ' in Zustand $(\Phi,\mu)^T$ zu den noch anzufahrenden Orten zählt und daher gilt, dass

$$\mu'\in\{p_j\in P\mid\varphi(j)=0\}\cup\{d_l\in D\mid\varphi(j)<1\}$$

Die Menge der in der ersten Zeile zu minimierenden Werte ist daher für beide zu minimierende Ausdrücke jeweils eine Teilmenge der in der zweiten Zeile zu minimierenden Strecken, so dass beide Minima der ersten Zeile jeweils größer oder gleich den entsprechenden Minima der nächsten Zeile sind. Die Beweise der anderen Fälle verlaufen analog, indem man Teilmengen-Beziehungen verdeutlicht, und seien hier nicht aufgeführt.

Somit gilt für jeden Auftrag i:

$$\overrightarrow{C_i}^+((\Phi,\mu)^T,(\Phi',\mu')^T) + \overrightarrow{C_i}^+((\Phi',\mu')^T,(\Phi^*_{2n},\mu^*_{2n})^T) \geq \overrightarrow{C_i}^+((\Phi,\mu)^T,(\Phi^*_{2n},\mu^*_{2n})^T)$$

Damit gilt auch für die Summe:

$$\begin{aligned}
\overrightarrow{C}^+((\Phi,\mu)^T,\{(\Phi',\mu')^T\}) \;+\; & \overrightarrow{C}^+((\Phi',\mu')^T,{}_{2n}V^Z_{n,k^*}) \\
&= \sum_{i=1}^{n} \overrightarrow{C_i}^+((\Phi,\mu)^T,(\Phi',\mu')^T) + \sum_{i=1}^{n} \overrightarrow{C_i}^+((\Phi',\mu')^T,(\Phi^*_{2n},\mu^*_{2n})^T) \\
&\geq \sum_{i=1}^{n} \overrightarrow{C_i}^+((\Phi,\mu)^T,(\Phi^*_{2n},\mu^*_{2n})^T) \\
&= \overrightarrow{C}^+((\Phi',\mu')^T,{}_{2n}V^Z_{n,k^*})
\end{aligned}$$

Somit erfüllt die Minimalzeitensumme $\overrightarrow{C}^+$ die geforderte Dreiecksungleichung.

□

Die Minimalzeitensumme $\overrightarrow{C}^+$ kann also prinzipiell als Schätzfunktion für den A*-Algorithmus verwendet werden. Da $\overrightarrow{C}^+$ jedoch bei geographisch nicht gleichverteilten noch anzufahrenden Orten auch erheblich von der tatsächlich noch verbleibenden Fahrzeit abweichen kann, muss in Testläufen festgestellt werden, ob tatsächlich bessere Ergebnisse erreicht werden.

3.4.1.1 Ergebnisse der Testläufe

Für diese Testläufe wurde die Anzahl der zu erzeugenden Knoten wieder mit einer Million nach oben beschränkt, ebenso wurde die Testumgebung wieder ereignisdiskret eingestellt. Als Vergleichswerte werden die Ergebnisse der Testläufe bei Verwendung des Zustandsgraphen und der Schätzfunktion „Maximalauftragszeit“ aus Abschnitt 3.3 herangezogen.

Zum besseren Verständnis wird die Struktur der Ergebnistabellen beibehalten, d. h. es werden die Anzahl der bearbeiteten Aufträge und der gelösten Instanzen, die Anzahl der erzeugten, unzulässigen, expandierten und Duplikat-Knoten sowie die Antwortzeiten sowohl als absolute Werte als auch im Vergleich zu den Ergebnissen aus Abschnitt 3.3 wiedergegeben:

Tabelle 3.12 zeigt, dass bei der Anzahl durchschnittlich bearbeiteter Aufträge nur eine leichte Steigerung von 0,05% bis 2,23% gegenüber den Testläufen bei Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ erzielt werden konnte. Die Anzahl der gelösten Testinstanzen konnte nicht verbessert werden.

Problemgruppe		Aufträge		gelöste Instanzen	
Name	Aufträge	absolut	Δ	absolut	Δ
100	51,86	33,46	0,05%	28	0,00%
200	103,08	44,50	0,98%	14	0,00%
400	205,66	66,34	1,27%	8	0,00%
600	308,97	74,65	1,45%	4	0,00%
800	411,75	83,02	1,41%	0	-
1000	514,98	94,05	2,23%	0	-

Tabelle 3.12: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{+}$ und des Zustandsgraphen G^{Z}_{2n,k^*} im Vergleich mit den Antwortzeiten bei Verwendung von $\vec{C}^{\triangleright}$ auf G^{Z}_{2n,k^*}

Problem-	♯ Knoten absolut			
gruppe	erzeugt	unzulässig	expandiert	Duplikate
100	385.688,20	171.560,91	68.801,05	132.987,04
200	642.034,02	293.064,50	121.395,67	203.634,67
400	728.949,71	367.256,19	148.783,15	182.359,93
600	811.936,97	422.693,08	168.763,93	186.883,10
800	822.759,42	445.273,90	176.960,24	170.623,90
1000	840.270,62	431.674,50	184.526,79	180.783,26

Tabelle 3.13: Die Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{+}$ und des Zustandsgraphen G^{Z}_{2n,k^*}

Die Tabellen 3.13 und 3.14 zeigen die Anzahl der erzeugten, unzulässigen, expandierten und Duplikat-Knoten, wobei Tabelle 3.13 der Vollständigkeit halber jeweils die absolute Anzahl Knoten angibt und 3.14 die für den Vergleich benötigte Anzahl der Knoten bei Beschränkung der Aufträge auf die in den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ bearbeiteten Aufträge sowohl absolut als auch relativ zur Anzahl Knoten des Testlaufs mit $\overrightarrow{C}^{\triangleright}$ enthält. Die Tabellen zeigen, dass die Anzahl der Knoten in allen Bereichen leicht reduziert werden konnte.

Diese Reduktion sollte sich auch in den Antwortzeiten zeigen, die in Tabelle 3.15 dargestellt werden. Bei den durchschnittlichen Antwortzeiten je Problemgruppe kann tatsächlich eine Reduktion von 3,57% - 17,01% beobachtet werden, in den durchschnittlich maximalen Antwortzeiten sind es sogar 8,17% - 26,27%. Betrachtet man die global maximale Antwortzeit jeder Problemklasse, so wird in Problemgruppe „1000“ die beachtliche Reduktion von 36,43% festgestellt. Allerdings zeigen die beiden Problemgruppen „100“ und „200“ eine Zunahme der global maximalen Antwortzeit um 2,07% bzw. sogar 18,97%. Bei genauerer Betrachtung stellt sich heraus, dass von den 56 Testinstanzen der Problemgruppe „100“ 11 Instanzen eine um durchschnittlich 1,24 Sekunden höhere maximale Antwortzeit haben, von denen eine die ausschlaggebende 18,82 Sekunden höhere maximale Antwortzeit erreicht. In der Problemgruppe

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	380.068,41	-6,38%	168.058,54	-7,57%	68.008,71	-6,46%	131.816,88	-4,85%
200	594.816,65	-9,36%	276.752,02	-10,89%	113.666,52	-9,36%	184.367,88	-9,76%
400	653.634,12	-10,41%	340.986,49	-10,61%	136.660,02	-9,52%	151.825,47	-14,13%
600	738.423,32	-9,62%	395.591,58	-9,63%	156.677,95	-8,46%	160.234,55	-13,96%
800	769.352,22	-7,94%	425.148,56	-7,07%	167.278,36	-7,11%	152.152,76	-14,00%
1000	719.988,47	-11,51%	394.517,02	-11,01%	164.528,05	-9,70%	128.671,33	-20,25%

Tabelle 3.14: Die Anzahl Knoten bei Verwendung der Schätzfunktion $\vec{C}^{+}$ im Vergleich zur Verwendung von $\vec{C}^{\triangleright}$, jeweils auf dem Zustandsgraphen G^{Z}_{2n,k^*}

Problem-gruppe	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	134,15	134,15	2,07%	9,36	9,36	-9,53%	0,33	0,33	-5,94%
200	69,14	69,14	18,97%	9,38	8,58	-8,17%	0,29	0,25	-3,57%
400	128,64	46,06	-35,96%	9,15	5,19	-26,27%	0,19	0,12	-17,01%
600	95,37	95,37	-21,26%	10,89	8,57	-14,82%	0,20	0,16	-10,37%
800	80,97	80,97	-35,75%	8,47	6,62	-17,69%	0,15	0,12	-11,24%
1000	132,92	31,04	-36,43%	10,10	3,72	-21,12%	0,16	0,07	-13,97%

Tabelle 3.15: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{+}$ und des Zustandsgraphen G^{Z}_{2n,k^*}, im Vergleich mit den Antwortzeiten bei Verwendung von $\vec{C}^{\triangleright}$ auf G^{Z}_{2n,k^*}

„200“ zeigt sich ein ähnliches Bild: 15 der 60 Testinstanzen zeigen eine um durchschnittlich 1,30 Sekunden höhere maximale Antwortzeit, eine davon erreicht eine Zunahme von 23,52 Sekunden.

Die Testläufe haben somit gezeigt, dass die Schätzfunktion $\overrightarrow{C}^{+}$ in vielen Instanzen hinsichtlich der Anzahl Knoten und der benötigten Antwortzeiten bessere Ergebnisse liefert als die Schätzfunktion $\overrightarrow{C}^{\triangleright}$. Da jedoch noch keine eindeutige Überlegenheit für alle Testinstanzen gegeben ist, wird im folgenden versucht, für beide Schätzfunktionen Varianten zu entwickeln, die noch mehr Informationen des Entscheidungsproblems in die Berechnung einbeziehen und somit die Abweichung von der berechneten geschätzten Fahrzeit und der tatsächlich verbleibenden Fahrzeit zu verringern und somit eine weitere Reduktion der Antwortzeiten zu erreichen.

3.4.2 Varianten der Schätzfunktionen unter Beachtung der Servicezeiten

Zur Beachtung der Servicezeiten in einer Variante einer Schätzfunktion sind zwei Ansätze denkbar. Die intuitive Möglichkeit besteht darin, die bekannten Servicezeiten s_ν aller Orte $\nu \in V^{Orte}_{\Phi,\mu}$ mit in die Berechnung der verbleibenden Fahrzeit einzubeziehen. Dies hat bei den Schätzfunktionen „Minimalzeitensumme“ $\overrightarrow{C}^{+}$ und „Maximalauftragszeit“ $\overrightarrow{C}^{\triangleright}$ unterschiedliche Auswirkungen: Da bei der Schätzfunktion „Minimalzeitensumme“ die minimalen Fahrzeiten

zu allen Orten der Menge $\in V_{\Phi,\mu}^{Orte}$ addiert werden, führt die Einbeziehung der Servicezeiten s_ν in die berechnete geschätzte Fahrzeit dazu, dass die Servicezeit s_ν für jeden Ort $\nu \in V_{\Phi,\mu}^{Orte}$ und damit für jeden Auftrag i berücksichtigt wird. Nutzt man die Schätzfunktion „Maximalauftragszeit“, so werden die Servicezeiten s_ν zwar für jeden Auftrag i in die Berechnungen einbezogen, der zurückgegebene Wert beinhaltet jedoch nur die Fahrzeit eines Auftrags i^* und damit auch nur dessen Servicezeiten $s_{p_{i^*}}$ (bei $\varphi(i^*) = 0$) bzw. $s_{d_{i^*}}$.

Da alle Aufträge bedient werden müssen, können auch alle Servicezeiten $s_\nu, \nu \in V_{\Phi,\mu}^{Orte}$ in die Schätzfunktion $\overrightarrow{C}^{\triangleright}$ eingehen. Für die Schätzfunktion $\overrightarrow{C}^{\triangleright}$ kommt also noch eine weitere Möglichkeit der Beachtung der Servicezeiten in Frage: die Summe der Servicezeiten aller noch anzufahrenden Orte wird zum Wert der Schätzfunktion $\overrightarrow{C}^{\triangleright}$, der längsten Fahrzeit eines einzelnen Auftrags, addiert. Dabei darf jede Servicezeit s_ν nur einmal einbezogen werden, d. h. die Servicezeiten $s_{p_{i^*}}$ bzw. $s_{d_{i^*}}$, die evtl. in der Fahrzeit des Auftrags i^* mit maximaler Einzelfahrtzeit enthalten sind, dürfen in der Summe aller noch verbleibenden Servicezeiten nicht enthalten sein.

Im folgenden werden für alle drei Konstellationen die Varianten der Schätzfunktionen angegeben sowie die Ergebnisse der Testläufe interpretiert.

3.4.2.1 Variante der Minimalzeitensumme mit Servicezeiten

In der Schätzfunktion „Minimalzeitensumme“ werden die minimalen Fahrzeiten zum Erreichen jedes noch anzufahrenden Ortes $\nu \in V_{\Phi,\mu}^{Orte}$ addiert. Die tatsächlich verbleibende Fahrzeit kann damit nicht überschätzt werden. Zusätzlich zu den Fahrzeiten $c_{\nu,\nu'}$ sind jedoch auch die Servicezeiten s_ν der Aufträge an den noch anzufahrenden Orten $\nu \in V_{\Phi,\mu}^{Orte} \setminus \{\mu\}$ bekannt. Da die Aufträge bedient werden müssen, müssen auch alle Servicezeiten $s_\nu, \nu \in V_{\Phi,\mu}^{Orte} \setminus \{\mu\}$ eingehalten werden, so dass ihre Berücksichtigung nicht zu einer Überschätzung der Routenlänge führen kann. Die Berücksichtigung der Servicezeiten kann allerdings dazu führen, dass die berechnete geschätzte Fahrzeit genauer, d. h. die Differenz zwischen der tatsächlich verbleibenden Fahrzeit und dem berechneten geschätzten Wert reduziert wird. Insbesondere wird die Addition aller Servicezeiten $s_\nu, \nu \in V_{\Phi,\mu}^{Orte} \setminus \{\mu\}$ bei etwa gleich langen Servicezeiten s_ν für alle Orte $\nu \in V_{\Phi,\mu}^{Orte} \setminus \{\mu\}$ bewirken, dass die Knoten auf „höheren“ Stufen k des Zustandsgraphen G_{n,k^*}^Z tendenziell bevorzugt werden, weil mit jeder Stufe $k > k^*$ ein Ort weniger in der Menge der noch anzufahrenden Orte $V_{\Phi,\mu}^{Orte}$ enthalten ist und damit auch die Summe der Servicezeiten bei steigendem k sinkt.

Die Schätzfunktion „*Minimalzeitensumme mit Servicezeiten*“ $\overrightarrow{C}^{+S} : V_{n,k^*}^Z \times {}_{2n}V_{n,k^*}^Z \to \mathbb{R}^+$ soll die geschätzte minimale Fahrzeit von einem Knoten $(\Phi,\mu)^T \in V_{n,k^*}^Z$ zu einem der Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V_{n,k^*}^Z$ inklusive aller Servicezeiten zurückgeben. Ausgehend vom Zustandsvektor $(\Phi,\mu)^T$ wird daher nun für jeden Auftrag i die minimale Fahrzeit zu einem Zustandsvektor $(\Phi^*,\mu^*)^T \in {}_{2n}V_{n,k^*}^Z$ berechnet als:

$$\overrightarrow{C}_i^{+S}\left(\binom{\Phi}{\mu},\binom{\Phi^*}{\mu^*}\right) = \begin{cases} 0, & \varphi(i)=\varphi^*(i) \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j),\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\} + s_{p_i}, & \varphi(i)=0 \wedge \varphi^*(i)=1 \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j),\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu d_i}, c_{p_j d_i}, c_{d_l d_i}\} + s_{d_i}, & \varphi(i)=1 \wedge \varphi^*(i)=2 \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j)\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\} + s_{p_i} + & \\ \quad + \min\limits_{\substack{0=\varphi(j)<\varphi^*(j)\\ \varphi(l)<\varphi^*(l)=2}} \{c_{p_j d_i}, c_{d_l d_i}\} + s_{d_i}, & \varphi(i)=0 \wedge \varphi^*(i)=2 \end{cases}$$

Analog zur Definition der Minimalzeitensumme gibt nun die Summe

$$\sum_{i=1}^{n} \overrightarrow{C}_i^{+S}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

die geschätzte minimale Fahrzeit von $(\Phi,\mu)^T$ zum Zielknoten $(\Phi^*,\mu^*)^T$ inklusive der Servicezeiten an. Die „Minimalzeitensumme mit Servicezeiten“ als Minimum der zu jedem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ geschätzten minimalen Fahr- und Servicezeit $\overrightarrow{C}^{+S}((\Phi,\mu)^T,\{(\Phi^*,\mu^*)^T\})$ wird definiert als:

$$\overrightarrow{C}^{+S}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) = \min_{(\Phi^*,\mu^*)^T\in{}_{2n}V^Z_{n,k^*}} \sum_{i=1}^{n} \overrightarrow{C}_i^{+S}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

Mit dieser Definition können nun die Eigenschaften des A*-Algorithmus bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+S}$ angegeben werden:

Satz 3.7

Der A-Algorithmus findet bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+S}$, „Minimalzeitensumme mit Servicezeiten“, schnellstmöglich einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}.*

Beweis: Zu zeigen ist, dass der Wert $\overrightarrow{C}^{+S}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})$ trotz der Hinzunahme der Servicezeiten die tatsächlich verbleibende Fahrzeit nicht überschätzt und die Dreiecksungleichung weiterhin gilt.

Die Minimalzeitensumme mit Servicezeiten $\overrightarrow{C}^{+S}$ lässt sich mit Hilfe der Minimalzeitensumme $\overrightarrow{C}^{+}$ darstellen. Da für jeden Auftrag i gilt:

$$\overrightarrow{C}_i^{+S}((\Phi,\mu)^T,(\Phi',\mu')^T) = \overrightarrow{C}_i^{+}((\Phi,\mu)^T,(\Phi',\mu')^T) + \begin{cases} 0, & \varphi(i)=\varphi'(i) \\ s_{p_i}, & \varphi(i)=0 \wedge \varphi'(i)=1 \\ s_{d_i}, & \varphi(i)=1 \wedge \varphi'(i)=2 \\ s_{p_i}+s_{d_i}, & \varphi(i)=0 \wedge \varphi'(i)=2 \end{cases}$$

kann $\overrightarrow{C}^{+S}$ geschrieben werden als

$$\overrightarrow{C}^{+S}((\Phi,\mu)^T,\{(\Phi',\mu')^T\}) = \overrightarrow{C}^{+}((\Phi,\mu)^T,\{(\Phi',\mu')^T\}) + \sum_{\substack{\nu\in(V^{Orte}_{\Phi,\mu}\setminus\{\mu\})\setminus\\ (V^{Orte}_{\Phi',\mu'}\setminus\{\mu'\})}} s_\nu$$

Insbesondere gilt damit für den mit Hilfe der Minimalzeitensumme mit Servicezeiten berechneten geschätzten Wert der Fahrzeit von $(\Phi,\mu)^T$ zu einer der Senken des Zustandsgraphen:

$$\overrightarrow{C}^{+S}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \overrightarrow{C}^{+}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) + \sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu$$

Im Beweis zu Satz 3.6 wurde gezeigt, dass für die Minimalzeitensumme $\overrightarrow{C}^{+}$ im Vergleich zur Summe der Fahrzeiten c^*_ν einer fahrzeitminimalen Route von der Fahrzeugposition μ durch alle noch anzufahrenden Orte $\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}$ gilt:

$$\begin{aligned}\sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} c^*_\nu &\geq \sum_{i=1}^{n} \overrightarrow{C}^{+}_i((\Phi,\mu)^T, (\Phi^*_{2n},\mu^*_{2n})^T)\\ &= \overrightarrow{C}^{+}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})\end{aligned}$$

Auf der fahrzeitminimalen Route müssen jedoch auch die Servicezeiten an jedem Ort eingehalten werden, so dass sich die Fahrzeit des Fahrzeugs dementsprechend verlängert. Daher ergibt sich für die tatsächlich verbleibende Fahrzeit der Route mindestens der Wert $\sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}}(c^*_\nu + s_\nu)$. Es gilt:

$$\begin{aligned}\sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} (c^*_\nu + s_\nu) &= \sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} c^*_\nu + \sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu\\ &\geq \overrightarrow{C}^{+}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) + \sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu\\ &= \overrightarrow{C}^{+S}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})\end{aligned}$$

Die tatsächlich verbleibende Fahrzeit kann somit von $\overrightarrow{C}^{+S}$ nicht überschätzt werden.

Bleibt noch zu zeigen, dass $\overrightarrow{C}^{+S}$ die Dreiecksungleichung

$$\overrightarrow{C}^{+S}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \overrightarrow{C}^{+S}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) \geq \overrightarrow{C}^{+S}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*})$$

erfüllt. Dazu wird ausgenutzt, dass die Dreiecksungleichung für $\overrightarrow{C}^{+}$ gilt:

$$\begin{aligned}&\overrightarrow{C}^{+S}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \overrightarrow{C}^{+S}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*})\\ &= \overrightarrow{C}^{+}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \sum_{\substack{\nu\in(V^{Orte}_{\Phi,\mu}\setminus\{\mu\})\setminus\\ (V^{Orte}_{\Phi',\mu'}\setminus\{\mu'\})}} s_\nu + \overrightarrow{C}^{+}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) + \sum_{\nu\in V^{Orte}_{\Phi',\mu'}\setminus\{\mu'\}} s_\nu\\ &= \overrightarrow{C}^{+}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \overrightarrow{C}^{+}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) + \sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu\\ &\geq \sum_{i=1}^{n} \overrightarrow{C}^{+}_i((\Phi,\mu)^T, \{(\Phi^*_{2n},\mu^*_{2n})^T\}) + \sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu\\ &= \overrightarrow{C}^{+S}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*})\end{aligned}$$

Problem-		Aufträge		gelöste Instanzen	
gruppe	**Aufträge**	**absolut**	**Δ**	**absolut**	**Δ**
100	51,86	34,71	3,79%	30	7,14%
200	103,08	49,43	12,18%	19	35,71%
400	205,66	75,86	15,81%	11	37,50%
600	308,97	87,92	19,48%	6	50,00%
800	411,75	97,56	19,17%	1	- [5]
1000	514,98	106,38	15,63%	0	-

Tabelle 3.16: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{+S}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

□

Die Minimalzeitensumme mit Servicezeiten kann also im A*-Algorithmus eingesetzt werden, ohne die Optimalität der Lösung oder das schnellstmögliche Finden einer Lösung zu gefährden.

Ergebnisse der Testläufe

Für diese Testläufe werden alle in Abschnitt 3.4.1 beschriebenen Einstellungen übernommen. Als Vergleichswerte werden weiterhin die Ergebnisse der Testläufe bei Verwendung des Zustandsgraphen und der Schätzfunktion „Maximalauftragszeit“ aus Abschnitt 3.4.1 herangezogen.

Tabelle 3.16 zeigt die Anzahl der bearbeiteten Aufträge und gelösten Instanzen. Hier konnten deutliche Steigerungen gegenüber den Testläufen unter Verwendung von $\overrightarrow{C}^{\triangleright}$ erzielt werden: die Anzahl durchschnittlich bearbeiteter Aufträge konnte um 3,79% bis 19,48% gesteigert werden, während die Anzahl der bis zum Erreichen von einer Million erzeugten Knoten gelösten Instanzen sogar um bis zu 50% verbessert werden konnte. In der Problemgruppe „800“ konnte sogar erstmals eine Instanz gelöst werden. Auffällig sind die im Vergleich zu den anderen Problemgruppen geringen Verbesserungen der Problemgruppe „100“: hier konnten nur 3,79% mehr Aufträge bearbeitet und nur 7,14% mehr Instanzen gelöst werden. Bei genauerer Betrachtung der Ergebnisse stellt man fest, dass alle Instanzen mit kurzem Planungshorizont, der nur ca. 10 Aufträge pro Fahrzeug zulässt, vollständig gelöst werden, während die Instanzen mit langem Planungshorizont und bis zu 30 möglichen Aufträgen pro Fahrzeug nicht gelöst werden. Eine Steigerung der Anzahl Aufträge ist daher in dieser Problemgruppe nur in den Instanzen mit langem Planungshorizont möglich, was zu den vergleichsweise niedrigen prozentualen Steigerungen führt. In diesen Testläufen mit der Schätzfunktion $\overrightarrow{C}^{+S}$ konnten jedoch deutlich bessere Ergebnisse erzielt werden als in den Testläufen mit der Funktion $\overrightarrow{C}^{+}$.

[5] Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Problem-gruppe	♯ Knoten absolut			
	erzeugt	unzulässig	expandiert	Duplikate
100	361.809,18	134.907,66	56.981,73	142.916,46
200	617.022,97	248.553,03	109.055,10	209.988,55
400	731.315,66	345.065,46	144.467,76	183.260,78
600	783.792,12	379.155,45	161.344,37	178.614,52
800	827.528,61	425.828,68	177.359,29	159.607,93
1000	851.191,03	412.334,00	187.064,84	183.039,95

Tabelle 3.17: bei Verwendung der Schätzfunktion $\vec{C}^{+S}$ im Vergleich zur Verwendung von $\vec{C}^{\triangleright}$

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl	Δ	♯ Vgl	Δ	♯ Vgl	Δ	♯ Vgl	Δ
100	248.989,95	-38,67%	98.416,23	-45,88%	42.095,68	-42,10%	94.892,39	-31,50%
200	422.308,78	-35,65%	187.459,97	-39,64%	78.198,20	-37,64%	132.077,85	-35,36%
400	495.815,97	-32,04%	259.867,53	-31,88%	103.571,56	-31,43%	103.968,73	-41,20%
600	578.697,55	-29,17%	303.559,92	-30,65%	121.061,28	-29,27%	120.361,42	-35,37%
800	607.576,20	-27,30%	338.657,08	-25,97%	132.615,05	-26,36%	104.717,17	-40,81%
1000	578.963,40	-28,84%	318.482,19	-28,16%	133.689,88	-26,62%	90.026,45	-44,20%

Tabelle 3.18: Die Anzahl Knoten bei Verwendung der Schätzfunktion $\vec{C}^{+S}$ im Vergleich zur Verwendung von $\vec{C}^{\triangleright}$

In den Tabellen 3.17 und 3.18 wird die Anzahl der erzeugten, unzulässigen, expandierten und Duplikat-Knoten dargestellt: Tabelle 3.17 gibt die absolute Anzahl Knoten und Tabelle 3.18 die für den Vergleich benötigte Anzahl der Knoten an, bei Beschränkung der Aufträge auf die in den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ bearbeiteten Aufträge. Die Vergleichswerte werden sowohl absolut als auch relativ zur Anzahl Knoten des Testlaufs mit $\overrightarrow{C}^{\triangleright}$ dargestellt. In Tabelle 3.17 zeigt die Anzahl von nur 361.809, 18 durchschnittlich in Problemgruppe „100“ erzeugten Knoten, dass in den gelösten Instanzen deutlich weniger als eine Million Knoten erzeugt werden. Tabelle 3.18 zeigt eine deutliche Reduktion der Anzahl Knoten von 25,97% bis 45,88% in allen Bereichen. Auch hier werden deutliche Verbesserungen gegenüber den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{+}$ erzielt, die eine maximale Reduktion von 11,51% der erzeugten Knoten aufweisen.

In Tabelle 3.19 werden die Antwortzeiten dargestellt. Die durchschnittlichen Antwortzeiten je Problemgruppe konnten um beachtliche 32,18% bis 41,29% reduziert werden. Bei den durchschnittlich maximalen Antwortzeiten sind es sogar 34,52% bis 47,03%. In den Ergebnissen für die global maximale Antwortzeit wird die beachtliche Reduktion von 53,55% für die Problemgruppe „800“ erzielt. Allerdings hat sich die global maximale Antwortzeit der Problemgruppe „200“ um 7,79% gegenüber den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ erhöht. Die Begründung hierfür liegt in der Tatsache, dass in 20 der 60 Testinstanzen eine um durch-

Problem-gruppe	global max. Antwortzeit sec	Vgl	Δ	Ø max. Antwortzeit sec	Vgl	Δ	Ø Antwortzeit sec	Vgl	Δ
100	193,35	123,66	-5,91%	16,63	6,63	-35,88%	0,56	0,23	-34,26%
200	129,80	62,64	7,79%	14,84	6,12	-34,52%	0,38	0,18	-32,18%
400	131,00	47,07	-34,56%	10,68	3,97	-43,64%	0,19	0,09	-39,33%
600	80,59	56,83	-53,07%	10,26	6,43	-36,09%	0,16	0,12	-33,47%
800	158,28	58,55	-53,55%	9,27	4,26	-47,03%	0,14	0,08	-41,29%
1000	106,26	24,69	-49,44%	9,98	2,64	-44,07%	0,14	0,05	-38,56%

Tabelle 3.19: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{+S}$ im Vergleich zur Verwendung von $\vec{C}^{\triangleright}$

schnittlich 0,71 Sekunden höhere maximale Antwortzeit erreicht wurde, in genau einer Instanz war die maximale Antwortzeit mit 9,56 Sekunden deutlich länger als im Testlauf dieser Instanz mit der Funktion $\overrightarrow{C}^{\triangleright}$. Auch hinsichtlich der Antwortzeiten wurden in den Testläufe mit der Schätzfunktion $\overrightarrow{C}^{+S}$ erheblich bessere Ergebnisse als in den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{+}$ erzielt.

Die Testläufe zeigen, dass die Verwendung der Schätzfunktion $\overrightarrow{C}^{+S}$ zu besseren Ergebnissen führt als die Verwendung der Schätzfunktion $\overrightarrow{C}^{+}$. In fast allen Fällen werden jetzt auch die Ergebnisse der Testläufe unter Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ übertroffen.

In den folgenden Abschnitten wird nun untersucht, inwieweit Varianten der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ unter Verwendung der Servicezeiten die Rechenzeiten positiv beeinflussen können.

3.4.2.2 Variante der Maximalauftragszeit mit Einzel-Servicezeit

In der Schätzfunktion „Maximalauftragszeit“, $\overrightarrow{C}^{\triangleright}$, wird für jeden noch zu bearbeitenden Auftrag i einzeln betrachtet, wie lange das Fahrzeug für die alleinige Vollendung dieses Auftrags brauchen würde. Anschließend wird das Maximum dieser Zeiten $\overrightarrow{C_i}^{\triangleright}$ aller Aufträge als Abschätzung für die insgesamt noch verbleibende Fahrzeit genommen. Da die Servicezeiten s_ν, die bei Ankunft an einem Pickup- oder Delivery-Ort $\nu \in V^{Orte}_{\Phi,\mu}$ anfallen, bekannt sind und eingehalten werden müssen, ist es nur natürlich, sie in die Abschätzung der Fahrzeit mit aufzunehmen, um so eine genauere Abschätzung zu erhalten.

Die „*Maximalauftragszeit mit Einzel-Servicezeit*“ $\overrightarrow{C}^{\triangleright E} : V^Z_{n,k^*} \times {}_{2n}V^Z_{n,k^*} \to \mathbb{R}^+$ gibt eine geschätzte minimale Fahr- und Servicezeit von einem Knoten $(\Phi,\mu)^T \in V^Z_{n,k^*}$ zu einem der Zielknoten $(\Phi^*,\mu*)^T \in {}_{2n}V^Z_{n,k^*}$ zurück. Dazu wird die für einen einzelnen Auftrag i zum Erreichen eines Zustandsvektors $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ verbleibende Fahrzeit definiert als:

$$\overrightarrow{C}_i^{\triangleright E}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) = \begin{cases} 0 & \text{falls } \varphi(i)=\varphi^*(i) \\ c_{\mu,d_i}+s_{d_i} & \text{falls } \varphi(i)=1 \wedge \varphi^*(i)=2 \\ c_{\mu,p_i}+s_{p_i} & \text{falls } \varphi(i)=0 \wedge \varphi^*(i)=1 \\ c_{\mu,p_i}+s_{p_i}+c_{p_i,d_i}+s_{d_i} & \text{falls } \varphi(i)=0 \wedge \varphi^*(i)=2 \end{cases}$$

Der Wert $\overrightarrow{C}_i^{\triangleright E}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$ bildet so die noch verbleibende Fahr- und Servicezeit bei alleiniger Bedienung des Auftrags i ab. Mit

$$\max_{i=1}^{n} \overrightarrow{C}_i^{\triangleright E}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

wird somit eine Abschätzung für die Fahrzeit zu einem Zielknoten $(\Phi^*,\mu^*)^T$ definiert. Damit kann die Maximalauftragszeit mit Einzel-Servicezeit als geschätzte Fahrzeit zu einem beliebigen Zielknoten der Menge $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ als Minimum der geschätzten Fahrzeiten zu einem der Zielknoten angegeben werden:

$$\overrightarrow{C}^{\triangleright E}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) \quad = \quad \min_{(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}} \max_{i=1}^{n} \overrightarrow{C}_i^{\triangleright E}((\Phi,\mu)^T,(\Phi^*,\mu*)^T)$$

Die Schätzfunktion $\overrightarrow{C}^{\triangleright E}$ hat folgende Eigenschaften:

Satz 3.8

Der A-Algorithmus findet mit der Schätzfunktion „Maximalauftragszeit mit Einzel-Servicezeiten", $\overrightarrow{C}^{\triangleright E}$, einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}. Das schnellstmögliche Auffinden der Lösung ist nicht garantiert.*

Beweis: Der Beweis verläuft in großen Teilen analog zum Beweis von Satz 3.3. Allerdings wird hier der Zustandsgraph G^Z_{n,k^*} zugrunde gelegt, während in 3.3 der Statusvektorbaum G^S_{n,k^*} benutzt wurde.

Die Optimalität der Lösung des A*-Algorithmus hängt davon ab, ob der Wert der Schätzfunktion $\overrightarrow{C}^{\triangleright E}((\Phi,\mu)^T,{}_{2n}V^{Orte}_{\Phi,\mu})$ die tatsächlich noch anfallende Fahrzeit von einem Knoten $(\Phi,\mu)^T$ zu einem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ überschätzen kann. Für einen Zustandsvektor $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ gilt: $\Phi^* = (2,\ldots,2)^T$, so dass $\overrightarrow{C}_i^{\triangleright E}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$ dargestellt werden kann als:

$$\overrightarrow{C}_i^{\triangleright E}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) = \begin{cases} 0 & \text{falls } \varphi(i)=2 \\ c_{\mu,d_i} & \text{falls } \varphi(i)=1 \\ c_{\mu,p_i}+c_{p_i,d_i} & \text{falls } \varphi(i)=0 \end{cases}$$

Da sich die Zustandsvektoren $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ nur im Ort der Fahrzeugposition μ^* unterschieden, der jedoch nicht in $\overrightarrow{C}^{\triangleright E}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})$ eingeht, kann auch $\overrightarrow{C}^{\triangleright E}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})$ einfacher geschrieben werden:

$$\overrightarrow{C}^{\triangleright E}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) = \max_{i=1\ldots,n} \overrightarrow{C}_i^{\triangleright E}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

Man betrachte nun, analog zum Beweis von Satz 3.7, für den Zustandsvektor $(\Phi, \mu)^T$ den gewichteten Graphen $G^{Orte}_{\Phi,\mu}$ mit der Knotenmenge $V^{Orte}_{\Phi,\mu}$, bestehend aus dem Ort μ der aktuellen Fahrzeugposition und allen noch anzufahrenden Orten, und der Kantenmenge $E^{Orte}_{\Phi,\mu}$, die alle möglichen Reihenfolgen der Orte in einer Route beginnend im Ort μ abbildet. Das Gewicht $c^{Orte}_{\Phi,\mu}(\nu, \nu')$ einer Kante $(\nu, \nu') \in V^{Orte}_{\Phi,\mu}$ entspreche der Fahrzeit $c_{\nu,\nu'}$ von Ort ν zu Ort ν'. Analog zum Beweis von Satz 3.7 bezeichne c^*_ν die Fahrzeit der Kante $(\nu', \nu)^*$, die zwei Orte der fahrzeitminimalen Route $(\mu, \nu_1, \nu_2, \ldots, \nu_l)$ von der Fahrzeugposition μ durch alle noch anzufahrenden Orte $\nu_1, \ldots \nu_l \in V^{Orte}_{\Phi,\mu}$ verbindet. Die Fahrzeit der minimalen Route beträgt somit $\sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} c^*_\nu$.

Die Maximalauftragszeit mit Einzel-Servicezeit, $\overrightarrow{C}^{\triangleright E}((\Phi, \mu)^T, {}_{2n}V^Z_{n,k^*})$, bildet das Maximum der $\overrightarrow{C}_i^{\triangleright E}((\Phi, \mu)^T, (\Phi^*, \mu^*)^T)$ aller Aufträge i. Es sei i' ein Auftrag mit $\varphi(i') = 2$ und maximalem $\overrightarrow{C}_i^{\triangleright E}((\Phi, \mu)^T, (\Phi^*, \mu^*)^T)$. Dann gilt für die Fahr- und Servicezeiten unter Betrachtung der Teil-Routen $(\mu, \nu_1, \nu_2 \ldots, p_i)$, $(p_i, \ldots, d_i)$ und $(d_i, \ldots, \nu_l)$:

$$\begin{aligned}
\overrightarrow{C}^{\triangleright E}((\Phi, \mu)^T, {}_{2n}V^Z_{n,k^*}) &= c_{\mu,p_{i'}} + s_{p_{i'}} + c_{p_{i'},d_{i'}} + s_{d_{i'}} \\
&\leq c^*_{\nu_1} + s_{\nu_1} + c^*_{\nu_2} + s_{\nu_2} + \cdots + c^*_{p_{i'}} + s_{p_{i'}} + \cdots + c^*_{d_{i'}} s_{d_{i'}} \\
&\leq \sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} (c^*_\nu + s_\nu)
\end{aligned}$$

da die Gültigkeit der Dreiecksungleichung für die Fahrzeiten $c_{\nu,\nu'}$ vorausgesetzt wurde (vgl. Abschnitt 2.3). Diese Eigenschaft kann analog für Aufträge i' mit $\varphi(i') < 2$ nachgewiesen werden. Die Schätzfunktion $\overrightarrow{C}^{\triangleright E}$ kann also die verbleibende Fahr- und Servicezeit nicht überschätzen.

Die für das schnellstmögliche Finden einer Lösung benötigte Gültigkeit der Dreieckgsungleichung

$$\overrightarrow{C}^{\triangleright E}((\Phi, \mu)^T, \{(\Phi', \mu')^T\}) + \overrightarrow{C}^{\triangleright E}((\Phi', \mu')^T, {}_{2n}V^S_{n,k^*}) \geq \overrightarrow{C}^{\triangleright E}((\Phi, \mu)^T, {}_{2n}V^S_{n,k^*})$$

ist nicht erfüllt. Das Gegenbeispiel aus Abschnitt 3.2.3, dass die Gültigkeit der Dreicksungleichung für die Schätzfunktion $\overrightarrow{C}^{\triangleright}$ auf dem Statusvektorbaum G^S_{n,k^*} widerlegt, kann hier übernommen werden: man betrachte dazu den Fall, dass alle Servicezeiten Null sind. Da das Gegenbeispiel in Abschnitt 3.2.3 auf dem Graphen G^{Orte}_Φ basiert, der bei gleichem Statusvektor mit dem hier betrachteten Graphen $G^{Orte}_{\Phi,\mu}$ übereinstimmt, kann es auch hier verwendet werden.

□

Die Schätzfunktion $\overrightarrow{C}^{\triangleright E}$ kann also benutzt werden, ohne die Optimalität der Lösung zu gefährden. Das schnellstmögliche Finden einer Lösung kann zwar nicht garantiert werden, dennoch kann eine Verbesserung der Rechenzeiten erwartet werden.

Problem-		Aufträge		gelöste Instanzen	
gruppe	Aufträge	absolut	Δ	absolut	Δ
100	51,86	33,45	0,00%	28	0,00%
200	103,08	44,30	0,53%	14	0,00%
400	205,66	65,83	0,49%	8	0,00%
600	308,97	74,12	0,72%	4	0,00%
800	411,75	82,88	1,24%	0	-
1000	514,98	93,33	1,44%	0	-

Tabelle 3.20: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright E}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-	♯ Knoten absolut			
gruppe	erzeugt	unzulässig	expandiert	Duplikate
100	377.989,79	164.525,27	66.621,11	133.069,88
200	649.437,55	297.629,65	122.374,80	208.676,53
400	722.144,46	368.667,59	148.061,98	181.766,42
600	808.493,42	429.669,68	168.186,68	184.657,68
800	834.410,71	449.993,64	178.822,51	180.559,14
1000	821.185,24	436.802,09	182.111,59	168.880,45

Tabelle 3.21: bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright E}$ im Vergleich zur Verwendung von $\vec{C}^{\triangleright}$

Ergebnisse der Testläufe

Die Testläufe basieren auf den in Abschnitt 3.4.1 beschriebenen Einstellungen. Als Vergleichswerte dienen weiterhin die Ergebnisse der Testläufe aus Abschnitt 3.4.1 unter Verwendung des Zustandsgraphen und der Schätzfunktion $\vec{C}^{\triangleright}$.

Tabelle 3.20 zeigt die Anzahl der bearbeiteten Aufträge und gelösten Instanzen. Hier können nur schwache Verbesserungen festgestellt werden: in Problemgruppe „100“ werden keine zusätzlichen Aufträge bearbeitet, die größten Zuwächse verzeichnet die Problemgruppe „1000“ mit nur 1,44%.

Auch bei den erzeugten Knoten, die in den Tabellen 3.21 und 3.22 dargestellt werden, können keine wesentlichen Verbesserungen festgestellt werden, die Einsparungen bewegen sich zwischen 2,69% und 9,52% der Knoten.

Bei den Antwortzeiten in Tabelle 3.23 werden zum Teil gute Verbesserungen ersichtlich: während sich die durchschnittliche Antwortzeit um 4,90% bis 11,21% verringert, kann bei der durchschnittlichen maximalen Antwortzeit eine Reduktion um 5,34% bis 14,19% erzielt werden. Die global maximale Antwortzeit kann in der Problemgruppe „800“ um beachtliche 33,75% gesenkt werden, während sie in den Problemgruppen „200“ und „400“ leicht um 4,74% bzw. 2,70% ansteigt: In den 60 Testinstanzen der Problemruppe 200 wird die maximale Antwortzeit

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl	Δ	♯ Vgl	Δ	♯ Vgl	Δ	♯ Vgl	Δ
100	377.989,79	-6,89%	164.525,27	-9,52%	66.621,11	-8,37%	133.069,88	-3,95%
200	627.934,00	-4,31%	290.873,75	-6,35%	118.757,58	-5,30%	198.816,27	-2,69%
400	701.514,93	-3,85%	364.165,41	-4,53%	144.492,53	-4,34%	170.237,75	-3,71%
600	794.436,87	-2,76%	422.450,63	-3,49%	165.127,03	-3,53%	181.169,28	-2,72%
800	808.958,31	-3,20%	439.695,10	-3,89%	173.356,59	-3,73%	171.746,83	-2,92%
1000	782.987,17	-3,77%	421.490,19	-4,92%	174.569,74	-4,18%	155.232,72	-3,79%

Tabelle 3.22: Die Anzahl Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright E}$ im Vergleich zur Verwendung von $\vec{C}^{\triangleright}$

Problem-gruppe	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	120,56	120,56	-8,28%	9,11	9,11	-11,95%	0,31	0,31	-11,21%
200	86,90	60,87	4,74%	10,25	8,84	-5,34%	0,28	0,25	-4,90%
400	105,97	73,88	2,70%	7,89	6,18	-12,19%	0,16	0,13	-10,08%
600	92,31	92,31	-23,78%	9,24	9,23	-8,30%	0,17	0,17	-6,47%
800	83,49	83,49	-33,75%	7,11	6,90	-14,19%	0,13	0,12	-10,74%
1000	74,36	44,76	-8,34%	5,62	4,39	-6,92%	0,10	0,08	-5,47%

Tabelle 3.23: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright E}$ im Vergleich zur Verwendung von $\vec{C}^{\triangleright}$

durchschnittlich um 0,5 Sekunden gesenkt, in einer Testinstanz sogar um 21,2 Sekunden. In 23 dieser Instanzen wird jedoch ein absoluter Anstieg der maximalen Antwortzeit um durchschnittlich 0,25 Sekunden verzeichnet, der größte Anstieg liegt bei 2,76 Sekunden in einer Instanz. In der Problemgruppe „400“ ist es ähnlich: von den 59 Instanzen haben 28 kürzere maximale Antwortzeiten verzeichnet, wobei die größte Einsparung 37,9 Sekunden beträgt, während in den anderen 31 Instanzen durchschnittlich um 0,09 Sekunden längere maximale Antwortzeiten gemessen wurden, bei einer maximalen Abweichung von 1,94 Sekunden.

Mit der Schätzfunktion $\overrightarrow{C}^{\triangleright E}$ können die Antwortzeiten ebenfalls verringert werden, im Vergleich zur Schätzfunktion $\overrightarrow{C}^{+S}$ sind die Verbesserungen allerdings nicht besonders hoch. Im folgenden Abschnitt wird daher versucht, auch die Summe der Servicezeiten in eine Variante der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ zu integrieren.

3.4.2.3 Variante der Maximalauftragszeit mit Summe der Servicezeiten

Bei der Maximalauftragszeit mit Einzel-Servicezeit $\overrightarrow{C}^{\triangleright E}$ werden nur die Servicezeiten $s_{p_{i'}}, s_{d_{i'}}$ des Auftrags i' mit maximaler noch anfallender Fahrt- und Servicezeit $\overrightarrow{C_{i'}}^{\triangleright E}$ berücksichtigt. Da jedoch alle Servicezeiten s_ν der noch anzufahrenden Orte $\nu \in V_{\Phi,\mu}^{Orte}$ eingehalten werden

müssen und unabhängig von der gewählten Route sind, können auch alle Servicezeiten in die Berechnung der Schätzfunktion aufgenommen werden. Damit soll erreicht werden, dass Knoten auf „höheren“ Stufen k des Zustandsgraphen G^Z_{n,k^*} tendenziell eher expandiert werden, da die Anzahl der noch anzufahrenden Orte $\mid V^{Orte}_{\Phi,\mu} \mid = 2n - k$ beträgt und somit für steigende k abnimmt, womit auch die Summe der Servicezeiten aller noch anzufahrenden Orten sinkt.

Die „*Maximalauftragszeit mit Summe der Servicezeiten*“ $\overrightarrow{C}^{\triangleright S} : V^Z_{n,k^*} \times {}_{2n}V^Z_{n,k^*} \rightarrow \mathbb{R}^+$ gibt eine geschätzte minimale Fahr- und Servicezeit von einem Knoten $(\Phi,\mu)^T \in V^Z_{n,k^*}$ zu einem der Zielknoten $(\Phi^*,\mu*)^T \in {}_{2n}V^Z_{n,k^*}$ zurück. Zur Definition von $\overrightarrow{C}^{\triangleright S}$ wird auf die Maximalauftragszeit $\overrightarrow{C}^{\triangleright}$ zurückgegriffen, zusätzlich wird die Summe aller Servicezeiten an den noch anzufahrenden Orten $\nu \in V^{Orte}_{\Phi,\mu}$ eingefügt:

$$\overrightarrow{C}^{\triangleright S}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \overrightarrow{C}^{\triangleright}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) + \sum_{\nu \in V^{Orte}_{\Phi,\mu}} s_\nu$$

Für diese Schätzfunktion gilt:

Satz 3.9
Der A-Algorithmus findet mit der Schätzfunktion $\overrightarrow{C}^{\triangleright S}$, der Maximalauftragszeit mit Summe der Servicezeiten, einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}. Das schnellstmögliche Auffinden der Lösung ist nicht garantiert.*

Beweis: Zum Beweis der Optimalität der Lösung muss gezeigt werden, dass der Wert $\overrightarrow{C}^{\triangleright S}((\Phi,\mu)^T, {}_{2n}V^S_{n,k^*})$ die tatsächlich verbleibende Fahrt- und Servicezeit des kürzesten Wegs von $(\Phi,\mu)^T$ zu einem der Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^S_{n,k^*})$ nicht überschätzt. Die tatsächlich verbleibende Fahrt- und Servicezeit lässt sich schreiben als

$$\sum_{\nu \in V^{Orte}_{\Phi,\mu}} (c^*_\nu + s_\nu),$$

wobei die Werte c^*_ν die in die fahrtzeitminimale Route von der Fahrzeugposition μ durch alle noch anzufahrenden Orte $\nu \in V^{Orte}_{\Phi,\mu}$ eingehenden Fahrzeiten der Kanten $(\nu',\nu)*$ und die Werte s_ν die an diesen Orten einzuhaltenden Servicezeiten darstellen.

Für die Schätzfunktion $\overrightarrow{C}^{\triangleright}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})$ gilt, dass sie die verbleibende Fahrzeit von einem Knoten $(\Phi,\mu)^T \in V^Z_{n,k^*}$ zu einer der Senken $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ nicht überschätzt, d. h.

$$\begin{aligned} & \overrightarrow{C}^{\triangleright}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) \leq \sum_{\nu \in V^{Orte}_{\Phi,\mu}} c^*_\nu \\ \Leftrightarrow \quad & \overrightarrow{C}^{\triangleright S}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) \leq \sum_{\nu \in V^{Orte}_{\Phi,\mu}} (c^*_\nu + s_\nu) \end{aligned}$$

Der A*-Algorithmus findet also garantiert den zeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}.

Problem-		Aufträge		gelöste Instanzen	
gruppe	Aufträge	absolut	Δ	absolut	Δ
100	51,86	34,57	3,36%	30	7,14%
200	103,08	47,92	8,74%	18	28,57%
400	205,66	72,29	10,35%	11	37,50%
600	308,97	82,40	11,98%	5	25,00%
800	411,75	92,49	12,98%	1	- [6]
1000	514,98	101,41	10,23%	0	-

Tabelle 3.24: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright S}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Die Dreiecksungleichung

$$\overrightarrow{C}^{\triangleright S}((\Phi,\mu)^T,\{(\Phi',\mu')^T\}) + \overrightarrow{C}^{\triangleright S}((\Phi',\mu')^T,{}_{2n}V^S_{n,k^*}) \geq \overrightarrow{C}^{\triangleright S}((\Phi,\mu)^T,{}_{2n}V^S_{n,k^*})$$

wird auch für die Schätzfunktion $\overrightarrow{C}^{\triangleright S}$ nicht erfüllt. Man betrachte dazu den Spezialfall, dass für die Servicezeiten gilt: $s_\nu = 0 \; \forall \; \nu \in V^{Orte}_{\Phi,\mu}$, und wende das Gegenbeispiel aus dem Beweis von Satz 3.3 an.

□

Ergebnisse der Testläufe

Diese Testläufe basieren ebenfalls auf den in Abschnitt 3.4.1 beschriebenen Einstellungen. Als Vergleichswerte dienen weiterhin die Ergebnisse der Testläufe aus Abschnitt 3.4.1 unter Verwendung des Zustandsgraphen und der Schätzfunktion $\overrightarrow{C}^{\triangleright}$.

In Tabelle 3.24 wird die Anzahl der durchschnittlich bis zum Erreichen der Grenze von einer Million Knoten bearbeiteten Aufträge und gelösten Testinstanzen dargestellt. Die Tabelle zeigt deutliche Steigerungen der bearbeiteten Aufträge von 3,36% bis 12,98%. Im Vergleich zu den mit der Schätzfunktion $\overrightarrow{C}^{+S}$ erzielten Steigerungen von 3,79% bis 19,48% sind diese Werte jedoch deutlich geringer, d. h. die Schätzfunktion $\overrightarrow{C}^{\triangleright S}$ konnte die Ergebnisse aus Abschnitt 3.4.2.1 nicht erreichen. Das zeigt sich auch in der Anzahl gelöster Instanzen: trotz der erreichten Steigerungen von 7,14% bis 37,50% wird in den Problemgruppen „200“ und „600“ je eine Instanz weniger gelöst als in den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{+S}$.

Diese Ergebnisse spiegeln sich auch in der Anzahl der erzeugten Knoten wider, die in den Tabellen 3.25 und 3.26 dargestellt werden. Im Vergleich zu den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ kann eine deutliche Reduktion um 17,26% bis 39,73% der Knoten verzeichnet werden, in den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{+S}$ aus Abschnitt 3.4.2.1 wurden jedoch Einsparungen von 25,97% bis 45,88% der Knoten erzielt.

[6] Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Problem-gruppe	♯ Knoten absolut			
	erzeugt	unzulässig	expandiert	Duplikate
100	353.193,66	139.762,52	57.754,80	132.418,57
200	605.646,12	261.409,60	110.374,13	199.121,57
400	746.836,81	347.458,63	148.607,27	202.156,59
600	811.344,48	404.256,62	166.893,30	190.989,80
800	848.525,56	433.543,63	181.453,80	184.181,58
1000	842.124,67	424.274,52	186.379,43	177.929,29

Tabelle 3.25: Die Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright S}$

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	271.974,57	-33,00%	109.583,63	-39,73%	46.306,05	-36,31%	102.165,05	-26,26%
200	500.083,03	-23,79%	221.616,27	-28,65%	92.571,52	-26,18%	161.107,30	-21,15%
400	568.705,12	-22,05%	293.765,32	-22,99%	117.609,39	-22,14%	128.706,63	-27,20%
600	668.519,13	-18,17%	349.063,50	-20,26%	137.979,30	-19,39%	148.716,87	-20,15%
800	686.946,31	-17,80%	373.103,68	-18,44%	147.833,86	-17,90%	135.860,64	-23,21%
1000	669.536,83	-17,71%	356.811,95	-19,51%	150.747,38	-17,26%	125.674,64	-22,11%

Tabelle 3.26: Die Anzahl erzeugter Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright S}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-gruppe	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	113,83	113,83	-13,39%	11,96	6,77	-34,56%	0,39	0,23	-34,61%
200	158,68	48,02	-17,37%	10,69	6,73	-27,92%	0,28	0,19	-27,96%
400	156,83	53,68	-25,37%	13,84	4,41	-37,28%	0,23	0,09	-35,96%
600	77,36	64,45	-46,78%	9,82	7,18	-28,63%	0,16	0,13	-27,21%
800	247,04	69,30	-45,01%	11,03	5,22	-35,10%	0,16	0,09	-32,39%
1000	67,92	41,74	-14,52%	7,52	3,60	-23,84%	0,11	0,06	-24,30%

Tabelle 3.27: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright S}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Die Antwortzeiten der Testläufe mit $\overrightarrow{C}^{\triangleright S}$ werden in Tabelle 3.27 dargestellt. Auch hier können beachtliche Verbesserungen gegenüber den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright S}$ festgestellt werden: die durchschnittliche Antwortzeit kann um 24,30% bis 35,96% gesenkt werden, während die durchschnittliche maximale Antwortzeit um 23,84 bis 37,28% und die global maximale Antwortzeit um 13,39% bis 46,78% verringert werden. Im Gegensatz zur Anzahl der Knoten und Anzahl der bearbeiteten Aufträge sind bei den Antwortzeiten die Ergebnisse der Schätzfunktion $\overrightarrow{C}^{\triangleright S}$ den Ergebnissen der Schätzfunktion $\overrightarrow{C}^{+S}$ fast ebenbürtig: die durchschnittlichen Antwortzeiten der Testläufe zu $\overrightarrow{C}^{\triangleright S}$ sind um maximal 0,01 Sekunden höher als die Antwortzeiten in den Testläufen zu $\overrightarrow{C}^{+S}$, die durchschnittlich maximalen Antwortzeiten sind maximal 0,96 Sekunden länger, und die global maximalen Antwortzeiten sind in den Problemgruppen „100“, „200“ und „400“ in den Testläufen zu $\overrightarrow{C}^{\triangleright S}$ um bis zu 13,5 Sekunden geringer als in den Testläufen zu $\overrightarrow{C}^{+S}$, in den Problemgruppen „600“, „800“ und „1000“ liegen sie jedoch bis zu 17,05 Sekunden darüber. Die trotz der höheren Anzahl erzeugter Knoten relativ geringen Antwortzeiten sind auf die im Vergleich zur Schätzfunktion $\overrightarrow{C}^{+S}$ geringe Rechenzeit zur Berechnung der Schätzfunktion $\overrightarrow{C}^{\triangleright S}$ zurückzuführen.

3.4.2.4 Fazit

Die Beachtung der Servicezeiten in Varianten der Schätzfunktionen hat gute Ergebnisse erzielt. Bei beiden Schätzfunktionen $\overrightarrow{C}^{\triangleright}$ und $\overrightarrow{C}^{+}$ konnte durch die Hinzunahme der Summe der Servicezeiten in den Testläufen eine deutliche Reduktion der Anzahl erzeugter Knoten und der Antwortzeiten erreicht werden. Die Testläufe unter Verwendung der Schätzfunktion $\overrightarrow{C}^{+S}$ zeigen dabei die besten Ergebnisse in Bezug auf die Anzahl der erzeugten Knoten, die durchschnittlichen und die durchschnittlich maximalen Antwortzeiten, während die global maximalen Antwortzeiten in jeweils der Hälfte der untersuchten Problemgruppen die niedrigsten Werte bei Verwendung von $\overrightarrow{C}^{+S}$ bzw. $\overrightarrow{C}^{\triangleright S}$ erreichten.

Insgesamt kann daher geschlossen werden, dass der A*-Algorithmus bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+S}$ in weniger Schritten zu einer optimalen Lösung findet. Die Schätzfunktion $\overrightarrow{C}^{+S}$ kann aber wegen der aufwändigeren Berechnung der geschätzten Werte in den Antwortzeiten noch nicht vollständig überzeugen.

3.4.3 Varianten der Schätzfunktionen mit Hilfe der Zeitfenster

Das Single Vehicle Routing wird auch durch die Zeitfenster $[\underline{t}_\nu, \overline{t}_\nu]$ an den noch anzufahrenden Orten ν beeinflusst. Je nach Größe der Zeitfenster ist dieser Einfluß mehr oder weniger stark. Der Einfluss der Zeitfenster kann sich auf zwei Arten auswirken:

- die untere Zeitfenstergrenze $\underline{t}_\nu$ kann zu einer Wartezeit ω_ν an Ort ν führen
- die obere Zeitfenstergrenzen $\overline{t}_\mu$ einer Fahrzeugposition μ kann dazu führen, dass der Knoten $(\Phi, \mu)^T$ des Zustandsgraphen unzulässig ist, weil die bisher zurückgelegte Fahr-, Service- und Wartezeit $\overleftarrow{C}$ die Zeitfenstergrenze $\overline{t}_\mu$ übersteigt

Zunächst wird der Einbezug der Wartezeiten in Varianten der Schätzfunktion betrachtet:

Es bezeichne $\omega_{\Phi,\mu}$ die Länge der Wartezeit in der Fahrzeugposition μ eines Knotens $(\Phi,\mu)^T$ des Zustandsgraphen. $\omega_{\Phi,\mu}$ hängt von der unteren Zeitfenstergrenze $\underline{t}_\mu$ und der bisherigen Fahrt-, Warte- und Servicezeit $\overleftarrow{C}((\Phi,\mu)^T)$ ab. Die Wartezeit eines Ortes ν könnte indirekt in die Berechnung einer Schätzfunktion für den Knoten $(\Phi,\mu)^T$ eingehen, indem überprüft wird, ob die geschätzte Fahrzeit von der Fahrzeugposition μ zu diesem Ort ν zusammen mit der bekannten Zeit $\overleftarrow{C}((\Phi,\mu)^T)$ die untere Zeitfenstergrenze $\underline{t}_\nu$ erreicht.

Angenommen, es gäbe einen Auftrag i^* mit maximaler unterer Zeitfenstergrenze $\underline{t}_{d_{i^*}}$ für das Delivery, die so hoch sei, dass auf jeden Fall eine Wartezeit $\omega_{d_{i^*}} > 0$ entsteht. Für die Maximalauftragszeit $\overrightarrow{C}^{\triangleright}((\Phi,\mu)^T)$ eines Knotens $(\Phi,\mu)^T$ mit $\varphi(i^*) < 2$ führt das Einbeziehen der Wartezeit dazu, dass für den Auftrag i^* die Differenz aus der unteren Zeitfenstergrenze $\underline{t}_{d_{i^*}}$ und der tatsächlich erfolgten Fahrzeit $\overleftarrow{C}((\Phi,\mu)^T)$ von der Quelle des Graphen G^Z_{n,k^*} bis zum Knoten $(\Phi,\mu)^T$ als Abschätzung für die Strecke von Auftrag i^* berechnet wird. Da die untere Zeitfenstergrenze des Auftrags i^* maximal ist und positive Wartezeit entsteht, ist die Maximalauftragszeit $\overrightarrow{C}^{\triangleright}((\Phi,\mu)^T)$ gleich der Abschätzung für die Strecke für Auftrag i^*. Das gilt für jeden Knoten $(\Phi,\mu)^T$ mit $\varphi(i^*) < 2$. Dies hat zur Folge, dass als geschätzte Gesamtkosten der Route $\overleftarrow{C}((\Phi,\mu)^T) + \overrightarrow{C}^{\triangleright}((\Phi,\mu)^T)$ gerade die untere Zeitfenstergrenze $\underline{t}_{d_{i^*}}$ berechnet wird. Da der A*-Algorithmus immer denjenigen Knoten $(\Phi',\mu')^T$ mit minimalen geschätzten Gesamtkosten $\overleftarrow{C}((\Phi',\mu')^T) + \overrightarrow{C}^{\triangleright}((\Phi',\mu')^T)$ expandiert, kann zumindest eine Zeit lang kein Knoten $(\Phi^*,\mu^*)^T$ bevorzugt expandiert werden, da die minimalen Gesamtkosten solange der maximalen unteren Zeitfenstergrenze $\underline{t}_{d_{i^*}}$ entsprechen, bis Auftrag i^* in allen für die Expansion zur Verfügung stehenden Knoten $(\Phi^*,\mu^*)^T$ den Status $\varphi(i^*) = 2$ erreicht hat. Für den A*-Algorithmus entfällt damit zumindest partiell die Entscheidungshilfe zur Auswahl des als nächsten zu expandierenden Knoten.

Bei der Minimalzeitensumme $\overrightarrow{C}^{\triangleright}$ können die Wartezeiten nicht so einfach eingebracht werden, da die in die Minimalzeitensumme $\overrightarrow{C}^{\triangleright}((\Phi,\mu)^T)$ eines Knotens $(\Phi,\mu)^T)$ eingehenden Fahrzeiten nicht unbedingt die Fahrzeiten der tatsächlich zu fahrenden Route vom Ort μ durch alle noch anzufahrenden Orte $\nu \in V^{Orte}_{\Phi,\mu}$ abbilden. Die Wartezeiten könnten aber insofern eingebracht werden, dass die maximale untere Zeitfenstergrenze $\underline{t}_{d_{i^*}}$ mit der Summe $\overleftarrow{C}((\Phi,\mu)^T) + \overrightarrow{C}^{+}((\Phi,\mu)^T)$ verglichen und die Differenz als geschätzte Wartezeit interpretiert wird. Auch dieses Vorgehen würde jedoch dazu führen, dass bei Auftreten von positiven Wartezeiten viele Knoten $(\Phi,\mu)^T$ die gleichen geschätzten Gesamtkosten $\underline{t}_{d_{i^*}}$ hätten. Wiederum würde durch dieses Vorgehen die Auswahl eines zu expandierenden Knotens $(\Phi^*,\mu^*)^T$ durch den A*-Algorithmus nicht unterstützt, sondern erschwert werden. Das Einbeziehen von Wartezeiten in die Schätzfunktion ist daher nicht zielführend und wird daher hier nicht weiter untersucht.

Erfolgsversprechender ist hingegen die Überprüfung, ob die Zeitfenster im weiteren Verlauf einer Route überhaupt noch eingehalten werden können. Da die Reihenfolge der Orte in einer zu berechnenden Route natürlicherweise nicht bekannt ist, kann zwar keine abschließende

Antwort auf diese Frage gegeben werden, trotzdem können durch einfache Berechnungen eventuell schon Zeitfenster gefunden werden, die auf jeden Fall verletzt werden. Dann kann durch extrem hohe geschätzte Kosten eine weitere Expandierung des Knotens verhindert werden, was insbesondere bei Knoten auf niedrigen Stufen des Zustandsgraphen eine erhebliche Rechenzeitersparnis bewirken kann. Diese Überlegung ist im Falle der Schätzfunktion „Maximalauftragszeit“ intuitiv anzuwenden, indem bei der Berechnung der noch zu fahrenden Strecke für jeden Auftrag direkt die Einhaltung der oberen Zeitfenstergrenzen überprüft wird. Für die Minimalzeitensumme ist eine solche in die Abschätzung eingebettete Vorgehensweise nicht möglich, nichts desto trotz kann aber mit etwas zusätzlicher Rechenzeit dasselbe Vorgehen wie in der Abschätzung mit Hilfe der Maximalauftragszeit ebenfalls angewendet werden. Die Überprüfung aller Zeitfenster wird daher für beide Abschätzungen in den folgenden Abschnitten erläutert und anhand der Testläufe bewertet.

3.4.3.1 Variante der Maximalauftragszeit mit Zeitfensterüberprüfung

Mit der Schätzfunktion „Maximalauftragszeit“ wird zunächst für jeden Auftrag i die Fahrzeit $\overrightarrow{C}_i^{\triangleright}$ für seine alleinige Abarbeitung berechnet, und danach das Maximum der $\overrightarrow{C}_i^{\triangleright}$ aller Aufträge $i = 1, \ldots, n$ als Abschätzung $\overrightarrow{C}^{\triangleright}$ zurückgegeben. Dieses Vorgehen kommt der Überprüfung aller Zeitfenster sehr entgegen. Bei der Berechnung der verbleibenden Fahrzeit $\overrightarrow{C}_i^{\triangleright}$ eines einzelnen Auftrags i kann direkt berechnet werden, ob dessen Zeitfenster $[\underline{t}_{p_i}, \overline{t}_{p_i}]$ (für $\varphi(i) < 2$) bzw. $[\underline{t}_{d_i}, \overline{t}_{d_i}]$ noch eingehalten werden können. Dazu wird direkt bei der Berechnung überprüft, ob die Summe aus der bisher zurückgelegten Fahrzeit $\overleftarrow{C}$ und der Zeit bis zum Erreichen eines Pickup- oder Delivery-Ortes des Auftrags i noch kleiner als die entsprechende obere Zeitfenstergrenze ist.

Die „*Maximalauftragszeit mit Zeitfensterüberprüfung*“ $\overrightarrow{C}^{\triangleright T} : V^Z_{n,k^*} \times {}_{2n}V^Z_{n,k^*} \to \mathbb{R}^+$ gibt eine geschätzte minimale Fahrzeit von einem Knoten $(\Phi, \mu)^T \in V^Z_{n,k^*}$ zu einem der Zielknoten $(\Phi^*, \mu*)^T \in {}_{2n}V^Z_{n,k^*}$ zurück, wobei ein nicht einzuhaltendes Zeitfenster mindestens eines Auftrags i mit einem Rückgabewert von ∞ angezeigt wird. Dazu wird die für einen einzelnen Auftrag i zum Erreichen eines Zustandsvektors $(\Phi^*, \mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ verbleibende Fahrzeit definiert als:

$$\overrightarrow{C}_i^{\triangleright T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) = \begin{cases} 0, & \varphi(i) = \varphi^*(i) \\ c_{\mu,d_i}, & \varphi(i) = 1 \wedge \varphi^*(i) = 2 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,d_i} \leq \overline{t}_{d_i} \\ c_{\mu,p_i}, & \varphi(i) = 0 \wedge \varphi^*(i) = 1 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} \leq \overline{t}_{p_i} \\ c_{\mu,p_i} + c_{p_i,d_i}, & \varphi(i) = 0 \wedge \varphi^*(i) = 2 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} \leq \overline{t}_{p_i} \wedge \\ & \qquad \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} + c_{p_i,d_i} \leq \overline{t}_{d_i} \\ \infty, & \text{sonst} \end{cases}$$

Der Wert $\overrightarrow{C}_i^{\triangleright T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$ bildet so die noch verbleibende Fahrzeit unter Berücksichtigung der Zulässigkeit der Zeitfenster bei alleiniger Bedienung eins Auftrags i ab. Mit

$$\max_{i=1\ldots,n} \overrightarrow{C_i}^{\triangleright T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

wird somit eine Abschätzung für die Fahrzeit vom Knoten $(\Phi,\mu)^T$ zu einem Zielknoten $(\Phi^*,\mu^*)^T$ definiert, die den Wert ∞ enthält, sobald mindestens ein Zeitfenster nicht eingehalten werden kann. Die Maximalauftragszeit mit Zeitfensterüberprüfung wird daher als Minimum der geschätzten Fahrzeiten von $(\Phi,\mu)^T$ zu einem der Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ definiert:

$$\overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) = \min_{(\Phi^*,\mu^*)^T\in{}_{2n}V^Z_{n,k^*}} \max_{i=1}^{n} \overrightarrow{C_i}^{\triangleright T}((\Phi,\mu)^T,(\Phi^*,\mu*)^T)$$

Für die Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ gilt:

Satz 3.10

Der A-Algorithmus findet mit der Schätzfunktion „Maximalauftragszeit mit Zeitfensterüberprüfung“, $\overrightarrow{C}^{\triangleright T}$, einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}, falls es mindestens einen zulässigen Weg gibt. Das schnellstmögliche Auffinden der Lösung ist nicht garantiert.*

Beweis: Zum Beweis der Optimalität muss gezeigt werden, dass der Wert der Schätzfunktion $\overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T,{}_{2n}V^{Orte}_{\Phi,\mu})$ die tatsächlich noch anfallende Fahrzeit von einem Knoten $(\Phi,\mu)^T$ zu einem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ nicht überschätzt. $\overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T,{}_{2n}V^{Orte}_{\Phi,\mu})$ kann wegen $\Phi^* = (2,\ldots,2)^T \;\forall\; (\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ dargestellt werden als:

$$\overrightarrow{C_i}^{\triangleright T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) = \begin{cases} 0, & \varphi(i)=2 \\ c_{\mu,d_i}, & \varphi(i)=1 \wedge \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,d_i} \le \overline{t}_{d_i} \\ c_{\mu,p_i}+c_{p_i,d_i}, & \varphi(i)=0 \wedge \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,p_i} \le \overline{t}_{p_i} \wedge \\ & \quad \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,p_i}+c_{p_i,d_i} \le \overline{t}_{d_i} \\ \infty, & \text{sonst} \end{cases}$$

$$\overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) = \max_{i=1\ldots,n} \overrightarrow{C_i}^{\triangleright T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

Für den Fall $\overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) < \infty$ gilt:

$$\overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) = \overrightarrow{C}^{\triangleright}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})$$

Für $\overrightarrow{C}^{\triangleright}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})$ gilt, dass die tatsächlich verbleibende Fahrzeit nicht überschätzt wird.

Man betrachte daher den Fall $\overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) = \infty$. Es gibt also einen Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$, einen Auftrag i' mit $\overrightarrow{C_i}^{\triangleright T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) = \infty$ und einen Ort $\nu_{i'}$ mit $\overleftarrow{C}((\Phi,\mu)^T) + \overrightarrow{C_{i'}}^{\triangleright}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) > \overline{t}_{\nu_{i'}}$. Angenommen, es gäbe einen Weg vom Knoten $(\Phi,\mu)^T$ zu einer der Senken $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$, der alle Zeitfensterrestriktionen einhält. Dann gibt es auch eine Route $(\mu,\nu_1,\ldots,\nu_l)$ von der Fahrzeugposition μ durch alle noch anzufahrenden Orte $\nu_1,\ldots,\nu_l \in V^{Orte}_{\Phi,\mu}$, die die Zeitfensterrestriktionen einhält. Man betrachte nun die

Teilroute $(\mu, \nu_1, \ldots, \nu_{i'})$. Offensichtlich muss gelten, dass die Fahrzeit dieser Teilroute kleiner als $\overrightarrow{C_{i'}}^{\triangleright}((\Phi,\mu)^T, (\Phi^*,\mu^*)^T)$ ist:

$$\begin{aligned} c^*_{\nu_1} + c^*_{\nu_2} + \cdots + c^*_{\nu_{i'}} &\leq \overline{t}_{\nu_{i'}} \\ &< \overrightarrow{C_{i'}}^{\triangleright}((\Phi,\mu)^T, (\Phi^*,\mu^*)^T) \\ &= \begin{cases} 0, & \varphi(i') = 2 \\ c_{\mu,\nu_{i'}}, & \varphi(i') = 1 \\ c_{\mu,p_{i'}} + c_{p_{i'},\nu_{i'}}, & \varphi(i') = 0 \end{cases} \end{aligned}$$

Das steht im Widerspruch zur geltenden Dreiecksungleichung für die Fahrzeit zwischen je zwei Orten. Somit ist gezeigt, dass die Schätzfunktion $\overrightarrow{C_i}^{\triangleright T}$ die Fahrzeit eines zulässigen Wegs nicht überschätzt.

Die für das schnellstmögliche Finden einer Lösung benötigte Gültigkeit der Dreieckgsungleichung ist nicht erfüllt. Man betrachte dazu das Gegenbeispiel aus Abschnitt 3.2.3 und den Spezialfall, dass alle Zeitfenster $[\underline{t}_\nu, \overline{t}_\nu]$ an den noch anzufahrenden Orten ν dem Planungshorizont entsprechen.

□

Die Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ kann daher im A*-Algorithmus eingesetzt werden, ohne die Optimalität der Lösung zu gefährden. Wenn in vielen Knoten $(\Phi,\mu)^T$ Zeitfensterverletzungen bemerkt werden, könnte die Zahl der Knoten und damit die Antwortzeit deutlich gesenkt werden.

Bemerkung:
Mit der Einführung der Kosten $\overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \infty$ für Knoten $(\Phi,\mu)^T$, deren weitere Expandierung sich wegen Unzulässigkeit mindestens eines Zeitfensters nicht lohnt, kann auch ein neues Abbruchkriterium für den Algorithmus angegeben werden. Bisher terminiert der Algorithmus mit der optimalen Lösung, sobald er eine Senke $(\Phi^*,\mu^*)^T$ zur Expansion auswählt, d. h. eine Senke $(\Phi^*,\mu^*)^T$ mit minimalen geschätzten Gesamtkosten $\overleftarrow{C}((\Phi^*,\mu^*)^T) + \overrightarrow{C}^{\triangleright T}((\Phi^*,\mu^*)^T, {}_{2n}V^Z_{n,k^*}) = \overleftarrow{C}((\Phi^*,\mu^*)^T)$ gefunden wurde. Andernfalls wird mit dem Ergebnis „nicht lösbar“ abgebrochen, wenn keine Knoten mehr zur Expansion zur Verfügung stehen, also alle noch nicht expandierten erzeugten Knoten unzulässig waren. Nun kann auch schon abgebrochen werden, sobald ein Knoten $(\Phi,\mu)^T$ mit geschätzten Gesamtkosten von $\overleftarrow{C}((\Phi,\mu)^T) + \overrightarrow{C}^{\triangleright T}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \infty$ zur Expansion ausgewählt wird, da dann alle noch für die Expansion zur Verfügung stehenden Knoten $(\Phi',\mu')^T$ geschätzte Kosten in Höhe von $\overleftarrow{C}((\Phi',\mu')^T) + \overrightarrow{C}^{\triangleright T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) = \infty$ besitzen und somit keine zulässige Lösung mehr gefunden werden kann.

Ergebnisse der Testläufe

Für diese Testläufe werden die in Abschnitt 3.4.1 beschriebenen Einstellungen übernommen. Als Vergleichswerte dienen weiterhin die Ergebnisse der Testläufe aus Abschnitt 3.4.1 unter

Problem-		Aufträge		gelöste Instanzen	
gruppe	Aufträge	absolut	Δ	absolut	Δ
100	51,86	40,75	21,84%	39	39,29%
200	103,08	69,75	58,28%	35	150,00%
400	205,66	122,81	87,48%	29	262,50%
600	308,97	177,00	140,54%	30	650,00%
800	411,75	230,41	181,45%	29	- [7]
1000	514,98	276,43	200,47%	27	- [7]

Tabelle 3.28: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright T}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-	♯ Knoten absolut			
gruppe	erzeugt	unzulässig	expandiert	Duplikate
100	216.090,09	108.342,20	30.393,54	75.993,45
200	368.190,97	208.613,20	52.959,22	104.956,02
400	481.459,24	286.644,44	69.577,90	121.962,17
600	541.182,93	330.490,82	82.008,40	124.339,10
800	554.131,71	347.500,41	87.509,05	110.186,83
1000	649.395,74	423.798,57	105.129,71	112.161,45

Tabelle 3.29: Die Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright T}$

Verwendung des Zustandsgraphen und der Schätzfunktion $\overrightarrow{C}^{\triangleright}$.

In Tabelle 3.28 wird die Anzahl der bearbeiteten Aufträge und gelösten Instanzen dargestellt. Die Tabelle zeigt beachtliche Steigerungen im Vergleich zur Schätzfunktion $\overrightarrow{C}^{\triangleright}$: Bei der Anzahl bearbeiteter Aufträge werden Zuwächse von 21,84% bis 200,47% der bei Verwendung von $\overrightarrow{C}^{\triangleright}$ bearbeiteten Aufträge erzielt, d. h. in Problemgruppe „1000“ konnte die Anzahl bearbeiteter Aufträge im Vergleich zu den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ verdoppelt werden. Die relativ geringen Steigerungen in den Problemklassen „100“ und „200“ von „nur“ 21,84% bzw. 58,28% sind durch die geringe Anzahl Aufträgen in diesen Problemgruppen zu erklären, da eine Verbesserung nur solange anhand der Anzahl bearbeiteter Aufträge ersichtlich wird, wie weitere Aufträge zur Bearbeitung zur Verfügung stehen. Für die Anzahl der gelösten Instanzen werden Steigerungen um 39,39% bis zu 650% verzeichnet: in allen Problemgruppen können jeweils mindestens 27 Instanzen gelöst werden, wohingegen bei den Testläufen unter Verwendung von $\overrightarrow{C}^{\triangleright}$ in den Problemgruppen „800“ und „1000“ keine Instanz gelöst werden kann.

Die Steigerungen der Anzahl bearbeiteter Aufträge und gelöster Instanzen spiegeln sich erwartungsgemäß in der Anzahl der erzeugten Knoten wider, die in den Tabellen 3.29 und

[7] Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	74.970,75	-81,53%	25.894,41	-85,76%	13.265,04	-81,76%	34.832,46	-74,86%
200	114.201,02	-82,60%	40.626,82	-86,92%	20.589,28	-83,58%	51.895,98	-74,60%
400	103.010,12	-85,88%	43.831,19	-88,51%	19.750,64	-86,92%	37.426,10	-78,83%
600	126.033,15	-84,57%	50.586,50	-88,44%	24.220,82	-85,85%	48.656,73	-73,87%
800	122.208,69	-85,38%	49.108,03	-89,27%	25.146,61	-86,04%	44.131,14	-75,06%
1000	111.388,79	-86,31%	49.404,16	-88,86%	25.844,19	-85,81%	31.612,10	-80,41%

Tabelle 3.30: Die Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright T}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-gruppe	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	34,65	11,62	-91,16%	2,84	0,97	-90,66%	0,09	0,03	-90,28%
200	19,48	19,48	-66,49%	2,76	1,56	-83,31%	0,05	0,04	-83,96%
400	72,25	44,77	-37,76%	6,15	1,22	-82,69%	0,06	0,02	-84,51%
600	34,36	30,94	-74,45%	4,04	1,86	-81,56%	0,03	0,03	-82,74%
800	23,18	16,87	-86,61%	2,93	1,28	-84,02%	0,02	0,02	-83,59%
1000	46,05	20,81	-57,38%	3,87	0,69	-85,46%	0,02	0,01	-85,92%

Tabelle 3.31: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright T}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

3.30 dargestellt werden: die Anzahl der erzeugten Knoten kann um 81,53% bis 89,31% in Problemgruppe „1000“ gesenkt werden, die Anzahl der als unzulässig erkannten Knoten wird sogar um bis zu 89,27% verringert.

Bei den Antwortzeiten in Tabelle 3.31 zeigen sich ebenfalls beachtliche Verbesserungen: die durchschnittlichen Antwortzeiten sinken um 82,74% bis 90,28%, die durchschnittlich maximalen Antwortzeiten werden um 81,56% bis 90,66% verringert und die global maximalen Antwortzeiten können um 37,76% bis 91,16% reduziert werden. Insbesondere ist anzumerken, dass die absoluten, global maximalen Antwortzeiten dieser Testläufe nur noch 19,48 bis 72,26 Sekunden betragen, so dass hier eine beachtliche Verbesserung festgestellt werden kann.

3.4.3.2 Variante der Minimalzeitensumme mit Zeitfensterüberprüfung

In die Schätzfunktion „Minimalzeitensumme“ $\overrightarrow{C}^{+}$ geht für jeden noch anzufahrenden Ort $\nu \in V_{\Phi,\mu}^{Orte}$ die minimale Fahrzeit $\min_{\nu \in V_{\Phi,\mu}^{Orte} \setminus \{\mu,\nu\}} c_{\mu\nu}$ zum Erreichen des Ortes ein, ohne dass auf den Zusammenhang – im graphentheoretischen Sinne – der Kanten geachtet wird. Die so entstehende Menge von Teilstrecken hat mit der tatsächlich zu fahrenden Route daher nicht viel gemeinsam, so dass mit ihrer Hilfe die Zeitfenster nicht überprüft werden können. Wohl aber kann man den Test der Zeitfenster analog zum vorgestellten Test in der Schätzfunk-

tion $\overrightarrow{C}^{rhd}$ zusätzlich durchführen, was natürlich die benötigte Zeit für die Berechnung der Schätzfunktion $\overrightarrow{C}^{+T}$ erhöht.

Die Schätzfunktion „*Minimalzeitensumme mit Zeitfensterüberprüfung*“ $\overrightarrow{C}^{+T} : V^Z_{n,k^*} \times {}_{2n}V^Z_{n,k^*} \rightarrow \mathbb{R}^+$ gibt die geschätzte minimale Fahrzeit von einem Knoten $(\Phi,\mu)^T \in V^Z_{n,k^*}$ zu einem der Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ zurück, wobei ein nicht mehr einzuhaltendes Zeitfenster mindestens eines Auftrags i bei alleiniger Betrachtung dieses Auftrags mit einem Rückgabewert von ∞ angezeigt wird. Ausgehend vom Zustandsvektor $(\Phi,\mu)^T$ wird daher nun für jeden Auftrag i die minimale Fahrzeit zu einem Zustandsvektor $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ berechnet als:

$$\overrightarrow{C}_i^{+T}\left(\binom{\Phi}{\mu},\binom{\Phi^*}{\mu^*}\right)=$$

$$\begin{cases} 0, & \varphi(i)=\varphi^*(i) \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j),\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\}, & \varphi(i)=0 \wedge \varphi^*(i)=1 \wedge \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,p_i} \leq \overline{t}_{p_i} \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j),\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu d_i}, c_{p_j d_i}, c_{d_l d_i}\}, & \varphi(i)=1 \wedge \varphi^*(i)=2 \wedge \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,d_i} \leq \overline{t}_{d_i} \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j)\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\} & \\ \quad + \min\limits_{\substack{0=\varphi(j)<\varphi^*(j)\\ \varphi(l)<\varphi^*(l)=2}} \{c_{p_j d_i}, c_{d_l d_i}\}, & \varphi(i)=0 \wedge \varphi^*(i)=2 \wedge \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,p_i} \leq \overline{t}_{p_i} \wedge \\ & \qquad \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,p_i}+c_{p_i,d_i} \leq \overline{t}_{d_i} \\ \infty, & \text{sonst} \end{cases}$$

Analog zur Definition der Minimalzeitensumme gibt nun die Summe

$$\sum_{i=1}^{n} \overrightarrow{C}_i^{+T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

die geschätzte minimale Fahrzeit von $(\Phi,\mu)^T$ zum Zielknoten $(\Phi^*,\mu^*)^T$ an, wobei ein nicht einzuhaltendes Zeitfenster den Wert ∞ herbeiführt. Die „Minimalzeitensumme mit Zeitfensterüberprüfung“ wird als Minimum der zu jedem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ geschätzten minimalen Fahrzeit $\overrightarrow{C}^{+T}((\Phi,\mu)^T,\{(\Phi^*,\mu^*)^T\})$ definiert:

$$\overrightarrow{C}^{+T}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) = \min_{(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}} \sum_{i=1}^{n} \overrightarrow{C}_i^{+T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$$

Nun können die Eigenschaften des A*-Algorithmus bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+T}$ angegeben werden:

Satz 3.11

Der A-Algorithmus findet bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+T}$, „Minimalzeitensumme mit Zeitfensterüberprüfung“, schnellstmöglich einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}.*

BEWEIS: Für den Beweis der Optimalität muss gezeigt werden, dass $\overrightarrow{C}^{+T}$ die tatsächlich verbleibenden Fahrtkosten von einem Knoten $(\Phi,\mu)^T$ zu einem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ nicht überschätzt. Man betrachte dazu einen Knoten $(\Phi,\mu)^T$ mit $\overrightarrow{C}^{+T}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) < \infty$. In diesem Fall gilt, dass

$$\overrightarrow{C}^{+T}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \overrightarrow{C}^{+}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})$$

so dass die tatsächlich verbleibende Fahrzeit nach Satz 3.6 nicht überschätzt wird.

Für einen Knoten $(\Phi,\mu)^T$ mit $\overrightarrow{C}^{+T}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \infty$ gilt analog zum Beweis von Satz 3.10, dass es keinen Weg vom Knoten $(\Phi,\mu)^T$ zu einem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ geben kann, der alle Zeitfensterrestriktionen einhält. Somit kann $\overrightarrow{C}^{+T}$ die tatsächlich verbleibende Fahrzeit eines fahrzeitminimalen Wegs nicht überschätzen.

Zum Beweis, dass der A*-Algorithmus schnellstmöglich die optimale Lösung findet, muss die Gültigkeit der Dreiecksungleichung

$$\overrightarrow{C}^{+T}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) \geq \overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*})$$

für jeden Knoten $(\Phi,\mu)^T$ und jeden von $(\Phi,\mu)^T$ erreichbaren Knoten $(\Phi',\mu')^T$ gezeigt werden.

Laut Satz 3.6 gilt die Dreiecksungleichung für den Fall, dass alle drei in die Dreiecksungleichung eingehenden Elemente $\overrightarrow{C}^{+T}((\Phi,\mu)^T, \{(\Phi',\mu')^T\})$, $\overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*})$ und $\overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) < \infty$ sind.

Falls $\overrightarrow{C}^{+T}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) = \infty$ oder $\overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) = \infty$ gilt, ist die Dreiecksungleichung trivialerweise erfüllt.

Man betrachte daher einen Knoten $(\Phi,\mu)^T$ mit $\overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) = \infty$. Dann muss gelten, dass

$$\begin{aligned} & \overrightarrow{C}^{+T}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) &=& \infty \\ \vee \quad & \overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) &=& \infty \end{aligned}$$

Da $\overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) = \infty$, muss es einen Ort ν_{i^*} mit verletztem Zeitfenster $\overline{t}_{\nu_{i^*}}$ geben. Man betrachte daher den Status des Auftrags i^* in den Knoten $(\Phi,\mu)^T$ und $(\Phi',\mu')^T$. Für die Fall $\varphi(i^*) = \varphi'(i^*)$ gilt:

$$\overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) = \overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) = \infty$$

Für $\varphi(i^*) = 0, \varphi'(i^*) = 1$ gilt, dass das verletzte Zeitfenster $\overline{t}_{\nu_{i^*}}$ im Falle eines Pickup-Zeitfensters $\overline{t}_{p_{i^*}}$ in $\overrightarrow{C}^{+T}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) = \infty$ eingeht. Im Falle eines Delivery-Zeitfensters $\overline{t}_{d_{i^*}}$ gilt wegen der geltenden Dreiecksungleichung für die Fahrzeiten zwischen je zwei Orten:

$$\begin{aligned} & \overleftarrow{C}((\Phi,\mu)^T) && +c_{\mu,p_i} + c_{p_i,d_i} && > \overline{t}_{d_{i^*}} \\ \Rightarrow \quad & \overleftarrow{C}((\Phi,\mu)^T) && +c_{\mu,\nu_1}+,\ldots,c_{\nu_l,p_i}+,\ldots,+c_{\nu_j,\mu'} + c_{\mu',d_i} && > \overline{t}_{d_{i^*}} \\ \Rightarrow \quad & \overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) = \infty \end{aligned}$$

Problem-		Aufträge		gelöste Instanzen	
gruppe	Aufträge	absolut	Δ	absolut	Δ
100	51,86	40,75	21,84%	39	39,29%
200	103,08	70,52	60,02%	36	157,14%
400	205,66	123,20	88,07%	30	275,00%
600	308,97	177,07	140,63%	30	650,00%
800	411,75	232,58	184,10%	29	- [8]
1000	514,98	276,59	200,64%	27	- [8]

Tabelle 3.32: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{+T}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Die beiden Fälle $\varphi(i^*) = 1, \varphi'(i^*) = 2$ und $\varphi(i^*) = 0, \varphi'(i^*) = 2$ lassen sich analog zeigen.

Somit ist bewiesen, dass der A*-Algorithmus schnellstmöglich eine Lösung findet.

□

Bemerkung: Falls keine zulässige Lösung existiert, werden im Laufe des Algorithmus keine Knoten $(\Phi, \mu)^T$ mit Kosten $\overrightarrow{C}^{+T}((\Phi, \mu)^T, {}_{2n}V^Z_{n,k^*}) < \infty$ mehr zur Expansion zur Verfügung stehen. Wie im vorhergehenden Abschnitt 3.4.3.1 geschildert, kann dementsprechend ein weiteres Abbruchkriterium eingeführt werden, dass den Algorithmus abbricht, sobald ein Knoten mit Kosten ∞ expandiert werden soll.

Ergebnisse der Testläufe

Für die Testläufe gelten weiterhin die in Abschnitt 3.4.1 beschriebenen Einstellungen. Als Vergleichswerte dienen die Ergebnisse der Testläufe aus Abschnitt 3.4.1 unter Verwendung des Zustandsgraphen und der Schätzfunktion $\overrightarrow{C}^{\triangleright}$.

Tabelle 3.32 zeigt die Anzahl der bearbeiteten Aufträge und gelösten Instanzen. Im Vergleich zur Schätzfunktion $\overrightarrow{C}^{\triangleright}$ werden auch hier beachtliche Steigerungen verzeichnet: die Anzahl bearbeiteter Aufträge wächst um 21,84% bis 200,64%. Für die Anzahl der gelösten Instanzen werden Steigerungen um 39,29% bis zu 650% erreicht: in allen Problemgruppen können jeweils mindestens 27 Instanzen gelöst werden, wohingegen bei den Testläufen unter Verwendung von $\overrightarrow{C}^{\triangleright}$ in den Problemgruppen „800“ und „1000“ keine Instanz gelöst werden kann. Im Vergleich zu den Testläufen zur Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ können leichte Verbesserungen festgestellt werden: es werden mindestens genauso viele Aufträge bearbeitet, maximal sind es in einer Problemgruppe durchschnittlich 2,17 Aufträge mehr. Für die Anzahl der gelösten Instanzen gilt, dass in den Problemgruppen „200“ und „400“ jeweils eine Instanz mehr gelöst werden kann als mit der Schätzfunktion $\overrightarrow{C}^{\triangleright T}$.

[8] Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Problem-gruppe	♯ Knoten absolut			
	erzeugt	unzulässig	expandiert	Duplikate
100	210.891,66	104.559,93	29.536,14	75.037,66
200	369.448,10	207.225,92	52.366,23	105.820,25
400	470.738,20	276.651,03	67.982,49	120.175,80
600	545.096,88	330.680,25	81.539,97	124.879,10
800	570.489,32	361.503,63	88.725,07	106.465,00
1000	651.967,95	423.851,79	104.564,14	112.627,12

Tabelle 3.33: Die Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{+T}$

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	71.726,36	-82,33%	23.606,29	-87,02%	12.713,77	-82,51%	34.171,84	-75,33%
200	108.653,53	-83,44%	38.563,97	-87,58%	19.444,45	-84,49%	48.990,97	-76,02%
400	95.410,36	-86,92%	41.337,93	-89,16%	18.292,68	-87,89%	33.155,14	-81,25%
600	118.775,07	-85,46%	46.557,92	-89,36%	22.970,62	-86,58%	46.344,42	-75,12%
800	114.519,76	-86,30%	47.486,36	-89,62%	23.686,44	-86,85%	39.152,15	-77,87%
1000	100.626,55	-87,63%	45.972,90	-89,63%	23.843,95	-86,91%	26.116,86	-83,81%

Tabelle 3.34: Die Anzahl er erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{+T}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Die Tabellen 3.33 und 3.34, in denen die Anzahl der erzeugten Knoten dargestellt wird, zeigen ebenso gute Ergebnisse: die Anzahl der erzeugten Knoten sinkt im Vergleich zu den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ um 82,33% bis 87,63%, wodurch auch eine Verbesserung gegenüber den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ um durchschnittlich einen Prozentpunkt erreicht wird. In der Anzahl der unzulässigen, expandierten und Duplikat-Knoten spiegelt sich dieses Verhältnis wider, bei den Duplikat-Knoten kann jedoch im Vergleich zu den Testläufen mit der Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ sogar eine Reduktion von durchschnittlich 1,96 Prozentpunkten festgestellt werden.

Die Antwortzeiten in Tabelle 3.35 zeigen im Vergleich zu den Testläufen mit $\overrightarrow{C}^{\triangleright}$ ebenfalls beachtliche Verbesserungen: die durchschnittlichen Antwortzeiten werden um 80,53% bis 88,22% reduziert, die durchschnittlich maximalen Antwortzeiten sinken um 81,11% bis 89,25% und die global maximalen Antwortzeiten können um 65,11% bis 90,49% verringert werden. Im Vergleich mit den Testläufen zur Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ zeigt sich jedoch ein anderes Bild: die durchschnittlichen Antwortzeiten sind für beide Schätzfunktionen ähnlich niedrig, während sich bei den durchschnittlich maximalen Antwortzeiten die Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ in fünf der sechs Problemgruppen mit einer um durchschnittlich 0,14 Sekunden niedrigeren Antwortzeit überlegen zeigt. Die Testläufe unter Verwendung der Schätzfunktion $\overrightarrow{C}^{+T}$ erreichen allerdings in der Problemgruppe „400“ eine um 0,3 Sekunden niedrigere durchschnittliche maximale

Problem-	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
gruppe	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	37,08	12,50	-90,49%	3,29	1,11	-89,25%	0,11	0,04	-88,22%
200	20,28	20,28	-65,11%	3,50	1,76	-81,11%	0,08	0,05	-81,26%
400	75,76	17,92	-75,09%	6,57	0,92	-86,95%	0,08	0,02	-86,67%
600	35,25	34,03	-71,91%	4,96	1,98	-80,28%	0,05	0,03	-80,53%
800	23,80	20,48	-83,75%	3,76	1,50	-81,31%	0,03	0,03	-81,52%
1000	45,96	16,32	-66,57%	4,62	0,70	-85,24%	0,03	0,01	-86,72%

Tabelle 3.35: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{+T}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Antwortzeit als die Testläufe mit $\overrightarrow{C}^{\triangleright T}$. Bei den global maximalen Antwortzeiten erreicht die Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ in vier der sechs Problemgruppen bessere Zeiten, wohingegen mit der Schätzfunktion $\overrightarrow{C}^{+T}$ in Problemgruppe „400“ die global maximale Antwortzeit der Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ um 60% reduziert und somit eine um 10 Sekunden niedrigere maximale Antwortzeit über alle Problemgruppen erreicht werden kann.

3.4.3.3 Fazit

Durch die Überprüfung der Zeitfenster in Varianten der Schätzfunktionen konnten sehr gute Ergebnisse erzielt werden. Bei beiden Schätzfunktionen $\overrightarrow{C}^{\triangleright}$ und $\overrightarrow{C}^{+}$ konnte in diesen Varianten durch die zusätzliche Überprüfung, ob ausgehend von einem Knoten (Φ, μ^T) die obere Zeitfenstergrenze $\overline{t}_{p_i}$ bzw. $\overline{t}_{d_i}$ bei alleiniger Fertigstellung eines Auftrags i schon überschritten wird, und das Setzen von „Strafkosten“ für jeden Knoten mit einer Zeitfensterüberschreitung eine beachtliche Reduktion der Anzahl erzeugter Knoten und der Antwortzeiten von jeweils ca. 80% erreicht werden. Die Testläufe unter Verwendung der Schätzfunktion $\overrightarrow{C}^{+T}$ zeigen dabei – in Analogie zu den Testläufen der Varianten der Schätzfunktionen mit Servicezeiten – die besten Ergebnisse in Bezug auf die Anzahl der erzeugten Knoten. Bei den Antwortzeiten erreicht die Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ in den meisten Problemklassen etwas bessere Werte, während die Schätzfunktion $\overrightarrow{C}^{+T}$ die niedrigste globale Antwortzeit über alle Problemgruppen aufweisen kann.

Insgesamt findet der A*-Algorithmus bei Verwendung der Schätzfunktionen $\overrightarrow{C}^{+T}$ oder $\overrightarrow{C}^{\triangleright T}$ in deutlich weniger Schritten eine optimale Lösung. Eine Entscheidung, welche der beiden Schätzfunktionen besser geeignet wäre, kann an dieser Stelle jedoch nicht getroffen werden.

3.4.4 Varianten der Schätzfunktion durch Kombinationen bisheriger Schätzfunktionen

Bisher wurde für beide Schätzfunktionen „Maximalauftragszeit“ $\overrightarrow{C}^{\triangleright}$ und „Minimalzeitensumme“ $\overrightarrow{C}^{+}$ untersucht, wie sie den A*-Algorithmus beeinflussen, wenn entweder die Servicezeiten

oder die Zeitfenster in Varianten der Schätzfunktion eingehen. Da sich jedoch die Überprüfung der Zeitfenster und die Addition der Servicezeiten in einer Variante der Schätzfunktion nicht gegenseitig ausschließen, können auch beide Ansätze gleichzeitig in eine Variante der Schätzfunktion einfließen. Insbesondere könnte das Beachten der Einzel-Servicezeit eines Auftrags i während der Zeitfensterüberprüfung dazu führen, dass mehr verletzte Zeitfenster erkannt werden und daher im A*-Algorithmus weniger Knoten expandiert werden. In den Abschnitten 3.4.4.1 und 3.4.4.2 werden daher Varianten der Schätzfunktionen $\overrightarrow{C}^{\triangleright}$ und „Minimalzeitensumme“ $\overrightarrow{C}^{+}$ definiert und getestet, in die Kombinationen aus der Zeitfensterüberprüfung und der Addition der Servicezeiten eingehen.

Die Maximalauftragszeit $\overrightarrow{C}^{\triangleright}$ kann tendenziell eher bei wenigen noch zu bedienenden Aufträgen oder einer Häufung der noch anzufahrenden Orte eine gute Schätzung bilden, während die Minimalzeitensumme $\overrightarrow{C}^{+}$ bessere Ergebnisse bei vielen gleichverteilten Orten erreicht. Mit der Schätzfunktion $\overrightarrow{C}^{\triangleright}T$ können bessere Antwortzeiten erreicht werden, während mit $\overrightarrow{C}^{+T}$ die Anzahl der Knoten deutlich niedriger, wegen der komplexeren Berechnung von $\overrightarrow{C}^{+T}$ jedoch die Antwortzeit etwas höher ist. Es liegt daher nahe, eine Kombination aus beiden Schätzfunktionen zu bilden, um gegebenenfalls die Vorteile beider Funktionen nutzen zu können, falls der zusätzlich entstehende Rechenaufwand nicht die Vorteile kompensiert. Da bei der Berechnung der „Minimalzeitensumme mit Zeitfensterüberprüfung“ $\overrightarrow{C}^{+T}$ ein Teil der zur Berechnung der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ nötigen Rechenschritte ohnehin eingeht, wird sich die Erhöhung der Rechenzeiten in Grenzen halten. Diese Variante wird im Abschnitt 3.4.4.3 genauer betrachtet.

In Abbildung 3.9 werden alle Abschätzungen in einer Übersicht zusammengestellt.

3.4.4.1 Variante der Maximalauftragszeit mit Servicezeiten und Zeitfensterüberprüfung

In der Schätzfunktion „Maximalauftragszeit“, $\overrightarrow{C}^{\triangleright}$, wird für jeden noch zu bearbeitenden Auftrag i einzeln betrachtet, wie lange das Fahrzeug für die alleinige Vollendung dieses Auftrags benötigt. Für die Zeitfensterüberprüfung wird in $\overrightarrow{C}^{\triangleright T}$ ebenfalls für jeden Auftrag i berechnet, ob bei alleiniger Bedienung dieses Auftrags die oberen Zeitfenstergrenzen $\overline{t}_{p_i}$ bzw. $\overline{t}_{d_i}$ eingehalten werden können. Bei der Einbeziehung der Servicezeiten s_ν wurden zwei Varianten getestet: in der Schätzfunktion $\overrightarrow{C}^{\triangleright E}$ werden die Servicezeiten bei der Berechnung der maximalen für die Bedienung eines einzelnen Auftrags i benötigten Fahrzeit berücksichtigt, während in der $\overrightarrow{C}^{\triangleright S}$ die Summe der Servicezeiten aller noch zu bedienenden Orte $\nu \in V_{\Phi,\mu}^{Orte}$ zur maximalen Fahrzeit eines einzelnen Auftrags addiert wird.

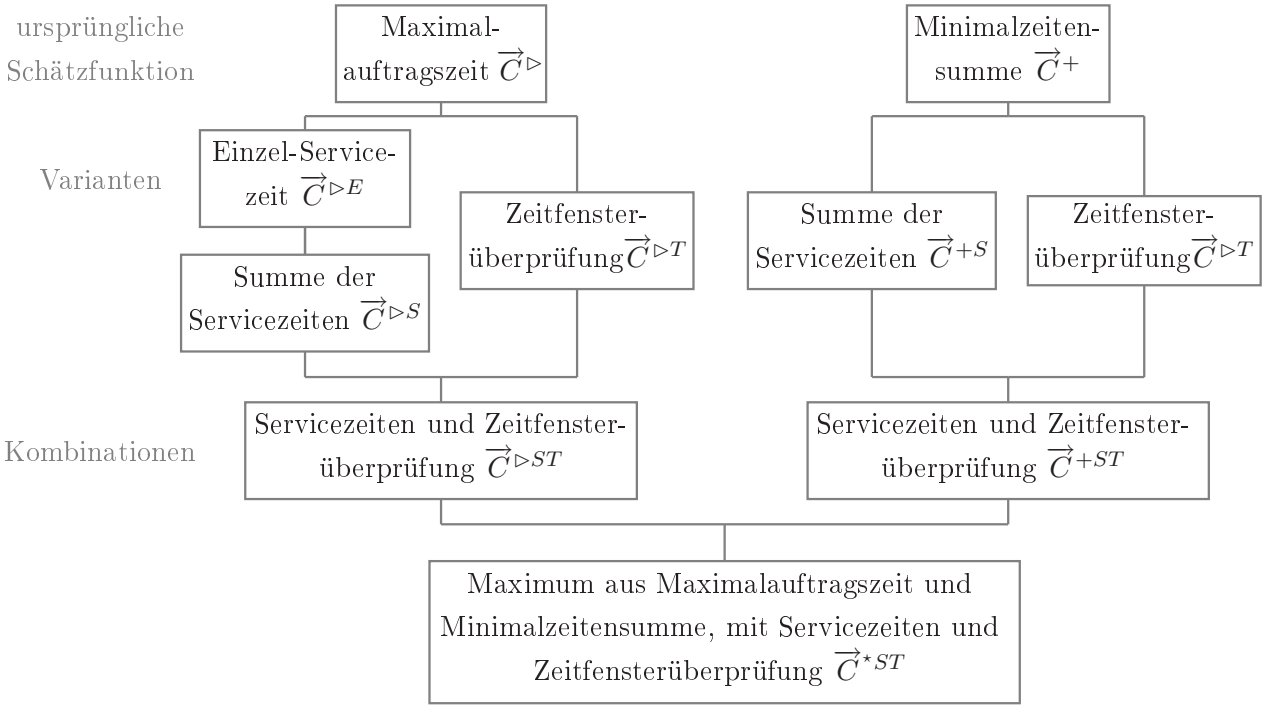

Abbildung 3.9: Schematische Übersicht aller getesteten Abschätzungen.

Betrachtet man die anfallende Servicezeit s_{p_i} eines einzelnen Auftrags i mit $\varphi(i) = 0$ bereits in der Zeitfensterüberprüfung, so könnten zusätzliche Zeitfensterverletzungen erkannt werden. Um jedoch die Summe der Servicezeiten $\sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} s_\nu$ trotzdem addieren zu können, wird die Einzel-Servicezeit nur zum Überprüfen der Zeitfenster genutzt, nicht jedoch in die Berechnung der maximalen für einen Auftrag i noch anfallenden Fahrzeit aufgenommen.

Die Schätzfunktion „*Maximalauftragszeit mit Servicezeiten und Zeitfensterüberprüfung*“ $\overrightarrow{C}^{\triangleright ST} : V^Z_{n,k^*} \times {}_{2n}V^Z_{n,k^*} \to \mathbb{R}^+$ gibt eine geschätzte minimale Fahrzeit von einem Knoten $(\Phi, \mu)^T \in V^Z_{n,k^*}$ zu einem der Zielknoten $(\Phi^*, \mu*)^T \in {}_{2n}V^Z_{n,k^*}$ zurück, wobei ein nicht einzuhaltendes Zeitfenster mindestens eines Auftrags i mit einem Rückgabewert von ∞ angezeigt wird. Dazu wird die für einen einzelnen Auftrag i zum Erreichen eines Zustandsvektors $(\Phi^*, \mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ verbleibende Fahrzeit definiert als:

$$\overrightarrow{C}_i^{\triangleright ST}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) = \begin{cases} 0, & \varphi(i) = \varphi^*(i) \\ c_{\mu,d_i}, & \varphi(i) = 1 \wedge \varphi^*(i) = 2 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,d_i} \leq \overline{t}_{d_i} \\ c_{\mu,p_i}, & \varphi(i) = 0 \wedge \varphi^*(i) = 1 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} \leq \overline{t}_{p_i} \\ c_{\mu,p_i} + c_{p_i,d_i}, & \varphi(i) = 0 \wedge \varphi^*(i) = 2 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} \leq \overline{t}_{p_i} \wedge \\ & \quad \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} + s_{p_i} + c_{p_i,d_i} \leq \overline{t}_{d_i} \\ \infty, & \text{sonst} \end{cases}$$

Die Funktion $\overrightarrow{C_i}^{\triangleright ST}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)$ unterscheidet sich nur in der Überprüfung der oberen Zeitfenstergrenze am Ort d_i für einen Auftrag i mit $\varphi(i)=0$ und $\varphi^*(i)=2$, $\overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,p_i}+s_{p_i}+c_{p_i,d_i}\geq \overline{t}_{d_i}$, von der Funktion $\overrightarrow{C_i}^{\triangleright T}$. Sie bildet so die noch verbleibende Fahrzeit unter Berücksichtigung der Zulässigkeit der Zeitfenster bei alleiniger Bedienung eins Auftrags i ab. Mit

$$\max_{i=1}^{n}\left\{\ \overrightarrow{C_i}^{\triangleright ST}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)\right\}\ +\sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu$$

wird somit eine Abschätzung für die Fahrzeit vom Knoten $(\Phi,\mu)^T$ zu einem Zielknoten $(\Phi^*,\mu^*)^T$ angegeben, die alle Servicezeiten der Orte $\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}$ beinhaltet und den Wert ∞ enthält, sobald mindestens ein Zeitfenster nicht eingehalten werden kann. Die Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$ wird als Minimum der geschätzten Fahrzeiten von $(\Phi,\mu)^T$ zu einem der Zielknoten $(\Phi^*,\mu^*)^T\in {}_{2n}V^Z_{n,k^*}$ definiert:

$$\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})=\min_{(\Phi^*,\mu^*)^T\in {}_{2n}V^Z_{n,k^*}}\left\{\max_{i=1}^{n}\ \overrightarrow{C_i}^{\triangleright ST}((\Phi,\mu)^T,(\Phi^*,\mu*)^T)\right\}\ +\sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu$$

Dann gilt:

Satz 3.12

Der A-Algorithmus findet mit der Schätzfunktion „Maximalauftragszeit mit Servicezeiten und Zeitfensterüberprüfung“ $\overrightarrow{C}^{\triangleright ST}$ einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}, falls es mindestens einen zulässigen Weg gibt. Das schnellstmögliche Auffinden der Lösung ist nicht garantiert.*

Beweis: Für die Optimalität der Lösung ist zu zeigen, dass der Wert der Schätzfunktion $\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T,{}_{2n}V^{Orte}_{\Phi,\mu})$ die tatsächlich noch anfallende Fahr- und Servicezeit von einem Knoten $(\Phi,\mu)^T$ zu einem Zielknoten $(\Phi^*,\mu^*)^T\in {}_{2n}V^Z_{n,k^*}$ nicht überschätzt. Analog zu den Beweisen von Satz 3.9 und Satz 3.10 kann $\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T,{}_{2n}V^{Orte}_{\Phi,\mu})$ geschrieben werden als:

$$\overrightarrow{C_i}^{\triangleright ST}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)=\begin{cases}0, & \varphi(i)=2\\ c_{\mu,d_i}, & \varphi(i)=1\wedge \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,d_i}\leq \overline{t}_{d_i}\\ c_{\mu,p_i}+c_{p_i,d_i}, & \varphi(i)=0\wedge \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,p_i}\leq \overline{t}_{p_i}\wedge\\ & \quad \overleftarrow{C}((\Phi,\mu)^T)+c_{\mu,p_i}+s_{p_i}+c_{p_i,d_i}\leq \overline{t}_{d_i}\\ \infty, & \text{sonst}\end{cases}$$

$$\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})=\max_{i=1}^{n}\overrightarrow{C_i}^{\triangleright ST}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T)+\sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu$$

Für den Fall $\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})<\infty$ gilt:

$$\begin{aligned}\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*}) &= \overrightarrow{C}^{\triangleright}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})+\sum_{\nu\in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} s_\nu\\ &= \overrightarrow{C}^{\triangleright S}((\Phi,\mu)^T,{}_{2n}V^Z_{n,k^*})\end{aligned}$$

Für $\overrightarrow{C}^{\triangleright}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})$ gilt nach Satz 3.9, dass die tatsächlich verbleibende Fahrzeit nicht überschätzt wird.

Falls $\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \infty$ gilt, ist zu zeigen, dass es keinen Weg von Knoten $(\Phi,\mu)^T$ zu einem der Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ geben kann. Analog zum Beweis von Satz 3.10 wird dies durch einen Widerspruchsbeweis gezeigt. Wegen $\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \infty$ gibt es einen Auftrag i' mit $\overrightarrow{C_i}^{\triangleright T}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) = \infty$ und einen Ort $\nu_{i'}$ mit $\overleftarrow{C}((\Phi,\mu)^T) + \overrightarrow{C_{i'}}^{\triangleright}((\Phi,\mu)^T,(\Phi^*,\mu^*)^T) > \overline{t}_{\nu_{i'}}$. Man betrachte den Fall, dass $\nu_{i'} \in D$ ein Delivery-Ort ist, d. h. die obere Grenze $\overline{t}_{d_i}$ des Delivery-Zeitfensters von Ort i' verletzt wird. Angenommen, es gäbe einen Weg vom Knoten $(\Phi,\mu)^T$ zu einer der Senken $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$, der alle Zeitfensterrestriktionen einhält. Dann gibt es auch eine Route $(\mu,\nu_1,\ldots,\nu_l)$ von der Fahrzeugposition μ durch alle noch anzufahrenden Orte $\nu_1,\ldots,\nu_l \in V^{Orte}_{\Phi,\mu}$, die die Zeitfensterrestriktionen einhält. Man betrachte nun die Teilroute $(\mu,\nu_1,\ldots,d_{i'})$. Offensichtlich muss gelten, dass die Fahr- und Servicezeit dieser Teilroute kleiner als $\overline{t}_{d_{i'}}$ ist, so dass gilt:

$$\begin{aligned} c^*_{\nu_1} + s^*_{\nu_1} + c^*_{\nu_2} + s^*_{\nu_2} + \cdots + c^*_{d_{i'}} &\leq \overline{t}_{d_{i'}} \\ &< \begin{cases} c_{\mu,d_{i'}}, & \varphi(i') = 1 \\ c_{\mu,p_{i'}} + s_{p_{i'}} + c_{p_{i'},d_{i'}}, & \varphi(i') = 0 \end{cases} \end{aligned}$$

Das steht jedoch im Widerspruch zur geltenden Dreiecksungleichung für die Fahrzeit zwischen je zwei Orten.

Für den Fall, dass $\nu_{i'} \in P$ ein Pickup-Ort ist, kann der Nachweis analog geführt werden. Somit ist gezeigt, dass die Schätzfunktion $\overrightarrow{C_i}^{\triangleright T}$ die Fahrzeit eines zulässigen Wegs nicht überschätzt.

Die für das schnellstmögliche Finden einer Lösung benötigte Gültigkeit der Dreiecksungleichung ist nicht erfüllt. Man betrachte dazu das Gegenbeispiel aus Abschnitt 3.2.3 und den Spezialfall, dass bei allen noch anzufahrenden Orten ν die Servicezeit $s_\nu = 0$ gilt und das Zeitfenster $[\underline{t}_\nu, \overline{t}_\nu]$ dem Planungshorizont entspricht.

□

Die Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$ kann daher im A*-Algorithmus eingesetzt werden, ohne die Optimalität der Lösung zu gefährden. Wenn mit dieser Schätzfunktion in erheblich mehr Knoten $(\Phi,\mu)^T$ als Zeitfensterverletzungen bemerkt werden, als mit der Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ oder sich die Addition der Summe der Servicezeiten auch in diesem Fall positiv auswirkt, könnte die Zahl der Knoten und damit die Antwortzeit deutlich gesenkt werden.

Problem-		Aufträge		gelöste Instanzen	
gruppe	**Aufträge**	**absolut**	**Δ**	**absolut**	**Δ**
100	51,86	41,20	23,17%	40	42,86%
200	103,08	72,17	63,77%	37	164,29%
400	205,66	128,69	96,46%	31	287,50%
600	308,97	182,68	148,27%	32	700,00%
800	411,75	239,75	192,86%	30	- [9]
1000	514,98	279,29	203,58%	27	- [9]

Tabelle 3.36: Anzahl der bearbeiteten Aufträge und vollständig gelöster Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Ergebnisse der Testläufe

Für diese Testläufe werden die in Abschnitt 3.4.1 beschriebenen Einstellungen übernommen. Als Vergleichswerte dienen weiterhin die Ergebnisse der Testläufe aus Abschnitt 3.4.1 unter Verwendung des Zustandsgraphen und der Schätzfunktion $\overrightarrow{C}^{\triangleright}$.

In Tabelle 3.36 kann eine beachtliche Steigerung der Anzahl bearbeiteter Aufträge und gelöster Instanzen im Vergleich zur Schätzfunktion $\overrightarrow{C}^{\triangleright}$ erkannt werden: Bei der Anzahl bearbeiteter Aufträge werden Steigerungen um 23,17% bis 203,58% erzielt, für die Anzahl der gelösten Instanzen werden Zuwächse von 42,86% bis 700% verzeichnet: in allen Problemgruppen können jeweils mindestens 27 Instanzen gelöst werden, wohingegen bei den Testläufen unter Verwendung von $\overrightarrow{C}^{\triangleright}$ in den Problemgruppen „800“ und „1000“ keine Instanz gelöst werden kann. Auch im Vergleich zu den Testläufen bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+T}$, bei denen bisher die größte Anzahl bearbeiteter Aufträge und gelöster Instanzen erreicht wurde, können hier bessere Werte erzielt werden: die Anzahl der pro Problemgruppe bearbeiteten Aufträge kann durchschnittlich um 4,44 Aufträge gesteigert werden, bei den Instanzen wird ein Anstieg von durchschnittlich 1,33 zusätzlich gelösten Instanzen verzeichnet.

Ähnliche Verbesserungen sind erwartungsgemäß in der Anzahl der erzeugten Knoten zu erkennen, die in den Tabellen 3.37 und 3.38 dargestellt werden: die Anzahl der erzeugten Knoten kann im Vergleich mit den Testläufen zur Schätzfunktion $\overrightarrow{C}^{\triangleright}$ um 85,68% bis 88,77% gesenkt werden, die Anzahl der als unzulässig erkannten Knoten wird sogar um bis zu 91,37% verringert. Auch hier können die bisher besten Ergebnisse der Testläufe zur Schätzfunktion $\overrightarrow{C}^{+T}$ weiter verbessert werden: in allen Problemgruppen werden durchweg weniger Knoten erzeugt: bei der Anzahl erzeugter Knoten kann eine Reduktion von durchschnittlich 2,88 Prozentpunkten gegenüber den Testläufen zur Schätzfunktion $\overrightarrow{C}^{+T}$ erreicht werden, bei der Anzahl der Duplikat-Knoten wird sogar eine Verringerung um 5,02 Prozentpunkten erzielt.

Die in Tabelle 3.31 dargestellten Antwortzeiten zeigen ebenfalls eine beachtliche Verbes-

[9] Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Problem-gruppe	♯ Knoten absolut erzeugt	unzulässig	expandiert	Duplikate
100	203.287,43	101.694,95	26.316,41	72.398,93
200	366.439,72	203.595,88	49.747,73	107.534,45
400	447.414,24	270.273,22	62.032,49	107.102,17
600	514.451,50	312.875,70	76.323,45	114.425,52
800	570.810,93	349.313,61	88.450,15	112.732,86
1000	654.875,40	424.319,28	103.593,90	112.410,74

Tabelle 3.37: Die Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-gruppe	erzeugt ♯ Vgl.	Δ	unzulässig ♯ Vgl.	Δ	expandiert ♯ Vgl.	Δ	Duplikate ♯ Vgl.	Δ
100	57.811,20	-85,76%	18.398,29	-89,88%	9.956,18	-86,31%	28.133,34	-79,69%
200	93.999,17	-85,68%	34.038,80	-89,04%	16.175,92	-87,10%	41.753,80	-79,56%
400	81.903,98	-88,77%	36.497,42	-90,43%	15.309,47	-89,86%	27.223,27	-84,60%
600	108.379,12	-86,73%	43.602,00	-90,04%	20.682,22	-87,92%	40.912,40	-78,03%
800	96.211,56	-88,49%	39.466,69	-91,37%	20.144,71	-88,81%	32.037,05	-81,89%
1000	96.453,09	-88,15%	42.657,21	-90,38%	22.784,86	-87,49%	25.881,36	-83,96%

Tabelle 3.38: Die Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-gruppe	global max. Antwortzeit sec	Vgl	Δ	Ø max. Antwortzeit sec	Vgl	Δ	Ø Antwortzeit sec	Vgl	Δ
100	54,12	10,08	-92,33%	3,22	0,77	-92,57%	0,10	0,03	-92,38%
200	109,79	19,95	-65,67%	5,31	1,41	-84,86%	0,09	0,04	-86,03%
400	68,75	10,18	-85,84%	5,53	0,64	-90,98%	0,05	0,01	-91,46%
600	30,21	21,99	-81,84%	3,48	1,43	-85,81%	0,03	0,02	-86,64%
800	249,71	15,71	-87,53%	8,01	0,93	-88,41%	0,04	0,02	-88,58%
1000	42,78	19,76	-59,53%	3,47	0,60	-87,29%	0,02	0,01	-88,48%

Tabelle 3.39: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{\triangleright ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

serung im Vergleich zu den Testläufen bei Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright}$: die durchschnittlichen Antwortzeiten sinken um 86,03% bis 92,38%, die durchschnittlich maximalen Antwortzeiten werden um 84,86% bis 92,57% verringert und die global maximalen Antwortzeiten können um 59,53% bis 92,33% reduziert werden. Im Vergleich zu den bisher besten erreichten Antwortzeiten in den Testläufen bei Verwendung der Schätzfunktionen $\overrightarrow{C}^{\triangleright T}$ bzw. $\overrightarrow{C}^{+T}$ werden bei den durchschnittlichen und durchschnittlich maximalen Antwortzeiten durchweg etwas bessere Ergebnisse erzielt, wobei die Reduktion im Bereich von weniger als $\frac{1}{100}$ Sekunde bzw. von maximal 0,43 Sekunden liegen. Bei den global maximalen Antwortzeiten werden nur in den Problemgruppen „200“ und „1000“ die bisher besten Ergebnisse nicht unterboten: in der Problemgruppe „200“ wird eine global maximale Antwortzeit von 19,95 Sekunden erreicht und damit die global maximale Antwortzeit dieser Problemgruppe bei den Testläufen zur Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ um 0,47 Sekunden überboten, während in der Problemgruppe „1000“ die bisher beste global maximale Antwortzeit bei Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright T}$ von 16,32 Sekunden gleich um 3,54 Sekunden verfehlt wird. Trotz dieser Steigerungen kann im Durchschnitt über alle Problemgruppen eine Reduktion von durchschnittlich 2,58 Sekunden gegenüber den bisher besten erzielten Ergebnissen erzielt werden. Dazu tragen insbesondere die Ergebnisse der Problemgruppen „400“ und „600“ bei: hier können die bisher besten global maximalen Antwortzeiten um 7,74 bzw. 8,95 Sekunden verringert werden.

3.4.4.2 Variante der Minimalzeitensumme mit Servicezeiten und Zeitfensterüberprüfung

Die Schätzfunktion „Minimalzeitensumme“ $\overrightarrow{C}^{+}$ gibt die Summe der für jeden noch anzufahrenden Ort $\nu \in V_{\Phi,\mu}^{Orte} \setminus \{\mu\}$ benötigten minimalen Fahrzeit von einem anderen Ort $\nu' \in V_{\Phi,\mu}^{Orte} \setminus \{\mu, \nu\}$ an. Für die Zeitfensterüberprüfung wird in $\overrightarrow{C}^{+T}$ zusätzlich für jeden Auftrag i berechnet, ob bei alleiniger Bedienung dieses Auftrags die oberen Zeitfenstergrenzen $\overline{t}_{p_i}$ bzw. $\overline{t}_{d_i}$ eingehalten werden können. Auch hier könnte von der Einbeziehung der Einzel-Servicezeiten in die Zeitfensterüberprüfung profitiert werden, wie sie im vorhergehenden Abschnitt beschrieben wurde. Für die Einbeziehung der Servicezeiten s_ν in die Variante $\overrightarrow{C}^{+S}$ der Schätzfunktion $\overrightarrow{C}^{+}$ wurde die Summe der Servicezeiten aller noch zu bedienenden Orte $\nu \in V_{\Phi,\mu}^{Orte}$ zur Summe der minimalen Fahrzeiten zu den noch anzufahrenden Orten $\nu \in V_{\Phi,\mu}^{Orte} \setminus \{\mu\}$ addiert.

Die Schätzfunktion „*Minimalzeitensumme mit Servicezeiten und Zeitfensterüberprüfung*“ $\overrightarrow{C}^{+ST} : V_{n,k^*}^{Z} \times {}_{2n}V_{n,k^*}^{Z} \rightarrow \mathbb{R}^{+}$ gibt die geschätzte minimale Fahrzeit von einem Knoten $(\Phi, \mu)^T \in V_{n,k^*}^{Z}$ zu einem der Zielknoten $(\Phi^*, \mu^*)^T \in {}_{2n}V_{n,k^*}^{Z}$ zurück, wobei die Summe der Servicezeiten in die Schätzfunktion eingeht und ein nicht mehr einzuhaltendes Zeitfenster mindestens eines Auftrags i bei alleiniger Betrachtung dieses Auftrags bei Beachtung der Servicezeiten mit einem Rückgabewert von ∞ angezeigt wird. Ausgehend vom Zustandsvektor $(\Phi, \mu)^T$ wird daher nun für jeden Auftrag i die minimale Fahrzeit zu einem Zustandsvektor

$(\Phi^*, \mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ definiert als:

$$\overrightarrow{C}_i^{+ST}\left(\binom{\Phi}{\mu},\binom{\Phi^*}{\mu^*}\right)=$$

$$\begin{cases} 0, & \varphi(i) = \varphi^*(i) \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j),\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\} + s_{p_i}, & \varphi(i) = 0 \wedge \varphi^*(i) = 1 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} \leq \overline{t}_{p_i} \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j),\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu d_i}, c_{p_j d_i}, c_{d_l d_i}\} + s_{d_i}, & \varphi(i) = 1 \wedge \varphi^*(i) = 2 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,d_i} \leq \overline{t}_{d_i} \\ \min\limits_{\substack{0=\varphi(j)<\varphi^*(j)\\ \varphi(l)<\varphi^*(l)=2}} \{c_{\mu p_i}, c_{p_j p_i}, c_{d_l p_i}\} + s_{p_i} & \\ \quad + \min\limits_{\substack{0=\varphi(j)<\varphi^*(j)\\ \varphi(l)<\varphi^*(l)=2}} \{c_{p_j d_i}, c_{d_l d_i}\} + s_{d_i}, & \varphi(i) = 0 \wedge \varphi^*(i) = 2 \wedge \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} \leq \overline{t}_{p_i} \wedge \\ & \qquad \overleftarrow{C}((\Phi,\mu)^T) + c_{\mu,p_i} + s_{p_i} + c_{p_i,d_i} \leq \overline{t}_{d_i} \\ \infty, & \text{sonst} \end{cases}$$

Analog zu vorhergehenden Definitionen gibt die Summe

$$\sum_{i=1}^{n} \overrightarrow{C}_i^{+ST}((\Phi,\mu)^T, (\Phi^*,\mu^*)^T)$$

einen Wert für eine geschätzte minimale Fahr- und Servicezeit von $(\Phi,\mu)^T$ zum Zielknoten $(\Phi^*,\mu^*)^T$ an, wobei ein nicht einzuhaltendes Zeitfenster den Wert ∞ herbeiführt. Die Schätzfunktion $\overrightarrow{C}^{+ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})$ wird als Minimum der zu jedem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ geschätzten minimalen Fahrzeit $\overrightarrow{C}^{+T}((\Phi,\mu)^T, \{(\Phi^*,\mu^*)^T\})$ definiert:

$$\overrightarrow{C}^{+ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) = \min_{(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}} \{\sum_{i=1}^{n} \overrightarrow{C}_i^{+ST}((\Phi,\mu)^T, (\Phi^*,\mu^*)^T)\}$$

Für die Schätzfunktion $\overrightarrow{C}^{+ST}$ gilt:

Satz 3.13
Der A-Algorithmus findet bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+ST}$, „Minimalzeitensumme mit Servicezeiten und Zeitfensterüberprüfung“, schnellstmöglich einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}.*

Beweis: Zum Beweis der Optimalität der Lösung wird $\overrightarrow{C}^{+ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})$ mit Hilfe der Schätzfunktion $\overrightarrow{C}^{+T}$ und der Summe der Servicezeiten dargestellt, so dass gilt:

$$\begin{aligned} \overrightarrow{C}^{+ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) &= \overrightarrow{C}^{+T}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) + \sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} s_\nu \\ &\leq \sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} c^*_\nu + \sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} s_\nu \\ &= \sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} (c^*_\nu + s_\nu) \end{aligned}$$

Für die Schätzfunktion $\overrightarrow{C}^{+T}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})$ gilt, dass sie die noch verbleibende Fahrzeit $c^*_{v_1} + c^*_{v_2} + \cdots + c^*_{v_l}$ einer optimalen Route nicht überschätzt (vgl. den Beweis zu Satz 3.11). Da die Servicezeiten aller Orte $\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}$ einzuhalten sind, kann $\overrightarrow{C}^{+ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})$ die verbleibende Fahr- und Servicezeit ebenfalls nicht überschätzen.

Für den Beweis der Dreiecksungleichung

$$\overrightarrow{C}^{+ST}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \overrightarrow{C}^{+ST}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) \geq \overrightarrow{C}^{+ST}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*})$$

wird ausgenutzt, dass die Dreiecksungleichung für $\overrightarrow{C}^{+T}$ gilt:

$$\begin{aligned}
&\overrightarrow{C}^{+ST}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \overrightarrow{C}^{+ST}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) \\
&\quad = \overrightarrow{C}^{+T}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \sum_{\substack{\nu \in (V^{Orte}_{\Phi,\mu} \setminus \{\mu\}) \setminus \\ (V^{Orte}_{\Phi',\mu'} \setminus \{\mu'\})}} s_\nu + \overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) + \sum_{\nu \in V^{Orte}_{\Phi',\mu'} \setminus \{\mu'\}} s_\nu \\
&\quad = \overrightarrow{C}^{+T}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \overrightarrow{C}^{+T}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) + \sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} s_\nu \\
&\quad \geq \sum_{i=1}^{n} \overrightarrow{C}_i^{+}((\Phi,\mu)^T, \{(\Phi^*_{2n}, \mu^*_{2n})^T\}) + \sum_{\nu \in V^{Orte}_{\Phi,\mu} \setminus \{\mu\}} s_\nu \\
&\quad = \overrightarrow{C}^{+ST}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*})
\end{aligned}$$

□

Die Minimalzeitensumme mit Servicezeiten und Zeitfensterüberprüfung kann also im A*-Algorithmus eingesetzt werden, ohne die Optimalität der Lösung oder das schnellstmögliche Finden einer Lösung zu gefährden.

Ergebnisse der Testläufe

Für diese Testläufe werden die in Abschnitt 3.4.1 beschriebenen Einstellungen übernommen. Als Vergleichswerte dienen weiterhin die Ergebnisse der Testläufe aus Abschnitt 3.4.1 unter Verwendung des Zustandsgraphen und der Schätzfunktion $\overrightarrow{C}^{\triangleright}$.

Tabelle 3.40 zeigt wiederum eine beachtliche Steigerung der Anzahl bearbeiteter Aufträge und gelöster Instanzen im Vergleich zur Schätzfunktion $\overrightarrow{C}^{\triangleright}$: Die Anzahl bearbeiteter Aufträge steigt um 23,38% bis 210,19%, für die Anzahl der gelösten Instanzen werden Steigerungen von 42,86% bis 725% erreicht. Im Vergleich zu den Testläufen bei Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$, die bisher die größte Anzahl bearbeiteter Aufträge und gelöster Instanzen erzielten, können mit der Schätzfunktion $\overrightarrow{C}^{+ST}$ in allen Problemklassen im Durchschnitt 3,00 zusätzliche Aufträge bearbeitet werden, bei der Anzahl gelöster Instanzen wird in drei Problemgruppen jeweils eine zusätzliche Instanz gelöst.

Problem-		Aufträge		gelöste Instanzen	
gruppe	Aufträge	absolut	Δ	absolut	Δ
100	51,86	41,27	23,38%	40	42,86%
200	103,08	73,05	65,77%	38	171,43%
400	205,66	130,02	98,47%	32	300,00%
600	308,97	188,73	156,49%	33	725,00%
800	411,75	243,36	197,27%	30	- ^{10}
1000	514,98	285,38	210,19%	27	- ^{10}

Tabelle 3.40: Anzahl der bearbeiteten Aufträge und vollständig gelösten Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{+ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-	♯ Knoten absolut			
gruppe	erzeugt	unzulässig	expandiert	Duplikate
100	219.593,18	108.642,21	27.281,07	78.190,30
200	359.345,72	198.824,42	46.779,33	102.240,35
400	437.736,73	260.824,80	59.817,25	104.660,59
600	516.269,53	312.525,27	75.389,88	111.361,42
800	549.307,93	350.092,42	83.924,98	92.775,59
1000	643.374,40	418.981,72	101.133,67	103.253,26

Tabelle 3.41: Anzahl der Knoten bei Verwendung der Schätzfunktion $\vec{C}^{+ST}$

Problem-	erzeugt		unzulässig		expandiert		Duplikate	
gruppe	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	53.727,95	-86,77%	16.992,29	-90,65%	9.186,09	-87,37%	26.166,59	-81,11%
200	82.841,75	-87,38%	30.213,17	-90,27%	14.121,10	-88,74%	36.505,62	-82,13%
400	72.539,71	-90,06%	32.657,46	-91,44%	13.700,36	-90,93%	23.777,83	-86,55%
600	97.554,90	-88,06%	38.560,32	-91,19%	18.831,40	-89,00%	37.027,13	-80,12%
800	81.479,95	-90,25%	35.454,88	-92,25%	17.637,56	-90,21%	23.990,73	-86,44%
1000	81.418,79	-89,99%	37.577,43	-91,52%	20.097,74	-88,97%	18.800,67	-88,35%

Tabelle 3.42: Anzahl der Knoten bei Verwendung der Schätzfunktion $\vec{C}^{+ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-gruppe	global max. Antwortzeit sec	Vgl	Δ	Ø max. Antwortzeit sec	Vgl	Δ	Ø Antwortzeit sec	Vgl	Δ
100	124,36	10,78	-91,80%	6,24	0,86	-91,66%	0,19	0,03	-91,13%
200	98,82	18,37	-68,39%	5,78	1,40	-85,04%	0,11	0,04	-85,64%
400	66,50	10,37	-85,58%	5,94	0,58	-91,76%	0,07	0,01	-91,28%
600	32,47	22,70	-81,26%	4,07	1,49	-85,24%	0,04	0,03	-85,60%
800	36,77	19,84	-84,26%	3,65	0,88	-89,06%	0,03	0,01	-89,43%
1000	43,78	8,51	-82,58%	3,63	0,47	-89,99%	0,03	0,01	-90,78%

Tabelle 3.43: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{+ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Die Tabellen 3.41 und 3.42 zeigen für die Anzahl der erzeugten Knoten ebenfalls bedeutende Verbesserungen im Vergleich zur Schätzfunktion $\overrightarrow{C}^{\triangleright}$: die Anzahl der erzeugten Knoten kann um 86,77% bis 90,06% gesenkt werden, die Anzahl der Duplikat-Knoten wird bis zu 88,35% geringer. Die bisher besten Ergebnisse der Testläufe zur Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$ können weiter verbessert werden: in allen Problemgruppen werden durchweg weniger Knoten erzeugt. Bei der Anzahl erzeugter Knoten kann gegenüber den Testläufen zur Schätzfunktion $\overrightarrow{C}^{+T}$ eine Verringerung von durchschnittlich 1,49 Prozentpunkten erreicht werden, bei der Anzahl der Duplikat-Knoten wird sogar eine weitere Reduktion um 2,83 Prozentpunkte erzielt.

Im Vergleich zu den Testläufen bei Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ zeigen die in Tabelle 3.35 dargestellten Antwortzeiten ebenfalls deutliche Verbesserungen: die durchschnittlichen Antwortzeiten sinken um 85,60% bis 91,28%, die durchschnittlich maximalen Antwortzeiten werden um 85,04% bis 91,76% verringert und die global maximalen Antwortzeiten können um 58,39% bis 91,80% reduziert werden. Im Vergleich zu den bisher besten erreichten Antwortzeiten in den Testläufen bei Verwendung der Schätzfunktionen $\overrightarrow{C}^{\triangleright ST}$, $\overrightarrow{C}^{\triangleright T}$ und $\overrightarrow{C}^{+T}$ werden bei den durchschnittlichen Antwortzeiten nur bei den Problemgruppen „800“ und „1000“ leichte Verbesserungen erzielt, die Unterschiede zwischen den durchschnittlichen Antwortzeiten bei Verwendung der Schätzfunktionen $\overrightarrow{C}^{\triangleright ST}$ bzw. $\overrightarrow{C}^{+ST}$ liegen jedoch unter $\frac{1}{100}$ Sekunde. Bei den durchschnittlich maximalen Antwortzeiten werden mit der Schätzfunktion $\overrightarrow{C}^{+ST}$ in den Problemgruppen „200“, „800“ und „1000“ etwas niedrigere Zeiten erreicht, im Durchschnitt über alle Problemgruppen kann die durchschnittliche Antwortzeit um 0,02 Sekunden gesenkt werden. Für die global maximalen Antwortzeiten wird in den Problemgruppen „200“ und „1000“ das bisher beste Ergebnis um 1,11 bzw. sogar 7,81 Sekunden unterboten. Die durchschnittliche Verringerung der global maximalen Antwortzeit bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+ST}$ gegenüber der Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$ über alle Problemgruppen beträgt 0,52 Sekunden.

[10] Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Auffällig ist, dass die Schätzfunktionen $\overrightarrow{C}^{+ST}$ und $\overrightarrow{C}^{\triangleright ST}$ ihre besten Ergebnisse jeweils in unterschiedlichen Problemgruppen verzeichnen, was darauf schließen lässt, dass eine Kombination beider Schätzfunktionen zu einer weiteren Verbesserung der Ergebnisse führen könnte.

3.4.4.3 Variante der Schätzfunktion: Maximum aus Maximalauftragszeit und Minimalzeitensumme mit Servicezeiten und Zeitfensterüberprüfung

Mit den bisher untersuchten Varianten der Schätzfunktionen „Maximalauftragszeit“ $\overrightarrow{C}^{\triangleright}$ und „Minimalzeitensumme“ $\overrightarrow{C}^{+}$ konnte schon eine deutliche Reduktion der Laufzeit des A*- Algorithmus erreicht werden. Für beiden Schätzfunktionen werden in der Variante “mit Zeitfensterüberprüfung und Servicezeiten“ ähnlich gute Rechenzeiten erzielt, wobei die Schätzfunktionen auf unterschiedlichen Gebieten jeweils die niedrigsten Rechenzeiten erreichen. Für die Schätzfunktion „Maximalauftragszeit“ $\overrightarrow{C}^{\triangleright}$ und ihre Varianten wird jeweils die noch für nur einen Auftrag i benötigte Fahrzeit berechnet, so dass mit diesen Funktionen eher bei wenigen noch zu bedienenden Aufträgen oder bei einer Häufung der noch anzufahrenden Orte $\nu \in V_{\Phi,\mu}^{Orte}$ eine „gute“ Schätzung erreicht wird. In der Schätzfunktion „Minimalzeitensumme“ $\overrightarrow{C}^{+}$ und ihren Varianten hingegen werden die mindestens zur Erreichung jedes noch anzufahrenden Orts $\nu \in V_{\Phi,\mu}^{Orte}$ benötigten Fahrzeiten addiert, so dass bei vielen gleichverteilten Orten genauere Schätzungen berechnet werden als mit der Schätzfunktion „Maximalauftragszeit“. Wünschenswert wäre eine Schätzfunktion, die die positiven Eigenschaften sowohl der „Maximalauftragszeit“ als auch der „Minimalzeitensumme“ und ihrer Varianten kombiniert, ohne jedoch den Rechenaufwand zur Berechnung des Werts der Schätzfunktion deutlich zu erhöhen.

Um eine optimale Lösung garantieren zu können, darf die tatsächliche minimale Fahrzeit von einem Knoten $(\Phi,\mu)^T$ zu einer Senke $(\Phi^*,\mu^*)^T \in {}_{2n}V_{n,k^*}^Z$ des Zustandsgraphen G_{n,k^*}^Z nicht überschätzt werden. Eine Summe aus den Schätzfunktionen $\overrightarrow{C}^{+}$ und $\overrightarrow{C}^{\triangleright S}$ kann diese Bedingung nicht erfüllen. Zur Kombination beider Schätzfunktionen bzw. ihrer Varianten wird daher das Maximum beider Schätzfunktionen berechnet. Dabei genügt es, die Zeitfensterüberprüfung nur einmal durchzuführen, da die Einhaltung der oberen Zeitfenstergrenzen in beiden Varianten $\overrightarrow{C}^{+T}$ und $\overrightarrow{C}^{\triangleright T}$ identisch berechnet wird.

Die Schätzfunktion „*Maximum aus Maximalauftragszeit und Minimalzeitensumme mit Servicezeiten und Zeitfensterüberprüfung*“ $\overrightarrow{C}^{\star ST} : V_{n,k^*}^Z \times {}_{2n}V_{n,k^*}^Z \rightarrow \mathbb{R}^+$ gibt die geschätzte minimale Fahrzeit von einem Knoten $(\Phi,\mu)^T \in V_{n,k^*}^Z$ zu einem der Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V_{n,k^*}^Z$ zurück, wobei die Summe der Servicezeiten in die Schätzfunktion eingeht und ein nicht mehr einzuhaltendes Zeitfenster mindestens eines Auftrags i bei alleiniger Betrachtung dieses Auftrags bei Beachtung der Servicezeiten mit einem Rückgabewert von ∞ angezeigt wird. Für einen Knoten $(\Phi,\mu)^T \in V_{n,k^*}^Z$ und eine Menge von Zielknoten ${}_{2n}V_{n,k^*}^Z$ wird $\overrightarrow{C}^{\star ST}$ definiert als:

$$\overrightarrow{C}^{\star ST}((\Phi,\mu)^T, {}_{2n}V_{n,k^*}^Z) = \max\left\{\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T, {}_{2n}V_{n,k^*}^Z), \overrightarrow{C}^{+S}((\Phi,\mu)^T, {}_{2n}V_{n,k^*}^Z)\right\}$$

Für $\overrightarrow{C}^{\star ST}((\Phi,\mu)^T, {}_{2n}V_{n,k^*}^Z)$ gilt der folgende Satz:

Satz 3.14

Der A-Algorithmus findet mit der Schätzfunktion „Maximum aus Maximalauftragszeit und Minimalzeitensumme mit Servicezeiten und Zeitfensterüberprüfung“, $\overrightarrow{C}^{\star ST}$, einen fahrzeitminimalen Weg von der Quelle zu einer der Senken des Graphen G^Z_{n,k^*}, falls es mindestens einen zulässigen Weg gibt. Das schnellstmögliche Auffinden der Lösung ist nicht garantiert.*

Beweis: Für die Optimalität der Lösung ist zu zeigen, dass der Wert der Schätzfunktion $\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})$ die tatsächlich auf einer fahrzeitminimalen Route anfallende Fahr- und Servicezeit $\sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}}(c^*_\nu + s_\nu)$ von einem Knoten $(\Phi,\mu)^T$ zu einem Zielknoten $(\Phi^*,\mu^*)^T \in {}_{2n}V^Z_{n,k^*}$ nicht überschätzt. Nach Satz 3.7 und Satz 3.10 gilt:

$$\begin{aligned}\overrightarrow{C}^{+S}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) &\leq \sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} (c^*_\nu + s_\nu)\\ \overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) &\leq \sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} (c^*_\nu + s_\nu)\end{aligned}$$

Daher gilt auch für das Maximum dieser Schätzfunktionen:

$$\begin{aligned}\overrightarrow{C}^{\star ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}) &= \max\left\{\overrightarrow{C}^{\triangleright ST}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*}), \overrightarrow{C}^{+S}((\Phi,\mu)^T, {}_{2n}V^Z_{n,k^*})\right.\\ &\leq \sum_{\nu \in V^{Orte}_{\Phi,\mu}\setminus\{\mu\}} (c^*_\nu + s_\nu)\end{aligned}$$

Für das schnellstmögliche Auffinden einer Lösung müsste die Dreiecksungleichung

$$\overrightarrow{C}^{\star ST}((\Phi,\mu)^T, \{(\Phi',\mu')^T\}) + \overrightarrow{C}^{\star ST}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*}) \geq \overrightarrow{C}^{\star ST}((\Phi',\mu')^T, {}_{2n}V^Z_{n,k^*})$$

gelten. Da jedoch $\overrightarrow{C}^{\triangleright ST}$ die Dreiecksungleichung nicht unbedingt erfüllt, ist sie auch für die Schätzfunktion $\overrightarrow{C}^{\star ST}$ nicht erfüllt.

□

Die folgenden Ergebnisse der Testläufe werden zeigen, ob das Single Vehicle Routing bei Verwendung dieser Schätzfunktion $\overrightarrow{C}^{\star ST}$ für den A*-Algorithmus weiter beschleunigt werden kann.

Ergebnisse der Testläufe

Die in Abschnitt 3.4.1 beschriebenen Einstellungen werden weiterhin übernommen. Als Vergleichswerte dienen ebenfalls die Ergebnisse der Testläufe aus Abschnitt 3.4.1 unter Verwendung des Zustandsgraphen und der Schätzfunktion $\overrightarrow{C}^{\triangleright}$.

[11] Bisher war die Anzahl gelöster Instanzen Null, die prozentuale Veränderung kann daher nicht berechnet werden.

Problem-		Aufträge		gelöste Instanzen	
gruppe	**Aufträge**	**absolut**	**Δ**	**absolut**	**Δ**
100	51,86	41,27	23,38%	40	42,86%
200	103,08	73,07	65,81%	38	171,43%
400	205,66	130,07	98,55%	32	300,00%
600	308,97	190,27	158,57%	34	750,00%
800	411,75	244,42	198,57%	30	- [11]
1000	514,98	286,86	211,81%	28	- [11]

Tabelle 3.44: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Schätzfunktion $\vec{C}^{\star ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-	♯ Knoten absolut			
gruppe	**erzeugt**	**unzulässig**	**expandiert**	**Duplikate**
100	219.152,43	108.471,61	27.187,27	78.049,54
200	361.402,23	199.911,67	46.917,10	102.765,62
400	436.590,93	260.367,71	59.471,08	104.352,78
600	513.053,83	311.559,20	74.629,35	110.340,13
800	544.760,14	348.711,56	82.965,59	91.546,92
1000	647.684,22	422.092,24	101.357,29	103.698,40

Tabelle 3.45: Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\star ST}$

Auch in diesen Testläufen zeigt Tabelle 3.44 eine deutliche Steigerung der Anzahl bearbeiteter Aufträge und gelöster Instanzen im Vergleich zur Schätzfunktion $\overrightarrow{C}^{\triangleright}$. Die Anzahl der bearbeiteten Aufträge kann um 23,38% bis 211,84% gesteigert werden, während die Anzahl der gelösten Instanzen um 42,86% bis 750% erhöht werden kann. Im Vergleich zu den Testläufen bei Verwendung der Schätzfunktion $\overrightarrow{C}^{+ST}$ mit der bisher höchsten Anzahl bearbeiteter Aufträge und gelöster Instanzen werden mit der Schätzfunktion $\overrightarrow{C}^{\star ST}$ in allen Problemklassen im Durchschnitt 0,69 zusätzliche Aufträge bearbeitet und in zwei Problemgruppen jeweils eine zusätzliche Instanz gelöst.

In den Tabellen 3.45 und 3.46 kann anhand der Anzahl erzeugter Knoten ebenfalls eine bedeutende Verbesserungen im Vergleich zur Schätzfunktion $\overrightarrow{C}^{\triangleright}$ festgestellt werden: die Anzahl der erzeugten Knoten sinkt um 86,80% bis 90,12%. Die bisher niedrigste Anzahl erzeugter Knoten, die in den Testläufen zur Schätzfunktion $\overrightarrow{C}^{+ST}$ erzielt wurde, kann weiter unterboten werden: in allen Problemgruppen werden durchweg etwas weniger Knoten erzeugt, allerdings werden im Durchschnitt über alle Problemgruppen nur Einsparungen in Höhe von 0,06 Prozentpunkten erreicht.

Im Vergleich zu den Testläufen bei Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ zeigen die in Tabelle 3.35 dargestellten Antwortzeiten erwartungsgemäß beachtliche Verbesserungen: die durchschnittlichen Antwortzeiten sinken um 85,61% bis 91,32%, die durchschnittlich maxima-

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	53.568,77	-86,80%	16.912,00	-90,70%	9.145,54	-87,42%	26.136,29	-81,13%
200	82.352,70	-87,45%	30.103,50	-90,31%	14.001,63	-88,83%	36.372,32	-82,20%
400	72.099,53	-90,12%	32.435,73	-91,50%	13.605,32	-90,99%	23.676,63	-86,61%
600	97.332,38	-88,09%	38.459,83	-91,21%	18.776,12	-89,03%	36.991,73	-80,14%
800	80.761,15	-90,34%	35.289,14	-92,29%	17.482,29	-90,29%	23.784,07	-86,56%
1000	81.027,14	-90,04%	37.340,90	-91,58%	20.014,47	-89,01%	18.718,22	-88,40%

Tabelle 3.46: Anzahl der Knoten bei Verwendung der Schätzfunktion $\vec{C}^{\star ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

Problem-gruppe	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	122,08	10,93	-91,68%	6,11	0,86	-91,68%	0,18	0,03	-91,16%
200	94,58	18,39	-68,35%	5,71	1,41	-84,91%	0,11	0,04	-85,61%
400	68,37	11,27	-84,34%	6,03	0,59	-91,64%	0,06	0,01	-91,32%
600	31,08	22,43	-81,48%	4,00	1,49	-85,20%	0,04	0,03	-85,65%
800	36,14	18,99	-84,93%	3,59	0,85	-89,40%	0,03	0,01	-89,88%
1000	42,33	8,97	-81,64%	3,55	0,48	-89,88%	0,02	0,01	-90,91%

Tabelle 3.47: Die Antwortzeiten bei Verwendung der Schätzfunktion $\vec{C}^{\star ST}$ im Vergleich zur Verwendung der Schätzfunktion $\vec{C}^{\triangleright}$

len Antwortzeiten werden um 84,91% bis 91,68% verringert und die global maximalen Antwortzeiten können um 68,35% bis 91,68% reduziert werden. Vergleicht man aber die erzielten Antwortzeiten mit den bisher besten erreichten Antwortzeiten der Testläufen bei Verwendung der Schätzfunktionen $\overrightarrow{C}^{\triangleright ST}$ und $\overrightarrow{C}^{\triangleright ST}$, so können nur bei den durchschnittlichen Antwortzeiten der Problemgruppen „800“ und „1000“ minimale Reduktionen in Höhe von weniger als $\frac{1}{100}$ Sekunde erzielt werden. Bei den durchschnittlich maximalen Antwortzeiten werden mit der Schätzfunktion $\overrightarrow{C}^{\star ST}$ in den Problemgruppe „800“ eine um 0,03 Sekunden niedrigere durchschnittliche Antwortzeit als in den bisherigen Testläufen erreicht, im Durchschnitt über alle Problemgruppen kann die Zeit nur gegenüber den Testläufen zur Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$ weiter gesenkt werden. Bei den global maximalen Antwortzeiten wird kein neues Minimum in der Antwortzeit erreicht, jedoch liegen die mit der Schätzfunktion $\overrightarrow{C}^{\star ST}$ erreichten global maximalen Antwortzeiten immer zwischen dem besseren und dem schlechteren Wert der Testläufe zu den Schätzfunktionen $\overrightarrow{C}^{\triangleright ST}$ und $\overrightarrow{C}^{\triangleright ST}$, so dass ein ausgewogeneres Bild entsteht.

Da diese Testläufe keinen eindeutigen Schluss zulassen, welche der Schätzfunktionen $\overrightarrow{C}^{\triangleright ST}$, $\overrightarrow{C}^{+ST}$ und $\overrightarrow{C}^{\star ST}$ für den A*-Algorithmus zur Lösung des Single Vehicle Routings Problems auf dem Zustandsgraphen G^{Z}_{n,k^*} am besten geeignet ist, werden weitere Testläufe durchgeführt. Um zu erkennen, mit welcher Schätzfunktion auch bei „größeren“ Graphen gute Ergebnisse

Problemgruppe			♯ Aufträge		gelöste Instanzen	
Name	**Aufträge**	**Instanzen**	**abs**	**%**	**abs**	**%**
100	51,86	56	41,27	73,69%	40	71,43%
200	103,08	60	73,07	121,78%	38	63,33%
400	205,66	59	130,34	220,91%	32	54,24%
600	308,97	60	190,40	317,33%	34	56,67%
800	411,75	59	244,64	414,65%	30	50,85%
1000	514,98	58	287,26	495,27%	28	48,28%

Tabelle 3.48: Anzahl der bearbeiteten Aufträge und gelösten Instanzen bei Verwendung einer der Schätzfunktionen $\vec{C}^{\triangleright ST}$, $\vec{C}^{+ST}$ und $\vec{C}^{\star ST}$

Problem-	erzeugt			unzulässig		
gruppe	$\vec{C}^{\triangleright ST}$	$\vec{C}^{+ST}$	$\vec{C}^{\star ST}$	$\vec{C}^{\triangleright ST}$	$\vec{C}^{+ST}$	$\vec{C}^{\star ST}$
100	274.750,27	219.593,18	219.152,43	129.046,30	108.642,21	108.471,61
200	1.008.145,52	363.507,32	361.402,23	428.554,65	200.718,15	199.911,67
400	715.305,73	439.684,75	436.945,47	427.860,53	262.246,41	260.655,80
600	727.253,18	517.833,55	513.386,93	435.596,93	313.509,15	311.836,68
800	820.447,44	554.121,24	545.442,34	444.937,73	352.423,92	349.265,71
1000	815.667,74	658.383,84	648.447,90	508.312,07	429.801,62	422.753,59

Tabelle 3.49: Anzahl der erzeugten und der unzulässigen Knoten bei Verwendung der Schätzfunktionen $\vec{C}^{\triangleright ST}$, $\vec{C}^{+ST}$ und $\vec{C}^{\star ST}$ im Vergleich

erzeugt werden, werden die Testläufe mit anderen Vergleichswerten wiederholt: zu Grunde gelegt wird jetzt nicht mehr die Anzahl der Aufträge, die unter Verwendung der Schätzfunktion $\overrightarrow{C}^{\triangleright}$ bis zum Errechnen von einer Million erzeugter Knoten bearbeitet wurden, sondern die pro Testinstanz maximale Anzahl Aufträge, die bei einem der Testläufe zur Verwendung der Schätzfunktionen $\overrightarrow{C}^{\triangleright ST}$, $\overrightarrow{C}^{+ST}$ und $\overrightarrow{C}^{\star ST}$ erreicht wurden. In diesen Testläufen wurden die folgenden Ergebnisse erzielt:

In der Tabelle 3.48 wird die Anzahl der bearbeiteten Aufträge und gelösten Instanzen dargestellt. Da für die Testläufe festgelegt war, wie viele Instanzen bearbeitet werden sollten, sind hier keine Unterschiede darzustellen. Es ist jedoch zu erkennen, dass nun bis zur Grenze von einer Million Knoten immerhin 55,78% - 79,58% der Aufträge abgearbeitet wurden und immerhin 48,28% bis 71,43% der Instanzen gelöst werden.

Die Tabellen 3.49 und 3.50 zeigen die bei den Testläufen unter Verwendung der Schätzfunktionen $\overrightarrow{C}^{\triangleright ST}$, $\overrightarrow{C}^{+ST}$ und $\overrightarrow{C}^{starST}$ erreichte Anzahl der je Problemgruppe durchschnittlich erzeugten, als unzulässig erkannten, expandierten und Duplikat-Knoten. Hier ist deutlich zu erkennen, dass mit der Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$ in jeder Problemklasse sowohl für die Anzahl der erzeugten Knoten als auch für die Anzahl der unzulässigen, der expandierten und der Duplikat-Knoten die besten Ergebnisse erzielt werden. Die Ergebnisse der Schätzfunktion sind nur geringfügig schlechter, während die Ergebnisse der Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$ zum Teil

Problem-gruppe	expandiert			Duplikate		
	$\overrightarrow{C}^{\triangleright ST}$	$\overrightarrow{C}^{+ST}$	$\overrightarrow{C}^{\star ST}$	$\overrightarrow{C}^{\triangleright ST}$	$\overrightarrow{C}^{+ST}$	$\overrightarrow{C}^{\star ST}$
100	34.384,57	27.281,07	27.187,27	105.713,52	78.190,30	78.049,54
200	126.116,60	47.350,48	46.917,10	390.095,33	103.225,07	102.765,62
400	95.622,03	60.054,02	59.522,22	169.168,73	104.848,47	104.365,12
600	105.812,15	75.659,13	74.667,35	165.544,40	111.452,35	110.357,38
800	122.982,08	84.876,34	83.067,36	217.300,36	93.440,19	91.570,25
1000	128.030,93	103.072,81	101.440,40	156.360,02	104.872,17	103.715,34

Tabelle 3.50: Anzahl der erzeugten Knoten bei Verwendung der Schätzfunktionen $\vec{C}^{\triangleright ST}$, $\vec{C}^{+ST}$ und $\vec{C}^{\star ST}$ im Vergleich

Problem-gruppe	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
	$\overrightarrow{C}^{\triangleright ST}$	$\overrightarrow{C}^{+ST}$	$\overrightarrow{C}^{\star ST}$	$\overrightarrow{C}^{\triangleright ST}$	$\overrightarrow{C}^{+ST}$	$\overrightarrow{C}^{\star ST}$	$\overrightarrow{C}^{\triangleright ST}$	$\overrightarrow{C}^{+ST}$	$\overrightarrow{C}^{\star ST}$
100	1.307,61	124,59	120,26	30,33	6,21	6,07	0,78	0,19	0,18
200	21.200,26	99,03	100,57	359,32	5,94	5,84	17,03	0,12	0,11
400	8.246,95	66,33	67,78	146,00	5,95	5,94	1,18	0,07	0,06
600	76,87	32,88	31,44	5,23	4,08	3,99	0,05	0,04	0,04
800	4.260,45	37,17	37,76	97,75	3,69	3,60	0,45	0,03	0,03
1000	273,93	44,78	42,35	8,50	3,73	3,53	0,04	0,03	0,02

Tabelle 3.51: Die Antwortzeiten bei Verwendung der Schätzfunktionen $\vec{C}^{\triangleright ST}$, $\vec{C}^{+ST}$ und $\vec{C}^{\star ST}$ im Vergleich

sehr deutlich abweichen: in der Problemgruppe „200“ beträgt die durchschnittliche Anzahl erzeugter Knoten das 2,8-fache der Anzahl erzeugter Knoten bei Verwendung von $\overrightarrow{C}^{+ST}$.

Die mit den Schätzfunktionen $\overrightarrow{C}^{\triangleright ST}$, $\overrightarrow{C}^{+ST}$ und $\overrightarrow{C}^{\star ST}$ jeweils erreichten Antwortzeiten in den Problemgruppen sind in Tabelle 3.51 dargestellt. Es ist sofort zu erkennen, dass mit der Schätzfunktion $\overrightarrow{C}^{\triangleright ST}$ deutlich schlechtere Antwortzeiten erreicht werden als bei Verwendung von $\overrightarrow{C}^{+ST}$ oder $\overrightarrow{C}^{\star ST}$. In den Testläufen unter Verwebung der Schätzfunktion $\overrightarrow{C}^{+ST}$ können in den Problemgruppen „200“, „400“ und „800“ die besten global maximalen Antwortzeiten erzielt werden, in allen anderen Fällen werden die besten global maximalen, die besten durchschnittlich maximalen und die besten durchschnittlichen Antwortzeiten der Problemgruppen in den Testläufen unter Verwendung der Schätzfunktion $\overrightarrow{C}^{starST}$ erzielt. Dabei gilt, dass mit der Schätzfunktion $\overrightarrow{C}^{+ST}$ bei den durchschnittlichen Antwortzeiten Abweichungen um 12,88%, bei den durchschnittlich maximalen Antwortzeiten um maximal 5,46% und bei den global maximalen Antwortzeiten Abweichungen um maximal 5,73% von den besseren Werten je Problemgruppe der Testläufe mit $\overrightarrow{C}^{\star ST}$ erreicht wurden, während die Schätzfunktion $\overrightarrow{C}^{\star ST}$ nur um maximal 2,18% von den global maximalen Antwortzeiten der Problemgruppen, die mit der Schätzfunktion $\overrightarrow{C}^{+ST}$ bessere Werte erzielten, abweicht.

Diese Testläufe haben gezeigt, dass mit der Schätzfunktion $\overrightarrow{C}^{\star ST}$ durchschnittlich die schnellsten Lösungen des Single Vehicle Routing Problems zu erwarten sind. Alle folgenden Testläufe zu weitergehenden Verbesserungen werden daher mit dieser Schätzfunktion durchgeführt.

3.5 Fazit

In diesem Kapitel wurde das Single Vehicle Routing Problem und der von Caramia et al. entwickelte Algorithmus zu seiner optimalen Lösung vorgestellt. Aufbauend auf einer Analyse des in [11] verwendeten Statusvektorbaums zur Lösung des Single Vehicle Routing Problems mit Hilfe des A*-Algorithmus wird der Zustandsgraph entwickelt. Sowohl die theoretische Analyse des Zustandsgraphen als auch die Testläufe zeigen, dass mit Hilfe des A*-Algorithmus auf dem Zustandsgraphen deutlich schneller eine optimale Lösung für das Single Vehicle Routing Problem berechnet werden kann. Anschließend wird eine neue Schätzfunktion zur Verwendung im A*-Algorithmus entwickelt, die die Effizienz des Algorithmus weiter deutlich erhöht.

Es ist gezeigt worden, dass das Single Vehicle Routing Problem mit Hilfe des Zustandsgraphen und der entwickelten Schätzfunktion für den A*-Algorithmus wesentlich effizienter gelöst werden kann, so dass nun auch in das dynamische Vehicle Routing Problem mit Zeitfenstern eine optimale Lösung des Single Vehicle Routing Problems eingehen kann.

Kapitel 4

Eine Heuristik zur Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern

Nachdem im Kapitel 3 das Teilproblem des Single Vehicle Routings analysiert und ein Algorithmus zur schnellen Lösung dieses Problems zur Verfügung gestellt wurde, kann nun mit der Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern fortgefahren werden.

Im Entscheidungsproblem des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern wird von einem dynamischen Auftragseingang ausgegangen, d. h. die Aufträge gehen im Verlauf der Planungsperiode ein, ohne dass im Vorhinein Informationen über die Anzahl der eingehenden Aufträge, die zeitliche Verteilung der Zeitpunkte des Auftragseingangs, die geographische Verteilung der für die eingehenden Aufträge anzufahrenden Orte oder die Länge der Zeitfenster an diesen Orten bekannt sind. Ein zum Zeitpunkt τ_κ eingehender Auftrag κ kann akzeptiert werden, wenn ein zulässiger Routenplan zur Abarbeitung aller bisher akzeptierten Aufträge $i \in N^{\tau_\kappa}$ zuzüglich des neuen Auftrags κ existiert, ansonsten muss der Auftrag abgelehnt werden. Da im Entscheidungsproblem der Auftragseingang online erfolgt und innerhalb weniger Sekunden eine Entscheidung über die Akzeptanz oder die Ablehnung des Auftrags κ erfolgen muss, wird eine Heuristik zur Berechnung eines zulässigen Routenplans eingesetzt, die auf die optimale Lösung des Teilproblems „Single Vehicle Routing“ zurückgreift.

Die Beschränkung der Rechenzeit gilt jedoch nur für die Zeit bis zur Entscheidung, ob ein neu eingehender Auftrag κ akzeptiert oder abgelehnt wird. Ist diese Entscheidung gefallen und liegt kein neuer Auftragseingang vor, kann die Zeit bis zum Eingang des nächsten Auftrags κ'

genutzt werden, um die heuristische Zuordnung der noch zu bedienenden Aufträge $i \in N^{\tau_\kappa}$ zu den Fahrzeugen $f \in F$ zu verbessern. Dies geschieht mit Hilfe einer Tabu Suche, die in [26] eingeführt wurde. Auch die Tabu Suche benötigt die im vorhergehenden Kapitel beschriebene Lösung des Single Vehicle Routing Problems.

Die Heuristik zur Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems kann folgendermaßen skizziert werden (vgl. auch Abbildung 2.4):

1. Überprüfe, ob ein Auftragseingang vorliegt.

2. Falls ein Auftragseingang vorliegt: führe die Heuristik zur Berechnung eines zulässigen Routenplans durch.
   Sonst: Führe das Verfahren der Tabu Suche durch.

3. Falls das Ende des Planungshorizonts erreicht ist: Stopp. Sonst: Gehe zu Schritt 1.

Die Heuristik zur Berechnung eines zulässigen Routenplans wird in Abschnitt 4.1 vorgestellt. Da die Antwortzeiten auch von dieser Heuristik beeinflusst werden, werden weitere Strategien zur Reduzierung der Antwortzeiten vorgestellt und ihre Auswirkungen auf die Antwortzeiten anhand von umfangreichen Testläufen dokumentiert. Anschließend wird in Abschnitt 4.2 das in [26] entwickelte Verfahren der Tabu Suche erläutert, um anschließend den Algorithmus im dynamischen Umfeld testen zu können. Die umfangreichen Testläufe werden analysiert. Abschließend wird ein Fazit gezogen.

4.1 Die Heuristik zur Berechnung eines zulässigen Routenplans

Die Zuordnung der Aufträge zu den Fahrzeugen bildet den heuristischen Teil des Algorithmus zur Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern. Wegen der Komplexität des Entscheidungsproblems und der geforderten kurzen Rechenzeiten kann auf einen derartigen heuristischen Ansatz nicht verzichtet werden. Da das Teilproblem „Single Vehicle Routing“ exakt gelöst wird, nimmt die Berechnung von Lösungen des Teilproblems einen Großteil der zur Verfügung stehenden Rechenzeit in Anspruch. Trotz der im vorhergehenden Kapitel beschriebenen Möglichkeiten zur Beschleunigung der Lösung des Single Vehicle Routing Problems und den in den Testläufen erzielten deutlich verbesserten Laufzeiten, erreichen die maximalen Antwortzeiten immer noch zu hohe Werte für eine akzeptable Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern. Die Antwortzeiten können jedoch eventuell weiter deutlich verringert werden, wenn die Heuristik zur Berechnung eines zulässigen Routenplans den Algorithmus zur Lösung des Single Vehicle Routing Problems effizienter einsetzen könnte.

Dieser Abschnitt beschäftigt sich daher mit der Heuristik zur Berechnung eines zulässigen Routenplans, mit deren Hilfe eine Zuordnung der Aufträge zu den Fahrzeugen mit möglichst niedriger Summe der Fahrzeiten der für diese Zuordnung optimalen Routen aller Fahrzeuge $f \in$

F gefunden werden soll. Zunächst wird die Heuristik beschrieben, bevor mögliche Änderungen zur Reduktion der Antwortzeiten vorgestellt werden.

Die Heuristik zur Berechnung eines zulässigen Routenplans wird jeweils zum Zeitpunkt τ_κ des Eingang eines neuen Auftrags κ durchgeführt und berechnet eine Lösung für das statische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern (vgl. Abschnitt 2.3.3). Sie besteht aus einer „testweisen" Zuordnung des neuen Auftrags κ zu jedem der Fahrzeuge $f \in F$ unter Beibehaltung der bisherigen Zuordnung aller anderen bereits akzeptierten Aufträge $i \in N^\tau$ und der Berechnung einer optimalen zukünftigen Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ für jedes Fahrzeug $f \in F$, für das eine zukünftige Route zur Bedienung aller von Fahrzeug f zum Zeitpunkt τ_κ noch zu bedienenden Aufträge $i \in N_f^\tau$ zuzüglich des neuen Auftrags κ existiert. Wird für wenigstens ein Fahrzeug f eine zulässige Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ gefunden, so wird Auftrag Nr. κ demjenigen Fahrzeug f^* mit minimalen zusätzlichen Fahrzeiten zugeordnet, sonst wird er abgelehnt. Die zusätzlichen Fahrzeiten bestehen dabei aus der Differenz der Fahrzeit $C(\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}})$ der neuen Route bei Bedienung von Auftrag κ und der Fahrzeit $C(\overrightarrow{R}_{N_f^\tau})$ der bisherigen Route (vgl. Abschnitt 2.4).

Die Heuristik kann folgendermaßen skizziert werden:

1. Berechne für jedes Fahrzeug $f \in F$ eine neue optimale Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$, falls eine solche Route existiert
2. Wenn für kein Fahrzeug $f \in F$ eine zulässige Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ existiert: lehne Auftrag κ ab. Sonst gehe zu Schritt 3.
3. Berechne für jedes Fahrzeug $f \in F$ mit zulässiger Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ die zusätzliche Fahrzeit $C_f^\kappa = C(\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}) - C(\overrightarrow{R}_{N_f^\tau})$
4. Wähle das Fahrzeug $f^* \in \{f \in F \mid \exists \overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}\}$ mit minimaler zusätzlicher Fahrzeit C_f^κ und ordne Auftrag κ dem Fahrzeug f^* zu.
5. Akzeptiere Auftrag κ.

Diese Herangehensweise lässt einigen Spielraum, der zur Reduktion der Antwortzeiten genutzt werden kann. In diesem Abschnitt werden zwei verschiedene Ansätze betrachtet:

Einerseits könnte eine Identifikation von Fahrzeugen $f \in F$, für die die Existenz einer zulässigen Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ ausgeschlossen werden kann, die Antwortzeit senken, da die Durchführung des Algorithmus zur Lösung des Single Vehicle Routing Problems für diese Fahrzeuge entfallen kann. Andererseits kann die Antwortzeit möglicherweise gesenkt werden, indem die Entscheidung über Akzeptanz des Auftrags schon dann getroffen wird, sobald für ein Fahrzeug eine Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}, f \in F$ gefunden wurde, da ein neu eingegangener Auftrag κ genau dann akzeptiert wird, wenn für mindestens ein Fahrzeug f eine Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ gefunden wird. Das

Ablehnen des Auftrags kann allerdings auch weiterhin erst nach Überprüfung aller Fahrzeuge $f \in F$ auf zulässige Routen $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ geschehen.

Eine auf der Überprüfung von Zeitfenstern basierende Möglichkeit zum Auffinden von Fahrzeugen $f \in F$, für die die Existenz einer zulässigen Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ ausgeschlossen werden kann, wird in Abschnitt 4.1.1 beschrieben. Die Ergebnisse werden anhand von Testläufen dokumentiert. Abschnitt 4.1.2 beschäftigt sich anschließend mit der Möglichkeit der vorgezogenen Akzeptanz eines Auftrags. Dazu werden Sortierkriterien für die Fahrzeuge entwickelt, um den Algorithmus zur Lösung des Single Vehicle Routing Problems für ein Fahrzeug f in einer bestimmten Reihenfolge der Fahrzeuge f durchzuführen, so dass möglichst schnell die erste zulässige Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ gefunden und somit der Auftrag $i*$ akzeptiert wird. Die Auswirkungen der vorgezogenen Akzeptanz und der Einfluss der Sortierkriterien auf die Antwortzeit werden anhand von Testläufen bewertet und untereinander verglichen. Abschließend wird ein Fazit gezogen.

4.1.1 Beschleunigung durch Überprüfung der Zeitfensterkompatibilität

Dieser Abschnitt beschäftigt sich mit der Identifikation von Fahrzeugen $f \in F$, für welche die Existenz einer zulässigen Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ unter Einbeziehung des neuen Auftrags κ ausgeschlossen werden kann. Für diese Fahrzeuge $f \in F$ kann auf die Lösung des Single Vehicle Routing Problems verzichtet werden, so dass zum Teil erhebliche Rechenzeit eingespart werden kann. Die Zeitfenster der Aufträge $i \in N_f^\tau \cup \{\kappa\}$ sind für die Zulässigkeit einer Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ von großer Bedeutung. Je enger die Zeitfenster, desto weniger zulässige Routen gibt es für das Fahrzeug f. Wünschenswert wäre es, mit wenig Rechenaufwand Fahrzeuge $f \in F$ zu identifizieren, bei denen die Zeitfenster der noch zu bedienenden Aufträge $i \in N_f^\tau$ keine zulässige Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ bei der zusätzlichen Zuordnung des neuen Auftrags κ ermöglichen.

Die Frage, ob die Zeitfenster aller von einem Fahrzeug f noch zu bedienenden Aufträge $i \in N_f^\tau$ und die Zeitfenster des neuen Auftrags κ in einer Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ eingehalten werden können, kann nur durch die Lösung des Single Vehicle Routing Problems beantwortet werden. Es kann allerdings relativ einfach überprüft werden, ob die Zeitfenster $[\underline{t}_{p_\kappa}, \overline{t}_{p_\kappa}]$ und $[\underline{t}_{d_\kappa}, \overline{t}_{d_\kappa}]$ am Pickup-Ort p_κ bzw. am Delivery-Ort d_κ des neuen Auftrags κ mit den Zeitfenstern eines anderen vom Fahrzeug f noch zu bedienenden Auftrags $i \in N_f^\tau$ zu vereinbaren sind. Ist dies der Fall, so werden die Zeitfenster $[\underline{t}_{p_\kappa}, \overline{t}_{p_\kappa}]$, $[\underline{t}_{d_\kappa}, \overline{t}_{d_\kappa}]$ des Auftrags κ *kompatibel* zu den Zeitfenstern der noch zu bedienenden Aufträge $i \in N_f^\tau$ genannt. Für die Überprüfung der Zeitfensterkompatibilität müssen je nach Status $\varphi^\tau(i) \in \{0, 1\}$ des Auftrags i zum Zeitpunkt τ zwei Fälle unterschieden werden: Falls $\varphi^\tau(i) = 0$ gilt, müssen für die Bedienung der Aufträge i und κ die Pickup- und Delivery-Orte p_i, d_i, p_κ und d_κ angefahren werden, wobei die Reihenfolgebedingung „erst Pickup, dann Delivery“ für beide Aufträge eingehalten werden muss. Die Anzahl der möglichen Routen entspricht damit der Anzahl Wege von der Wurzel $\binom{0}{0}$ des Statusvektorbaums $G_{2,0}^S$ zur Menge der Blätter ${}_4V_{2,0}^S$. Für den Fall $\varphi^\tau(i) = 1$ müssen nur

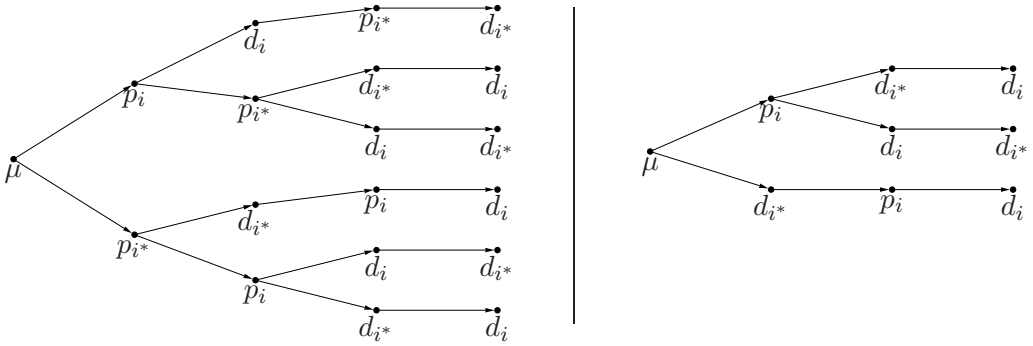

Abbildung 4.1: Entscheidungsbäume für die mögliche Anordnung der anzufahrenden Orte für zwei Aufträge i, i^* mit Status $\varphi(i^*) = 0$ und 1.

die Orte d_i, p_κ und d_κ betrachtet werden, die möglichen Routen entsprechen den Sequenzen der Fahrzeugpositionen, die auf einem Weg von der Wurzel $\binom{1}{0}$ eines Statusvektorbaums $G^S_{2,1}$ zur Menge der Blätter ${}_4V^S_{2,0}$ erreicht werden. Abbildung 4.1 zeigt die Sequenzen der Fahrzeugpositionen sowohl für einen Auftrag i mit $\varphi^\tau(i) = 0$ als auch für einen Auftrag i mit Status $\varphi^\tau(i) = 1$.

Die Überprüfung der Zeitfensterkompatibilität entspricht also der Suche nach einem zulässigen Weg von der Wurzel $\binom{0}{0}$ bzw. $\binom{1}{0}$ zu einem Blatt eines Statusvektorbaums $G^S_{2,0}$ bzw. $G^S_{2,1}$. Da es sich nur um Statusvektorbäume für zwei Aufträge handelt, kann die Überprüfung in sehr kurzer Zeit stattfinden: im schlechtesten Fall werden für einen Auftrag $i \in N^\tau_f$ mit $\varphi^\tau(i) = 0$ 18 Additionen der Fahr- und Servicezeiten und 18 Vergleiche mit einer zugehörigen oberen Zeitfenstergrenze benötigt, für einen Auftrag mit $\varphi^\tau(i) = 1$ sind höchstens 8 Additionen und 8 Vergleiche durchzuführen.

Für die Überprüfung der Kompatibilität der Zeitfenster eines neuen Auftrags κ mit den Zeitfenstern der noch zu bedienenden Aufträge $i \in N^\tau_f$ eines Fahrzeug f kann folgendermaßen vorgegangen werden:

i) Wähle einen Auftrag $i \in N^\tau_f$, der noch nicht überprüft wurde.

ii) Falls es keinen solchen Auftrag i gibt: Abbruch. Die Zeitfenster sind kompatibel.

iii) Suche einen zulässigen Weg von der Wurzel $\binom{\varphi^\tau(i)}{0}$ mit Fahrzeugposition μ zu einer der Senken im Graphen $G^S_{2,\varphi^\tau(i)}$.

iv) Falls es keinen solchen Weg gibt: Abbruch. Die Zeitfenster sind nicht kompatibel.

v) Gehe zu Schritt i).

Mit Hilfe der Überprüfung der Zeitfensterkompatibilität kann nun der Schritt 1 in der Heuristik zur Berechnung eines zulässigen Routenplans geändert werden in:

Problem-		Aufträge		gelöste Instanzen	
gruppe	Aufträge	absolut	Δ	absolut	Δ
100	51,86	41,27	0,00%	40	0,00%
200	103,08	73,08	0,02%	38	0,00%
400	205,66	130,69	0,48%	33	3,13%
600	308,97	192,32	1,08%	35	2,94%
800	411,75	251,02	2,70%	31	3,33%
1000	514,98	292,05	1,81%	29	3,57%

Tabelle 4.1: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Überprüfung der Zeitfensterkompatibilität im Vergleich zur bisherigen Heuristik

1. Führe für jedes Fahrzeug $f \in F$ aus:
   (a) Überprüfe die Zeitfensterkompatibilität des Auftrags κ zu den noch zu bedienenden Aufträgen $i \in N_f^\tau$ des Fahrzeugs f
   (b) Falls die Zeitfenster kompatibel sind: Berechne eine neue optimale Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$, falls sie existiert

Die Auswirkungen dieser Änderung wird im Folgenden anhand von Testläufen untersucht.

Ergebnisse der Testläufe

Für diese Testläufe wird die Testumgebung weiterhin ereignisdiskret eingestellt und die Grenze von einer Million maximal zu erzeugenden Knoten beibehalten. Zum Vergleich werden die Ergebnisse von Testläufen mit der Heuristik zur Berechnung eines zulässigen Routenplans ohne Überprüfung der Zeitfensterkompatibilität angegeben.

Tabelle 4.1 zeigt die Anzahl der bearbeiteten Aufträge und der gelösten Instanzen. Während in den Problemgruppen „100“ und „200“ die Zahl der bearbeiteten Aufträge beinahe gleich bleibt, kann in den Problemgruppen mit einer höheren Anzahl Aufträgen ein Zuwachs von bis zu 2,70% erreicht werden. Die größten Zuwächse werden in den Problemgruppen „800“ und „1000“ mit durchschnittlich 6,59 bzw. 5,19 zusätzlichen Aufträgen verzeichnet. Die Anzahl der gelösten Instanzen kann sogar in den Problemgruppen „400“„600“, „800“ und „1000“ um jeweils eins erhöht werden.

Für die in den Tabellen 4.2 und 4.3 dargestellte Anzahl der erzeugten, unzulässigen und expandierten Knoten können deutliche Reduzierungen verzeichnet werden, wobei die prozentuale Einsparungen mit der Anzahl Aufträge der Problemgruppen steigt: die Zahl der erzeugten Knoten kann in Problemgruppe „100“ um 0,52% gesenkt werden, in Problemgruppe „1000“ ist eine deutliche Reduktion von 27,63% zu konstatieren. Der kontinuierliche Anstieg der prozentualen Verbesserung lässt sich dadurch erklären, dass in den Problemgruppen mit vielen Aufträgen einerseits die Wahrscheinlichkeit des Verletzens der Zeitfensterkompatibilität

Problem-gruppe	♯ Knoten absolut			
	erzeugt	unzulässig	expandiert	Duplikate
100	218.022,36	107.649,20	26.920,30	78.039,84
200	351.812,00	191.904,27	45.376,67	102.623,73
400	415.194,80	242.844,80	55.808,98	103.869,88
600	444.307,98	257.347,05	63.639,93	107.157,37
800	460.072,32	278.096,34	68.861,95	90.959,76
1000	519.822,22	315.276,07	79.813,00	103.706,66

Tabelle 4.2: Anzahl der erzeugten Knoten bei Überprüfung der Zeitfensterkompatibilität

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl	Δ	♯ Vgl	Δ	♯ Vgl	Δ	♯ Vgl	Δ
100	218.022,36	-0,52%	107.649,20	-0,76%	26.920,30	-0,98%	78.039,84	-0,01%
200	347.495,65	-3,85%	188.991,25	-5,46%	45.015,38	-4,05%	101.817,03	-0,92%
400	405.360,00	-7,15%	236.988,05	-8,98%	54.416,71	-8,50%	101.882,75	-2,37%
600	429.475,22	-16,29%	247.332,62	-20,61%	62.002,52	-16,92%	104.250,08	-5,52%
800	432.405,56	-20,62%	259.554,61	-25,57%	65.145,05	-21,48%	86.450,71	-5,57%
1000	468.738,36	-27,63%	279.228,84	-33,85%	74.161,22	-26,83%	95.561,86	-7,85%

Tabelle 4.3: Anzahl der erzeugten Knoten bei Überprüfung der Zeitfensterkompatibilität im Vergleich zur bisherigen Heuristik

wächst, andererseits aber auch bei einer Verletzung der Zeitfensterkompatibilität auf die Berechnung des Single Vehicle Routings auf größeren Zustandsgraphen verzichtet werden kann. Bei der Anzahl Duplikat-Knoten zeichnet sich ein anderes Bild: die prozentualen Einsparungen wachsen zwar auch mit der Anzahl Aufträge der Problemgruppen, liegen jedoch nur bei 0,01% bis 7,85%. Dies lässt den Schluss zu, dass bei der Berechnung von Lösungen für das Single Vehicle Routing von den Fahrzeugen, deren Zeitfenster nicht kompatibel mit den Zeitfenstern eines neu eingegangenen Auftrags sind, nur sehr wenige Duplikat-Knoten erzeugt werden. Die Überprüfung der möglichen Einhaltung der Zeitfenster in der Schätzfunktion des A*-Algorithmus muss daher schon auf einer relativ niedrigen Stufe des Zustandsgraphen zu einem Abbruch führt, da Duplikat-Knoten erst auf höheren Stufen erzeugt werden.

Die in 4.4 dargestellten Änderungen der Antwortzeiten sind sehr unterschiedlich: während die prozentuale Verringerung der durchschnittlichen und durchschnittlich maximalen Antwortzeit ebenfalls in den Problemgruppen mit zunehmender Anzahl Aufträgen eher steigt und Reduzierungen von 4,71% bis 38,28% bzw. 4,70% bis 31,63% zu verzeichnen sind, wird in der Problemgruppe „100“ sogar eine Zunahme der maximalen Antwortzeit um eine Sekunde und in den anderen Problemgruppen eine Reduktion um nur 0,65% bis 5,19% realisiert. Bei der global maximalen Antwortzeit ist zudem auffallend, dass bei den in den Testläufen mit Überprüfung der Zeitfensterkompatibilität zusätzlich bearbeiteten Aufträgen keine glo-

Problem-	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
gruppe	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	122,77	122,77	0,90%	5,82	5,82	-4,70%	0,17	0,17	-4,71%
200	99,45	99,45	-5,19%	5,60	5,56	-5,62%	0,10	0,10	-11,10%
400	65,83	65,83	-4,84%	4,30	4,30	-27,18%	0,05	0,05	-28,22%
600	30,15	30,15	-3,86%	3,38	3,31	-16,75%	0,03	0,03	-25,26%
800	36,53	36,53	-4,09%	2,48	2,46	-31,63%	0,02	0,02	-38,28%
1000	40,57	40,57	-0,65%	3,28	2,81	-20,13%	0,02	0,02	-36,90%

Tabelle 4.4: Antwortzeiten bei Überprüfung der Zeitfensterkompatibilität im Vergleich zur bisherigen Heuristik

bal maximalen Antwortzeiten erreicht wurden, da sich die absoluten Werte in den Spalten „sec“ und „Vgl“ nicht unterscheiden. Da die global maximalen Antwortzeiten den „Worst Case“ abbilden, ist nicht zu erwarten, dass hier unbedingt große Verbesserungen zu erzielen sind. Die erwarteten Verbesserungen zeigen sich jedoch in den durchschnittlichen und durchschnittlich maximalen Antwortzeiten. Im Gegensatz zur Auswertung der Anzahl Knoten fließt hier auch die zur Überprüfung der Zeitfensterkompatibilität benötigte Rechenzeit ein. Da die Antwortzeiten pro Auftrag gemessen werden und für jeden Auftrag entsprechend der Anzahl zur Verfügung stehender Fahrzeuge mehrere Single Vehicle Routing Probleme gelöst werden, liegt hier eine andere Durchschnittsbildung als bei der Anzahl der pro Testlauf erzeugten Knoten vor, so dass die Zunahme der prozentualen Einsparungen der Rechenzeit nicht wie die Anzahl Knoten monoton mit steigender Anzahl Aufträge der Problemgruppen wächst.

Insgesamt kann eine z. T. deutliche Beschleunigung der Heuristik festgestellt werden, ohne ein Veränderung der Lösungen in Kauf nehmen zu müssen.

4.1.2 Beschleunigung durch veränderten Entscheidungszeitpunkt

In der Heuristik zur Berechnung eines zulässigen Routenplans werden bei Eingang eines neuen Auftrags κ zunächst die zusätzlichen Fahrzeiten C_f^κ für alle Fahrzeuge $f \in F$ berechnet, bevor der Auftrag bei Existenz mindestens einer neuen optimalen Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Hinzunahme des Auftrag κ demjenigen Fahrzeug $f \in F$ mit minimalen zusätzlichen Kosten C_f^κ zugeordnet und anschließend akzeptiert bzw. andernfalls abgelehnt wird.

Da das Fuhrunternehmen den Auftrag möglichst schnell annehmen oder ablehnen will, die Akzeptanz nur von der Existenz eines zulässigen Routenplans unter Hinzunahme des neuen Auftrags κ abhängt und keine Informationen über die geplante Route weitergegeben werden, kann der Auftrag κ akzeptiert werden, sobald für ein Fahrzeug $f \in F$ eine optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Einbeziehung des Auftrags κ gefunden wurde. Um die Qualität des berechneten Routenplans nicht zu beeinträchtigen, wird anschließend mit der Berechnung von zulässigen fahrzeitminimalen Routen $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ fortgefahren, so dass Auftrag κ wiederum

demjenigen Fahrzeug mit minimaler zusätzlicher Fahrzeit C_f^κ zugeordnet wird. Wird für kein Fahrzeug $f \in F$ eine Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ gefunden, so muss der Auftrag abgelehnt werden. Somit kann weder die Zeit zur Ablehnung eines Auftrags noch die tatsächlich benötigte Rechenzeit auf diese Weise reduziert werden. Es wird davon ausgegangen, dass die Zeit bis zum Eingang des nächsten eingehenden Auftrags $\kappa + 1$ ausreicht, um die Rechenzeit abzudecken.

Die größten Einsparungen in den Antwortzeiten könnten natürlich erzielt werden, wenn grundsätzlich zuerst eine Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ für ein Fahrzeug $f \in F$ mit zulässiger Route unter Hinzunahme dieses Auftrags κ berechnet würde. Die Information, für welche Fahrzeuge f eine Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ existiert, liegt natürlich nicht vor. Es wäre jedoch eventuell möglich, die Fahrzeuge nach bestimmten Kriterien zu sortieren, so dass für die Fahrzeuge f mit „guten Chancen“ auf eine optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ bei Hinzunahme des neuen Auftrags κ zuerst neue Routen $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ berechnet werden.

Die Heuristik zur Berechnung einer neuen zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität und vorgezogener Akzeptanz-Entscheidung kann folgendermaßen skizziert werden:

0. Sortiere die Fahrzeuge $f \in F$ entsprechend eines gegebenen Sortierkriteriums S_f.
1. Führe für jedes Fahrzeug $f \in F$ in der gegebenen Reihenfolge der Fahrzeuge aus:
   (a) Überprüfe die Zeitfensterkompatibilität des Auftrags κ zu den noch zu bedienenden Aufträgen $i \in N_f^{\tau_\kappa}$ des Fahrzeugs f
   (b) Falls die Zeitfenster kompatibel sind: Berechne eine neue optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$, falls sie existiert
   (c) Falls für Fahrzeug f eine neue optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ existiert und Auftrag κ noch nicht akzeptiert wurde: Akzeptiere Auftrag κ.
2. Wenn für kein Fahrzeug $f \in F$ eine optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ existiert: lehne Auftrag κ ab. Sonst gehe zu Schritt 3.
3. Berechne für jedes Fahrzeug $f \in F$ mit optimaler Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ die zusätzliche Fahrzeit $C_f^\kappa = C(\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}) - C(\overrightarrow{R}_{N_f^{\tau_\kappa}})$
4. Wähle das Fahrzeug $f^* \in \{f \in F \mid \exists \overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}\}$ mit minimaler zusätzlicher Fahrzeit C_f^κ und ordne Auftrag κ dem Fahrzeug f^* zu.

Im Unterschied zur Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität wird ein Schritt 0 hinzugefügt, in dem die Fahrzeuge entsprechend einem gegebenen Sortierkriteriums sortiert werden. Im ersten Schritt wird ein Auftrag κ schon akzeptiert, wenn die erste optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ für ein Fahrzeug f gefunden wurde; der letzte Schritt entfällt.

Im folgenden werden drei mögliche Sortierkriterien untersucht. Ein ganz einfacher und intuitiver Ansatz besteht in der Hoffnung, dass für ein Fahrzeug f, dem bisher nur wenige Aufträge $i \in N_f$ zugeordnet sind, eher eine optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ gefunden werden kann als für ein Fahrzeug f', dass bereits mehr Aufträge $i' \in N_{f'}$ zu erledigen hat. Die Fahrzeuge $f \in F$ könnten also entsprechend der Anzahl $|N_f|$ der einem Fahrzeug f bereits zugeordneten Aufträge aufsteigend sortiert werden. Dieses Vorgehen wird in Abschnitt 4.1.2.1 beschrieben und analysiert.
Da aber ein Fahrzeug f mit mehreren Aufträgen $i \in N_f$, zu deren Bedienung nur kurze Fahrzeiten benötigt werden, grundsätzlich auch einen neuen Auftrag κ bedienen könnte, wird als weiteres Sortierkriterium die Summe der Wartezeiten eines Fahrzeugs f herangezogen. Auf diese Vorgehensweise wird in Abschnitt 4.1.2.2 näher eingegangen.
Letztendlich sind es allerdings häufig die Zeitfenster der dem Fahrzeug f zugeordneten Aufträge $i \in N_f$, die die Existenz einer optimalen Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ verhindern. Interessant für die Zuordnung könnte daher auch sein, inwieweit die Zeitfenster der Aufträge $i \in N_f$ groß genug sind und noch einen „Umweg“ für einen neuen Auftrag κ ermöglichen. Als Kriterium für die Reihenfolge der Berechnung neuer Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ für die Fahrzeuge $f \in F$ könnte daher auch die Verschiebbarkeit der Pickups und Deliverys interessant sein, was in Abschnitt 4.1.2.3 dargestellt wird. Abschließend wird ein Fazit gezogen.

4.1.2.1 Sortierkriterium „Anzahl der Aufträge“

Dieser Abschnitt behandelt das Sortierkriterium „Anzahl der Aufträge“ für die Reihenfolge der Berechnung neuer zulässiger Routen $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ für die Fahrzeuge $f \in F$. Die Idee besteht darin, dass ein Fahrzeug f, dem bisher noch kein oder nur wenige Aufträge $i \in N_f$ zugeordnet sind, prinzipiell eher eine optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Einbeziehung des neuen Auftrags besitzt als ein Fahrzeug f' mit vielen schon zugeordneten Aufträgen $i' \in N_{f'}$. Das Sortierkriterium S_f^A „Anzahl der Aufträge“ wird daher für ein Fahrzeug $f \in F$ definiert als

$$S_f^A = |N_f^{\tau_\kappa}|$$

Die Fahrzeuge $f \in F$ können nun entsprechend S_f^A aufsteigend sortiert werden, bevor neue Routen $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ berechnet werden. Sobald eine zulässige Lösung gefunden wird, wird der Auftrag akzeptiert. Kann kein Fahrzeug f eine neue optimale Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ finden, wird der Auftrag abgelehnt.

Problemgruppe		Aufträge		gelöste Instanzen	
Name	Aufträge	absolut	Δ	absolut	Δ
100	51,86	41,27	0,00%	40	0,00%
200	103,08	73,35	0,36%	38	0,00%
400	205,66	130,37	-0,25%	32	-3,03%
600	308,97	193,08	0,40%	35	0,00%
800	411,75	258,29	2,90%	32	3,23%
1000	514,98	313,19	7,24%	31	6,90%

Tabelle 4.5: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^A im Vergleich mit den Ergebnissen aus 4.1.1

Ergebnisse der Testläufe

Für diese Testläufe wird die Testumgebung weiterhin ereignisdiskret eingestellt und die Grenze von einer Million maximal zu erzeugenden Knoten beibehalten. Für die Vergleiche werden die Ergebnisse der Testläufe zur Verwendung der Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität aus Abschnitt 4.1.1 verwendet. Da wegen der Sortierung der Fahrzeuge $f \in F$ vor der Berechnung neuer Routen $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ nun bei gleichen minimalen zusätzlichen Fahrzeiten C_f^κ möglicherweise ein anderes Fahrzeug f gewählt wird, können dann im weiteren Verlauf andere Aufträge abgelehnt bzw. akzeptiert und andere Zielfunktionswerte erreicht werden. Für die Vergleiche werden daher Testläufe herangezogen, die in jeder Testinstanz exakt die gleiche Anzahl Aufträge bearbeiten wie die Testläufe in Abschnitt 4.1.1.

Tabelle 4.5 zeigt die Anzahl der akzeptierten Aufträge und gelösten Instanzen bei Verwendung des vorgezogenen Entscheidungszeitpunkts und dem Sortierkriterium S_f^A. Die Anzahl der bearbeiteten Aufträge steigt in den Problemgruppen „800“ und „1000“ mit einer Zunahme von 2,90% bzw. 7,24% deutlich an, während sie in den Problemgruppen mit weniger Aufträgen beinahe konstant bleibt. In Problemgruppe„400“ konnten 0,25% weniger Auftrage bearbeitet werden. Bei der Anzahl gelöster Instanzen zeigt sich ein ähnliches Bild: während in Problemgruppe „400“ eine Instanz weniger gelöst werden konnte, wurden in Problemgruppe „800“ eine und in Problemgruppe „1000“ sogar zwei zusätzliche Instanzen gelöst.

In Tabelle 4.6 werden die durchschnittliche Anzahl der akzeptierten Aufträge und die durchschnittlichen Gesamtzeiten der Problemgruppen angegeben. In der Spalte „absolut“ wird das Ergebnis des Testlaufs mit maximal einer Million erzeugter Knoten dargestellt, während Spalte „Vgl“ das Ergebnis der Testläufe mit Bearbeitung der gleichen Aufträge wie in den Testläufen aus Abschnitt 4.1.1 und Spalte „Δ“ die prozentuale Abweichung dieser Werte von den Werten der Testläufe aus Abschnitt 4.1.1 wiedergeben. Im Ergebnis der Problemgruppe „100“ wird die Anzahl der akzeptierten Aufträge um 0,31% bei gleichbleibender durchschnitt-

Problem-gruppe	akzeptierte Aufträge			Gesamtzeit		
	absolut	Vgl	Δ	absolut	Vgl	Δ
100	35,23	35,23	0,31%	4.424,59	4.424,59	0,00%
200	57,22	56,95	-0,32%	9.562,91	9.517,89	0,00%
400	97,53	97,59	-0,36%	19.253,72	19.245,99	-0,19%
600	141,45	140,65	-0,61%	34.044,48	33.858,60	0,13%
800	189,15	182,10	-0,12%	53.880,25	52.631,25	-0,01%
1000	229,45	212,71	-0,27%	72.366,63	68.382,60	0,41%

Tabelle 4.6: Die Anzahl akzeptierter Aufträge und die Gesamtzeit bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^A im Vergleich mit den Ergebnissen aus 4.1.1

licher Gesamtzeit gesteigert, während in Problemgruppe „200“ genau das Gegenteil eintritt: bei gleichbleibender Gesamtzeit geht die Anzahl der akzeptierten Aufträge um 0,32% zurück. In Problemgruppe „200“ übersteigt jedoch die durchschnittliche Gesamtzeit der Spalte „absolut“ die durchschnittliche Gesamtzeit der Spalte „Vgl“, da durchschnittlich 0,27 zusätzliche Aufträge akzeptiert wurden. In der Problemgruppe „400“ werden bei diesen Testläufen im Vergleich 0,77% mehr Aufträge abgelehnt und damit eine um 0,19% geringere Gesamtzeit erreicht. In den Problemgruppen „400“ und „800“ wurde die Anzahl der akzeptierten Aufträge im Vergleich leicht um 0,36% bzw. bis 0,12% gesenkt. Die entsprechenden Gesamtzeiten sanken in Problemgruppe „400“ um 0,19%, während in Problemgruppe „800“ eine Reduktion von nur 0,01% zu verzeichnen ist. In den Problemgruppen „600“ und „1000“ wird hingegen trotz einer um 0,61% bzw. 0,27% gefallenen durchschnittlichen Anzahl akzeptierter Aufträge die Gesamtzeit im Schnitt um 0,13% bzw. 0,41% erhöht. Das liegt an der geänderten Zuordnung der Aufträge zu den Fahrzeugen und der daraus folgenden Akzeptanz oder Ablehnung anderer Aufträge als in den Testläufen in Abschnitt 4.1.1. In den Spalten mit den absolut erreichten Werten bei alleiniger Beschränkung der Berechnungen durch maximal eine Million zu erzeugender Knoten zeigt sich jedoch, dass nur in den Problemgruppen „400“ und „600“ im Durchschnitt 0,42 bzw. 0,07 weniger Aufträge akzeptiert wurden, während in den anderen Problemgruppen 0,08 bis maximal 16,17 Aufträge mehr akzeptiert wurden als in den Testläufen in Abschnitt 4.1.1. Die durchschnittlich pro akzeptiertem Auftrag anfallende Gesamtzeit ist in beiden Testläufen beinahe identisch: in den Problemgruppen „200“, „400“ und „600“ werden Steigerungen um absolut 0,55 bzw. 1,74 Zeiteinheiten erreicht, während in den Problemgruppen „100“, „800“ und „1000“ eine Reduktion um 0,39 bis 3,93 Zeiteinheiten festgestellt wird.

Insgesamt zeigt die Tabelle, dass durch die Sortierung der Fahrzeuge vor der Berechnung der Routen $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ und der anschließenden Auswahl des nach der Sortierung ersten Fahrzeugs mit minimaler zusätzlicher Fahrzeit C_f^κ durchaus andere Ergebnisse erzielt werden, ohne dass eine Überlegenheit einer der beiden Heuristiken bezüglich der erreichten Zielfunktions-

Problem-	glob. max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
gruppe	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	0,029	0,029	-99,98%	0,0014	0,0014	-99,98%	0,00025	0,00025	-99,86%
200	0,172	0,170	-99,83%	0,0098	0,0096	-99,83%	0,00092	0,00090	-99,10%
400	0,063	0,484	-99,26%	0,0045	0,0127	-99,70%	0,00063	0,00072	-98,45%
600	0,087	0,101	-99,67%	0,0061	0,0067	-99,80%	0,00056	0,00057	-97,94%
800	0,116	0,115	-99,69%	0,0085	0,0074	-99,70%	0,00066	0,00073	-95,56%
1000	0,096	0,091	-99,78%	0,0053	0,0052	-99,84%	0,00060	0,00055	-96,86%

Tabelle 4.7: Die Antwortzeiten bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^A im Vergleich mit den Ergebnissen aus 4.1.1

werte, der Anzahl der akzeptierten Aufträge und der Gesamtzeit, festgestellt werden kann.

Das Ziel der Einführung des vorgezogenen Entscheidungszeitpunkts und der Sortierung der Fahrzeuge war ja auch nicht die Verbesserung der Zielfunktionswerte, sondern die Reduktion der Antwortzeiten bei qualitativ gleich bleibenden Lösungen. Die erreichten Antwortzeiten werden in Tabelle 4.7 dargestellt. Hier zeigt sich eine bemerkenswerte Verbesserung: die durchschnittlichen Antwortzeiten sinken um 95,56% bis 99,86% auf unter $\frac{1}{1000}$ Sekunde, die durchschnittlich maximalen Antwortzeiten erreichen mit Werten unter $\frac{2}{100}$ Sekunden eine Reduktion um 99,70% bis 99,98% und die global maximalen Antwortzeiten können um 99,26% bis 99,98% auf maximal 0,17 Sekunden gesenkt werden.

Das Einführen des vorgezogenen Entscheidungszeitpunkts und der Einsatz dieses einfachen Sortierkriteriums haben somit zu einer erheblichen Reduktion der Antwortzeiten geführt. Daher stellt sich die Frage, ob nicht mit einem geschickteren Sortierkriterium die Ergebnisse noch verbessert werden können. Im folgenden Abschnitt werden daher die Wartezeiten des Fahrzeuges auf ihre Eignung als Sortierkriterium der Fahrzeuge untersucht.

4.1.2.2 Sortierkriterium „Leerlaufzeit des Fahrzeugs“

Wird nur auf die Anzahl der Aufträge n_f eines Fahrzeugs geachtet, so wird ignoriert, dass Aufträge je nach „Entfernung“ der anzufahrenden Orte $\mu \in P_f \cup D_f$ sehr unterschiedliche Auswirkungen auf die Auslastung eines Fahrzeugs $f \in F$ haben können. Insbesondere bei mehreren Aufträgen pro Fahrzeug können die Orte μ mit sehr unterschiedlichen Fahr- und Wartezeiten zu erreichen sein, so dass die Anzahl $|N_f|$ der Aufträge allein nicht viel über die Möglichkeiten der Zuordnung eines neuen Auftrags κ aussagt. Interessanter wäre an dieser Stelle eine Aussage über die Zeit, in der das Fahrzeug f nicht beschäftigt ist, die Leerlaufzeit. Die *Leerlaufzeit* eines Fahrzeuges $f \in F$ wird definiert als die gesamte Zeit vom Startzeitpunkt $\underline{T}_f$ des Fahrzeugs bis zum Ende $\overline{T}_f$ der Arbeitszeit des Fahrzeugs f, in der das Fahrzeug weder gerade von einem Ort μ zum nächsten Ort ν fährt noch im Be- oder Entladevorgang

Problemgruppe		Aufträge		gelöste Instanzen	
Name	Aufträge	absolut	Δ	absolut	Δ
100	51,86	41,27	0,00%	40	0,00%
200	103,08	73,17	0,11%	38	0,00%
400	205,66	130,41	-0,22%	32	-3,03%
600	308,97	191,67	-0,34%	34	-2,86%
800	411,75	250,66	-0,14%	31	0,00%
1000	514,98	303,55	3,94%	29	0,00%

Tabelle 4.8: Die Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^L im Vergleich mit den Ergebnissen aus 4.1.1

ist. Das Sortierkriterium S_f^L summiert daher für alle noch anzufahrenden Orte $\mu \in P_f \cup D_f$ des Fahrzeugs f die Wartezeiten ω_μ, bestehend aus der Differenz der geplanten Ankunftszeit α_μ im Ort μ und der geplanten Abfahrtszeit beim vorhergehenden Auftrag ν zuzüglich der Fahrzeit von μ nach ν. Mit der Notation des Modells für das statische Pickup and Delivery Vehicle Routing Problem ergibt sich

$$S_f^L = \sum_{\mu \in P_f \cup D_f:\ \alpha_\mu > \underline{T}_f} \omega_\mu + \overline{T}_f - \max_{\mu \in D_f} \{\alpha_\mu + s_\mu\}$$

wobei in der Summe nur diejenigen Orte $\mu \in P_f \cup D_f$ betrachtet werden, die von Fahrzeug f nach Zeitpunkt $\underline{T}_f$ noch angefahren werden müssen. Zusätzlich wird die noch verbleibende Zeit $\overline{T}_f - \max_{\mu \in D_f}\{\alpha_\mu + s_\mu\}$ zwischen der geplanten Ankunftszeit α_μ zuzüglich der dort anfallenden Servicezeit s_μ am letzten Ort $\mu \in D_f$ seiner Route $\overrightarrow{R}_{N_f^{\tau_\kappa}}$ und der oberen Zeitfenstergrenze $\overline{T}_f$ der Arbeitszeit des Fahrzeugs f hinzugefügt, da eine freie Zeitspanne am Ende der Route $\overrightarrow{R}_{N_f^{\tau_\kappa}}$ genau so potentielle Arbeitszeit darstellt wie Wartezeiten ω_μ während der geplanten Route $\overrightarrow{R}_{N_f^{\tau_\kappa}}$.

Ergebnisse der Testläufe

Für die Testläufe wird die Testumgebung weiterhin ereignisdiskret eingestellt und die Grenze von einer Million maximal zu erzeugenden Knoten beibehalten. Für die Vergleiche werden wie in den Testläufen im vorhergehenden Abschnitt 4.1.2.1 die Ergebnisse der Testläufe zur Verwendung der Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität aus Abschnitt 4.1.1 verwendet.

In Tabelle 4.8 werden die Anzahl der akzeptierten Aufträge und die Anzahl gelöster Instanzen bei Verwendung des vorgezogenen Entscheidungszeitpunkts und dem Sortierkriterium S_f^L dargestellt. In Problemgruppe „1000“ steigt die Anzahl der bearbeiteten Aufträge mit einer Zunahme um 3,94% deutlich an, erreicht aber nicht die in den Testläufen mit dem Sortier-

Problem-gruppe	akzeptierte Aufträge			Gesamtzeit		
	absolut	Vgl	Δ	absolut	Vgl	Δ
100	35,09	35,09	-0,10%	4.412,86	4.412,86	-0,27%
200	57,37	57,28	0,26%	9.541,08	9.513,22	-0,05%
400	97,49	97,66	-0,29%	19.234,68	19.210,33	-0,38%
600	140,83	141,50	-0,01%	33.694,22	33.779,69	-0,10%
800	182,83	182,47	0,08%	52.382,62	52.424,13	-0,41%
1000	220,62	212,95	-0,15%	69.666,32	67.850,81	-0,37%

Tabelle 4.9: Die Anzahl akzeptierter Aufträge und die Gesamtzeit bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^L im Vergleich mit den Ergebnissen aus 4.1.1

kriterium S_f^A erzielte Steigerung um 7,24%. In den Problemgruppen „400“, „600“ und „800“ geht die Anzahl der durchschnittlich bearbeiteten Aufträge um 0,14% bis 0,34% zurück, während sie in Problemgruppe „100“ konstant bleibt und in Problemgruppe „200“ leicht um 0,11% zunimmt. Bei der Anzahl gelöster Instanzen werden in den Problemgruppen „400“ und „600“ jeweils eine Instanz weniger gelöst, während sich die Anzahl gelöster Instanzen in allen anderen Problemgruppen nicht verändert.

Tabelle 4.9 zeigt die durchschnittliche Anzahl der akzeptierten Aufträge und die durchschnittlichen Gesamtzeiten der Problemgruppen, wobei die Spalte „absolut“ das Ergebnis des Testlaufs mit maximal einer Million erzeugter Knoten dargestellt enthält, während Spalte „Vgl“ das Ergebnis der Testläufe mit Bearbeitung der gleichen Aufträge wie in den Testläufen aus Abschnitt 4.1.1 und Spalte „Δ“ die prozentuale Abweichung dieser Werte von den Werten der Testläufe aus Abschnitt 4.1.1 zeigen. Die Ergebnisse der Problemgruppen „100“, „400“, „600“, „1000“ zeigen im Vergleich eine Reduktion der durchschnittlichen Anzahl akzeptierter Aufträge um 0,01% bis 0,29% bei gleichzeitig um 0,10% bis 0,38% sinkender Gesamtzeit gegenüber den Testläufen ohne vorgezogenen Entscheidungszeitpunkt. In den Problemgruppen „200“ und „800“ kann hingegen trotz einer Steigerung der durchschnittlichen Anzahl akzeptierter Aufträge um 0,26% bzw. 0,08% eine Reduktion der durchschnittlichen Gesamtzeit um 0,05% bzw. 0,41% konstatiert werden. Die Anzahl der akzeptierten Aufträge in den nur durch die Begrenzung durch eine Million maximal zu erzeugenden Knoten eingeschränkten Testläufen liegt erreicht ähnliche Werte wie die Testläufe mit pro Testinstanz festgelegter Anzahl zu bearbeitender Aufträge.

Im Vergleich zu den Testläufen mit vorgezogenem Entscheidungszeitpunkt und dem Sortierkriterium S_f^A kann in allen Problemgruppen „200“ bis „1000“ eine leichte Verbesserung in der Anzahl akzeptierter Aufträge in den Testläufen mit pro Testinstanz festgelegten Anzahl zu bearbeitender Aufträge erreicht werden, während für die Testläufe, die nur durch eine Million maximal zu erzeugender Knoten beschränkt waren, in den Problemgruppen “100“ und „400“ bis „1000“ mit bis zu 9 Aufträgen zum Teil deutlich weniger Aufträge akzeptiert werden.

Problem-	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
gruppe	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	125,39	127,40	3,77%	4,39	4,36	-25,07%	0,1235	0,1226	-29,83%
200	12,45	12,25	-87,68%	0,70	0,69	-87,67%	0,0174	0,0166	-83,42%
400	25,60	27,40	-58,37%	0,49	0,52	-88,02%	0,0072	0,0072	-84,31%
600	10,38	10,51	-65,13%	0,50	0,56	-83,31%	0,0058	0,0066	-76,04%
800	37,05	38,29	4,84%	1,75	1,81	-26,89%	0,0094	0,0098	-40,21%
1000	42,62	41,09	1,26%	1,07	1,03	-68,57%	0,0066	0,0062	-64,54%

Tabelle 4.10: Die Antwortzeiten bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^L im Vergleich mit den Ergebnissen aus 4.1.1

Eine eindeutige Überlegenheit eines der beiden Sortierkriterien kann daher nicht festgestellt werden.

Tabelle 4.10 zeigt die erreichten Antwortzeiten. Die durchschnittlichen Antwortzeiten können um 29,83% bis 84,31% auf unter $\frac{2}{100}$ Sekunde gesenkt werden, die durchschnittlich maximalen Antwortzeiten erzielen mit Werten unter 2 Sekunden eine Reduktion um 25,07% bis 88,02%. Die global maximalen Antwortzeiten verzeichnen jedoch in den Problemgruppe „100“, „800“ und „1000“ leichte Zunahmen um 1,26% bis 3,77% auf Werte bis zu 127,4 Sekunden, während in den Problemgruppen „200“, „400“ und „600“ die maximalen Antwortzeiten um 58,37% bis 87,68% auf maximal 27,4 Sekunden gesenkt werden. Die Zunahme der maximalen Antwortzeit bei gleichzeitig deutlicher Reduktion der durchschnittlich maximalen Antwortzeit zeigt, dass es einige Fälle gibt, in denen das Sortierkriterium nicht wie erhofft dazu führt, dass zuerst die Routen $\overrightarrow{R}_{N_f^{\tau_\kappa}}$ der Fahrzeuge f mit optimalen Routen $\overrightarrow{R}_{N_f^{\tau_\kappa}}$ berechnet werden.

Im Vergleich zu den Testläufen mit vorgezogenem Entscheidungszeitpunkt und dem Sortierkriterium S_f^A werden somit deutlich schlechtere Ergebnisse erzielt. Zusätzlich zu den Leerlaufzeiten hängt die mögliche Zuordnung eines Auftrags zu einem Fahrzeug in besonderem Maße von den Zeitfenstern ab, deren Erfüllung strikt vorgeschrieben ist. Im folgenden Abschnitt wird daher ein Sortierkriterium vorgestellt, dass die Flexibilität der Route im Hinblick auf die Zeitfenster bestimmen soll.

4.1.2.3 Sortierkriterium „Verschiebbarkeit der Pickups und Deliverys“

Um einen neuen Auftrag κ in eine bestehende Route $\overrightarrow{R}_{N_f^{\tau_\kappa}}$ eines Fahrzeugs f aufzunehmen, müssen wahrscheinlich die Ankunftszeiten α_μ an den Pickup- und Delivery-Orten $\mu \in P_f \cup D_f$ der anderen Aufträge $i \in N_f$ zumindest teilweise geändert werden. Wegen der zwingend zu erfüllenden Zeitfenster $[\underline{t}_\mu, \overline{t}_\mu]$ ist es wahrscheinlich, dass die Route $\overrightarrow{R}_{N_f^{\tau_\kappa}}$ zumindest in Teilen erhalten bleibt und die für den neuen Auftrag κ anzufahrenden Orte p_κ und d_κ in die Sequenz $(\mu_1, \mu_2, \ldots, \mu_l)$ der Orte $\mu_1, \ldots, \mu_l \in P_f \cup D_f$ der bestehenden Route $\overrightarrow{R}_{N_f^{\tau_\kappa}}$ eingefügt

werden, ohne große Veränderungen in der Reihenfolge der Orte zu bewirken: $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}} \approx (\mu_1, \mu_2, \ldots, p_\kappa, \ldots, d_\kappa, \ldots, \mu_l)$. Das bedeutet, dass die Ankunftszeiten α_μ in den nach p_κ anzufahrenden Orten nach hinten verschoben werden. Im Algorithmus zur Lösung des Single Vehicle Routing Problems werden die geplanten Ankunftszeiten immer frühestmöglich gesetzt, so dass eine Verschiebung der Ankunftszeiten α_μ bei Beibehaltung der Reihenfolge der Orte in der Route nach vorne nicht möglich ist.

Interessant für eine Einschätzung der möglichen Existenz einer neuen Route $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ ist demnach, um wie viel die Ankunftszeiten α_μ nach hinten verschoben werden können, ohne die zugehörigen Zeitfenster $[\underline{t}_\mu, \overline{t}_\mu]$ zu verletzen. Die Differenz von oberer Zeitfenstergrenze $\overline{t}_\mu$ für die Ankunft in einem Ort μ und der geplanten Ankunftszeit α_μ kann als Maß für die Verschiebbarkeit der Ankunft in diesem Ort betrachtet werden. Summiert man die Zeiten für die Verschiebbarkeit der Ankunft aller noch anzufahrenden Orte $\mu \in P_f \cup D_f$, so erhält man ein Maß für die Flexibilität der Route. Da diese Summe insbesondere von der Anzahl der zu besuchenden Orte μ abhängt, wird sie relativ zur Anzahl der Orte betrachtet. Für das Sortierkriterium „Verschiebbarkeit der Pickups und Deliverys“ S_f^V eines Fahrzeugs f erhalten wir also

$$S_f^V = \frac{\sum_{\mu \in P_f \cup D_f, \alpha_\mu > \tau_\kappa} (\overline{t}_\mu - \alpha_\mu)}{|\{\mu \in P_f \cup D_f \mid \alpha_\mu > \tau_\kappa\}|}$$

wobei N_F die Menge der dem Fahrzeug f zugeordneten Aufträge darstellt und P_f bzw. D_f die Pickup- und Delivery-Orte dieser Aufträge beinhalten.

Ergebnisse der Testläufe

Auch für diese Testläufe wird die Testumgebung ereignisdiskret eingestellt und die Grenze von einer Million maximal zu erzeugenden Knoten beibehalten. Für die Vergleiche werden wie in den Testläufen in den beiden vorhergehenden Abschnitten die Ergebnisse der Testläufe zur Verwendung der Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität aus Abschnitt 4.1.1 verwendet.

Tabelle 4.11 stellt die Anzahlen akzeptierter Aufträge und gelöster Instanzen bei Verwendung des vorgezogenen Entscheidungszeitpunkts und dem Sortierkriterium S_f^V dar. In den Problemgruppen „200“, „400“ und „600“ geht die Anzahl der durchschnittlich bearbeiteten Aufträge um 0,55% bis 1,56% zurück, während sie in Problemgruppe „100“ konstant bleibt und in den Problemgruppe „800“ und „1000“ sogar um 4,35% bzw. 8,22% steigt. Im Vergleich zu den bisherigen Testläufen unter Verwendung des vorgezogenen Entscheidungszeitpunkts mit den Sortierkriterien S_f^A bzw. S_f^L wurden somit in den Problemgruppen „800“ und „1000“ die besten, in den Problemgruppen „200“, „400“ und „600“ die schlechtesten Werte erzielt. Bei der Anzahl gelöster Instanzen werden in den Problemgruppen „200“ und „400“ jeweils eine und in Problemgruppe „600“ sogar zwei Instanzen weniger gelöst als in den Testläufen oh-

Problemgruppe		Aufträge		gelöste Instanzen	
Name	Aufträge	absolut	Δ	absolut	Δ
100	51,86	41,27	0,00%	40	0,00%
200	103,08	72,68	-0,55%	37	-2,63%
400	205,66	128,66	-1,56%	32	-3,03%
600	308,97	190,53	-0,93%	33	-5,71%
800	411,75	261,93	4,35%	33	6,45%
1000	514,98	316,07	8,22%	31	6,90%

Tabelle 4.11: Die Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^V im Vergleich mit den Ergebnissen aus 4.1.1

Problem-	akzeptierte Aufträge			Gesamtzeit		
gruppe	absolut	Vgl	Δ	absolut	Vgl	Δ
100	35,18	35,18	0,15%	4.425,25	4.425,25	0,01%
200	56,60	56,80	-0,58%	9.498,87	9.467,26	-0,53%
400	96,42	97,42	-0,54%	19.128,82	19.235,62	-0,25%
600	140,18	141,18	-0,24%	33.728,55	33.805,09	-0,03%
800	190,64	182,12	-0,11%	54.318,13	52.827,98	0,36%
1000	231,22	212,67	-0,28%	72.614,14	67.904,95	-0,29%

Tabelle 4.12: Die Anzahl akzeptierter Aufträge und die Gesamtzeit bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^L im Vergleich mit den Ergebnissen aus 4.1.1

ne vorgezogenen Entscheidungszeitpunkt, während sich die Anzahl gelöster Instanzen in den Problemgruppen „800“ und „1000“ um jeweils eine bzw. zwei Instanzen erhöht. Damit können die Ergebnisse der Testläufe mit dem Sortierkriterium S_f^A nur in Problemgruppe „800“ um eine zusätzlich gelöste Instanz übertroffen werden, während in den Problemgruppen „200“ und „600“ eine bzw. sogar zwei gelöste Instanzen weniger erreicht werden.

In Tabelle 4.12 werden die durchschnittliche Anzahl der akzeptierten Aufträge und die durchschnittlichen Gesamtzeiten der Problemgruppen vorgestellt. In den Problemgruppen „200“ bis „1000“ wird im Vergleich zu den Testläufen ohne Sortierung der Fahrzeuge eine Reduktion der durchschnittlichen Anzahl akzeptierter Aufträge um 0,24% bis 0,58% konstatiert, während in Problemgruppe „100“ die Anzahl der akzeptierten Aufträge um durchschnittlich 0,15% gestiegen ist. Diese Reduktion bzw. Zunahme der akzeptierten Aufträge spiegelt sich in der Reduktion bzw. Zunahme der Gesamtzeit wider, nur in Problemgruppe „800“ wird trotz einer um 0,11% sinkenden Anzahl akzeptierter Aufträge eine Zunahme der Gesamtzeit um 0,36% verzeichnet. Für die Testläufe bei festgelegter Anzahl zu bearbeitender Aufträge können in den Problemgruppen „600“ und „800“ mit durchschnittlich 141,18 bzw. 182,12 akzeptierten Auf-

Problem-	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
gruppe	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	129,87	127,46	3,82%	6,16	6,21	6,71%	0,1822	0,1832	4,89%
200	100,20	96,35	-3,12%	5,36	5,09	-9,10%	0,0980	0,0907	-9,27%
400	109,83	67,53	2,58%	7,47	5,57	29,58%	0,0699	0,0531	15,00%
600	30,34	29,85	-1,00%	2,75	2,68	-20,80%	0,0221	0,0206	-25,52%
800	13,85	14,26	-60,97%	1,13	1,06	-57,17%	0,0081	0,0078	-52,55%
1000	23,05	22,79	-43,84%	2,15	2,08	-36,48%	0,0099	0,0102	-41,88%

Tabelle 4.13: Die Antwortzeiten bei Verwendung der Überprüfung der Zeitfensterkompatibilität mit vorgezogenem Entscheidungszeitpunkt und Sortierkriterium S_f^V im Vergleich mit den Ergebnissen aus 4.1.1

trägen neue Maximalwerte der Testläufe mit vorgezogenem Entscheidungszeitpunkt erreicht werden; sie erreichen jedoch nicht die Ergebnisse der Testläufe ohne vorgezogenen Entscheidungszeitpunkt. In Problemgruppe „1000“ wird die durchschnittliche Fahrzeit pro akzeptiertem Auftrag um 5,28 Zeiteinheiten gesenkt und erreicht so einen neuen Bestwert, während in den anderen Problemgruppen bei der durchschnittlich pro akzeptiertem Auftrag erzielten Fahrzeit im Vergleich mit den anderen Testläufen schlechtere Werte erzielt werden. In den nur durch maximal eine Million zu erzeugenden Knoten beschränkten Testläufen konnten jedoch in den Problemgruppen „800“ und „1000“ mit durchschnittlich 190,64 bzw. 231,22 akzeptierten Aufträgen neue Bestwerte erzielt werden. Auch für das Sortierkriterium S_f^V kann daher anhand der erreichten Zielfunktionswerte keine eindeutige Überlegenheit festgestellt werden.

In Tabelle 4.13 werden die erreichten Antwortzeiten dargestellt. Die durchschnittlichen Antwortzeiten können in den Problemgruppen „200“ und „600“ bis „1000“ um 9,27% bis 52,55% gesenkt werden, während sie in den Problemgruppen „100“ und "400“ um 4,89% bzw. sogar 15,00% steigen. Die durchschnittliche Antwortzeit bleibt jedoch kleiner als $\frac{2}{10}$ Sekunden. Mit bis zu 6,21 Sekunden liegen die erzielten durchschnittlich maximalen Antwortzeiten in Problemgruppe „100“ um 6,71% höher als in den Testläufen ohne vorgezogenen Entscheidungszeitpunkt, in Problemgruppe „400“ wird sogar ein Anstieg um 29,58% verzeichnet. In den anderen Problemgruppen sinken die durchschnittlich maximalen Antwortzeiten um 9,10% bis 57,17%. Bei den global maximalen Antwortzeiten ist in den Problemgruppe „100“ und „400“ eine Zunahme um 3,82% bzw. 2,58% auf Werte bis zu 127,4 Sekunden festzustellen, während in den anderen Problemgruppen die maximalen Antwortzeiten um 1,00% bis 60,97% im schlechtesten Fall nur auf 96,37 Sekunden gesenkt werden. Die Zunahme aller dargestellten Antwortzeit-Werte in den Problemgruppen „100“ und „400“ zeigt, dass dieses Sortierkriterium häufig ungünstige Reihenfolgen für die Berechnung der Routen $\overrightarrow{R}_{N_f^{\tau_\kappa}}$ der Fahrzeuge f erzeugt. Auch die in den anderen Problemgruppen erreichten Reduzierungen der Antwortzeiten sind deutlich geringer als die in den anderen Testläufen zur Verwendung des vorgezogenen Entscheidungszeitpunkts erzielten Einsparungen.

In den Testläufen zur Verwendung des Sortierkriteriums S_f^V „Verschiebbarkeit der Pickups und Deliverys“ können somit zwar die Antwortzeiten gegenüber den Testläufen ohne vorgezogenen Entscheidungszeitpunkt weitgehend verringert werden, aber bei weitem nicht so gute Ergebnisse wie in den Testläufen mit dem Sortierkriterium S_f^A „Anzahl der noch zu bedienenden Aufträge“ erzielt werden.

Bemerkung:
Es gibt noch viele mögliche Arten, ein Maß für die Verschiebbarkeit der Pickups und Deliverys innerhalb einer Route zu definieren. Denkbar wäre zum Beispiel auch, die Verschiebbarkeit nur für diejenigen Pickups und Deliverys mit Ankunftszeiten $\alpha_\mu > \underline{t}_{p_\kappa}$ zu betrachten, wobei die Fahrzeit zum Erreichen des Ortes p_κ nicht betrachtet wird. Wegen der schlechten Ergebnisse des Sortierkriteriums S_f^V werden diese Kriterien jedoch nicht weiter untersucht.

4.1.3 Fazit

In diesem Abschnitt wurden zwei Möglichkeiten vorgestellt, die Heuristik zur Berechnung eines zulässigen Routenplans im Hinblick auf die benötigten Antwortzeiten effizienter zu gestalten.

Mit der Überprüfung der Zeitfensterkompatibilität der Zeitfenster des neu eingegangenen Auftrags κ und der bereits dem Fahrzeug f zugeordneten Aufträge $N_f^{\tau_\kappa}$ vor der Berechnung einer optimalen Route $\overrightarrow{R}_{N_f^{\tau_\kappa}\cup\{\kappa\}}$ kann die Heuristik zur Berechnung eines zulässigen Routenplans wegen der entfallenden Berechnungen optimaler Routen $\overrightarrow{R}_{N_f^{\tau_\kappa}\cup\{\kappa\}}$ für Fahrzeuge f, deren Zeitfensterkompatibilität nicht gegeben ist, mit deutlich geringeren Antwortzeiten durchgeführt werden, ohne die Qualität der Lösungen zu beeinflussen.

Der vorgezogene Entscheidungszeitpunkt hat weiter deutlich zur Reduzierung der Antwortzeiten beigetragen. Insbesondere das einfachste Sortierkriterium S_f^A, die Anzahl der dem Fahrzeug f zugeordneten noch zu bedienenden Aufträge $i \in N_f^\tau$, konnte deutliche Einsparungen von über 95,56% in allen Problemgruppen sowohl in der durchschnittlichen, der durchschnittlich maximalen und der global maximalen Antwortzeit erzielen. Die Zielfunktionswerte, die Anzahl der akzeptierten Aufträge und die Gesamtzeit zur Bearbeitung dieser Aufträge, werden von der Verwendung des vorgezogenen Entscheidungszeitpunkts und des Sortierkriteriums S_f^A nicht beeinträchtigt: in den Testläufen mit festgelegter Anzahl pro Testlauf zu bearbeitender Aufträge wurden durchschnittlich 0,35 Aufträge weniger akzeptiert, während in den nur durch die maximal eine Million zu erzeugenden Knoten begrenzten Testläufe durchschnittlich 3,76 zusätzliche Aufträge akzeptiert werden können. Die durchschnittlich pro akzeptiertem Auftrag benötigte Zeit konnte sogar wegen der in den Problemgruppen „800“ und „1000“ bedeutenden Einsparungen sowohl für die Testläufe mit festgelegter Anzahl zu bearbeitender Aufträge pro Instanz als auch für die nur durch die Anzahl zu erzeugender Knoten beschränkten Testläufe gesenkt werden.

Für die weiteren Testläufe wird daher die Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität und vorgezogenem Entscheidungszeitpunkt mit dem Sortierkriterium S_f^A, der Anzahl noch zu bedienender Aufträge, zu Grunde gelegt.

4.2 Der Algorithmus der Tabu Suche zur Verbesserung der heuristischen Zuordnung

Mit Hilfe der Heuristik zur Berechnung eines zulässigen Routenplans wird das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern gelöst, indem bei Eingang eines neuen Auftrags κ die Zuordnung der akzeptierten Aufträge $i \in N^\tau$ zu den Fahrzeugen und das Single Vehicle Routing der Fahrzeuge $f \in F$ zwar ineinander verschachtelt, aber prinzipiell eigenständig betrachtet werden. Die Lösung des Single Vehicle Routing Problems erfolgt mit einem schnellen exakten Algorithmus, während das Zuordnungsproblem nur heuristisch gelöst wird: bei Eingang eines neuen Auftrags κ wird für jedes Fahrzeug $f \in F$ die optimale Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ bestimmt, ohne dass an der bestehenden Zuordnung der Aufträge zu den Fahrzeugen etwas geändert wird. Der Auftrag κ wird demjenigen Fahrzeug f^* mit neuer zulässiger Route $\overrightarrow{R}_{N_{f^*}^\tau \cup \{\kappa\}}$ und minimaler zusätzliche Fahrzeit $C_{f^*}^\kappa$ durch Aufnahme des Auftrags κ zugeordnet, falls für mindestens ein Fahrzeug f eine zulässige Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ existiert. Falls für kein Fahrzeug $f \in F$ eine zulässige Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ existiert, wird er abgelehnt.

Diese Zuordnung ist eine „einfache“ Heuristik und muss daher keine besonders guten Lösungen erzeugen. So kann z. B. ein bereits bestehender Auftrag $i' \in N_{f'}^\tau$, der dem Fahrzeug $f' \in F$ zugeordnet ist, besonders „gut“ zum neu eingegangenen Auftrag κ „passen“. Wenn aber für Fahrzeug f' keine zulässige Route $\overrightarrow{R}_{N_{f'}^\tau \cup \{\kappa\}}$ unter Hinzunahme des neuen Auftrags κ existiert, wird der neue Auftrag κ – bei Existenz einer zulässigen Route $\overrightarrow{R}_{N_f^\tau \cup \{\kappa\}}$ – einem anderen Fahrzeug $f^* \in F \setminus \{f'\}$ zugeordnet. Auch wenn die Menge $N_{f^*}^\tau$ der dem Fahrzeug f^* zugeordneten Aufträge bisher noch leer war, wird der „gut“ dazu „passende“ Auftrag i' nicht in Fahrzeug f^* verschoben. Eine solche Überprüfung wäre natürlich wünschenswert, bedeutet aber einen erheblichen Rechenaufwand. Wegen der geforderten geringen Antwortzeiten wird daher die oben beschriebene einfache Zuordnung bis zur Akzeptanz oder Ablehnung eines Auftrags eingesetzt.

Allerdings gilt die strikte Rechenzeitbeschränkung nur für die Antwortzeit und damit für das erstmalige Zuordnen des neuen Auftrags κ. Sobald Auftrag κ zugeordnet wurde, wird in der Heuristik zur Berechnung eines zulässigen Routenplans nur auf den Eingang des nächsten Auftrags κ' gewartet. In der Zeit $[\tau_\kappa + \epsilon, \tau_{\kappa'}]$ zwischen der vollständig berechneten Zuordnung eines Auftrags κ und dem folgenden Auftragseingang von κ' kann die Rechnerkapazität daher genutzt werden, um die Zuordnung der Aufträge $i \in N^{\tau_\kappa}$ zu den Fahrzeugen $f \in F$ zu verbessern. Dabei muss jedoch darauf geachtet werden, dass zu jedem Zeitpunkt $\tau > \tau_\kappa$ eine aktuelle Lösung existiert, damit im Falle des Eingangs eines neuen Auftrags κ' die Heuristik zur Berechnung eines zulässigen Routenplans unter Hinzunahme des neuen Auftrags κ' sofort eingesetzt oder einem Fahrzeug $f \in F$, dass einen Pickup oder Delivery erledigt hat, der als nächstes anzufahrende Ort $\mu \in P_f \cup D_f$ mitgeteilt werden kann.

Zur Verbesserung der Zuordnung bietet sich daher ein Verfahren der *Tabu Suche* an. Ein Verfahren der Tabu Suche ist ein Verbesserungsverfahren, das speziell auf den Typ des Entscheidungsproblems zugeschnitten wird. Ausgehend von einer Startlösung wird versucht, bessere Lösungen in der „Umgebung“ dieser Lösung zu finden. Dazu wird zu einer gegebenen Lösung $\mathcal{L}$ eine Umgebung $U(\mathcal{L})$ von zulässigen Lösungen berechnet und die beste Lösung $\mathcal{L}^* \in U(\mathcal{L})$ ausgewählt, wobei einige Lösungen $\tilde{\mathcal{L}} \in \mathcal{B}$ „tabu“ sind und nicht ausgewählt werden dürfen. Ist der Zielfunktionswert der Lösung $\mathcal{L}^*$ besser als der Zielfunktionswert der bisherigen Lösung $\mathcal{L}$, so wird $\mathcal{L}^*$ als neue beste Lösung übernommen und das Verfahren der Tabu Suche in ihrer Umgebung $U(\mathcal{L}^*)$ fortgesetzt, soweit ein gegebenes Abbruchkriterium noch nicht erfüllt ist. Andernfalls terminiert das Verfahren der Tabu Suche mit der Lösung $\mathcal{L}^*$, einem – möglicherweise lokalen – Optimum. Die Qualität des Verfahrens der Tabu Suche hängt von der gewählten Umgebung $U(\mathcal{L})$ und der Wahl der Tabumengen $\mathcal{B}$ ab, die entsprechend dem Entscheidungsproblem definiert werden.

Die Tabu Suche kann im vorliegenden Fall angewandt werden, da sie zu jedem Zeitpunkt $\tau > \tau_\kappa$ eine beste bekannte Lösung $\mathcal{L}_\tau$ zur Verfügung stellen kann. Als Abbruchkriterium müssen sowohl ein möglicherweise lokales Optimum $\mathcal{L}^*$, der Eingang eines neuen Auftrags κ' oder die Statusänderung eines bereits akzeptierten Auftrags $i \in N^\tau$ angesetzt werden. In [26] wurden bereits verschiedene Umgebungen für die Tabu Suche in Verbindung mit dem Algorithmus von Caramia et al. untersucht. Die besten Ergebnisse erzielte dabei die Umgebung $U(\mathcal{L}_\tau)$, in der die Zuordnung eines einzelnen Auftrags $i \in N^\tau$ geändert wurde.Eine Lösung $\mathcal{L}_\tau$ zum Zeitpunkt τ besteht aus einer Zuordnung der Aufträge $i \in N^\tau$ zu den Fahrzeugen $f \in F$ und zulässigen Routen $\overrightarrow{R}_{N_f^\tau}$ für jedes Fahrzeug. Die Umgebung kann daher über die Änderung einer Route definiert werden als

$$U(\mathcal{L}_\tau) = \{\mathcal{L}'_\tau \mid \exists! \; i \in \bigcup_{f \in F} N_f^\tau : i \in N_f^\tau \wedge i \notin N_f'^\tau\}$$

wobei $N_f'^\tau$ die Menge der dem Fahrzeug f in der Lösung $\mathcal{L}'_\tau$ zugeordneten Aufträge darstellt. Da nur bisher nicht eingeladene Aufträge $i \in N^\tau$ zwischen zwei Fahrzeugen f und f' ausgetauscht werden dürfen, besteht die Umgebung $U(\mathcal{L}_\tau)$ aus maximal

$$\mid \{i \in N^\tau \mid \alpha_{p_i} > \underline{T}_f\} \mid (|F| - 1) \;\leq\; \mid N^\tau \mid (|F| - 1)$$

möglichen Lösungen. Die Anzahl der Lösungen in $U(\mathcal{L}_\tau)$ ist genau dann kleiner als $\mid \{i \in N^\tau \mid \alpha_{p_i} > \tau\} \mid (|F| - 1)$, wenn es bisher nicht abgeholte Aufträge $i \in N_f^\tau, f \in F$ und Fahrzeuge $f' \neq f$ gibt, für die keine zulässige Route $\overrightarrow{R}_{N_{f'}^\tau \cup \{i\}}$ existiert.

Die Umgebung $U(\mathcal{L}_\tau)$ kann erzeugt werden, indem für jedes Fahrzeug $f \in F$ und jeden Auftrag $i \in N_f^\tau$ mit $\alpha_{p_i} > \underline{T}_f$ die möglichen Routen $\overrightarrow{R}_{N_{f'}^\tau \cup \{i\}}$ für jedes Fahrzeug $f' \in F \setminus \{f\}$ mit Hilfe des Algorithmus zur Lösung des Single Vehicle Routing Problems berechnet werden. Die Berechnung der Umgebung $U(\mathcal{L}_\tau)$ kann daher folgendermaßen skizziert werden:

1. Wähle das erste Fahrzeug $f^* \in F$.
2. Wähle den ersten Auftrag $i \in N^\tau_{f^*}$ mit $\alpha_{p_i} > \underline{T}_{f^*}$ und berechne die Route $\overrightarrow{R}_{N^\tau_{f^*}\setminus\{i\}}$.
3. Berechne für jedes Fahrzeuge $f \in F \setminus \{f^*\}$ die Route $\overrightarrow{R}_{N^\tau_f \cup \{i\}}$, falls sie existiert, und füge bei Existenz von $\overrightarrow{R}_{N^\tau_f \cup \{i\}}$ eine Lösung $\mathcal{L}'_\tau$, die durch Ersetzen der Routen $\overrightarrow{R}_{N^\tau_{f^*}}$ und $\overrightarrow{R}_{N^\tau_f}$ der Lösung $\mathcal{L}_\tau$ durch die neu berechneten Routen $\overrightarrow{R}_{N^\tau_{f^*}\setminus\{i\}}$ und $\overrightarrow{R}_{N^\tau_{f^*}\cup\{i\}}$ entsteht, zur Umgebung $U(\mathcal{L}_\tau)$ hinzu.
4. Wähle den nächsten Auftrag $i \in N^\tau_{f^*}$ mit $\alpha_{p_i} > \underline{T}_{f^*}$ und gehe zu Schritt 3. Falls kein solcher Auftrag mehr vorliegt, gehe zu Schritt 5.
5. Wähle das nächste Fahrzeug $f^* \in F$ und gehe zu Schritt 2. Falls es kein weiteres Fahrzeug gibt: Abbruch. Alle Lösungen $\mathcal{L}'_\tau$ der Umgebung $U(\mathcal{L}_\tau)$ wurden berechnet.

Schon bei dieser einfachen Umgebung $U(\mathcal{L}_\tau)$ nimmt das Erzeugen aller möglichen Lösungen $\mathcal{L}'_\tau \in U(\mathcal{L}_\tau)$ sehr viel Rechenzeit in Anspruch: für jeden Auftrag $i \in N^\tau$, dessen Bedienung noch nicht begonnen wurde, muss die mögliche Zuordnung dieses Auftrags $i \in N^\tau_{f^*}$ zu jedem anderen Fahrzeug $f \in F\setminus\{f^*\}$ überprüft werden. Dazu müssen für alle Fahrzeuge $f \in F\setminus\{f^*\}$ neue Routen $\overrightarrow{R}_{N^\tau_f \cup \{i\}}$ und für Fahrzeug f^* die neue Route $\overrightarrow{R}_{N^\tau_{f^*}\setminus\{i\}}$ berechnet werden. Für jeden dieser Aufträge $i \in N^\tau$ mit $\alpha_{p_i} > \underline{T}_f$ entspricht die Rechenzeit zur Überprüfung der Zuordnung also in etwa der Rechenzeit bei Eingang eines neuen Auftrags.

Daher wird die Umgebung $U(\mathcal{L}_\tau)$ so klein wie möglich gehalten. Der zuletzt getauschte bzw. bei Start der Tabu Suche zuletzt zugeordnete Auftrag i^* kann ausgeschlossen werden, da eine Zuordnung dieses Auftrags zu einem anderen Fahrzeug keine Reduzierung der Fahrzeit zur Folge haben kann. Zusätzlich werden weitere Lösungen in die Tabumenge $\mathcal{B}_\tau$ aufgenommen: die Zuordnung einer festen Anzahl zuletzt zugeordneter oder getauschter Aufträge $i \in \mathcal{A}$ darf nicht geändert werden, so dass diese Lösungen für die Umgebung nicht berechnet werden müssen. Die zu erzeugende Umgebung $U(\mathcal{L}_\tau) \setminus \mathcal{B}_\tau$ einer Lösung $\mathcal{L}_\tau$ wird damit reduziert auf

$$U(\mathcal{L}_\tau) \setminus \mathcal{B}_\tau = \{\mathcal{L}'_\tau \mid \exists i \in \bigcup_{f \in F} N^\tau_f, i \notin \mathcal{A} : i \in N^\tau_f \wedge i \notin N'^\tau_f\}$$

Die Berücksichtigung der Tabumengen bei der Erzeugung einer Umgebung $U(\mathcal{L}_\tau)$ kann direkt im Schritt 2 überprüft werden:

2. Wähle den ersten Auftrag $i \in N^\tau_{f^*} \setminus \mathcal{A}$ mit $\alpha_{p_i} > \underline{T}_{f^*}$ und berechne die Route $\overrightarrow{R}_{N^\tau_{f^*}\setminus\{i\}}$.

Damit bleibt die Umgebung $U(\mathcal{L}_\tau)$ jedoch immer noch sehr groß und ist wegen der angestrebten Antwortzeiten bei Eingang eines neuen Auftrags κ nicht vollständig berechenbar. Daher werden immer nur Teile der Umgebung $U(\mathcal{L}_\tau)$ untersucht und bei Finden einer besseren

Lösung $\mathcal{L}'_\tau$ in einer solchen Teilumgebung diese sofort als neue beste Lösung akzeptiert und die Suche in Teilen der Umgebung $U(\mathcal{L}'_\tau)$ der neuen Lösung fortgesetzt.

Es liegt nahe, die Umgebung $U(\mathcal{L}_\tau)$ in Teilumgebungen $U_i(\mathcal{L}_\tau)$ aufzuteilen, so dass jede Teilumgebung alle Lösungen $\mathcal{L}'_\tau \in U(\mathcal{L}_\tau)$ umfasst, die die Zuordnung eines Auftrags $i \in N_f^\tau, \alpha_{p_i} > \underline{T}_f$ ändern:

$$\mathcal{L}'_\tau \in U_i(\mathcal{L}_\tau) \;\Leftrightarrow\; \mathcal{L}'_\tau \in U(\mathcal{L}_\tau) \wedge \exists!\, f \in F : i \in N_f^\tau \wedge i \notin N_f'^\tau$$

Das Vorgehen zur sukzessiven Berechnung der Teilumgebungen $U_i(\mathcal{L}_\tau)$ und der Abbruch der Tabu Suche bei Finden einer besseren Lösung $\mathcal{L}'_\tau \in \mathcal{L}_\tau$ kann direkt in der Berechnung einer Umgebung $U(\mathcal{L}_\tau)$ in Schritt 3 eingefügt werden:

3. Berechne für jedes Fahrzeuge $f \in F \backslash \{f^*\}$ die Route $\overrightarrow{R}_{N_f^\tau \cup \{i\}}$, falls sie existiert, und füge bei Existenz der Route $\overrightarrow{R}_{N_f^\tau \cup \{i\}}$ eine Lösung $\mathcal{L}'_\tau$, die durch Ersetzen der Routen $\overrightarrow{R}_{N_{f^*}^\tau}$ und $\overrightarrow{R}_{N_f^\tau}$ der Lösung $\mathcal{L}_\tau$ durch die neu berechneten Routen $\overrightarrow{R}_{N_{f^*}^\tau \backslash \{i\}}$ und $\overrightarrow{R}_{N_{f^*}^\tau \cup \{i\}}$ entsteht, zur Umgebung $U(\mathcal{L}_\tau)$ hinzu. Falls der Zielfunktionswert von $\mathcal{L}'_\tau$ besser ist als der Zielfunktionswert von $\mathcal{L}_\tau$: Stopp.

Das schnelle Auffinden einer möglichst guten neuen zulässigen Lösung $\mathcal{L}'_\tau$ hängt damit von der Reihenfolge ab, in der die Teilumgebungen $U_i(\mathcal{L}_\tau)$ erzeugt werden. Die Reihenfolge des Erzeugens der Teilumgebungen $U_i(\mathcal{L}_\tau)$ kann durch verschiedene Strategien bestimmt werden, die eine Reihenfolge für die Auswahl der möglicherweise zu tauschenden Aufträge $i \in N_f^\tau \setminus \mathcal{A}$ festlegen. Zur Auswahl einer geeigneten Strategie müssen Kriterien für einen wahrscheinlich erfolgreich zu tauschenden Auftrag $i \in N^\tau \setminus \mathcal{A}_\tau$ festgelegt werden. Eine Verringerung der Fahrzeiten $C(\mathcal{L})$ ist möglicherweise dann zu erwarten, wenn das Fahrzeug f mit $i \in N_f^\tau$ im Falle der Streichung der für Auftrag i anzufahrenden Orte p_i und d_i aus seiner Route $\overline{R}^\tau_{N_f^\tau}$ viel Fahrzeit einsparen kann - in der Hoffnung, dass der Auftrag i in einer anderen Route $\overline{R}^\tau_{N_{f'}^\tau \cup \{i\}}$ weniger zusätzliche Fahrzeit verursacht. Aus allen noch nicht abgeholten Aufträgen $i \in N^\tau$ mit $\alpha_{p_i} > \underline{T}_f, i \in N_f^\tau$ wird daher derjenige Auftrag $i^* \in N_{f^*}^\tau$ ausgewählt, bei dem der Wegfall seiner Pickup- und Delivery-Orte p_i und d_i aus der Route $\overrightarrow{R}_{N_{f^*}^\tau} = (\mu_1, \mu_2, \ldots, p_i, \ldots, d_i, \ldots \mu_l)$ unter Beibehaltung der Reihenfolge aller anderen Orte μ_j in der Route die größte Ersparnis $\mathcal{E}(i^*)$ bringen würde. Zur Berechnung der Ersparnis $\mathcal{E}(i)$ eines Auftrags $i \in N^\tau$ muss unterschieden werden zwischen Aufträgen $i \in N_f^\tau$, deren Pickup- und Delivery-Orte p_i und d_i in der Route $\overrightarrow{R}_{N_f^\tau}$ direkt aufeinander folgen ($x^f_{p_i d_i} = 1$) und Aufträgen $i \in N_f^\tau$, bei denen in der Route $\overrightarrow{R}_{N_f^\tau}$ zwischen dem Pickup-Ort p_i und dem Delivery-Ort d_i noch mindestens ein anderer Ort $\mu \notin \{p_i, d_i\}$ angefahren wird ($x^f_{p_i d_i} = 0$). Der Vorgänger-Ort eines Ortes μ in einer Route $\overrightarrow{R}_{N_f^\tau}$ wird dabei mit $pred(\mu)$, der nachfolgende Ort mit $suc(\mu)$ bezeichnet. Für einen Auftrag i in Fahrzeug f kann wird die Einsparung $\mathcal{E}(i)$ definiert als

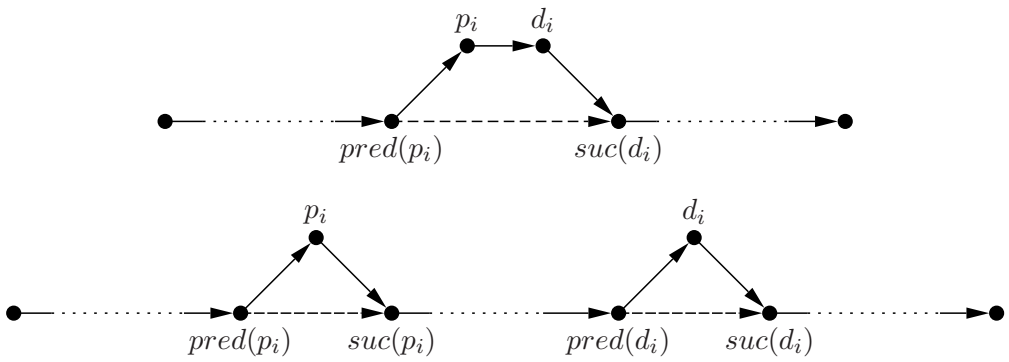

Abbildung 4.2: Heuristische Änderung der Route bei Eliminierung von Auftrag i für nicht direkt sowie direkt aufeinanderfolgende Pickup- und Delivery-Orte des Auftrags i

$$\mathcal{E}(i) = \begin{cases} c_{pred(p_i),p_i} + c_{p_i,d_i} + c_{d_i,suc(d_i)} - c_{pred(p_i),suc(d_i)} & \text{falls } \alpha_{p_i} > \tau \ \wedge \ x^f_{p_i d_i} = 1 \\ c_{pred(p_i),p_i} + c_{p_i,suc(p_i)} + c_{pred(d_i),d_i} + c_{d_i,suc(d_i)} - & \\ \quad - c_{pred(p_i),suc(p_i)} - c_{pred(d_i),suc(d_i)} & \text{falls } \alpha_{p_i} > \tau \wedge \ x^f_{p_i d_i} = 0 \\ 0 & \text{sonst} \end{cases}$$

Abbildung 4.2 stellt die Berechnung der Einsparung $\mathcal{E}(i)$ exemplarisch dar.

Aus allen noch nicht abgeholten Aufträgen $i \in N^\tau_f$ wird dann derjenige Auftrag i^* mit maximaler Einsparung $\mathcal{E}(i^*)$ gewählt:

$$\mathcal{E}(i^*) = \max_{i \in N^\tau : \alpha_{p_i} > \tau} \mathcal{E}(i)$$

Für diesen Auftrag i^* wird die Teilumgebung $U_{i^*}(\mathcal{L}_\tau)$ erzeugt und nach einer Lösung $\mathcal{L}'_\tau \in U_{i^*}(\mathcal{L}_\tau)$ mit geringerer Gesamtfahr-, -warte- und -servicezeit aller Fahrzeuge durchsucht.

Der Algorithmus zur Tabu Suche kann damit folgendermaßen skizziert werden:

1. Überprüfe, ob ein neuer Auftrag κ eingegangen ist. Falls ja: Abbruch der Tabu Suche.
2. Wähle den Auftrag $i^* \in N^\tau \setminus \mathcal{B}_\tau$ mit maximaler Einsparung $\mathcal{E}(i^*)$, und berechne die Teilumgebung $U_{i^*}(\mathcal{L}_\tau)$.
3. Durchsuche die Teilumgebung $U_{i^*}(\mathcal{L}_\tau)$ nach der Lösung $\mathcal{L}^*_\tau$ mit minimaler Gesamtzeit $C(\mathcal{L}^*_\tau)$. Ist $C(\mathcal{L}^*_\tau) < C(\mathcal{L}_\tau)$, übernimm die Lösung $\mathcal{L}^*_\tau$ als neue beste Lösung $\mathcal{L}_\tau$.
4. Gehe zu Schritt 1.

Im Folgenden werden die Ergebnisse der Heuristik unter Einbeziehung dieses Algorithmus der Tabu Suche vorgestellt.

Problemgruppe		Aufträge			gelöste Instanzen	
Name	**Aufträge**	**absolut**	**Δ**	**unterbr.**	**absolut**	**Δ**
100	51,86	51,86	25,66%	7,84	56	40,00%
200	103,08	103,08	41,05%	17,88	60	57,89%
400	205,66	205,66	57,36%	32,41	59	78,79%
600	308,97	308,97	60,66%	44,77	60	71,43%
800	411,75	411,75	64,03%	55,24	59	90,32%
1000	514,98	514,98	76,33%	65,98	58	100,00%

Tabelle 4.14: Die Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung des Algorithmus der Tabu-Suche

4.2.1 Testläufe und Auswertung

Für diese Testläufe wird die Testumgebung nun auf die kontinuierliche Simulation eingestellt, d. h. die Zeit zwischen dem Abschluss der Zuordnung eines Auftrags κ und der Anrufzeit des nächsten Auftrags $\kappa + 1$ wird nicht übersprungen, sondern kann für die Tabu-Suche genutzt werden. Statt der bisher benutzten einer Million maximal insgesamt zu erzeugenden Knoten wird nun eine Grenze von maximal 50.000 zu erzeugenden Knoten für die Zuordnung eines einzelnen Auftrags κ gesetzt. Wurde bereits vor Erreichen der Grenze von 50.000 Knoten eine zulässige Zuordnung gefunden, so wird der Auftrag akzeptiert. Eine Berechnung ohne eine solche Grenze ist leider wegen der auftretenden Rechenzeiten bei einigen Aufträgen nicht sinnvoll. Zum Vergleich werden die Ergebnisse der Testläufen mit der Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität und vorgezogenem Entscheidungszeitpunkt mit dem Sortierkriterium S_f^A, der Anzahl noch zu bedienender Aufträge, aus Abschnitt 4.1.2.1 herangezogen.

Tabelle 4.14 zeigt die Anzahl der bearbeiteten Aufträge und der vollständig gelösten Instanzen. Da die Begrenzung von einer Million maximal zu erzeugenden Knoten durch eine Begrenzung der für die Zuordnung eines Auftrags maximal zu erzeugenden Knoten ersetzt wurde, können nun alle Aufträge bearbeitet und somit auch alle Instanzen gelöst werden. In der Spalte Δ wird wie in den vorhergehenden Auswertungen die Steigerung im Vergleich zu den vorhergehenden Testläufen dargestellt. Spalte „unterbr.“ gibt die durchschnittliche Anzahl der Aufträge der Problemklasse an, bei denen die Zuordnung wegen der Überschreitung der 50.000 maximal pro Auftrag zu erzeugenden Knoten abgebrochen wurde. Die prozentuale Anzahl der Aufträge, deren Zuordnung unterbrochen wurde, liegt in allen Problemklassen zwischen 12,8% und 17,3% der eingegangenen Aufträge, wobei der höchste Anteil in der Problemgruppe „200“ erreicht wird und die Anteile für die Problemgruppen mit mehr Aufträgen danach kontinuierlich fallen. Dieses Phänomen lässt sich darin begründen, dass in den Problemgruppen mit mehr Aufträgen die Komplexität der Routenplanung für ein Fahrzeug zwar wegen der zunehmenden Anzahl bereits zugeordneter Aufträge zwar zunimmt, gleichzeitig jedoch die Wahrscheinlichkeit für das Finden einer zulässigen Lösung sinkt und wegen der

Problem-gruppe	♯ Knoten absolut erzeugt	unzulässig	expandiert	Duplikate
100	3.897.344,14	2.106.325,05	424.455,38	1.123.705,80
200	9.682.128,83	5.475.577,83	1.057.613,42	2.577.332,70
400	19.651.707,31	11.424.769,71	2.261.915,25	4.644.807,44
600	30.813.048,95	17.898.714,90	3.680.120,17	7.030.541,12
800	43.547.279,85	25.448.307,97	5.351.223,44	9.328.901,46
1000	60.646.231,76	37.428.455,60	7.457.210,72	11.618.064,03

Tabelle 4.15: Die Anzahl der Knoten bei Verwendung des Algorithmus der Tabu-Suche

Probl.-gruppe	erzeugt je Anfrage	erzeugt Δ	unzulässig je Anfrage	unzulässig Δ	expandiert je Anfrage	expandiert Δ	Duplikate je Anfrage	Duplikate Δ
100	75.155,40	1322,56%	40.617,84	1457,11%	8.185,09	332,83%	21.669,26	3221,82%
200	93.925,26	1851,15%	53.117,97	1922,90%	10.259,79	630,65%	25.002,42	3926,87%
400	95.553,88	2907,84%	55.551,46	2889,68%	10.998,27	1283,86%	22.584,77	5188,96%
600	99.729,36	4216,74%	57.930,89	4229,20%	11.911,06	2037,69%	22.755,01	6776,45%
800	105.762,55	5670,44%	61.805,88	5478,76%	12.996,43	3486,56%	22.656,95	8158,96%

Tabelle 4.16: Die Anzahl der Knoten bei Verwendung des Algorithmus der Tabu-Suche im Vergleich zur Heuristik ohne Tabu-Suche

Überprüfung der Zeitfensterkompatibilität sowohl vor Beginn als auch während der Routenplanung unzulässige Lösungen frühzeitig erkannt und nicht weiter betrachtet werden, so dass trotz höherer Komplexität des Routenplanungsproblems keine besonders große Anzahl Knoten benötigt wird.

In den Tabellen 4.15 und 4.16 werden die Anzahl der erzeugten, als unzulässig erkannten, der expandierten und der Duplikat-Knoten sowohl als absolute Werte als auch im Vergleich zu den Werten der Testläufe bei Verwendung der Heuristik ohne Tabu-Suche angegeben. Für den Vergleich wird nun allerdings wegen der geänderten Begrenzung der Anzahl zu erzeugender Knoten und der daraus folgenden unterschiedlichen Zuordnung der Aufträge der Quotient aus der absoluten Anzahl Knoten und der Anzahl der bearbeiteten Aufträge herangezogen. Tabelle 4.15 zeigt eine deutliche Steigerung der absoluten Anzahl Knoten: schon in Problemgruppe „100“ werden durchschnittlich fast 3,9 Millionen Knoten erzeugt. In Tabelle 4.16 wird in der Spalte „je Anfrage“ angegeben, wie viele Knoten je eingegangenem Auftrag erzeugt wurden, während die Spalte „Δ“ die prozentuale Steigerung dieses Werts gegenüber den Testläufen der Heuristik ohne Tabu-Suche wiedergibt. Die Anzahl der je Anfrage erzeugten Knoten liegt mit Werten von 75.155,40 bis 105.762,55 Knoten deutlich über der Grenze von 50.000 maximal pro Auftrag zu erzeugenden Knoten. Das liegt daran, dass der Algorithmus zur Tabu-Suche auch die Routenplanung durchführt und somit viele Aufträge häufiger als einmal in der Routenplanung berechnet werden. Die hier vorliegenden extremen Steigerungsraten sind daher nicht

Problem-gruppe	global max. Antwortzeit sec	Δ	Ø max. Antwortzeit sec	Δ	Ø Antwortzeit sec	Δ
100	37,067	-69,81%	2,4493	-57,93%	0,24509	40,33%
200	28,010	-71,83%	2,1967	-60,76%	0,22515	125,18%
400	26,569	-59,64%	2,0157	-53,12%	0,19180	315,72%
600	29,600	-1,83%	1,1912	-64,79%	0,11631	320,60%
800	32,413	-11,26%	1,4191	-42,75%	0,04926	200,43%
1000	14,317	-64,71%	0,8896	-72,85%	0,02124	20,75%

Tabelle 4.17: Die Antwortzeiten bei Verwendung des Algorithmus der Tabu-Suche

unbedingt kritisch für die Rechenzeiten, sondern zeigen die Nutzung der bisher freien Rechnerkapazität zwischen dem Abschluss der Zuordnung eines eingegangenen Auftrags κ und der Anrufzeit des folgenden Auftrags $\kappa + 1$ für die Tabu-Suche.

Tabelle 4.17 gibt die durchschnittliche, die durchschnittlich maximale und die maximale Antwortzeit der Aufträge in den Testinstanzen der Problemgruppen an. Im Gegensatz zu bisherigen Vergleichen der Antwortzeiten kann hier wegen des geänderten Abbruchkriteriums der maximal zu erzeugenden Knoten kein Vergleichswert für die gleichen Aufträge angegeben werden. In der Spalte „Delta“ wird daher die in diesen Testläufen erreichte Zeit direkt mit der in den Testläufen der Heuristik ohne Tabu-Suche erreichten Zeit verglichen. Die durchschnittliche Antwortzeit eines Auftrags liegt mit 0,02 bis 0,24 Sekunden höher als in den Testläufen zur Heuristik ohne Tabu-Suche mit anderem Abbruch-Kriterium, da nun alle eingehenden Aufträge bearbeitet werden und die vorher wegen der Überschreitung von einer Million Knoten nicht mehr betrachteten Aufträge wegen der bis dahin schon zugeordneten Menge von Aufträgen tendenziell längere Rechenzeiten benötigen. Die Steigerungsraten von bis zu 320,60% sind wegen der geringen absoluten Steigerung der Antwortzeiten jedoch nicht kritisch. Die absolut erreichten durchschnittlichen Antwortzeiten von maximal $\frac{1}{4}$ Sekunde sind zufriedenstellend, ebenso die durchschnittlich maximalen Antwortzeiten von 0,89 bis 2,45 Sekunden. Hier wurden wegen der Änderung des Abbruchkriteriums auf maximal 50.000 für die Zuordnung eines einzelnen Auftrags zu erzeugenden Knoten Reduktionen von 42,75% bis 72,85% im Vergleich zu den Testläufen aus Abschnitt 4.1.2.1 erzielt. Diese Änderung schlägt sich auch in den Ergebnissen der global maximalen Antwortzeiten nieder: mit absoluten Werten von nur 14,32 bis 37,07 Sekunden werden die absolut maximalen Rechenzeiten der Problemgruppen um bis zu 71,83% reduziert. Diese Ergebnisse sind zwar von den angestrebten „wenigen Sekunden“ für die Antwortzeit noch entfernt, bilden aber die Ausnahme und sind daher noch vertretbar.

In Tabelle 4.18 werden die Anzahl der akzeptierten Aufträge und die Gesamtfahrzeit der Fahrzeuge dargestellt. Die Anzahl der akzeptierten Aufträgen ist gegenüber den Testläufen der Heuristik ohne Tabu-Suche deutlich gestiegen. Diese Entwicklung ist jedoch zum Teil auch auf die geänderte Abbruchbedingung zurückzuführen ist. Dementsprechend ist auch die Gesamtfahr-, -warte- und -servicezeit der Fahrzeuge deutlich gestiegen. Zum Vergleich wird

Problem-gruppe	akzeptierte Aufträge absolut	Δ	Gesamtzeit in sec absolut	je akz. Auftrag	Δ
100	44,50	26,69%	5.019,82	112,80	-10,45%
200	78,97	38,21%	11.632,61	147,31	-11,57%
400	145,71	48,76%	25.101,07	172,27	-12,50%
600	213,15	50,62%	45.500,13	213,47	-10,66%
800	278,95	53,00%	70.494,86	252,72	-12,47%
1000	350,17	64,19%	99.636,46	284,54	-10,89%

Tabelle 4.18: Die Anzahl abgelehnter Aufträge und die Zielfunktionswerte bei Verwendung des Algorithmus der Tabu-Suche im Vergleich zur Verwendung der Heuristik ohne Tabu-Suche

Problem-gruppe	Tabu Suche Aufträge i^*	getauscht	in %
100	133,38	13,02	9,76%
200	401,28	25,93	6,46%
400	1.364,88	55,98	4,10%
600	2.760,18	80,48	2,92%
800	4.558,64	101,05	2,22%
1000	6.798,83	124,72	1,83%

Tabelle 4.19: Die Anzahl der während des Verfahrens der Tabu-Suche getauschten Aufträge

daher in der Spalte „je akz. Auftrag“ die pro akzeptiertem Auftrag benötigte Fahrt-, Warte- und Servicezeit als Quotient der absoluten Gesamtzeit und der Anzahl akzeptierten Aufträge betrachtet. Hier wird mit Hilfe der Tabu-Suche eine Reduktion um bedeutende 10,45% bis 12,50% im Vergleich zu den Testläufen aus Abschnitt 4.1.2.1 erreicht.

Abschließend stellt Tabelle 4.19 die Anzahl der Tauschvorgänge im Algorithmus der Tabu-Suche dar. In der Spalte „Aufträge i^*“ wird die durchschnittliche Anzahl der während einer Testinstanz für den Tausch ausgewählten Aufträge angegeben. Ein Auftrag, der während der Tabu-Suche mehrfach zum Tauschen ausgewählt wurde, wird an dieser Stelle mehrfach angegeben. Die Spalte „getauscht“ gibt die durchschnittliche Anzahl der erfolgreich zwischen zwei Fahrzeugen verschobenen Aufträgen wieder, und Spalte „in %“ zeigt die durchschnittliche Anzahl der erfolgreich verschobenen Aufträge im Verhältnis zu den zum Verschieben ausgewählten Aufträgen. Es fällt auf, dass die Anzahl der für den Tausch ausgewählten Aufträge i^* in den Problemgruppen mit mehr Aufträgen überproportional zunimmt, während die Anzahl der erfolgreich verschobenen Aufträge eher proportional wächst, was zu einem deutlichen Rückgang der relativen Werte in Spalte Δ führt.

4.2.1.1 Fazit

Mit der Einführung einer Tabu-Suche in der Zeit $[\tau_\kappa + \epsilon, \tau_{\kappa'}]$ zwischen der vollständig berechneten Zuordnung eines Auftrags κ und dem folgenden Auftragseingang von κ' konnte die Heuristik deutlich bessere durchschnittliche Fahrt-, Warte- und Servicezeiten pro akzeptiertem Auftrag erzielen. Die Anzahl der akzeptierten Aufträge konnte ebenfalls gesteigert werden.

Die Einführung des geänderten Abbruchkriteriums von maximal 50.000 zu erzeugenden Knoten für die Zuordnung eines einzelnen Auftrags führte dazu, dass nun eine Heuristik vorliegt, die alle Testinstanzen vollständig berechnet und trotzdem die gewünschten Antwortzeiten von nur wenigen Sekunden bis auf wenige Ausnahmen einhält. Die in allen Problemgruppen erreichte maximale Antwortzeit eines Auftrags von 37 Sekunden ist für ein Telefonat immer noch vertretbar. Das Ziel, einen effizienten Algorithmus zur Lösung des dynamischen Pickup und Delivery Vehicle Routing Problems mit Zeitfenstern zur Verfügung zu stellen, ist damit erreicht worden.

Die Tabu-Suche selbst kann wahrscheinlich effizienter gestaltet werden, indem

- die Umgebung anders definiert wird. Vorstellbar wäre zum Beispiel eine Einbeziehung der Zeitfensterrestriktionen in die Umgebungsdefinition, um eine exaktere Berechnung der tatsächlich möglichen Einsparungen bei Entfernen eines Auftrags i^* aus einem Fahrzeug oder eine frühzeitige Erkennung der wegen der Zeitfensterrestriktionen nicht in andere Fahrzeuge verschiebbaren Aufträge zu ermöglichen.
- mit Hilfe eines Matching-Verfahrens eine Tauschmöglichkeit von mehreren Aufträgen mit ähnlichen Zeitfenstern aus unterschiedlichen Fahrzeugen gesucht wird

Die Verbesserung der Tabu-Suche ist jedoch nicht Gegenstand dieser Arbeit.

Kapitel 5

Das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern und Ladebedingungen

Nachdem ein Algorithmus entwickelt wurde, mit dem das dynamische Pickup and Delivery Vehicle Routing Problem mit Zeitfenstern gelöst werden kann, können abschließend weitere für die Praxis relevante Einschränkungen betrachtet werden. In diesem Kapitel werden Ladebedingungen als weitere praxisrelevante Nebenbedingungen vorgestellt, ihre Auswirkungen auf die Anzahl der zulässigen Lösungen und damit auf die Rechenzeiten analysiert und die Ergebnisse der Testläufe vorgestellt.

Ladebedingungen beschreiben die Regeln, die beim Be- und Entladen eines Fahrzeugs beachtet werden müssen:

Definition 5.1 (Ladebedingungen)
Ladebedingungen sind Nebenbedingungen, die Regeln für zulässiges Be- oder Entladen der Fahrzeuge beinhalten.

Die Notwendigkeit solcher Nebenbedingungen für den Transport von Gütern ist offensichtlich: das zuerst eingeladene Gut wird sich im Normalfall auf der Ladefläche weit hinten befinden, während die zuletzt eingeladenen Güter eher in der Nähe der Ladeluke liegen, so dass sie leichter entladen werden können. Wenn

- besonders schwere und/oder sperrige Güter
- besonders hochwertige und/oder empfindliche Güter

transportiert werden sollen, werden Ladebedingungen von besonderem Interesse sein, wobei natürlich der in Kauf genommene Umweg im Verhältnis zur Einsparung an Umladezeit bzw. Transportschäden durch Umladen stehen sollte.

In der Literatur finden sich einige Arbeiten zu statischen Vehicle Routing Problemen mit Backhauls (vgl. den Überblick in [77]). Diese Ladebedingung besagt, dass in einer ersten Phase zunächst alle geladenen Güter ausgeladen werden, bevor in der zweiten Phase Güter eingeladen werden. Sie wird häufig beim Massentransport eingesetzt, oft auch bei der Brief- und Paketzustellung. Im vorliegenden Fall müssen die Güter zunächst abgeholt, dann ausgeliefert werden. Die Nebenbedingung „Backhauls“ wird daher entsprechend definiert. Die Auswirkungen dieser Ladebedingung auf den entwickelten Algorithmus wird im Abschnitt 5.1 analysiert.

Bei der Routenplanung kann auch die Ladebedingung „LIFO“ (Last In, First Out) eine Rolle spielen, insbesondere beim Transport von schweren und/oder sperrigen Gütern. Betrachtet man den Transport von Gütern auf Europaletten, die sich im Logistikgewerbe durchgesetzt haben, so kommt noch eine weitere Variante in Frage: da mehrere Paletten nebeneinander auf einem Fahrzeug Platz finden, können mehrere Güter für den nächsten Entladevorgang in Frage kommen. Sollen die q zuletzt eingeladenen Güter ausgeladen werden dürfen, wird die Ladebedingung im Folgenden mit „LIFO-q“ bezeichnet. Die Ladebedingungen „LIFO“ und „LIFO-q“ werden im Abschnitt 5.2 untersucht und ihre Auswirkungen auf die Laufzeit des Algorithmus beschrieben.

Im Abschnitt 5.3 wird abschließend ein Fazit gezogen.

5.1 Backhauls

Dieser Abschnitt behandelt die Ladebedingung „Backhauls“. Sie wird folgendermaßen definiert:

Definition 5.2 (Backhauls)
Eine Route genügt der Ladebedingung „Backhauls“, wenn sie sich in genau eine Linehaul- und genau eine Backhaul-Phase unterteilen lässt, wobei das Fahrzeug während der Linehaul-Phase nur beladen und in der anschließenden Backhaul-Phase nur entladen wird.
Ein Routenplan genügt der Ladebedingung „Backhauls“, wenn alle Fahrzeuge die Ladebedingung erfüllen.

Im dynamischen Umfeld kann diese Ladebedingung natürlich immer nur für den zukünftigen Routenplan gelten. Geht ein Auftrag κ ein, so dürften bei strikter Einhaltung der Backhauls-Bedingung bis zum Zeitpunkt τ_κ keine Aufträge ausgeliefert worden sein. Diese strikte Regel ist jedoch nur in Spezialfällen sinnvoll, da dann alle Fahrzeuge nur einladen und dann warten müssten, bis sie keine neuen Aufträge mehr annehmen können, bevor sie mit dem Entladen beginnen können. Die Nebenbedingung „Backhauls“ gilt daher immer nur für die zukünftigen Routen.

Die Ladebedingung „Backhauls“ verlangt, dass erst alle Aufträge eingeladen werden, bevor mit dem Entladen begonnen wird. Sie bildet damit eine zusätzliche Nebenbedingung für das statische Teilproblem (vgl. Abschnitt 2.3.3), das bei Eingang eines neuen Auftrags κ zu lösen

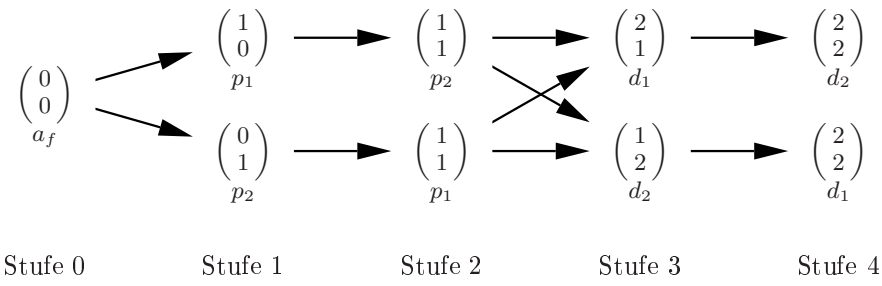

Abbildung 5.1: Der Zustandsgraph $G^B_{2,0}$ mit Backhauls für zwei nicht bearbeitete Aufträge.

ist. Diese Nebenbedingung kann für das statische Modell aus Abschnitt 2.3.3.2 formuliert werden, indem das Anfahren eines Pickup-Ortes p_i nach dem Anfahren eines Delivery-Ortes d_i untersagt wird:

$$\sum_{\mu \in D \setminus \{a_f\}} \sum_{\nu \in P} x_{\mu,\nu} = 0$$

Die Heuristik zur Berechnung eines zulässigen Routenplans löst das statische Teilproblem bei Eingang eines neuen Auftrags κ, indem für jedes Fahrzeug $f \in F$ mit festgestellter Zeitfensterkompatibilität der bisher zugeordneten Aufträge $i \in N_f^{\tau_\kappa}$ und dem neu eingegangenen Auftrag κ die Routen $\overrightarrow{R}_{N_f^{\tau_\kappa} \cup \{\kappa\}}$ berechnet werden. Das Verfahren der Tabu Suche sucht nach besseren Lösungen, indem für einen Auftrag $i \in N_f^\tau$ die Routen $\overrightarrow{R}_{N_f^\tau \setminus \{ß\}}$ für Fahrzeug f und $\overrightarrow{R}_{N_{f'}^\tau \cup \{ß\}}$ für jedes Fahrzeug $f' \neq f$ berechnet werden. Die Berechnung einer solchen Route $\overrightarrow{R}$ entspricht der Lösung des Single Vehicle Routing Problems mit Hilfe des A*-Algorithmus auf einem Zustandsgraphen, der nun zusätzlich zur Reihenfolgebedingung auch die Ladebedingung in seiner Struktur verankern soll.

Es wird daher in 5.1.1 der Zustandsgraph mit Backhauls $G^B = (V^B_{n_f,k^*}, E^B_{n_f,k^*}, c^B_{n_f,k^*})$ eingeführt und die Anzahl seiner Knoten und Kanten analysiert, um Aussagen über die maximale Laufzeit machen und den Vergleich zur maximalen Laufzeit des Algorithmus ohne Ladebedingungen herstellen zu können. Anschließend werden in 5.1.2 die Implementierung der Ladebedingung und die Ergebnisse der Testläufe vorgestellt.

5.1.1 Der Zustandsgraph mit Backhauls

Die Ladebedingung „Backhauls“ verlangt, dass erst alle Aufträge eingeladen werden, bevor mit dem Entladen begonnen wird. Für die Struktur des Zustandsgraphen mit Backhauls $G^B = (V^B_{n_f,k^*}, E^B_{n_f,k^*}, c^B_{n_f,k^*})$ impliziert dies, dass in der Linehaul-Phase bis zur Stufe n_f für alle Aufträge nur die Zustände 0 und 1 zulässig sind, während in der Backhaul-Phase auf den Stufen $n_f + 1$ bis $2n_f$ nur die Zustände $\varphi(i) = 1$ und $\varphi(i) = 2$ möglich sind.

Die Knotenmenge $V^B_{n_f,k^*}$ besteht aus der Vereinigung der Knotenmengen ${}_kV^B_{n_f,k^*}$ der Stufen $k = k^*,\ldots,2n_f$, wobei die Knotenmenge der Stufe k^* als einzigen Knoten die Quelle des Graphen enthält. O.B.d.A. seien die Aufträge so nummeriert, dass auf Stufe k^* die ersten k^* Aufträge bereits den Status 1 haben. Diese k^* ersten Einträge des Statusvektors Φ werden im Folgenden der obere Bereich genannt, während die restlichen $n_f - k^*$ Einträge den unteren Bereich des Statusvektors bilden.

Für die Knotenmengen der Linehaul-Stufen $k = k^*,\ldots,n_f$ des Graphen gilt, dass die ersten k^* Einträge des Statusvektors Φ den Wert 1 haben und von den restlichen $n_f - k^*$ Einträgen genau $k - k^*$ den Wert 1 besitzen, alle anderen Einträge sind 0. Für die Fahrzeugposition μ gilt, dass es ein Pickup-Ort p_i sein muss, wobei der zugehörige Auftrag i zu den unteren Aufträgen mit Status $\varphi(i) = 1$ gehört. Die Knotenmenge ${}_kV^B_{n_f,k^*}$ lässt sich schreiben als:

$$
\begin{aligned}
{}_kV^B_{n_f,k^*} = \Big\{(\Phi,\mu)\Big|&\varphi(1) = \cdots = \varphi(k^*) = 1;\ \varphi(i) \in \{0;1\}\ \forall i = k^*+1,\ldots,n_f;\\
&\sum_{i=k*+1}^{k} \varphi(i) = k - k^*;\ \mu \in \{p_i | i \in \{k^*+1,\ldots,n_f\}, \varphi(i) = 1\}\Big\}, \qquad k = k^*,\ldots,n_f
\end{aligned}
$$

Auf den Backhaul-Stufen $k = n_f + 1,\ldots,2n_f$ sind nur die Einträge 1 oder 2 im Statusvektor zulässig. Die Summe aller Einträge des Statusvektors muss k entsprechen. Für die Fahrzeugposition μ kommen alle Delivery-Orte d_i in Frage, deren zugehörige Aufträge den Status 2 haben:

$$
{}_kV^B_{n_f,k^*} = \Big\{(\Phi,\mu)\Big|\varphi(i) \in \{1;2\}\ \forall i = 1,\ldots,n_f;\ \sum_{i=1}^{n_f}\varphi(i) = k;\ \mu \in \{d_i | i \in \{1,\ldots,n_f\}; \varphi(i) = 2\}\Big\},
\qquad k = k^*+1,\ldots,2n_f
$$

Mit

$$
V^B_{n_f,k^*} = \bigcup_{k=k^*}^{2n_f} {}_kV^B_{n_f,k^*}
$$

ist die Knotenmenge des Graphen $G^B_{n_f,k^*}$ definiert.

Für die Kanten des Zustandsgraphen mit Backhauls gilt: zwei Knoten $(\Phi,\mu),(\Phi',\mu') \in {}_kV^B_{n_f,k^*}$ sind genau dann mit einer Kante verbunden, wenn es einen Einheitsvektor e_i gibt, so dass der Statusvektor Φ' durch Addition von e_i zum Statusvektor Φ entsteht und die Fahrzeugposition μ' entsprechend $\varphi(i)$ den Pickup- oder Delivery-Ort, p_i bzw. d_i, des Auftrags i enthält. Die Kantenmenge $E^B_{n_f,k^*}$ kann daher definiert werden als:

$$
E^B_{n_f,k^*} = \Big\{\big((\Phi,\mu),(\Phi',\mu')\big) \in V^B_{n_f,k^*} \times V^B_{n_f,k^*}\Big|\exists\, i \in \{1\ldots n_f\} : \Phi' = \Phi + e_i, \mu' \in \{p_i, d_i\}\Big\}
$$

Die Kostenfunktion $c^B_{n_f,k^*} : E^B_{n_f,k^*} \to \mathbb{R}$ entspricht der Kostenfunktion $c^Z_{n_f,k^*}$ des Zustandsgraphen $G^Z_{n_f,k^*}$. Sie beinhaltet die Fahrzeit $c_{\mu,\mu'}$ für den Weg von Ort μ zu Ort μ' und die

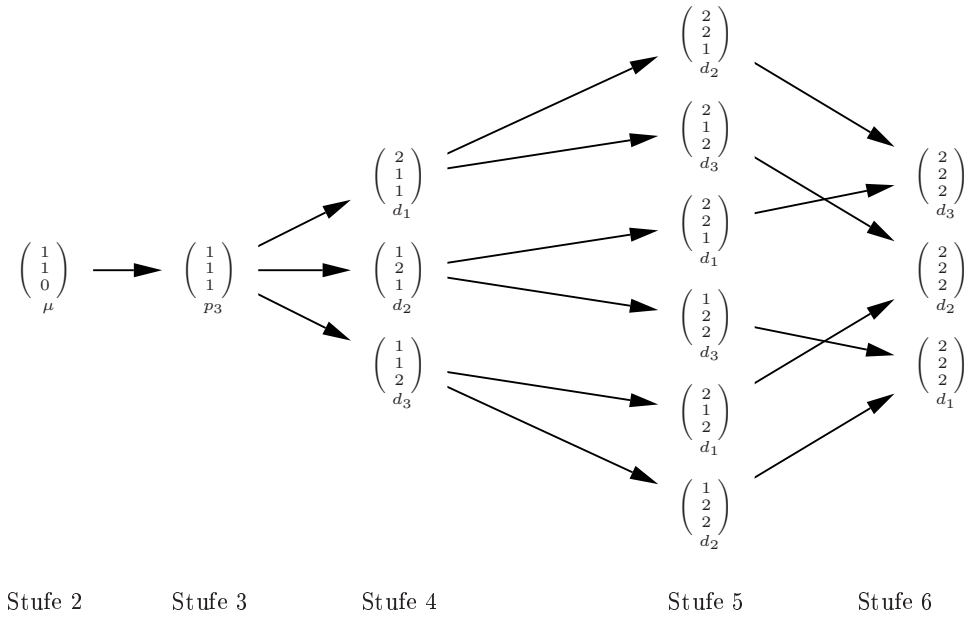

Abbildung 5.2: Der Zustandsgraph $G^B_{3,2}$ mit Backhauls für drei Aufträge mit Quelle auf Stufe $k^* = 2$.

Wartezeit $\omega_{\mu'}$ bis zum Beginn des für den Ort μ' geltenden Zeitfensters $[\underline{t}_{\mu'}, \overline{t}_{\mu'}]$. Damit ist der Graph $G^B_{n_f,k^*}$ definiert.

Die Abbildungen 5.1 und 5.2 zeigen beispielhaft Zustandsgraphen mit Backhauls. In 5.1 wird $G^B_{2,0}$ dargestellt, der Zustandsgraph für zwei Aufträge beginnend auf Stufe 0, während Abbildung 5.2 den Zustandsgraphen mit Backhauls $G^B_{3,2}$ für drei Aufträge mit Quelle auf Stufe 2 zeigt. Im Folgenden wird der Zustandsgraph mit Backhauls $G^B_{n_f,k^*}$ analysiert, um anhand der Anzahl Knoten $|V^B_{n_f,k^*}|$ und der Anzahl Kanten $|E^B_{n_f,k^*}|$ Rückschlüsse auf die maximale Rechenzeit des A*-Algorithmus ziehen zu können.

Satz 5.1
Es bezeichne $|V^B_{n_f,k^*}|$ die Anzahl der Knoten und $|E^B_{n_f,k^*}|$ die Anzahl der Kanten im Zustandsgraphen mit Backhauls für n_f Aufträge, dessen Quelle auf Stufe k^* liegt. Dann gilt:

$$| V^B_{n_f,k^*} | = 1 + \sum_{k=k^*+1}^{n_f} \frac{(n_f - k^*)!}{(k - k^* - 1)!(n_f - k)!} + \sum_{k=n_f+1}^{2n_f} \frac{n_f!}{(k - n_f - 1)!(2n_f - k)!}$$

$$|V^B_{n_f,k^*}| = n_f - k^* + \sum_{k=k^*+1}^{n_f-1} \frac{(n_f - k^*)!}{(k - k^* - 1)!(n_f - k - 1)!} + (n_f - k^*)n_f + \sum_{k=n_f+1}^{2n_f-1} \frac{n_f!}{(k - n_f - 1)!(2n_f - k - 1)!}$$

Beweis: Der Beweis erfolgt konstruktiv mit Hilfe kombinatorischer Überlegungen. Es wird zunächst die Formel für die Anzahl $|V^B_{n_f,k^*}|$ der Knoten und anschließend die Formel für die Anzahl $|E^B_{n_f,k^*}|$ der Kanten bewiesen.

Die Anzahl $|V^B_{n_f,k^*}|$ der Knoten im Zustandsgraphen mit Backhauls $G^B_{n_f,k^*}$ wird analog zur Definition der Knotenmenge V^B des Graphen stufenweise berechnet und für eine Stufe $k \in \{k^*, \ldots, 2n_f\}$ mit $|_kV^B_{n_f,k^*}|$ bezeichnet. Auf Stufe k^* gibt es genau eine Quelle, so dass $|_{k^*}V^B_{n_f,k^*}| = 1$ gilt. Für die Linehaul-Stufen $k = k^* + 1, \ldots, n_f$ gilt, dass die Einträge der Statusvektoren im oberen Bereich konstant $\varphi(i) = 1$ sind, während sich die $n_f - k^*$ Einträge im unteren Bereich aus $k - k^*$ Einträgen 1 und $n_f - k$ Einträgen 0 zusammensetzen. Es gibt daher auf Stufe k mit $k^* < k \leq n_f$

$$\frac{(n_f - k^*)!}{(k - k^*)!(n_f - k)!}$$

verschiedene Statusvektoren. Die Fahrzeugposition μ kann nur Pickup-Orte von Aufträgen aus dem unteren Bereich des Statusvektors Φ mit Status $\varphi(i) = 1$ enthalten. Da im unteren Bereich auf Stufe k genau $k-k^*$ Einträge 1 sind, sind für jeden Statusvektor $k-k^*$ Fahrzeugpositionen zulässig. Auf einer Linehaul-Stufe k gibt es daher

$$|_kV^B_{n_f,k^*}| = \frac{(n_f - k^*)!(k - k^*)}{(k - k^*)!(n_f - k)!} = \frac{(n_f - k^*)!}{(k - k^* - 1)!(n_f - k)!}, \qquad k = k^* + 1, \ldots, n_f$$

verschiedene Knoten.

Auf den Backhauls-Stufen $k = n_f + 1, \ldots, 2n_f$ gilt für die n_f Einträge des Statusvektors, dass genau $k - n_f$ Einträge den Wert $\varphi(i) = 2$ haben, für die restlichen $2n_f - k$ Einträge gilt $\varphi(i) = 1$. Es gibt also

$$\frac{n_f!}{(k - n_f)!(2n_f - k)!}$$

verschiedene Statusvektoren. Als Fahrzeugposition kommen die Delivery-Orte aller Aufträge mit Status 2 in Frage, also für jeden Statusvektor $k - n_f$ verschiedene Orte. Somit gibt es auf einer Backhauls-Stufe k

$$|_kV^B_{n_f,k^*}| = \frac{n_f!(k - n_f)}{(k - n_f)!(2n_f - k)!} = \frac{n_f!}{(k - n_f - 1)!(2n_f - k)!}, \qquad k = n_f + 1, \ldots, 2n_f$$

verschiedene Knoten. Insgesamt erhält man damit

$$|V^B_{n_f,k^*}| = 1 + \sum_{k=k^*+1}^{n_f} \frac{(n_f - k^*)!}{(k - k^* - 1)!(n_f - k)!} + \sum_{k=n_f+1}^{2n_f} \frac{n_f!}{(k - n_f - 1)!(2n_f - k)!}$$

Knoten im Zustandsgraphen G^B.

Für die Berechnung der Anzahl $|E^B_{n_f,k^*}|$ der Kanten wird der Graph ebenfalls in Linehaul- und Backhaul-Stufen unterteilt. Die Anzahl Kanten von Stufe k zu Stufe $k + 1$ werden mit $|_kE^B_{n_f,k^*}|$ bezeichnet. Von der Quelle auf Stufe k^* geht für jeden Status $\varphi(i) = 0$ eine Kante zur Stufe $k^* + 1$, also gilt: $|_{k^*}E^B_{n_f,k^*}| = n_f - k^*$. Für die Stufen $k = k^* + 1, \ldots, n_f - 1$ gilt, dass jeder Knoten genauso viele Nachfolger wie Einträge $\varphi(i) = 0$ im Statusvektor Φ hat. Für

jeden der $n_f - k$ Einträge $\varphi(i) = 0$ eines Statusvektors auf Stufe k existiert daher eine Kante zur nächsten Stufe $k+1$. Mit Hilfe der bereits bekannten Anzahl $|{}_kV^B_{n_f,k^*}|$ der Knoten auf einer Linehaul-Stufe k lässt sich die Anzahl der ausgehenden Kanten daher berechnen als

$$|{}_kE^B_{n_f,k^*}| = \frac{(n_f - k^*)!(n_f - k)}{(k - k^* - 1)!(n_f - k)!} = \frac{(n_f - k^*)!}{(k - k^* - 1)!(n_f - k - 1)!}, \quad k = k^* + 1, \ldots, n_f - 1$$

Im Übergang von der letzten Linehaul-Stufe n_f zur ersten Backhaul-Stufe $n_f + 1$ hat jeder Knoten $(\Phi, \mu)^T$ der Stufe n_f genau n_f nachfolgende Knoten $(\Phi', \mu')^T$ auf Stufe $n_f + 1$, da jeder Status von $\varphi(i) = 1$ auf $\varphi'(i) = 2$ geändert werden darf. Da es auf Stufe n_f genau $n_f - k^*$ Knoten gibt, erhält man $(n_f - k^*)n_f$ Kanten von Stufe n_f zu Stufe $n_f + 1$. Auf den Backhauls-Stufen $k = n_f + 1, \ldots, 2n_f - 1$ gilt, dass jeder Eintrag $\varphi(i) = 1$ im Statusvektor Φ eine Kante zu einem Nachfolgeknoten impliziert. Da jeder Statusvektor der Stufe k genau $2n_f - k$ Einträge 1 besitzt, gehen von Stufe k insgesamt

$$|{}_kE^B_{n_f,k^*}| = \frac{n_f!(2n_f - k)}{(k - n_f - 1)!(2n_f - k)!} = \frac{n_f!}{(k - n_f - 1)!(2n_f - k - 1)!}, \quad k = n_f + 1, \ldots, 2n_f$$

Kanten aus. Die Stufe $2n_f$ des Graphen enthält die Senken, so dass keine weiteren Kanten zu betrachten sind. Insgesamt erhält man somit

$$|E^B_{n_f,k^*}| = 1 + \sum_{k=k^*+1}^{n_f-1} \frac{(n_f - k^*)!}{(k - k^* - 1)!(n_f - k - 1)!} + (n_f - k^*)n_f + \sum_{k=n_f+1}^{2n_f-1} \frac{n_f!}{(k - n_f - 1)!(2n_f - k - 1)!}$$

Kanten.

□

Die Tabelle 5.1 zeigt beispielhaft die Anzahl Knoten und Kanten im Zustandsgraphen $G^B_{n_f,k^*}$ mit Backhauls für $n_f = 1, \ldots, 10$ Aufträge, immer beginnend auf Stufe $k^* = 0$. Im Vergleich zum Zustandsgraphen ohne Ladebedingungen (Tabelle 3.6) ist die Anzahl der Knoten und Kanten erwartungsgemäß deutlich gesunken, so dass Einsparungen bei den Rechenzeiten zu erwarten sind. Im folgenden Abschnitt werden die Ergebnisse der Testläufe vorgestellt und analysiert.

5.1.2 Testläufe und Auswertung

Für diese Testläufe wird die Testumgebung analog zu den Testläufen bis einschließlich Abschnitt 4.1.2.1 ereignisdiskret eingestellt; zur Beschränkung der Rechenzeit wird wieder auf die Grenze von einer Million maximal insgesamt zu erzeugenden Knoten zurückgegriffen. Für die Vergleiche werden die Ergebnisse der Testläufe zur Verwendung der Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität und Sortierung der Fahrzeuge nach dem Sortierkriterium S^A_f aus Abschnitt 4.1.2.1 verwendet.

♯ Aufträge n_f	$\lvert V^B_{n_f,0}\rvert$	Δ	$\lvert E^B_{n_f,0}\rvert$	Δ
1	3	0,00%	2	0,00%
2	9	-30,77%	10	-37,50%
3	25	-54,55%	36	-64,71%
4	65	-70,05%	116	-78,68%
5	161	-80,15%	350	-86,38%
6	385	-86,80%	1.002	-91,04%
7	897	-91,21%	2.744	-94,03%
8	2.049	-94,14%	7.240	-96,00%
9	4.609	-96,10%	18.522	-97,31%
10	10.241	-97,40%	46.190	-98,19%

Tabelle 5.1: Anzahl der Knoten und Kanten in $G^B_{n_f,k^*}$ für n_f Aufträge, beginnend auf Stufe 0, im Vergleich mit der Anzahl Knoten und Kanten des Zustandsgraphen $G^Z_{n_f,k^*}$.

Problemgruppe		**Aufträge**		**gelöste Instanzen**	
Name	**Aufträge**	**absolut**	Δ	**absolut**	Δ
100	51,86	46,30	12,20%	46	15,00%
200	103,08	83,12	13,32%	44	15,79%
400	205,66	157,92	20,44%	42	31,25%
600	308,97	214,18	10,91%	37	5,71%
800	411,75	294,00	12,89%	39	21,88%
1000	514,98	361,21	14,98%	38	22,58%

Tabelle 5.2: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung von $G^B_{n_f,k^*}$ im Vergleich zur Verwendung von $G^Z_{n_f,k^*}$

Tabelle 5.2 zeigt die Anzahl der bearbeiteten Aufträge und gelösten Instanzen bei Einhaltung der Backhauls-Ladebedingung. Die Anzahl der bearbeiteten Aufträge steigt in allen Problemgruppen um 10,91% bis 20,44% an, die Anzahl der vollständig gelösten Instanzen steigt ebenfalls in allen Problemgruppen mit Steigerungsraten von 5,71% bis 31,25%. Diese Steigerungen liegen in der deutlichen Reduktion der Anzahl Knoten im Zustandsgraphen $G^B_{n_f,k^*}$ mit Backhauls-Ladebedingung im Vergleich zum Zustandsgraphen $G^Z_{n_f,k^*}$ begründet (vgl. Tabelle 5.1).

In den Tabellen 5.3 und 5.4 werden die durchschnittlichen Anzahlen der erzeugten, der als unzulässig erkannten, der expandierten und der Duplikat-Knoten angegeben. Die absolut erzeugten Knoten in Tabelle 5.3 liegen wegen der hohen Anzahl vollständig gelöster Instanzen in allen Problemgruppen weit unter der maximalen Grenze von einer Million maximal zu erzeugenden Knoten. In Tabelle 5.4 zeigt sich wiederum die deutliche Reduktion des zugrunde

Problem-gruppe	♯ Knoten absolut			
	erzeugt	unzulässig	expandiert	Duplikate
100	169.940,89	28.222,39	32.238,96	108.048,98
200	272.894,70	55.241,02	52.697,73	161.910,58
400	321.734,22	92.160,78	68.448,95	157.632,39
600	396.128,10	108.338,25	86.144,40	195.395,33
800	403.703,63	129.691,68	91.529,34	175.921,15
1000	387.908,91	132.829,53	93.196,34	154.034,79

Tabelle 5.3: Anzahl der Knoten gesamt bei Verwendung von $G^B_{n_f,k^*}$

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	12.856,84	-94,10%	3.282,25	-96,95%	3.695,46	-86,27%	5.386,46	-93,10%
200	30.318,48	-91,62%	12.182,02	-93,99%	8.078,80	-82,59%	9.371,57	-90,77%
400	48.507,25	-89,25%	18.722,34	-92,90%	14.574,41	-75,58%	13.730,42	-88,04%
600	75.533,98	-82,25%	36.097,63	-85,16%	20.867,57	-66,05%	15.584,67	-85,11%
800	119.763,36	-74,17%	43.691,69	-84,72%	32.613,85	-52,94%	38.966,24	-55,17%
1000	96.727,41	-82,77%	38.999,48	-88,64%	32.059,47	-63,26%	20.905,53	-80,70%

Tabelle 5.4: Anzahl der Knoten bei Verwendung von $G^B_{n_f,k^*}$ im Vergleich zur Verwendung von $G^Z_{n_f,k^*}$

liegenden Zustandsgraphen $G^B_{n_f,k^*}$ mit Backhauls-Ladebedingung im Vergleich zum Zustandsgraphen $G^Z_{n_f,k^*}$: die Anzahl der erzeugten Knoten sinkt um 74,17% bis 94,10%. Die Reduktion der als unzulässig erkannten, der expandierten und der Duplikat-Knoten liegt ebenfalls in diesem Bereich.

Die erreichten Antwortzeiten werden in Tabelle 5.5 dargestellt. Hier zeigen sich bemerkenswerte Veränderungen im Vergleich zu den Testläufen aus Abschnitt 4.1.2.1: die durchschnittlichen Antwortzeiten verschlechtern sich in den Problemgruppen „100“ und „600“ um 17,56% bzw. 18,34%, während sie sich in den anderen Problemgruppen um 8,93% bis 29,13% verbessern. Die durchschnittlichen Antwortzeiten sind jedoch mit Werten unter $\frac{1}{100}$ Sekunden so gering, dass diese relativen Änderungen nur minimale Auswirkungen haben. Bei den durchschnittlich maximalen Antwortzeiten ist in den Problemgruppen „100“ und „600“ eine Steigerung um 198,95% bzw. 26,63% zu verzeichnen, während in den anderen Problemgruppen die durchschnittlich maximalen Antwortzeiten um 38,11% - 52,88% reduziert werden. Allerdings liegen auch hier die absoluten Werte noch unter $\frac{1}{100}$ Sekunde. Die global maximale Antwortzeit nimmt in den Problemgruppen „100“ und „600“ wegen jeweils einer Testinstanz mit deutlich höherer maximaler Antwortzeit um 327,43% bzw. 163,94% zu; in allen anderen Problemgruppen wird sie um 37,7% bis 71,73% verringert, wobei alle Antwortzeiten im Vergleich unter 0,3 Sekunden liegen, während die absolut erreichten Antwortzeiten maximal 7,43

Problem-	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
gruppe	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	7,43	0,12	327,43%	0,2011	0,0041	198,95%	0,01008	0,00029	17,56%
200	4,92	0,10	-37,77%	0,1466	0,0055	-41,98%	0,00548	0,00063	-29,13%
400	1,37	0,14	-71,73%	0,0592	0,0063	-49,94%	0,00296	0,00065	-8,93%
600	1,69	0,27	163,94%	0,0766	0,0084	26,63%	0,00200	0,00067	18,34%
800	1,84	0,07	-39,08%	0,0462	0,0039	-52,88%	0,00110	0,00049	-28,86%
1000	0,05	0,05	-46,98%	0,0053	0,0032	-38,11%	0,00062	0,00045	-23,73%

Tabelle 5.5: Die Antwortzeiten bei Verwendung von $G^B_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^Z_{n_{f,k^*}}$

Sekunden betragen und somit bemerkenswert gering bleiben.

Die sehr unterschiedlichen Veränderungen in den einzelnen Problemgruppen sind darauf zurückzuführen, dass durch die Einführung der Ladebedingung „Backhauls'' die Menge der möglichen Lösungen deutlich verringert wird, so dass bisher „gut" zu lösende Routenplanungsprobleme nun eventuell keine zulässige oder eine für den A*-Algorithmus nur mit mehr Rechenaufwand zu findende Lösung mehr besitzen können.

Die Einführung der Ladebedingung „Backhauls" führt bei Anpassung des zugrunde liegenden Zustandsgraphen zu einer deutlichen Verringerung der Rechenzeit. Eine genaue Anpassung der Schätzfunktion für den A*-Algorithmus an die speziellen Eigenschaften dieser Ladebedingung wie zum Beispiel die Überprüfung der Zulässigkeit der Backhauls-Phase schon während der Linehaul-Phase könnte noch zu deutlich besseren Ergebnissen führen. Diese Untersuchung soll jedoch nicht Gegenstand dieser Arbeit sein.

5.2 LIFO

Dieser Abschnitt behandelt die Ladebedingungen „LIFO" und „LIFO-q". Da die Ladebedingung „LIFO" einen Spezialfall der Bedingung „LIFO-q" für $q = 1$ darstellt, werden beide Ladebedingungen in diesem Abschnitt zusammen betrachtet. Sie werden folgendermaßen definiert:

Definition 5.3 (LIFO)
Eine Route genügt der Ladebedingung „LIFO", wenn für jeden Entladevorgang gilt, dass das auszuladende Transportgut von allen sich zu diesem Zeitpunkt auf dem Fahrzeug befindlichen Gütern als letztes eingeladen wurde.
Ein Routenplan genügt der Ladebedingung „LIFO", wenn alle Fahrzeuge die Ladebedingung erfüllen.

Definition 5.4 (LIFO-q)
Eine Route genügt der Ladebedingung „LIFO-q“, wenn für jeden Entladevorgang gilt, dass das auszuladende Transportgut zu den q zuletzt eingeladenen Gütern gehört, wobei nur die noch auf dem Fahrzeug befindlichen Güter gezählt werden.
Ein Routenplan genügt der Ladebedingung „LIFO-q“, wenn alle Fahrzeuge die Ladebedingung erfüllen.

Im Gegensatz zur Ladebedingung „Backhauls“ ist das Einhalten dieser Ladebedingungen für den gesamten Routenplan möglich.

Die Ladebedingung „LIFO-q“ verlangt, dass nur die q zuletzt eingeladenen Aufträge ausgeladen werden dürfen und bildet damit eine zusätzliche Nebenbedingung für das statische Teilproblem (vgl. Abschnitt 2.3.3), für das bei Eingang eines neuen Auftrags κ mit der Heuristik zur Berechnung eines zulässigen Routenplans eine Lösung berechnet und diese anschließend mit dem Verfahren der Tabu Suche verbessert wird. Dazu werden von beiden Verfahren jeweils Lösungen für mehrere Single Vehicle Routing Probleme mit Hilfe des A*-Algorithmus auf einem Zustandsgraphen berechnet, der nun zusätzlich zur Reihenfolgebedingung auch die Ladebedingung in seiner Struktur verankern soll.

Im Folgenden wird zunächst in Abschnitt 5.2.1 der LIFO-Zustandsgraph definiert und analysiert. Anschließend werden in 5.2.1 die theoretischen Resultate mit den Ergebnissen der Testläufe untermauert. Danach erfolgt in 5.2.2 die Definition des allgemeineren Zustandsgraphen mit LIFO-q-Bedingung. In Abschnitt 5.2.2 werden die Implementierung und die Ergebnisse der Testläufe zur LIFO-q-Bedingung vorgestellt.

5.2.1 Der LIFO-Zustandsgraph

Die Ladebedingung „LIFO“ fordert, dass nur der zuletzt eingeladene Auftrag ausgeladen werden darf, während das Einladen weiterer Aufträge nicht beschränkt wird. Für den LIFO-Zustandsgraphen $G^L = (V^L_{n_f,k^*}, E^L_{n_f,k^*}, c^L_{n_f,k^*})$ zur Darstellung der möglichen Zustände eines einzelnen Fahrzeugs $f \in F$ bedeutet dies, dass zusätzlich zum Statusvektor Φ und der Fahrzeugposition μ die Pickup-Reihenfolge der Aufträge $i \in N_f$ mit Status $\varphi(i) = 1$ zur Charakterisierung eines Zustands benötigt wird. Daher wird der *Reihenfolgevektor* $\rho \in \mathbb{N}_0^{n_f}$ eingeführt, der für jeden Auftrag $i \in \{1, \ldots, n_f\}$ mit Status $\varphi(i) = 1$ angibt, an wievielter Stelle aller Aufträge $i' \in N_f$ mit Status $\varphi(i') = 1$ Auftrag i eingeladen wurde. Falls Auftrag i den Status $\varphi(i) = 0$ oder $\varphi(i) = 2$ besitzt, ist $\rho(i) = 0$. Ein Knoten des LIFO-Zustandsgraphen besteht also aus dem Tripel $(\Phi, \mu, \rho)^T \in \{0;1;2\}^{n_f} \times W \times \mathbb{N}_0^{n_f}$.

Die Knotenmenge $V^L_{n_f,k^*}$ wird wiederum als Vereinigung der Knotenmengen ${}_kV^L_{n_f,k^*}$ der Stufen $k = k^*, \ldots, 2n_f$ des LIFO-Zustandsgraphen G^L definiert. Auf Stufe k^* existiert nur ein Knoten, die Quelle des Graphen. O. B. d. A. wird angenommen, dass die ersten k^* Aufträge $i = 1 \ldots k^*$ im Statusvektor Φ Status $\varphi(1) = \cdots = \varphi(k^*) = 1$ haben und entsprechend der

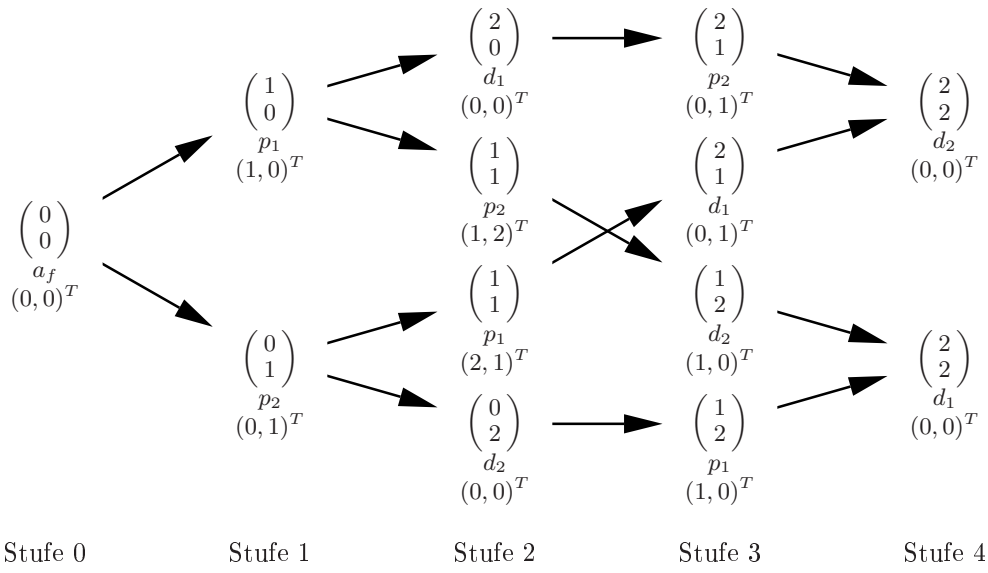

Abbildung 5.3: Der Zustandsgraph $G^L_{2,0}$ mit LIFO-Bedingung für zwei nicht bearbeitete Aufträge.

Reihenfolge des Einladens numeriert sind, d. h. $\rho(i) = i$ für alle Aufträge $i = 1 \dots k^*$:

$$ {}_{k^*}V^L_{n_f,k^*} = \left\{ \left((1,\dots,1,0,\dots,0)^T, p_{k^*}, (1,2,\dots,k^*,0,\dots,0)^T \right) \right\} $$

Wegen des durch die Ladebedingung beschränkten Übergangs von einem Knoten $(\Phi,\mu,\rho)^T$ zum nächsten wird die Knotenmenge ${}_kV^L_{n_f,k^*}$ einer Stufe $k > k^*$ rekursiv in Abhängigkeit der Knotenmenge ${}_{k-1}V^L_{n_f,k^*}$ definiert. Ein Knoten $(\Phi,\mu,\rho)^T$ auf Stufe k ist genau dann zulässig, wenn es einen Vorgängerknoten $(\Phi',\mu',\rho')^T$ auf Stufe $k-1$ und einen Auftrag $i \in \{1,\dots,n_f\}$ gibt, so dass

- der Statusvektor Φ durch Addition des Einheitsvektors e_i aus dem Statusvektor Φ' entsteht,
- für einen Auftrag i mit Status $\varphi'(i) = 1$ auf Stufe $k-1$ die LIFO-Bedingung eingehalten wird: $\rho'(i) = \max_{j=1}^{n_f} \rho'(j)$,
- der Reihenfolge-Vektor ρ entsprechend aktualisiert wird: falls $\varphi'(i) = 1$, muss $\rho(i) = 0$ sein; sonst muss $\rho(i) = \max_{j=1}^{n_f} \rho'(j) + 1$ gelten
- die Fahrzeugposition μ entsprechend $\varphi(i)$ den Pickup- bzw. Delivery-Ort des Auftrags i enthält

Die Knotenmenge ${}_kV^L_{n_f,k^*}$ auf Stufe k mit $k^* < k \leq 2n_f$ kann damit folgendermaßen definiert werden:

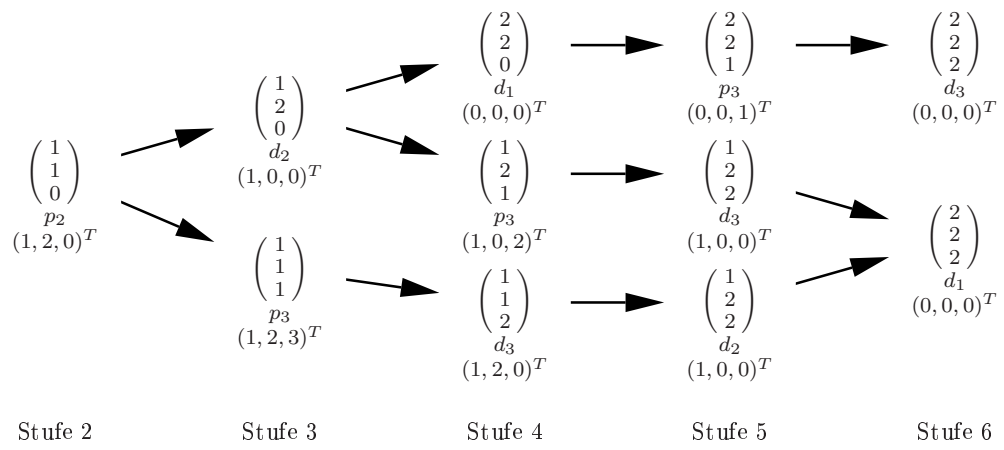

Abbildung 5.4: Der Zustandsgraph $G^L_{3,2}$ mit LIFO-Bedingung für drei Aufträge mit Quelle auf Stufe 2 und Fahrzeugposition p_2.

$$ {}_kV^L_{n_f,k^*} = \left\{ (\Phi,\mu,\rho)^T \middle| \exists (\Phi',\mu',\rho')^T \in {}_{k-1}V^L_{n_f,k^*},\ i \in \{1\dots n_f\} : \Phi = \Phi' + e_i,\ \Phi \in \{0;1;2\}^{n_f}, \right. $$
$$ \left. \mu = \begin{cases} p_i, \rho'(i)=0 \\ d_i, \rho'(i)>0 \end{cases}, \rho(i) = \begin{cases} \max\limits_{j=1}^{n_f} \rho'(j)+1, \rho'(i)=0 \\ 0, \qquad\qquad \rho'(i)>0 \end{cases}, \rho'(i) \in \{0; \max_{j=1}^{n_f} \rho'(j)\} \right\} $$

Die Knotenmenge $V^L_{n_f,k^*}$ des LIFO-Zustandsgraphen G^L ergibt sich daher als

$$ V^L_{n_f,k^*} = \bigcup_{k=k^*}^{2n_f} {}_kV^L_{n_f,k^*}. $$

Die Kantenmenge $E^L_{n_f,k^*}$ lässt sich schreiben als

$$ E^L_{n_f,k^*} = \left\{ \left((\Phi',\mu',\rho')^T, (\Phi,\mu,\rho)^T\right) \in V^L_{n_f,k^*} \times V^L_{n_f,k^*} \middle| \right. $$
$$ \left. \exists |\ i \in \{1,\dots,n_f\} :\ \Phi = \Phi' + e_i,\ \mu \in \{p_i, d_i\},\ \rho(i) \in \left\{0; \max_{j=1}^{n_f}\{\rho(j)\}\right\} \right\} $$

Die Kostenfunktion $c^L_{n_f,k^*} : E^L_{n_f,k^*} \to \mathbb{R}$ entspricht der Kostenfunktion $c^Z_{n_f,k^*}$ des Zustandsgraphen $G^Z_{n_f,k^*}$ ohne Ladebedingungen und beinhaltet die Fahrzeit $c_{\mu',\mu}$ von Ort μ' zu Ort μ sowie die Wartezeit ω_μ bis zum Beginn des Zeitfensters $[\underline{t}_\mu, \overline{t}_\mu]$ an Ort μ. Damit ist der Graph G^L eindeutig definiert.

Die Abbildungen 5.3 und 5.4 zeigen beispielhaft zwei LIFO-Zustandsgraphen G^L_{n,k^*}. In Abbildung 5.3 wird $G^L_{2,0}$ für zwei Aufträge und Quelle auf Stufe Null gezeigt, während in Abbildung 5.4 der LIFO-Zustandsgraph $G^L_{3,2}$ für drei Aufträge und Quelle auf Stufe zwei dargestellt wird. Im Folgenden wird der LIFO-Zustandsgraph analysiert, um anhand der Anzahl Knoten $|V^L_{n_f,k^*}|$ und der Anzahl Kanten $|E^L_{n_f,k^*}|$ im LIFO-Zustandsgraphen die maximalen Rechenzeiten des A*-Algorithmus einschätzen zu können.

Satz 5.2

Es bezeichne $|V^L_{n_f,k^*}|$ die Anzahl der Knoten und $|E^L_{n_f,k^*}|$ die Anzahl der Kanten im LIFO-Zustandsgraphen für n_f Aufträge, dessen Quelle auf Stufe k^* liegt. Dann gilt:

$$\left|V^L_{n_f,k^*}\right| =$$

$$\sum_{k=k^*}^{2n_f} \sum_{\substack{l_2=\max\{0,\\ k-2n_f+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j_2=\max\{0,\\ k-n_f-l_2\}}}^{\substack{\min\{n_f-k^*,\\ \frac{1}{2}(k-k^*-l_2)\}}} \frac{(n_f-k^*)!\,((1-\mathrm{sgn}(k-k^*-l_2-2j_2))\,\mathrm{sgn}(l_2)+j_2+\mathrm{sgn}(k-k^*-l_2-2j_2))}{(n_f-k+l_2-j_2)!j_2!}$$

$$\left|E^L_{n_f,k^*}\right| = n_f - k^* + \mathrm{sgn}(k^*) +$$

$$\sum_{k=k^*+1}^{2n_f-1} \sum_{\substack{l_2=\max\{0,\\ k-2n_f+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j_2=\max\{0,\\ k-n_f-l_2\}}}^{\substack{\min\{n_f-k^*,\\ \frac{1}{2}(k-k^*-l_2)\}}} \Big(\frac{(n_f-k^*)!\,((1-\mathrm{sgn}(k-k^*-l_2-2j_2))\,\mathrm{sgn}(l_2)+j_2+\mathrm{sgn}(k-k^*-l_2-2j_2))}{(n_f-k+l_2-j_2)!\;j_2!} \cdot$$

$$\cdot\,(j_0+\mathrm{sgn}(k-2j_2-2l_2))\Big)$$

Beweis: Der Beweis erfolgt konstruktiv mit Hilfe kombinatorischer Überlegungen. In einem ersten Schritt wird die Anzahl der Knoten $|V^L_{n_f,k^*}|$ bewiesen, anschließend die Anzahl der Kanten $|E^L_{n_f,k^*}|$.

Die Anzahl der Knoten $|V^L_{n_f,k^*}|$ wird zunächst für jede Stufe $k = k^*, \ldots, 2n_f$ einzeln berechnet. Dazu wird der Statusvektor Φ in einen „oberen“ und einen „unteren“ Bereich unterteilt, wobei die Aufträge $i = 1, \ldots, k^*$, die in der Quelle des Graphen den Status $\varphi(i) = 1$ haben, den oberen Bereich bilden, während die Aufträge $i = k^*+1, \ldots, n_f$ im unteren Bereich liegen. Für die Aufträge $i = 1, \ldots, k^*$ des oberen Bereichs gilt, dass nur die Status $\varphi(i) = 1$ und $\varphi(i) = 2$ zulässig sind und die Reihenfolge der möglichen Statusänderung von $\varphi(i) = 1$ auf $\varphi(i) = 2$ wegen der einzuhaltenden LIFO-Bedingung bereits gegeben ist. Die Anzahl der Einträge i des oberen Bereichs mit Status $\varphi(i) = 1$ wird im Folgenden mit l_1 bezeichnet, l_2 entspricht der Anzahl Einträge i des oberen Bereichs mit Status $\varphi(i) = 2$. Im unteren Bereich hingegen sind alle Status $\varphi(i) \in \{0; 1; 2\}$ zulässig, die Reihenfolge ist bisher nicht festgelegt. Hier gilt: die Anzahl der Einträge $i \in \{k^*+1, \ldots, n_f\}$ mit Status $\varphi(i) = 0$ wird mit j_0 bezeichnet, die Anzahl Einträge i mit Status $\varphi(i) = 1$ mit j_1 und die Anzahl Einträge i mit $\varphi(i) = 2$ mit j_2.

Für die Zulässigkeit des Statusvektors Φ auf Stufe k muss gelten, dass

$$l_1 + l_2 = k^* \tag{5.1}$$

$$l_1 + 2l_2 + j_1 + 2j_2 = k \tag{5.2}$$

$$l_1 + l_2 + j_0 + j_1 + j_2 = n_f \tag{5.3}$$

Die Formeln 5.1 und 5.3 garantieren, dass im oberen Bereich und im gesamten Statusvektor die Anzahl der Einträge der Größe des Bereichs bzw. des Vektors entspricht. Formel 5.2 gewährleistet, dass die Summe aller Einträge des Statusvektors auf Stufe k gerade k beträgt (vgl. den Beweis von Satz 3.5).

Zur Berechnung der Anzahl aller möglichen Statusvektoren auf einer Stufe k werden kombinatorische Überlegungen herangezogen. Zunächst wird nur der obere Bereich des Statusvektors Φ betrachtet, bevor im nächsten Schritt auch der untere Bereich hinzugezogen wird. Für den oberen Bereich gilt:

- Von der Quelle des Graphen auf Stufe k^* bis zur Stufe k werden $k - k^*$ Einträge des Statusvektors Φ um Eins erhöht. Für den oberen Bereich gilt daher auf Stufe k, dass maximal $k - k^*$ Einträge den Status $\varphi(i) = 2$ haben können, falls so viele Einträge im oberen Bereich vorhanden sind. Sonst wird die Anzahl der Einträge mit Status $\varphi(i) = 2$ natürlicherweise durch die gesamte Anzahl Einträge im oberen Bereich beschränkt:

$$l_2 \leq \min\{k - k^*; k^*\}$$

- Als untere Grenze für die Anzahl der Einträge $\varphi(i) = 2$ muss betrachtet werden, ob alle $k - k^*$ Statusänderungen im unteren Bereich vollzogen werden können. Im unteren Bereich gibt es $n_f - k^*$ Einträge, deren Status zweimal erhöht werden kann. Damit muss spätestens nach $2(n_f - k^*)$ Schritten und damit auf Stufe $k^* + 2(n_f - k^*)$ ein Auftrag $i \in \{1, \ldots, k^*\}$ im oberen Bereich den Status $\varphi(i) = 2$ besitzen. Für die Anzahl der Einträge mit $\varphi(i) = 2$ gilt also:

$$l_2 \geq \max\{0; k - 2n_f + k^*\}$$

- Für die Anzahl der möglichen Kombinationen von Einträgen $\varphi(i) = 1$ und $\varphi(i) = 2$ im oberen Bereich eines Statusvektors Φ bei gegebenen Anzahlen l_2 und $l_1 = k^* - l_2$ gilt: da die Pickup-Reihenfolge der Aufträge gegeben ist, ist auch die Delivery-Reihenfolge festgelegt. Es ist daher für jede Anzahl l_2 der Statusvektor im oberen Bereich eindeutig bestimmt.

Für den oberen Bereich erhält man somit

$$\sum_{l_2=\max\{0;k-2n_f+k^*\}}^{\min\{k-k^*;k^*\}} 1$$

verschiedene Statusvektoren.

Im unteren Bereich gibt es, in Abhängigkeit von l_2 aus dem oberen Bereich, mehrere Möglichkeiten zur Gestaltung des Statusvektors. Insgesamt sind $n_f - k^*$ Einträge vorhanden, bis Stufe k müssen $k - k^* - l_2$ Statuserhöhungen im unteren Bereich vollzogen worden sein. Hier ist die LIFO-Ladebedingung nur für die Kanten auschlaggebend, im LIFO-Zustandsgraphen G^L_{n,k^*} können im unteren Bereich des Statusvektors alle möglichen Kombinationen von Zuständen $\varphi(i) \in \{0; 1; 2\}$ erzeugt werden. Analog zum Beweis von Satz 3.5 können im unteren Bereich daher insgesamt

$$\sum_{\substack{j_2=\max\{0,\\ k-n_f-l_2\}}}^{\substack{\min\{n_f-k^*,\\ \frac{1}{2}(k-k^*-l_2)\}}} \frac{(n_f-k^*)!}{j_0!\; j_1!\; j_2!}$$

verschiedene Kombinationen der Status $\varphi(i) \in \{0;1;2\}$ in Abhängigkeit von der Anzahl l_2 der Einträge $\varphi(i) = 2$ im oberen Bereich entstehen.
Damit erhält man insgesamt

$$\sum_{\substack{l_2=\max\{0;\\ k-2n_f+k^*\}}}^{\substack{\min\{k^*\\ k-k^*\}}} \sum_{\substack{j_2=\max\{0,\\ k-n_f-l_2\}}}^{\substack{\min\{n_f-k^*,\\ \frac{1}{2}(k-k^*-l_2)\}}} \frac{(n_f-k^*)!}{j_0!\; j_1!\; j_2!}$$

verschiedene Statusvektoren Φ auf Stufe k des LIFO-Zustandsgraphen G^L_{n,k^*}.

Um die Anzahl der verschiedenen Zustände zu erhalten, fehlt nun noch die Anzahl der Fahrzeugpositionen μ und die Anzahl der Reihenfolgevektoren ρ, die für einen Statusvektor Φ der Stufe k mit l_2 Status $\varphi(i) = 2$ im oberen und j_2 Status $\varphi(i) = 2$ im unteren Bereich möglich sind. Für die möglichen Fahrzeugpositionen μ gelten folgende Überlegungen:

- Das Fahrzeug kann nur dann in einem Delivery-Ort d_i eines Auftrags $i \in \{1,\dots,k^*\}$ aus dem oberen Bereich des Statusvektors Φ stehen, wenn es laut LIFO-Bedingung genau diesen Auftrag i aus dem oberen Bereichs abladen darf. Das ist nur dann möglich, wenn der Auftrag i als letzter eingeladen wurde, also $j_1 = 0$ gilt. Dann kann im Fall $l_2 > 0$ genau ein Auftrag des oberen Bereichs ausgeladen werden, das Fahrzeug hat also eine eindeutige Position.
  Zusätzlich sind für einen Statusvektor Φ mit $j_1 = 0$ auch die j_2 Delivery-Orte der Auftrage $i \in \{k^*+1,\dots,n_f\}$ mit $\varphi(i) = 2$ als Fahrzeugpositionen möglich.
- Falls $j_1 > 0$, so kann sich das Fahrzeug nur in einem der j_2 Delivery-Orte eines Auftrags $i \in \{k^*+1,\dots,n_f\}$ des unteren Bereichs mit Status $\varphi(i) = 2$ oder in einem der j_1 Pickup-Orte eines Auftrags $i \in \{k^*+1,\dots,n_f\}$ des unteren Bereichs mit $\varphi(i) = 1$ befinden.

Man erhält somit zu einem gegebenen Statusvektor Φ $j_1 + j_2$ mögliche Fahrzeugpositionen μ sowie für den Fall $j_1 = 0$ und $l_2 > 0$ eine zusätzliche Fahrzeugposition, also insgesamt

$$(1 - \operatorname{sgn}(j_1))\operatorname{sgn}(l_2) + j_1 + j_2$$

verschiedene Fahrzeugpositionen pro Statusvektor Φ einer Stufe k.

Die Anzahl der möglichen Reihenfolgevektoren hängt von der Anzahl j_1 der Einträge $\varphi(i) = 1$ im unteren Bereich des Statusvektors Φ und der Fahrzeugposition μ ab:

- Falls die Fahrzeugposition $\mu \in D$ ein Delivery-Ort ist, gibt es $j_1!$ mögliche Permutationen der Ziffern $1,\dots,j_1$ für die j_1 Einträge $\rho(i) > 0$ des Reihenfolgevektors ρ. Die Anzahl Kombinationen aus Fahrzeugposition μ und Reihenfolgevektor ρ eines Statusvektors Φ der Stufe k beträgt daher

$$((1 - \operatorname{sgn}(j_1))\operatorname{sgn}(l_2) + j_2)\, j_1!$$

- Ist die Fahrzeugposition ein Pickup-Ort $\mu = p_i \in P$, so ist bereits festgelegt, dass Auftrag i zuletzt eingeladen wurde und daher der entsprechende Eintrag des Reihenfolgevektors $\rho(i) = j_1 > 0$ beträgt. Es gibt daher nur $(j_1 - 1)!$ mögliche Permutationen für den Reihenfolgevektor ρ. Die Anzahl Kombinationen aus Fahrzeugposition μ und Reihenfolgevektor ρ eines Statusvektors Φ der Stufe k beträgt in diesem Fall $(j_1 > 0)$:

$$j_1(j_1 - 1)! = j_1!$$

Insgesamt erhält man somit

$$((1 - \operatorname{sgn}(j_1))\operatorname{sgn}(l_2) + j_2)\, j_1! + \operatorname{sgn}(j_1) j_1! \quad = \quad ((1 - \operatorname{sgn}(j_1))\operatorname{sgn}(l_2) + j_2 + \operatorname{sgn}(j_1))\, j_1!$$

mögliche Kombinationen aus Fahrzeugposition μ und Reihenfolgevektor ρ für einen Statusvektors Φ der Stufe k, so dass sich

$$\sum_{\substack{l_2=\max\{0,\\ k-2n_f+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j_2=\max\{0,\\ k-n_f-l_2\}}}^{\substack{\min\{n_f-k^*,\\ \frac{1}{2}(k-k^*-l_2)\}}} \frac{(n_f-k^*)!\,((1-\operatorname{sgn}(j_1))\operatorname{sgn}(l_2) + j_2 + \operatorname{sgn}(j_1))\, j_1!}{j_0!\; j_1!\; j_2!}$$

mögliche Kombinationen aus Statusvektoren Φ, Fahrzeugpositionen μ und Reihenfolgevektoren ρ auf Stufe k ergeben.

Aus dem Gleichungssystem 5.1 - 5.3 ergibt sich:

$$l_1 = k^* - l_2 \tag{5.4}$$
$$j_1 = k - k^* - l_2 - 2j_2 \tag{5.5}$$
$$j_0 = n_f - k + l_2 - j_2 \tag{5.6}$$

Ersetzt man nun j_0 und j_1 entsprechend der Gleichungen 5.5 und 5.6 und kürzt $j_1!$, so erhält man

$$\left|{}_kV^L_{n,k^*}\right| =$$
$$\sum_{\substack{l_2=\max\{0,\\ k-2n_f+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j_2=\max\{0,\\ k-n_f-l_2\}}}^{\substack{\min\{n_f-k^*,\\ \frac{1}{2}(k-k^*-l_2)\}}} \frac{(n_f-k^*)!\,((1-\operatorname{sgn}(k-k^*-l_2-2j_2))\operatorname{sgn}(l_2) + j_2 + \operatorname{sgn}(k-k^*-l_2-2j_2))}{(n_f-k+l_2-j_2)!\; j_2!}$$

Knoten auf Stufe k. Summieren über alle Stufen $k = k^*, \ldots, 2n_f$ bei gleichzeitigem Ersetzen von l_2 durch l und j_2 durch j liefert die Anzahl Knoten $|V^L_{n,k^*}|$ der Behauptung.

Für die Berechnung der Anzahl $|E^L_{n,k^*}|$ der Kanten im Graphen $|G^L_{n,k^*}|$ wird die Anzahl der ausgehenden Kanten jeder Stufe k mit $k^* \leq k \leq 2n_f - 1$ betrachtet. Auf Stufe k^* gibt es genau einen Knoten, dessen Statusvektor aus k^* Einträgen $\varphi(i) = 1$ und $n_f - k^*$ Einträgen $\varphi(i) = 0$ besteht. Von den Einträgen $\varphi(i) = 1$ kann wegen der LIFO-Bedingung genau ein

n_f	$\lvert V^L_{n_f,0}\rvert$	Δ	$\lvert E^L_{n_f,0}\rvert$	Δ
1	3	0,00%	2	0,00%
2	13	0,00%	14	-12,50%
3	61	10,91%	87	-14,71%
4	305	40,55%	516	-5,15%
5	1.681	107,27%	3.125	21,60%
6	10.465	258,76%	20.358	82,03%
7	74.145	626,41%	146.951	219,92%
8	595.201	1.600,91%	1.187.336	556,71%
9	5.361.409	4.439,76%	10.715.913	1.455,48%
10	53.624.321	13.521,95%	107.233.290	4.090,76%

Tabelle 5.6: Die Anzahl der Knoten und Kanten des LIFO-Graphen $G^L_{n_f,0}$, für $n = 1,\ldots,10$ beginnend auf Stufe 0, sowie die prozentuale Veränderung im Vergleich zum Zustandsgraphen $G^Z_{n_f,0}$.

Eintrag auf $\varphi(i) = 2$ erhöht werden, also entsteht für $k^* > 0$ genau eine Kante, für $k^* = 0$ keine Kante zur Stufe $k^* + 1$. Die Status der Aufträge $i \in \{k^* + 1,\ldots,n_f\}$ können beliebig von $\varphi(i) = 0$ auf $\varphi(i) = 1$ erhöht werden, so dass hier $n_f - k^*$ Kanten zu Knoten der nächsten Stufe $k^* + 1$ entstehen. Insgesamt erhält man

$$n_f - k^* + \operatorname{sgn}(k^*)$$

Kanten von Stufe k^* zu Stufe $k^* + 1$.

Für die Stufen $k > k^*$ gilt, dass von jedem Knoten $(\Phi,\mu,\rho)^T$ der Stufe k für maximal einen Auftrag mit Status $\varphi(i) = 1$ sowie für jeden Auftrag mit Status $\varphi(i) = 0$ eine Kante zur Stufe $k+1$ führt. Die Anzahl $\lvert {}_kV^L_{n,k^*}\rvert$ der Knoten auf Stufe k ist bekannt, die Anzahl der ausgehenden Kanten eines Knotens $(\Phi,\mu,\rho)^T \in {}_kV^L_{n,k^*}$ beträgt $j_0 + \operatorname{sgn}(j_1 + l_1)$. Von Stufe $k > k^*$ zu Stufe $k+1$ verlaufen somit

$$\lvert {}_kE^L_{n,k^*}\rvert = \sum_{\substack{l_2=\max\{0,\\ k-2n_f+k^*\}}}^{\substack{\min\{k^*,\\ k-k^*\}}} \sum_{\substack{j_2=\max\{0,\\ k-n_f-l_2\}}}^{\substack{\min\{n_f-k^*,\\ \frac{1}{2}(k-k^*-l_2)\}}} \frac{(n_f-k^*)!\,((1-\operatorname{sgn}(j_1))\operatorname{sgn}(l_2)+j_2+\operatorname{sgn}(j_1))\,(j_0+\operatorname{sgn}(j_1+l_1))}{j_0!\;j_2!}$$

Kanten. Durch Summieren der Stufen $k = k^* \ldots 2n_f - 1$ und Ersetzen von l_1, j_0 und j_1 entsprechend 5.4 - 5.6 sowie l_2 durch l und j_2 durch j erhält man die Behauptung.

□

Tabelle 5.6 zeigt die Anzahl $\lvert V^L_{n_f,k^*}\rvert$ der Knoten und die Anzahl $\lvert E^L_{n_f,k^*}\rvert$ der Kanten in LIFO-Graphen $G^L_{n_f,0}$ für $n = 1,\ldots,10$ Aufträge mit Quelle auf Stufe 0 und gibt die prozentuale Steigerung im Vergleich zum Zustandsgraphen $G^Z_{n_f,0}$, ebenfalls beginnend auf Stufe 0, an.

Problemgruppe		Aufträge		gelöste Instanzen	
Name	Aufträge	absolut	Δ	absolut	Δ
100	51,86	43,38	4,86%	43	7,50%
200	103,08	77,92	5,86%	41	7,89%
400	205,66	141,44	7,30%	36	12,50%
600	308,97	214,57	10,00%	39	11,43%
800	411,75	285,12	8,66%	38	18,75%
1000	514,98	353,48	11,13%	36	16,13%

Tabelle 5.7: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung von $G^L_{n_f,k^*}$ im Vergleich zur Verwendung von $G^Z_{n_f,k^*}$

Die Anzahl $|V^L_{n_f,k^*}|$ der Knoten ist in den beiden kleinsten Graphen gleich und steigt danach wegen der Hinzunahme der Pickup-Reihenfolge als Charakteristik für einen Zustand deutlich an. Die Anzahl $|E^L_{n_f,k^*}|$ der Kanten geht in den „kleineren“ Graphen für bis zu 5 Aufträge zunächst etwas zurück, da einige Übergänge nicht mehr zulässig sind. Mit zunehmender Größe des Graphen wird dieser Effekt aber durch die stark ansteigende Anzahl Knoten kompensiert, der Unterschied in der Anzahl Kanten wächst lediglich langsamer als der Unterschied der Anzahl Knoten.

Testläufe und Auswertung

Für diese Testläufe wird die Testumgebung weiterhin ereignisdiskret eingestellt; zur Beschränkung der Rechenzeit wird die Grenze von einer Million maximal insgesamt zu erzeugenden Knoten verwendet. Den Vergleichen liegen weiterhin die Ergebnisse der Testläufe zur Verwendung der Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität und Sortierung der Fahrzeuge nach dem Sortierkriterium S^A_f aus Abschnitt 4.1.2.1 zu Grunde.

In Tabelle 5.7 werden die Anzahl der akzeptierten Aufträge und die Anzahl der gelösten Instanzen bei Einhaltung der LIFO-Ladebedingung dargestellt. Die Anzahl der bearbeiteten Aufträge wächst in allen Problemgruppen um 4,86% bis 11,13%, die Anzahl der vollständig gelösten Instanzen nimmt ebenfalls in allen Problemgruppen um 7,50% bis 18,75% zu. Diese Steigerungen sind bei Betrachtung der Tabelle 5.6 zunächst überraschend, da die Anzahl der Knoten im Zustandsgraphen $G^L_{n_f,k^*}$ mit LIFO-Ladebedingung im Vergleich zum Zustandsgraphen $G^Z_{n_f,k^*}$ deutlich ansteigt.

Die Begründung für die Zunahme der bearbeiteten Aufträge und gelösten Instanzen kann in den Tabellen 5.8 und 5.9 mit den durchschnittlichen Anzahlen der erzeugten, der als unzulässig erkannten, der expandierten und der Duplikat-Knoten abgelesen werden. Besonders deutlich wird die Veränderung im Algorithmus bei den Ergebnissen der Problemgruppe „1000“, deren Ergebnisse im Vergleich deutlich schlechter abschneiden als die Problemgruppen „100“

Problem-gruppe	♯ Knoten absolut			
	erzeugt	unzulässig	expandiert	Duplikate
100	203.909,02	78.223,30	43.600,23	77.939,88
200	308.816,85	117.974,05	67.842,55	111.943,15
400	369.718,75	170.167,81	83.514,76	101.666,00
600	405.505,80	192.088,83	93.426,48	101.986,30
800	445.153,80	212.019,32	102.395,61	108.177,54
1000	787.684,71	392.001,07	151.906,09	178.270,62

Tabelle 5.8: Anzahl der Knoten gesamt bei Verwendung von $G^L_{n_{f,k^*}}$

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	107.795,71	-50,54%	39.577,16	-63,21%	23.731,43	-11,80%	41.252,75	-47,13%
200	169.965,18	-53,03%	66.873,28	-66,99%	38.479,92	-17,07%	57.857,82	-42,99%
400	196.214,56	-56,51%	83.862,00	-68,21%	47.483,90	-20,43%	55.091,44	-52,00%
600	202.662,85	-52,38%	95.792,78	-60,61%	50.571,02	-17,73%	43.230,03	-58,70%
800	250.841,46	-45,91%	124.784,71	-56,35%	60.811,61	-12,25%	47.337,47	-45,53%
1000	612.108,64	9,03%	299.987,03	-12,65%	116.235,90	33,20%	135.223,33	24,86%

Tabelle 5.9: Anzahl der Knoten bei Verwendung von $G^L_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^Z_{n_{f,k^*}}$

bis „800“: die Anzahl der erzeugten Knoten nimmt in Problemgruppe 1000 um 9,03% zu, während sie in den anderen Problemgruppen um 45,91% bis 56,51% fallen. Der erhebliche Unterschied liegt in der unterschiedlichen Struktur der zulässigen Lösungen begründet, die in der Testinstanz „lrc1103“ dazu führte, dass zur Berechnung der für den Vergleich benötigten Anzahl von 89 bearbeiteten Aufträgen 17 Millionen Knoten erzeugt werden mussten, so dass dieses eine Testergebnis den Mittelwert erheblich beeinflusste. Trotzdem kann anhand der Ergebnisse der Problemgruppe „1000“ die Veränderung im Verhalten des Algorithmus gut erkannt werden: die Anzahl der expandierten Knoten nimmt im Vergleich überproportional gegenüber der Anzahl der erzeugten Knoten um 33,20% zu. Diese Veränderung liegt in der anderen Struktur des LIFO-Graphen $G^L_{n_f,k^*}$ begründet: das Verhältnis von Kanten zu Knoten ist im LIFO-Graphen $G^L_{n_f,k^*}$ deutlich kleiner als das Verhältnis von Kanten zu Knoten im Zustandsgraphen $G^Z_{n_f,k^*}$ mit wenigen Aufträgen. In den Testläufen werden daher bei jeder Expansion im Schnitt weniger Nachfolge-Knoten erzeugt, so dass die Anzahl der expandierten Knoten im Verhältnis zu den insgesamt erzeugten Knoten ansteigt. Diese Veränderung ist auch in den Problemgruppen „100“ bis „800“ erkennbar.

Die in Tabelle 5.10 dargestellten Antwortzeiten zeigen in allen Problemgruppen deutliche Verbesserungen. Die durchschnittlichen Antwortzeiten sinken im Vergleich zu den Testläufen aus Abschnitt 4.1.2.1 um 20,08% bis 35,50%, bei den durchschnittlich maximalen Antwort-

Problem-gruppe	global max. Antwortzeit sec	Vgl	Δ	Ø max. Antwortzeit sec	Vgl	Δ	Ø Antwortzeit sec	Vgl	Δ
100	0,226	0,010	-65,92%	0,0084	0,0009	-37,43%	0,00072	0,00017	-31,79%
200	12,730	0,061	-63,00%	0,2278	0,0046	-51,64%	0,00516	0,00058	-35,50%
400	0,473	0,104	-78,22%	0,0298	0,0041	-67,06%	0,00205	0,00054	-23,72%
600	0,150	0,028	-72,23%	0,0078	0,0029	-56,90%	0,00130	0,00045	-20,08%
800	0,593	0,053	-53,00%	0,0172	0,0034	-59,74%	0,00109	0,00047	-31,96%
1000	0,187	0,023	-76,52%	0,0077	0,0029	-45,30%	0,00069	0,00044	-25,34%

Tabelle 5.10: Die Antwortzeiten bei Verwendung von $G^L_{n_f,k^*}$ im Vergleich zur Verwendung von $G^Z_{n_f,k^*}$

zeiten ist eine Reduktion um 37,43% bis 67,06% zu verzeichnen und die global maximale Antwortzeit wird um 53,00% bis 78,22% verringert.

Die Testläufe zeigen, dass der A*-Algorithmus mit der eingeführten Schätzfunktion „Maximum aus Maximalauftragszeit und Minimalzeitensumme mit Servicezeiten und Zeitfensterüberprüfung“ trotz höherer Anzahl Knoten und Kanten in den LIFO-Zustandsgraphen $G^L_{n_f,k^*}$ für „größere“ Anzahl Aufträge n die optimale Route sogar schneller findet als im Zustandsgraphen $G^Z_{n_f,k^*}$ ohne Ladebedingungen.

5.2.2 Der LIFO-q-Zustandsgraph

Die Ladebedingung „LIFO-q“ besagt, dass von den Aufträgen $i \in \{1,\dots,n_f\}$ im Fahrzeug $f \in F$ nur die q zuletzt eingeladenen ausgeladen werden dürfen. Das Einladen ist nicht beschränkt.

Für den LIFO-q-Zustandsgraphen $G^{Lq} = (V^{Lq}_{n_f,k^*}, E^{Lq}_{n_f,k^*}, c^{Lq}_{n_f,k^*})$ gilt daher wiederum, dass zusätzlich zum Statusvektor Φ und der Fahrzeugposition μ der Reihenfolgevektor ρ zur Charakterisierung eines Zustands nötig ist. Der Vektor $\rho \in \mathbb{N}_0^{n_f}$ gibt für jeden Auftrag $i = 1 \dots n_f$ mit Status $\varphi(i) = 1$ an, an wievielter Stelle aller Aufträge mit Status 1 Auftrag i eingeladen wurde. Falls Auftrag i den Status $\varphi(i) = 0$ oder $\varphi(i) = 2$ besitzt, ist $\rho(i) = 0$. Ein Knoten des LIFO-q-Zustandsgraphen ist also ein Tripel $(\Phi, \mu, \rho)^T \in \{0;1;2\}^{n_f} \times W \times \mathbb{N}_0^{n_f}$.

Die Knotenmenge $V^{Lq}_{n_f,k^*}$ wird analog zur Knotenmenge des LIFO-Zustandsgraphen als Vereinigung der Knotenmengen ${}_kV^{Lq}_{n_f,k^*}$ der Stufen $k = k^*,\dots,2n_f$ des LIFO-q-Zustandsgraphen G^{Lq} definiert. Auf Stufe k^* gibt es wiederum genau einen Knoten. Ebenso wird wieder angenommen, dass die ersten k^* Aufträge im Statusvektor Φ Status 1 haben und entsprechend der Reihenfolge des Einladens numeriert sind, also $\varphi(i) = 1$ und $\rho(i) = i$ für alle Aufträge $i = 1,\dots,k^*$ gilt:

$$ {}_{k^*}V^{Lq}_{n_f,k^*} = \left\{ \left((1,\dots,1,0,\dots,0)^T, p_{k^*}, (1,2,\dots,k^*,0,\dots,0)^T \right) \right\} $$

Die Knotenmenge ${}_kV^{Lq}_{n_f,k^*}$ einer Stufe $k > k^*$ wird wieder rekursiv in Abhängigkeit der Knotenmenge ${}_{k-1}V^{Lq}_{n_f,k^*}$ definiert. Für die Zulässigkeit eines Knotens $(\Phi,\mu,\rho)^T$ auf Stufe k gelten bis auf das Ausladen dieselben Bedingungen wie im LIFO-Zustandsgraphen. Für das Ausladen gilt: Auftrag i darf genau dann ausgeladen werden, wenn er in der Pickup-Reihenfolge zu den q zuletzt eingeladenen Aufträgen gehört, also $\max\limits_{i=1\ldots n_f}\rho'(i)-q<\rho'(i)\leq\max\limits_{i=1\ldots n_f}\rho'(i)$ und auf Stufe $k-1$ den Status $\varphi'(i)=1$ hatte.

Allerdings muss im LIFO-q-Zustandsgraphen zusätzlich beachtet werden, dass die Ladebedingung nur so lange restriktiv ist, wie entsprechend viele Aufträge im Fahrzeug geladen oder noch abzuholen sind. Haben $j_0 < q$ Aufträge einen Status $\varphi(i) = 0$, dann dürfen die $q - j_0$ zuletzt eingeladenen Aufträge i auf jeden Fall in allen Folgeknoten ausgeladen werden, so dass die Ladebedingung für diese Aufträge nicht weiter beachtet werden muss. Für den LIFO-q-Graphen $G^{Lq}_{n_f,k^*}$ bedeutet diese Eigenschaft, dass Knoten $(\Phi,\mu,\rho)^T$ und $(\Phi,\mu,\rho')^T$ mit $j_0=|\{i\in N_f|\varphi(i)=0\}|\leq q,\rho(i)=\rho'(i)\ \forall i:\rho(i)<q-j_0$ als identisch betrachtet werden können. In der Knotenmenge $V^{Lq}_{n_f,k^*}$ des Zustandsgraphen $G^{Lq}_{n_f,k^*}$ wird daher im Fall $j_0 < q$ für jeden neu eingeladenen Auftrag i der Eintrag $\rho(i) = q - j_0$ gesetzt. Diese Festlegung der Einträge $\rho(i)$ des Reihenfolgevektors sorgt dafür, dass nicht mehrere Zustände $(\Phi,\mu,\rho)^T$ und $(\Phi,\mu,\rho')^T$ mit gleichen Folgezuständen existieren.

Damit kann die Knotenmenge auf Stufe k mit $k^* < k \leq 2n_f$ als Vereinigung von Knotenmengen mit der Eigenschaft $|\{i\in N_f|\varphi(i)<2\}|\leq q$ bzw. $|\{i\in N_f|\varphi(i)<2\}|>q$ definiert werden:

$$
{}_kV^{Lq}_{n_f,k^*}=\left\{(\Phi,\mu,\rho)^T\middle|\,|\{i\in N_f|\varphi(i)<2\}|>q\ \wedge\ \exists(\Phi',\mu',\rho')^T\in{}_{k-1}V^{L}_{n_f,k^*},i\in\{1\ldots n_f\}:\right.
$$

$$
\Phi=\Phi'+e_i,\Phi\in\{0;1;2\}^{n_f},\rho=\begin{cases}\rho'+(\max\limits_{j=1}^{n_f}\rho'(j))e_i, & \rho'(i)=0\wedge|\{j\in N_f|\varphi(j)=0\}|<q\\ \rho'+(1+\max\limits_{j=1}^{n_f}\rho'(j))e_i, & \rho'(i)=0\wedge|\{j\in N_f|\varphi(j)=0\}|\geq q\\ \rho'-\rho'(i)e_i-\sum\limits_{\substack{\{j\in N_f|\\ \rho(j)>\rho'(i)\}}}e_j, & \rho'(i)>0\end{cases},
$$

$$
\left.\mu=\begin{cases}p_i,\rho'(i)=0\\ d_i,\rho'(i)>0\end{cases},\ \rho'(i)\in\{0\}\cup\{\max\limits_{i=1}^{n_f}\rho'(i)-q+|\{j\in N_f|\rho'(j)=\max\limits_{i=1}^{n_f}\rho'(i)\}|-1,\ldots,\max\limits_{l=1}^{n_f}\rho'(l)\}\right\}
$$

Die gesamte Knotenmenge $V^{Lq}_{n_f,k^*}$ besteht aus der Vereinigung der Knotenmengen der Stufen $k=k^*\ldots 2n_f$:

$$
V^{Lq}_{n_f,k^*}=\bigcup_{k=k^*}^{2n_f}{}_kV^{Lq}_{n_f,k^*}.
$$

Die Kantenmenge $E^{Lq}_{n_f,k^*}$ lässt sich schreiben als

$$
E^{Lq}_{n_f,k^*}=\left\{\left((\Phi',\mu',\rho')^T,(\Phi,\mu,\rho)^T\right)\in V^{Lq}_{n_f,k^*}\times V^{Lq}_{n_f,k^*}\middle|\right.
$$

$$
\left.\exists\ i\in\{1\ldots n_f\}:\Phi=\Phi'+e_i,\ \mu\in\{p_i,d_i\},\ \rho(i)\in\left\{0;\max_{j=1}^{n_f}\{\rho(j)\}\right\}\right\}
$$

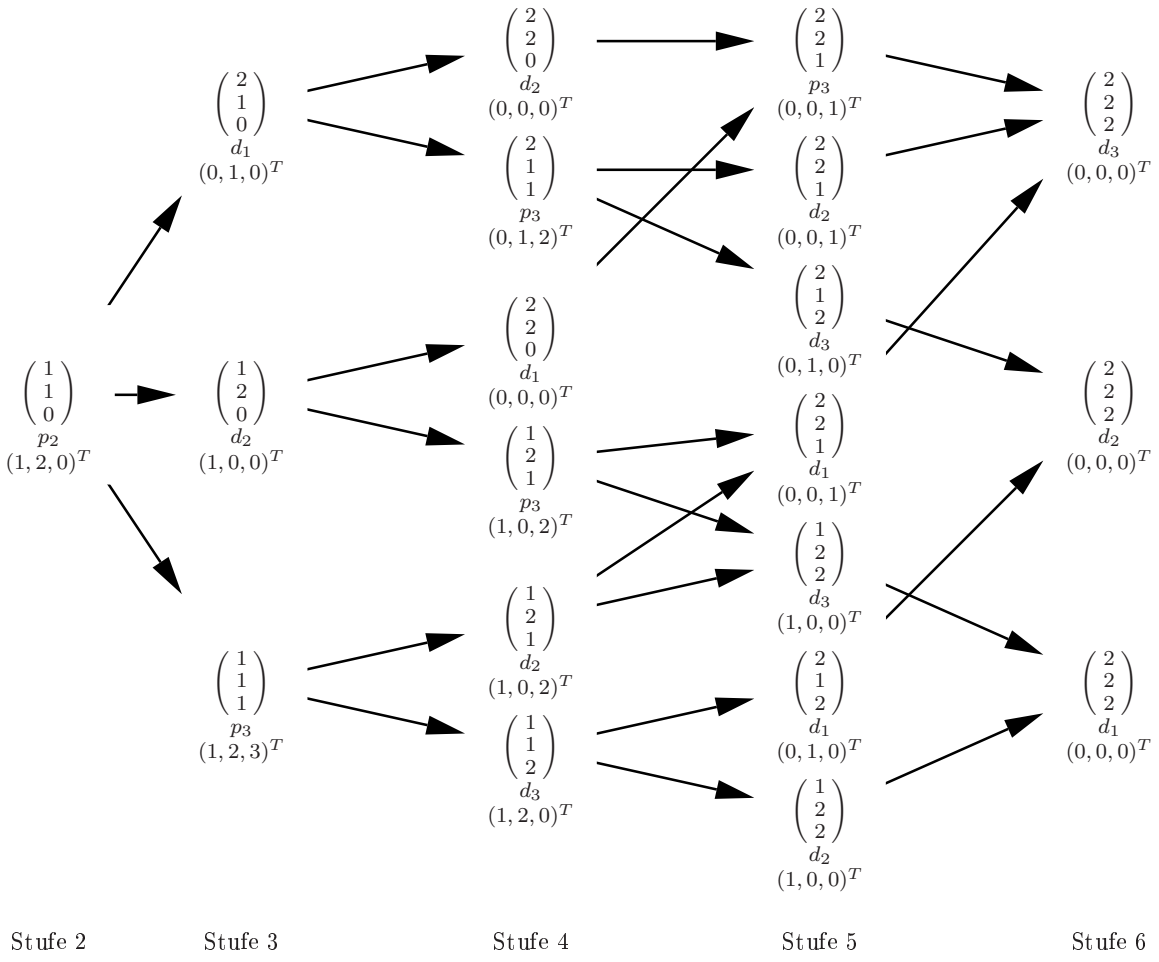

Abbildung 5.5: Der Zustandsgraph $G^{L2}_{3,2}$ mit LIFO-2-Bedingung für drei Aufträge mit Quelle auf Stufe 2 und Fahrzeugstartposition $a_f = p_2$.

Die Kostenfunktion $c^{Lq}_{n_f,k^*} : E^{Lq}_{n_f,k^*} \to \mathbb{R}$ entspricht der Kostenfunktion des Zustandsgraphen G_{n_f,k^*} ohne Ladebedingungen und beinhaltet die Fahrzeit $c_{\mu',\mu}$ von einem Ort μ' zu einem anderen Ort μ sowie die Wartezeit ω_μ bis zum Beginn des Zeitfensters $[\underline{t}_\mu, \overline{t}_\mu]$ an Ort μ. Damit ist der Graph $G^{Lq}_{n_f,k^*}$ eindeutig definiert.

In Abbildung 5.5 wird beispielhaft der LIFO-2-Zustandsgraph $G^{L2}_{3,2}$ für drei Aufträge und Quelle auf Stufe zwei dargestellt. Anhand der Abbildung wird deutlich, dass sich die Anzahl der Knoten und Kanten sowohl gegenüber dem Zustandsgraphen $G^{Z}_{3,2}$ ohne Ladebedingungen als auch gegenüber dem LIFO-Zustandsgraphen $G^{L}_{3,2}$ deutlich verändert. Im Folgenden werden daher die Anzahl $|V^{Lq}_{n_f,k^*}|$ der Knoten und die Anzahl $|E^{Lq}_{n_f,k^*}|$ der Kanten des LIFO-q-Zustandsgraphen analysiert, um eine Einschätzung der maximalen Rechenzeiten des Algorithmus zu ermöglichen.

Satz 5.3
Es bezeichne $|V^{Lq}_{n_f,0}|$ die Anzahl der Knoten und $|E^{Lq}_{n_f,k^*}|$ die Anzahl der Kanten im LIFO-Zustandsgraphen für n_f Aufträge mit Quelle auf Stufe 0. Dann gilt:

$$|_kV^{Lq}_{n_f,0}| = \sum_{k=0}^{2n_f} \sum_{\substack{j=\max\{0,\\ k-n_f\}}}^{\substack{\min\{\frac{k}{2},\\ n_f\}}} \frac{n_f!}{(n_f-k+j)!(k-2j)!j!} \cdot \cdot \left(\frac{\operatorname{sgn}(k-2j)(k-2j)!}{\max\{0, \min\{k-2j-1, q-n_f+k-j-1\}\}!} + \frac{j(k-2j)!}{\max\{0, \min\{k-2j, q-n_f+k-j\}\}!} \right)$$

$$|E^{Lq}_{n_f,0}| = \sum_{k=0}^{2n_f} \sum_{\substack{j=\max\{0,\\ k-n_f\}}}^{\substack{\min\{\frac{k}{2}\\ n_f\}}} \frac{n_f!}{(n_f-k+j)!(k-2j)!j!} (n_f - k + j + \min\{k-2j, q\}) \cdot \cdot \left(\frac{\operatorname{sgn}(k-2j)(k-2j)!}{\max\{0, \min\{k-2j-1, q-n_f+k-j-1\}\}!} + \frac{j(k-2j)!}{\max\{0, \min\{k-2j, q-n_f+k-j\}\}!} \right)$$

Beweis: Der Beweis erfolgt wiederum konstruktiv mit Hilfe kombinatorischer Überlegungen. Zuerst wird die Anzahl $|V^{Lq}_{n_f,0}|$ der Knoten betrachtet, anschließend die Anzahl $|E^{Lq}_{n_f,0}|$ der Kanten.

Die Anzahl $|V^{Lq}_{n_f,0}|$ der Knoten wird zunächst für jede Stufe $k = 0, \ldots, 2n_f$ separat berechnet. Da nur LIFO-q-Zustandsgraphen $G^{Lq}_{n_f,0}$ mit Beginn auf Stufe $k = 0$ betrachtet werden, sind für alle Einträge $\varphi(i), i = 1, \ldots, n$, des Statusvektors Φ die Status $\varphi(i) = 0$, $\varphi(i) = 1$ und $\varphi(i) = 2$ zulässig, der Vektor ρ der Pickup-Reihenfolge ist auf Stufe $k^* = 0$ der Nullvektor. Die Anzahl der Einträge $\varphi(i)$ mit Status $\varphi(i) = 0$ wird im Folgenden wiederum mit j_0 bezeichnet, die Anzahl Einträge $\varphi(i)$ mit Status $\varphi(i) = 1$ mit j_1 und die Anzahl Einträge $\varphi(i)$ mit $\varphi(i) = 2$ mit j_2. Für die Zulässigkeit des Statusvektors Φ auf Stufe k muss gelten, dass

$$2 + j_1 + 2j_2 = k \tag{5.7}$$

$$j_0 + j_1 + j_2 = n_f \tag{5.8}$$

Formel 5.7 gewährleistet, dass die Summe aller Einträge des Statusvektors auf Stufe k gerade k entspricht, während Formel 5.8 garantiert, dass der Statusvektor Φ aus genau n_f Einträgen besteht.
Auf einem Weg von der Quelle auf Stufe 0 des Graphen $G^{Lq}_{n_f,0}$ bis zu einem Knoten $(\Phi, \mu, \rho)^T$ auf Stufe k wird genau k mal jeweils ein Eintrag $\varphi(i)$ des Statusvektors Φ um genau 1 erhöht. Die LIFO-q-Ladebedingung ist bei einer Quelle auf Stufe 0 nur für die Kanten auschlaggebend, im LIFO-q-Zustandsgraphen können im Statusvektor alle möglichen Kombinationen von Zuständen $\varphi(i) = 0$, $\varphi(i) = 1$ und $\varphi(i) = 2$ entstehen. Es können daher insgesamt

$$\sum_{j_2=\max\{0,\ k-n_f\}}^{\min\{\frac{1}{2}k,\ n_f\}} \frac{n_f!}{j_0!j_1!j_2!}$$

verschiedene Statusvektoren auf Stufe k des LIFO-q-Zustandsgraphen entstehen.
Um die Anzahl der verschiedenen Zustände $(\Phi, \mu, \rho)^T$ auf Stufe k zu erhalten, fehlen nun noch die Anzahl der Fahrzeugpositionen μ und die Anzahl der zu unterscheidenden Pickup-Reihenfolgen ρ, die für einen Statusvektor Φ auf Stufe k möglich sind.

Für die Fahrzeugposition μ gilt: Das Fahrzeug kann sich in jedem der j_1 Pickup-Orte der Aufträge i mit Status $\varphi(i) = 1$ und in jedem der j_2 Delivery-Orte derjenigen Aufträge i mit Status $\varphi(i) = 2$ befinden. Damit erhält man auf Stufe k

$$\sum_{j_2=\max\{0,\ k-n_f\}}^{\min\{\frac{1}{2}(k),\ n_f\}} \frac{n_f!}{j_0!j_1!j_2!}(j_1 + j_2)$$

mögliche Kombinationen aus Statusvektor Φ und Fahrzeugposition μ. Im Vektor ρ, der die Pickup-Reihenfolge wiedergibt, sind nur die zu Aufträgen $i \in N_f$ mit Status $\varphi(i) = 1$ gehörenden Einträge von Null verschieden. Es gibt daher grundsätzlich $j_1!$ verschiedene Möglichkeiten, den Pickup-Reihenfolge-Vektor ρ an den entsprechenden Einträgen mit j_1 Werten von 1 bis j_1 zu füllen. Zu beachten sind allerdings zwei weitere Bedingungen:

- Wenn die Fahrzeugposition der Pickup-Ort p_i ist, muss Auftrag i im Pickup-Reihenfolge-Vektor den höchsten Eintrag besitzen: $\rho(i) = \max_{j=1}^{n_f} \rho(j)$.
- Wenn $j_0 \leq q$ gilt, so ist nur die Reihenfolge der $q - j_0$ zuerst eingeladenen Aufträge i relevant.

Für den LIFO-q-Graphen $G_{n_f}^{Lq}$ bedeutet die erste Bedingung, dass bei Knoten $(\Phi, \mu, \rho)^T$ mit Fahrzeugposition $\mu \in P$ in einem Pickup-Ort ein Eintrag des Reihenfolge-Vektors bereits festgelegt ist, so dass in diesem Fall nur $(j_1 - 1)!$ verschiedene Pickup-Reihenfolgen möglich sind. Im Fall $j_1 = 0$ ist allerdings $(j_1 - 1)!$ nicht definiert, das Ergebnis der möglichen Pickup-Reihenfolgen muss ebenfalls 0 betragen. Da die Anzahl Pickup-Reihenfolgen aber mit der Anzahl j_1 der Pickup-Orte multipliziert wird, kann stattdessen auch

$$\operatorname{sgn}(j_1)(j_1)!$$

geschrieben werden.

Die zweite Bedingung besagt, dass für den Fall $j_0 \leq q$ die Anzahl der für die Berechnung der zu unterscheidenden Reihenfolgevektoren entscheidenden Einträge $\rho(i) > 0$ von bisher $\min\{j_1, q\}$ auf $q - j_0$ sinkt. Die Anzahl der relevanten Einträge kann daher geschrieben werden als $\max\{0, \min\{j_1 - 1, q - j_0 - 1\}\}$ für den Fall $\mu \in P$ bzw. $\max\{0, \min\{j_1, q - j_0\}\}$ für den Fall $\mu \in D$. Es müssen daher nur

$$\frac{\operatorname{sgn}(j_1)(j_1)!}{\max\{0, \min\{j_1 - 1, q - j_0 - 1\}\}!} \quad \text{bzw.} \quad \frac{j_2 j_1!}{\max\{0, \min\{j_1, q - j_0\}\}!}$$

verschiedene Kombinationen aus Fahrzeugposition μ Reihenfolgevektor ρ betrachtet werden.

Auf Stufe k ergeben sich damit

$$|{}_kV_{n_f}^{Lq}| = \sum_{\substack{j_2=\max\{0,\\ k-n_f\}}}^{\min\{\frac{k}{2},\ n_f\}} \frac{n_f!}{j_0!j_1!j_2!}\left(\frac{\operatorname{sgn}(j_1)j_1!}{\max\{0,\min\{j_1-1,q-j_0-1\}\}!} + \frac{j_2j_1!}{\max\{0,\min\{j_1,q-j_0\}\}!}\right) \tag{5.9}$$

Zustände im LIFO-q-Zustandsgraphen.

Durch Auflösen der Formeln 5.8 und 5.7 nach j_0 bzw. j_1 und Ersetzen von j_0 und j_1 entsprechend dieser Formeln erhält man

$$\begin{aligned}|{}_kV_{n_f}^{Lq}| = \sum_{\substack{j_2=\max\{0,\\ k-n_f\}}}^{\min\{\frac{k}{2},\ n_f\}} \Big(&\frac{n_f!}{(n_f-k+j_2)!(k-2j_2)!j_2!}\cdot \\ &\left(\frac{\operatorname{sgn}(k-2j_2)(k-2j_2)!}{\max\{0,\min\{k-2j_2-1,q-n_f+k-j_2-1\}\}!}+\right. \\ &\left.\left.\frac{j_2(k-2j_2)!}{\max\{0,\min\{k-2j_2,q-n_f+k-j_2\}\}!}\right)\right)\end{aligned} \tag{5.10}$$

Die Summe der ${}_kV_{n_f}^{Lq}$ über alle Stufen $k = 0 \dots 2n_f$ bei gleichzeitigem Ersetzen von j_2 durch j entspricht der ersten Behauptung.

Zur Berechnung der Anzahl $|E_{n_f}^{Lq}|$ der Kanten werden jeweils die Mengen der von Stufe k zu Stufe $k+1$ führenden Kanten ${}_kE_{n_f}^{Lq}$ betrachtet, $k = 0 \dots 2n_f - 1$. Von Stufe $k = 0$ verlaufen n_f Kanten zu Stufe 1. Für eine Stufe $k > 0$ gilt, dass für jeden Eintrag $\varphi(i) = 0$ im Statusvektor Φ sowie für maximal q Einträge $\varphi(i) = 1$ von Φ jeweils eine Kante zur Stufe $k+1$ führt. Somit verlaufen von einem Knoten $(\Phi, \mu, \rho)^T$ auf Stufe k genau $j_0 + \min\{j_1, q\}$ Kanten zur nächsten Stufe $k+1$, so dass unter Verwendung der Anzahl Knoten $|{}_kV_{n_f}^{Lq}|$ aus Formel 5.9 gilt:

$$\begin{aligned}|{}_kE_{n_f}^{Lq}| = \sum_{\substack{j_2=\max\{0,\\ k-n_f\}}}^{\min\{\frac{k}{2},\ n_f\}} &\frac{n_f!}{j_0!j_1!j_2!}(j_0+\min\{j_1,q\})\cdot \\ &\left(\frac{\operatorname{sgn}(j_1)j_1!}{\max\{0,\min\{j_1-1,q-j_0-1\}\}!} + \frac{j_2j_1!}{\max\{0,\min\{j_1,q-j_0\}\}!}\right)\end{aligned}$$

Durch Ersetzen von j_0 und j_1 entsprechend der Formeln 5.8 und 5.7 und j_2 durch j erhält man

$$\begin{aligned}|{}_kE_{n_f}^{Lq}| = \sum_{\substack{j=\max\{0,\\ k-n_f\}}}^{\min\{\frac{k}{2},\ n_f\}} \Big(&\frac{n_f!}{(n_f-k+j)!(k-2j)!j!}(n_f-k+j_2+\min\{k-2j,q\})\cdot \\ &\cdot\left(\frac{\operatorname{sgn}(k-2j)(k-2j)!}{\max\{0,\min\{k-2j-1,q-n_f+k-j-1\}\}!}+\right. \\ &\left.\left.+\frac{j(k-2j)!}{\max\{0,\min\{k-2j,q-n_f+k-j\}\}!}\right)\right)\end{aligned} \tag{5.11}$$

n_f	$\lvert V^{L2}_{n_f,0}\rvert$	Δ	$\lvert E^{L2}_{n_f,0}\rvert$	Δ
1	3	0,00%	2	0,00%
2	13	0,00%	16	0,00%
3	58	5,45%	105	2,94%
4	281	29,49%	660	21,32%
5	1 531	88,78%	4 185	62,84%
6	9 505	225,85%	28 038	150,70%
7	67 320	559,55%	205 205	346,74%
8	540 433	1 444,40%	1 667 368	822,22%
9	4 868 245	4 022,17%	15 077 169	2 088,54%
10	48 692 321	12 269,10%	150 960 330	5 799,65%

Tabelle 5.11: Vergleich der Anzahl Knoten und Kanten des $G^{L2}_{n_f,0}$ und des $G^{Z}_{n_f,0}$.

n_f	$\lvert V^{L3}_{n_f,0}\rvert$	Δ	$\lvert E^{L3}_{n_f,0}\rvert$	Δ
1	3	0,00%	2	0,00%
2	13	0,00%	16	0,00%
3	55	0,00%	102	0,00%
4	237	9,22%	612	12,50%
5	1 181	45,62%	3 915	52,33%
6	7 045	141,52%	27 198	143,19%
7	49 295	382,95%	205 310	346,97%
8	394 665	1 027,84%	1 696 936	838,57%
9	3 553 561	2 908,97%	15 454 917	2 143,37%
10	35 540 921	8 928,31%	155 121 210	5 962,26%

Tabelle 5.12: Vergleich der Anzahl Knoten und Kanten des $G^{L3}_{n_f,0}$ und des $G^{Z}_{n_f,0}$.

Kanten. Die Behauptung erhält man durch Summieren der Anzahl ausgehender Kanten $\lvert {}_kE^{Lq}_{n_f}\rvert$ für die Stufen $k = 1 \ldots 2n_f - 1$ und Addition von n_f ausgehenden Kanten der Stufe $k = 0$.

□

Die Tabellen 5.11 und 5.12 stellen die Anzahl der Knoten und Kanten der LIFO-q-Zustandsgraphen $G^{L2}_{n_f,k^*}$ bzw. $G^{L3}_{n_f,k^*}$ für $k^* = 0, n_f = 1 \ldots 10$ dar und geben in der Spalte Δ die prozentuale Steigerung der Anzahlen im Vergleich zum Zustandsgraphen $G^{Z}_{n_f,0}$ an. Es ist gut zu erkennen, dass sich die Anzahl Knoten und Kanten für $n_f \leq q$ nicht unterscheidet, da in diesen Fällen alle Aufträge ausgeladen werden dürfen und die Pickup-Reihenfolge somit unerheblich ist. Für $n_f > q$ wird die Reihenfolgevektor ρ als zusätzliche Charakteristik eines Zustands benötigt, da wegen der LIFO-q-Ladebedingung nicht mehr alle Zustände $(\Phi, \mu, \rho)^T$ mit gleichem Statusvektor Φ und gleicher Fahrzeugposition μ auch zu den gleichen Zuständen auf den folgenden

Problemgruppe		Aufträge		gelöste Instanzen	
Name	Aufträge	absolut	Δ	absolut	Δ
100	51,86	42,21	2,29%	41	2,50%
200	103,08	76,38	4,14%	40	5,26%
400	205,66	139,15	6,13%	36	12,50%
600	308,97	201,40	4,29%	36	2,86%
800	411,75	275,17	5,66%	37	15,63%
1000	514,98	346,31	10,24%	35	12,90%

Tabelle 5.13: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung von $G^{L2}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

Problem-	♯ Knoten absolut			
gruppe	erzeugt	unzulässig	expandiert	Duplikate
100	225.478,39	94.171,80	39.295,59	86.995,45
200	311.062,52	138.835,08	56.037,32	104.133,37
400	396.665,42	195.763,75	72.637,64	113.878,08
600	420.695,42	202.173,83	81.087,87	119.964,85
800	524.700,85	295.994,63	93.483,44	113.978,34
1000	451.924,53	255.592,97	91.502,48	82.805,97

Tabelle 5.14: Anzahl der Knoten gesamt bei Verwendung von $G^{L2}_{n_{f,k^*}}$

Stufen führen. Die Anzahl $|V^{Lq}_{n_f,k^*}|$ der Knoten steigt daher deutlich an. Die Steigerung der Anzahl $|E^{Lq}_{n_f,k^*}|$ der Kanten ist ebenfalls erheblich, fällt aber wegen der Beschränkung auf maximal q ausladbare Aufträge etwas geringer aus.

Testläufe und Auswertung

Zur Auswertung der Auswirkung der LIFO-q Ladebedingung werden Testläufe mit $q = 2$, $q = 3$ und $q = 9$ durchgeführt. Für die Testläufe wird die Testumgebung weiterhin ereignisdiskret eingestellt; zur Beschränkung der Rechenzeit wird die Grenze von einer Million maximal insgesamt zu erzeugenden Knoten verwendet. Den Vergleichen liegen ebenfalls die Ergebnisse der Testläufe zur Verwendung der Heuristik zur Berechnung eines zulässigen Routenplans mit Überprüfung der Zeitfensterkompatibilität und Sortierung der Fahrzeuge nach dem Sortierkriterium S^A_f aus Abschnitt 4.1.2.1 zu Grunde. Hier werden zunächst die Ergebnisse der Testläufe mit $q = 2$ dargestellt.

In Tabelle 5.13 sind die Anzahl der akzeptierten Aufträge und die Anzahl der gelösten Instanzen bei Einhaltung der LIFO-2-Ladebedingung angegeben. Die Anzahl der bearbeiteten Aufträge nimmt im Vergleich mit den Ergebnissen aus Abschnitt 4.1.2.1 um 2,29% bis 10,24% zu und bleibt damit niedriger als in den Testläufen des vorhergehenden Abschnitts unter Verwendung der LIFO-Ladebedingung. Die Anzahl der vollständig gelösten Instanzen steigt

Problem-gruppe	erzeugt		unzulässig		expandiert		Duplikate	
	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	128.652,88	-40,97%	53.351,96	-50,41%	23.203,20	-13,77%	47.881,77	-38,64%
200	186.201,32	-48,54%	83.617,68	-58,72%	34.732,58	-25,14%	58.574,52	-42,29%
400	261.579,64	-42,03%	125.211,08	-52,54%	49.928,20	-16,34%	74.337,78	-35,22%
600	255.101,93	-40,06%	126.425,00	-48,01%	51.892,48	-15,58%	62.602,73	-40,19%
800	389.610,12	-15,98%	221.804,88	-22,40%	71.296,68	2,88%	78.176,85	-10,05%
1000	339.396,86	-39,55%	184.005,47	-46,42%	73.101,45	-16,23%	63.537,17	-41,33%

Tabelle 5.15: Anzahl der Knoten bei Verwendung von $G^{L2}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

Problem-gruppe	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	0,10	0,02	-43,76%	0,0030	0,0012	-14,54%	0,00048	0,00024	-4,05%
200	0,31	0,16	-2,88%	0,0148	0,0075	-21,34%	0,00113	0,00065	-27,79%
400	0,94	0,12	-75,79%	0,0400	0,0054	-57,47%	0,00158	0,00064	-9,56%
600	0,30	0,09	-16,68%	0,0126	0,0049	-25,92%	0,00083	0,00056	-1,96%
800	0,43	0,04	-67,90%	0,0146	0,0047	-43,72%	0,00083	0,00048	-29,41%
1000	0,03	0,03	-67,10%	0,0044	0,0035	-32,73%	0,00072	0,00049	-17,13%

Tabelle 5.16: Die Antwortzeiten bei Verwendung von $G^{L2}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

ebenfalls mit Zuwachsraten von 2,50% bis 15,63% weniger als in den LIFO-Testläufen.

In den Tabellen 5.14 und 5.15 mit den durchschnittlichen Anzahlen der erzeugten, der als unzulässig erkannten, der expandierten und der Duplikat-Knoten können ähnliche, jedoch nicht mehr so deutliche Veränderungen wie in den Testläufen zur LIFO-Ladebedingung festgestellt werden. Die in den Vergleichswerten der LIFO-Ladebedingung auffällige Testinstanz „lrc1103“ weist unter der LIFO-2-Ladebedingung wieder eine normale Anzahl erzeugter Knoten auf; allerdings wird in diesen Testläufen für die Testinstanz „lrc288“ aus der Problemgruppe „800“ ein Wert von 8,6 Millionen Knoten bis zur Erreichung der für den Vergleich mit den Testläufen aus Abschnitt 4.1.2.1 geforderten Anzahl bearbeiteter Aufträge erreicht, der den Mittelwert der im Vergleich erzeugten Knoten deutlich beeinflusst. Die Anzahl der erzeugten Knoten geht in den anderen Problemgruppen um 39,55% bis 48,54% im Vergleich zu den Testläufen ohne Ladebedingung zurück und erreicht somit wiederum höhere Anzahlen als die LIFO-Ladebedingung (ausgenommen die Problemgruppe „800“). Die strukturelle Veränderung, dass im Verhältnis zu den erzeugten Knoten mehr Knoten expandiert werden, lässt sich auch hier noch gut erkennen.

Die in Tabelle 5.16 dargestellten Antwortzeiten zeigen weiterhin zum Teil deutliche Verbesserungen gegenüber den Testläufen ohne Ladebedingung. Die durchschnittlichen Antwort-

Problemgruppe		Aufträge		gelöste Instanzen	
Name	Aufträge	absolut	Δ	absolut	Δ
100	51,86	41,70	1,04%	40	0,00%
200	103,08	76,08	3,73%	40	5,26%
400	205,66	136,47	4,08%	35	9,38%
600	308,97	199,78	3,45%	36	2,86%
800	411,75	272,47	4,63%	35	9,38%
1000	514,98	337,86	7,55%	34	9,68%

Tabelle 5.17: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung von $G^{L3}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

Problem-	♯ Knoten absolut			
gruppe	erzeugt	unzulässig	expandiert	Duplikate
100	241.483,07	108.373,63	36.247,00	91.606,38
200	335.917,33	167.681,27	52.581,75	104.888,38
400	452.989,41	241.776,08	70.364,29	127.079,76
600	434.145,18	231.015,85	72.488,62	113.062,35
800	456.161,44	258.134,36	78.028,66	99.060,08
1000	482.522,12	282.419,34	88.252,47	90.045,97

Tabelle 5.18: Anzahl der Knoten gesamt bei Verwendung von $G^{L3}_{n_{f,k^*}}$

zeiten sinken um 1,96% bis 29,41%, bei den durchschnittlich maximalen Antwortzeiten ist eine Reduktion um 14,54% bis 57,47% zu verzeichnen und die global maximale Antwortzeit wird um 2,88% bis 67,90% verringert. Alle Antwortzeiten liegen jedoch höher als in den Testläufen zur LIFO-Ladebedingung, wobei die absoluten Werte selbst der maximalen Antwortzeiten noch unter 0,2 Sekunden liegen.

In den Testläufen zur LIFO-3-Ladebedingung ($q = 3$) wurden folgende Ergebnisse erzielt:

Die Anzahl der akzeptierten Aufträge und die Anzahl der gelösten Instanzen bei Einhaltung der LIFO-3-Ladebedingung in Tabelle 5.17 zeigen noch leichte Steigerungen im Vergleich zu den Testläufen ohne Ladebedingung, liegen aber mit Zunahmen um 1,04% bis 7,55% bei der Anzahl der akzeptierten Aufträge und 0% bis 9,68% bei den gelösten Instanzen in allen Problemgruppen unter den Ergebnissen der LIFO-2-Testläufe.

In den Tabellen 5.18 und 5.19 mit den durchschnittlichen Anzahlen der erzeugten, der als unzulässig erkannten, der expandierten und der Duplikat-Knoten können ähnliche, jedoch nicht mehr so deutliche Veränderungen wie in den Testläufen zur LIFO-Ladebedingung festgestellt werden. Die Anzahl der erzeugten Knoten sinkt im Vergleich zu den Testläufen ohne Ladebedingung in allen Problemgruppen um 18,84% bis 34,60% und erzielt somit höhere Werte als die LIFO-2-Ladebedingung. Der Anteil der expandierten Knoten an den erzeugten Knoten nimmt wegen der größeren Anzahl Kanten im $G^{L3}_{n_f}$ im Vergleich zu den Testläufen mit LIFO-

Problem-	erzeugt		unzulässig		expandiert		Duplikate	
gruppe	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ	♯ Vgl.	Δ
100	155.596,79	-28,60%	70.192,46	-34,75%	23.918,86	-11,11%	56.873,36	-27,11%
200	236.636,85	-34,60%	115.364,20	-43,05%	38.301,07	-17,45%	73.052,57	-28,02%
400	366.188,90	-18,84%	201.274,47	-23,71%	57.718,34	-3,29%	95.495,59	-16,79%
600	305.213,73	-28,28%	158.234,93	-34,93%	54.586,02	-11,20%	76.977,62	-26,46%
800	369.199,69	-20,38%	209.288,63	-26,78%	65.567,76	-5,39%	74.177,08	-14,65%
1000	407.058,33	-27,50%	227.913,48	-33,64%	77.922,95	-10,71%	80.832,02	-25,37%

Tabelle 5.19: Anzahl der Knoten bei Verwendung von $G^{L3}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

Problem-	global max. Antwortzeit			Ø max. Antwortzeit			Ø Antwortzeit		
gruppe	sec	Vgl	Δ	sec	Vgl	Δ	sec	Vgl	Δ
100	0,29	0,02	-40,67%	0,0063	0,0012	-15,07%	0,00056	0,00022	-10,77%
200	0,28	0,08	-49,07%	0,0129	0,0072	-24,34%	0,00097	0,00077	-13,95%
400	0,11	0,10	-78,62%	0,0059	0,0057	-54,95%	0,00080	0,00058	-18,59%
600	0,08	0,07	-32,73%	0,0072	0,0054	-18,37%	0,00078	0,00057	0,24%
800	0,10	0,04	-68,89%	0,0062	0,0043	-48,74%	0,00065	0,00051	-25,56%
1000	0,04	0,03	-63,78%	0,0048	0,0035	-33,83%	0,00063	0,00050	-16,46%

Tabelle 5.20: Die Antwortzeiten bei Verwendung von $G^{L3}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

und LIFO-2- Ladebedingung zu.

Die in Tabelle 5.20 dargestellten Antwortzeiten zeigen unterschiedliche Veränderungen gegenüber den Testläufen mit der LIFO- bzw. LIFO-2-Ladebedingung. Die durchschnittlichen Antwortzeiten in Problemgruppe „600“ nehmen um 0,24% zu und sinken in den anderen Problemgruppen um 13,95% bis 25,56%, so dass bessere Ergebnisse als bei den Testläufen mit LIFO-2-Bedingung, jedoch deutlich geringere Verbesserungen als bei den Testläufen mit LIFO-Bedingung erreicht werden. Bei den durchschnittlich maximalen Antwortzeiten ist eine Reduktion um 15,07% bis 54,95% zu verzeichnen, was in etwa den Verbesserungen der LIFO-2-Testläufe entspricht. Die global maximale Antwortzeit wird jedoch um 32,73% bis 78,62% verringert, so dass wiederum zum Teil deutlich geringere Zeiten als mit der lIFO2-Bedingung, jedoch höhere absolute Werte als in den Testläufen mit der LIFO-Bedingung erzielt werden.

Zusätzlich wurden Testläufe mit LIFO-9-Ladebedingung ($q = 9$) durchgeführt. Da sich die Graphen $G^{L9}_{n_{f,k^*}}$ und $G^{Z}_{n_{f,k^*}}$ erst für mehr als neun einem Fahrzeug f zugeordneten Aufträgen ($n_f > q = 9$) unterscheiden, sollten ähnliche Ergebnisse wie in den Testläufen ohne Ladebedingungen erzielt werden:
In Tabelle 5.21 werden die Anzahl der akzeptierten Aufträge und die Anzahl der gelösten In-

Problemgruppe		Aufträge		gelöste Instanzen	
Name	Aufträge	absolut	Δ	absolut	Δ
100	51,86	41,27	0,00%	40	0,00%
200	103,08	73,35	0,00%	38	0,00%
400	205,66	131,12	0,00%	32	0,00%
600	308,97	193,45	0,17%	35	0,00%
800	411,75	260,42	0,00%	32	0,00%
1000	514,98	315,95	0,58%	31	0,00%

Tabelle 5.21: Anzahl bearbeiteter Aufträge und vollständig gelöster Instanzen bei Verwendung von $G^{L9}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

Problem-	♯ Knoten absolut			
gruppe	erzeugt	unzulässig	expandiert	Duplikate
100	217.883,09	107.552,11	26.904,98	78.016,50
200	362.902,98	203.009,43	46.567,62	101.947,87
400	450.583,69	263.279,34	59.639,37	114.731,27
600	423.444,55	241.477,77	61.453,03	104.262,93
800	460.896,51	283.092,17	69.246,24	86.892,88
1000	562.080,57	343.140,79	87.628,84	108.872,71

Tabelle 5.22: Anzahl der Knoten gesamt bei Verwendung von $G^{L9}_{n_{f,k^*}}$

stanzen bei Einhaltung der LIFO-9-Ladebedingung dargestellt. Im Vergleich zu den Testläufen ohne Ladebedingung sind nur leichte Änderungen in der Anzahl der bearbeiteten Aufträge mit Unterschieden von maximal 0,58% zu konstatieren, während die Anzahl der gelösten Instanzen in allen Problemgruppen mit der Anzahl gelöster Instanzen der Testläufe ohne Ladebedingung übereinstimmt.

Die Tabellen 5.22 und 5.23 stellen die durchschnittlichen Anzahlen der erzeugten, der als unzulässig erkannten, der expandierten und der Duplikat-Knoten dar. Erwartungsgemäß sind auch hier nur leichte Veränderungen im Vergleich zu den Testläufen ohne Ladebedingungen aus Abschnitt 4.1.2.1 festzustellen: Die Anzahl der erzeugten Knoten weicht nur um -1,22% bis 0,30% von der Anzahl erzeugter Knoten in den Testläufen ohne Ladebedingungen ab. Auch in der Anzahl unzulässiger, expandierter und Duplikat-Knoten sind die Ergebnisse ähnlich den Ergebnissen ohne Ladebedingungen.

Die in Tabelle 5.24 dargestellten Antwortzeiten zeigen für die Problemgruppe „1000“ Zuwächse um 17,90% bei den durchschnittlich maximalen und 29,14% bei den global maximalen Antwortzeiten. Dieser Zuwachs liegt jedoch nur an der maximalen Antwortzeit von 0,12 Sekunden in der Testinstanz „lrc2105“, die einen Ausreißer in der Menge der Antwortzeiten dieser Problemgruppe darstellt. Die Antwortzeiten aller anderen Problemgruppen zeigen in allen Kategorien um bis zu 3% schlechtere, aber prinzipiell ähnliche Werte wie in den Ergebnissen der

Problem-gruppe	erzeugt ♯ Vgl.	erzeugt Δ	unzulässig ♯ Vgl.	unzulässig Δ	expandiert ♯ Vgl.	expandiert Δ	Duplikate ♯ Vgl.	Duplikate Δ
100	217.883,09	-0,02%	107.552,11	-0,03%	26.904,98	-0,01%	78.016,50	-0,02%
200	362.902,98	0,30%	203.009,43	0,22%	46.567,62	0,36%	101.947,87	0,45%
400	450.583,69	-0,14%	263.279,34	-0,21%	59.639,37	-0,07%	114.731,27	-0,03%
600	420.377,05	-1,22%	238.806,87	-1,80%	61.240,28	-0,38%	104.080,97	-0,56%
800	460.896,51	-0,61%	283.092,17	-0,96%	69.246,24	-0,08%	86.892,88	-0,02%
1000	556.278,83	-0,92%	338.581,78	-1,41%	87.105,07	-0,18%	108.162,10	-0,13%

Tabelle 5.23: Anzahl der Knoten bei Verwendung von $G^{L9}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

Problem-gruppe	global max. Antwortzeit sec	global max. Antwortzeit Vgl	global max. Antwortzeit Δ	Ø max. Antwortzeit sec	Ø max. Antwortzeit Vgl	Ø max. Antwortzeit Δ	Ø Antwortzeit sec	Ø Antwortzeit Vgl	Ø Antwortzeit Δ
100	0,03	0,03	2,25%	0,0014	0,0014	3,40%	0,00026	0,00026	3,52%
200	0,17	0,17	3,98%	0,0099	0,0099	4,03%	0,00092	0,00092	3,35%
400	0,49	0,49	3,61%	0,0131	0,0131	3,77%	0,00073	0,00073	2,50%
600	0,11	0,11	4,43%	0,0070	0,0070	5,78%	0,00057	0,00057	0,76%
800	0,12	0,12	3,96%	0,0081	0,0081	-3,29%	0,00066	0,00066	-3,10%
1000	0,13	0,13	29,84%	0,0062	0,0062	18,92%	0,00059	0,00058	-1,69%

Tabelle 5.24: Die Antwortzeiten bei Verwendung von $G^{L9}_{n_{f,k^*}}$ im Vergleich zur Verwendung von $G^{Z}_{n_{f,k^*}}$

Testläufe ohne Ladebedingungen. Die durchschnittlichen Antwortzeiten sind mit weniger als einer Millisekunde bei einer maximalen Antwortzeit von nur $\frac{1}{4}$ Sekunde weiterhin sehr gut.

5.3 Fazit

In diesem Kapitel wurde untersucht, wie sich zusätzliche Nebenbedingungen in Form von Ladebedingungen auf die allgemeine Lösbarkeit der Problemstellung und insbesondere auf die Laufzeiten des entwickelten Algorithmus auswirken. Dazu wurden zu den Ladebedingungen Backhauls, LIFO und LIFO-q jeweils zunächst die Zustandsgraphen $G^{B}_{n_{f,k^*}}$, $G^{L}_{n_{f,k^*}}$ und $G^{Lq}_{n_{f,k^*}}$ definiert und die Anzahl der Knoten und Kanten als Maß für den Speicherplatzbedarf und die Rechenzeit im "Worst Case" angegeben. Anschließend wurde in Testläufen das tatsächliche Verhalten des Algorithmus überprüft.

Der Zustandsgraph $G^{B}_{n_{f,k^*}}$ zur Ladebedingung „Backhauls" besitzt deutlich weniger Knoten und Kanten als der Zustandsgraph $G^{Z}_{n_{f,k^*}}$ ohne Ladebedingungen. Erwartungsgemäß sind sowohl die Anzahl der erzeugten Knoten als auch die Antwortzeit bei Verwendung der vorgestellten Heuristik deutlich geringer als in den Testläufen ohne Ladebedingungen.

Für das Entscheidungsproblem mit der LIFO-Ladebedingung hat der Zustandsgraph $G^{L}_{n_{f,k^*}}$

für $n_f > 2$ deutlich mehr Knoten und für $n_f > 4$ auch deutlich mehr Kanten als der Zustandsgraph $G^Z_{n_{f,k^*}}$ ohne Ladebedingungen. Die Testläufe weisen jedoch trotzdem eine geringere Anzahl erzeugter Knoten und eine geringere Antwortzeit als in den Testläufen ohne Ladebedingungen auf. Die Anwendung des angepassten A*-Algorithmus auf der veränderten Graphenstruktur mit einem geringeren Anteil Kanten führt zu einem schnellen Auffinden der optimalen Lösung.

Die Definition der LIFO-q Graphen $G^{Lq}_{n_{f,k^*}}$ unterscheidet sich wegen des benötigten Reihenfolgevektors als zusätzliches Charakteristikum für einen Zustand von den vorhergehenden Zustandsgraphen. Die Anzahl der Knoten und Kanten ist für $n_f \leq q$ mit der Anzahl der Knoten und Kanten im Zustandsgraphen $G^Z_{n_{f,k^*}}$ identisch, steigt jedoch für $n_f > q$ deutlich an. In den Testläufen werden jedoch im Gegensatz dazu für $q = 2$ die kleinsten Anzahlen Knoten und die besten Antwortzeiten verzeichnet, während sich sowohl die Anzahl der erzeugten Knoten als auch die Antwortzeiten für $q = 9$ den Ergebnissen der Testläufe ohne Ladebedingung annähern.

Die entwickelte Heuristik ist demnach auch für Entscheidungsprobleme mit Ladebedingungen gut verwendbar. Durch eine geeignete Anpassung der Schätzfunktion des A*-Algorithmus sowie der Zeitfensterüberprüfung und des Sortierkriteriums in der Heuristik zur Berechnung eines zulässigen Routenplans können die Ergebnisse gegebenenfalls noch verbessert werden.

Kapitel 6

Fazit

Das Ziel der vorliegenden Arbeit besteht darin, einen Algorithmus zur Verfügung zu stellen, mit dem das vorgestellte dynamische Pickup und Delivery Vehicle Routing Problem mit Zeitfenstern mit einer Antwortzeit von wenigen Sekunden gelöst werden kann.

Dazu wurde nach der exakten Formulierung des Entscheidungsproblems der Algorithmus von Caramia et al. ([11]) vorgestellt, der für eine ähnliche Problemstellung entwickelt wurde, sich jedoch wegen der stark ansteigenden Rechenzeiten nur für kleine Testinstanzen mit wenigen Aufträgen pro Fahrzeug eignet. Auf Basis dieses Algorithmus, bestehend aus einer heuristischen Zuordnung der Aufträge zu den Fahrzeugen unter Einbezug einer optimalen Routenplanung, wird in den Kapiteln 3 und 4 ein schneller Algorithmus zur Lösung des vorgestellten Entscheidungsproblems entwickelt.

In Kapitel 3 wird zunächst die optimale Routenplanung von Caramia et al. bestehend aus der Anwendung einer Variante des A*-Algorithmus auf den Statusvektorbaum, analysiert, mit einer Testumgebung implementiert und getestet. Anschließend wird der Zustandsgraph als alternative zum Statusvektorbaum entwickelt, analysiert, in den Algorithmus integriert und die erwarteten Verbesserungen anhand von Testläufen in der gleichen Umgebung dokumentiert. Die Anzahl der erzeugten Knoten und die erreichten Rechenzeiten konnten bereits bedeutend um durchschnittlich ca. 70% bzw. 99% gesenkt werden.
Anschließend wurde eine spezielle Schätzfunktion für den A*-Algorithmus entwickelt, die für Routenplanungsprobleme unabhängig von der geographischen Verteilung der anzufahrenden Orte und der Anzahl der noch anzufahrenden Orte bei vertretbarem Rechenaufwand eine Abschätzung für die noch verbleibende Zeit einer Route berechnet. Unter Verwendung dieser Schätzfunktion konnten sowohl die Anzahl der erzeugten Knoten als auch die Antwortzeiten um weitere 85% gesenkt werden.

Im Kapitel 4 wurde die Heuristik zur Berechnung eines zulässigen Routenplans auf weitere Möglichkeiten zur Reduzierung der Rechenzeiten untersucht und anschließend das Ergebnis der Heuristik durch die in [26] eingeführte Tabu-Suche verbessert und in einer dynamischen Umgebung getestet.

Dazu wurde zunächst die Überprüfung der Zeitfensterkompatibilität eingeführt, um nicht lösbare Routenplanungsprobleme frühzeitig zu erkennen. Anschließend wurde der Antwortzeitpunkt gemäß dem vorgestellten Entscheidungsproblem verändert und die Antwortzeit durch die Wahl eines geeigneten Sortierkriteriums für die Reihenfolge der Fahrzeuge in der Zuordnung eines Auftrags weiter verringert. Diese Änderungen an der Heuristik führten zu einer geringen Reduzierung der Anzahl der erzeugten Knoten, die Antwortzeiten konnten jedoch nochmals um 99% verringert werden.
Anschließend wurde die in [26] eingeführte Tabu-Suche implementiert und die Testläufe in einer dynamischen Umgebung durchgeführt, wobei sowohl die Zuordnung als auch die Tabu-Suche bei wenigen, zu langen Berechnungen für einen Auftrag unterbrochen werden. Die Ergebnisse dieser Testläufe zeigen durchschnittliche Antwortzeiten von weniger als $\frac{1}{4}$ Sekunde, die maximale Antwortzeit aller 352 Testläufe betrug 37 Sekunden. Die durchschnittlich maximalen Antwortzeiten von höchstens 2,5 Sekunden zeigen jedoch, dass solche langen Antwortzeiten nur selten zu erwarten sind. Der entwickelte Algorithmus ist somit für das vorliegende dynamische Pickup und Delivery Vehicle Routing Problem mit Zeitfenstern gut geeignet.

Abschließend wurde in Kapitel 5 der Einfluß von Ladebedingungen untersucht. Dazu wurden entsprechende Zustandsgraphen unter Berücksichtigung der Ladebedingungen entwickelt, analysiert und implementiert. Die entwickelte Heuristik liefert auch bei Verwendung dieser Zustandsgraphen mit Ladebedingungen sehr gute Ergebnisse.

Die in Kapitel 1.1 geforderten zusätzlichen Eigenschaften des Algorithmus konnten ebenfalls erfüllt werden: zusätzliche Nebenbedingungen, mehrdimensionale Kapazitätsbeschränkungen der Fahrzeuge, eine andere Strukturierung der Kosten oder eine andere Zielfunktion können leicht implementiert werden.

Anhang

Implementierung: Eine Möglichkeit zur Umsetzung des Zustandsgraphen

Dieser Abschnitt behandelt die Frage, wie die Struktur des Zustandsgraphen effizient implementiert werden kann, ohne die Laufzeit des A*-Algorithmus zu erhöhen.

Bei der Verwendung des Statusvektorbaums kann bei der Anwendung des A*-Algorithmus der Graph problemlos erzeugt werden: In jedem Durchlauf wird ein offener Knoten mit minimalen Kosten ausgewählt, expandiert und als geschlossen markiert, während alle seine Nachfolger-Knoten als offen markiert werden. Dies wird iterativ fortgesetzt, bis eine Senke als Knoten mit minimalen Kosten ausgewählt wird (vgl. Abschnitt 3.2.3). Wegen der Baumstruktur des Graphen G^S_{n,k^*} ist dieses Vorgehen ohne weitere Überprüfung der neu erzeugten Nachfolger-Knoten möglich.

Im Zustandsgraphen können jetzt jedoch durch die Addition eines Einheitsvektors e_i zu einem Statusvektor Φ eines Knotens einer Stufe k *Duplikat-Knoten* erzeugt werden, d. h. Knoten $(\Phi', \mu')^T$ auf der Stufe k+1, die aus demselben Statusvektor Φ' und derselben Fahrzeugposition μ' wie ein bereits existierender Knoten der Stufe k+1 bestehen. Im Unterschied zum Statusvektorbaum G^S_{n,k^*} werden jedoch im Zustandsgraphen G^Z_{n,k^*} keine identischen Zustandsvektoren zugelassen (vgl. Abschnitt 3.3.1). In diesem Fall müssen daher so viele Duplikat-Knoten wieder gelöscht werden, dass nur ein Knoten $(\Phi', \mu')^T$ erhalten bleibt. Da der Weg mit minimaler Fahrzeit gesucht wird, können die Duplikat-Knoten mit nicht minimaler Fahrzeit entfernt werden, da sie nicht auf dem optimalen Weg von der Quelle zu einer der Senken liegen können. Der Knoten $(\Phi', \mu')^T$ mit kürzerer Fahrzeit kann dann als offen markiert und im Laufe des Algorithmus eventuell weiter expandiert werden.

Die Schwierigkeit für die algorithmische Behandlung einer solchen Situation besteht nun darin, eine Datenstruktur zu wählen, die sowohl die Suche nach solchen Duplikat-Knoten als auch die Suche nach fahrzeitminimalen Knoten unterstützt, da in jedem Durchlauf des Algorithmus zunächst ein Knoten mit minimaler Fahrzeit ausgewählt wird und anschließend jeder seiner Nachfolge-Knoten auf Duplikate überprüft werden muss. Um diesen Anforderungen gerecht zu werden, müssen folgende Voraussetzungen geschaffen werden:

- Die Zustandsvektoren $(\Phi, \mu)^T$ müssen effizient miteinander verglichen werden können.

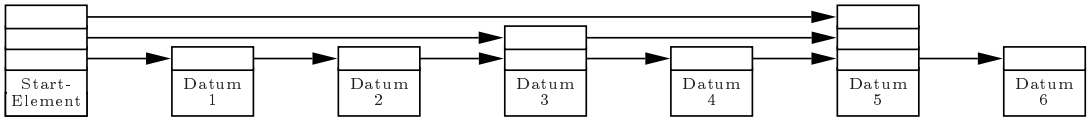

Abbildung 1: Eine Skip List mit sechs Daten-Elementen und bis zu drei Skip Level.

Da insbesondere der Vergleich von zwei Statusvektoren Φ, Φ' relativ aufwändig ist und somit die Rechenzeit negativ beeinflussen würde, sollte hier Abhilfe geschaffen werden.

- Die Datenstruktur muss zur Dynamik des wachsenden Zustandsgraphen passen, da mitunter sehr viele Knoten erzeugt und in die Datenstruktur eingefügt werden. Es muss daher sehr schnell möglich sein, neu erzeugte Knoten auf Duplikat-Knoten zu überprüfen, sie in die Datenstruktur einzufügen und eventuell andere Knoten zu entfernen und nach einem fahrzeitminimalen Element zu suchen.

Beide Forderungen können zunächst getrennt voneinander betrachtet werden. Zuerst wird hier auf die Forderung nach einer schnelleren Vergleichsmöglichkeit der Statusvektoren eingegangen. Ein Statusvektor Φ besteht aus n Einträgen $\varphi(i) \in \{0, 1, 2\}$, so dass er eineindeutig als Zahl im Dreiersystem interpretiert und im Zehnersystem dargestellt werden kann als:

$$Z(\Phi) = \sum_{i=1}^{n} \varphi(i) \cdot 3^{n-i-1}$$

Das Tupel $(Z(\Phi), \mu)$ wird im Folgenden die *Knotenidentität* des Knotens $(\Phi, \mu)^T$ genannt. Sie repräsentiert den Knoten eineindeutig und ermöglicht einen effizienten Vergleich von zwei Knoten, da dazu nur der Vergleich von zwei natürlichen Zahlen nötig ist. Die erste Forderung kann somit leicht erfüllt werden.

Nun muss noch eine dynamische Datenstruktur geschaffen werden, die die Suche nach zwei verschiedenen Eigenschaften eines Knotens $(\Phi, \mu)^T$ unterstützt. Zur effizienten Suche in Daten werden im Allgemeinen balancierte Bäume eingesetzt, um die Anzahl der Schritte bis zum Auffinden des gesuchten Datums zu minimieren. Balancierte Bäume sind jedoch für dynamische Datenstrukturen nicht geeignet, da das Einfügen und Löschen von Daten sehr aufwändig ist [47]. Abhilfe kann an dieser Stelle die Datenstruktur „*Skip List*“ schaffen, die 1990 von Pugh eingeführt wurde [61] und die Vorteile der balancierten Bäume mit den Möglichkeiten des „dynamischen Wachstums“ verbindet.

Eine Skip List ist eine mehrlagige sortierte Liste von Daten-Elementen, wobei die Lagen probabilistisch erstellt werden. Dazu werden die zu sortierenden Daten in sogenannten Daten-Elementen gespeichert, bestehend aus dem Datum selbst (mit dem Ordnungsmerkmal) und einer zufälligen Anzahl Skip Level, über die die effiziente Suche ermöglicht wird. Die Anzahl

der Skip Level eines Daten-Elements ist eine zufällige Größe, deren Wahrscheinlichkeit mit zunehmender Anzahl Level abnimmt. Eine Skip List besteht aus einem Startelement mit Datum Null und der maximalen Anzahl Skip Level sowie einer Menge von Daten-Elementen, die entsprechend dem Ordnungsmerkmal sortiert sind. Jedes Daten-Element hat mindestens den Skip Level 1, der einen Zeiger auf das laut Ordnungsmerkmal nachfolgende Daten-Element enthält. Somit ist die Skip List auf dem ersten Level eine einfach verkettete Liste. Zusätzlich ist jedes Daten-Element, das mehr als nur einen Skip Level besitzt, auf dem zweiten Level mit dem laut Ordnungsmerkmal nächsten Daten-Element mit mindestens zwei Skip Level verkettet. So wird für jeden Skip Level verfahren, so dass sich eine geordnete Liste wie in Abbildung 1 ergibt.

Beim Einfügen eines neuen Datums in die Liste wird zufällig die Anzahl der Skip Level für dieses Daten-Element bestimmt, es erhält aber mindestens ein Skip Level. Das neue Daten-Element muss nun in die Skip List einsortiert werden. Dazu muss zunächst das Daten-Element identifiziert werden, dass laut Ordnungsmerkmal der direkte Vorgänger des neuen Elements in der Liste wird. Die Suche nach diesem Element beginnt im Start-Element der Skip List: auf dem höchsten Level wird solange vom aktuellen Element η zum jeweils nachfolgenden Daten-Element η' „gesprungen“ (engl. skip), bis das Ordnungsmerkmal des dann nachfolgenden Daten-Elements η'' anzeigt, dass das neue Daten-Element vor diesem in die Liste eingefügt werden muss. Dann wird vom Element η' die Suche auf dem nächstniedrigeren Level analog fortgesetzt, bis schließlich der direkte Vorgänger des neu einzufügenden Elements beim Durchsuchen eines Listenteils auf Level 1 gefunden wurde.

Die zufällige Wahl der Anzahl Skip Level sorgt dafür, dass auch bei dynamischen Wachstum der Liste die Abstände der Elemente mit gleicher Anzahl Skip Level nicht überdurchschnittlich wachsen. Durch das Zufallselement wird eine vertretbare Verteilung erwartet, ohne dass Arbeitsaufwand für die neue Strukturierung der Liste beim Einfügen oder Löschen von Daten-Elementen benötigt wird.

Für die Anwendung des A*-Algorithmus auf den Zustandsgraphen G^Z_{n,k^*} bei gleichzeitiger Erzeugung des Graphen muss eine sortierte Liste mit den Knoten des Zustandsgraphen zuzüglich der kleinsten bisher bekannten Fahrzeit zur Erreichung dieses Knotens als Daten erstellt werden. Diese Liste muss sowohl nach einem Element mit minimaler bekannter Fahrzeit als auch nach einem Element mit bestimmter Knotenidentität durchsucht werden, d. h. es werden zwei verschiedene Ordnungsmerkmale angegeben. Die soeben vorgestellte Skip List kann jedoch nur nach einem Ordnungsmerkmal sortiert werden. Die Lösung besteht im Aufbau einer doppelten Skip List:

Dazu wird zunächst eine einfache Skip List wie oben beschrieben aufgebaut, z. B. mit dem Ordnungsmerkmal der Knotenidentität. Um nun auch die Suche nach dem fahrzeitminimalen Daten-Element zu unterstützen, bekommt jedes Daten-Element noch einmal eine zufällige Anzahl Skip Level zugeordnet. Wie oben beschrieben, werden auch diese Skip Level jeweils mit

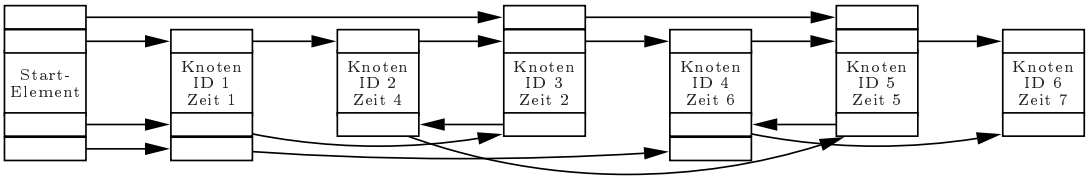

Abbildung 2: Eine zweifache Skip List mit sechs Daten-Elementen mit den Daten „Knoten mit Knotenidentität ID und kleinster bekannter Fahrzeit“ und bis zu drei Skip Level.

dem nach dem zweiten Ordnungsmerkmal nächsten Level derselben Höhe verkettet, also mit dem –laut Fahrzeit – nächsten Daten-Element mit mindestens genau so vielen Skip Leveln. Auf diese Weise kann mit relativ wenig Speicherplatzbedarf eine Struktur geschaffen werden, die trotz dynamischen Wachstums schnell nach den beiden Ordnungsmerkmalen Knotenidentität und Fahrzeit durchsucht werden kann und beim Einfügen und Löschen von Elementen leicht zu handhaben ist. Die Abbildung 2 zeigt beispielhaft eine solche Skip List, wobei die Anordnung der sechs Daten-Elemente nach der Sortierung der Knotenidentität (ID) erfolgt, so dass die oberen Verkettungen geordnet erscheinen. Mit Hilfe dieser Datenstruktur kann nun bei der Verwendung des A*-Algorithmus auch der Zustandsgraph problemlos erzeugt werden, indem beim Expandieren eines Knotens $(\Phi, \mu)^T$ jeder seiner Nachfolger $(\Phi', \mu')^T$ berechnet und dann die Skip List nach einem bereits existierenden Knoten mit der Identität $(Z(\Phi'), \mu')$ durchsucht wird. Wird solch ein Knoten gefunden, wird der fahrzeitminimale der beiden ansonsten identischen Knoten $(\Phi', \mu')^T$ als Daten-Element in der Skip List gespeichert, das andere Daten-Element mit dem „teureren“ Duplikat-Knoten wird gelöscht.

Liste der benutzten Variablen

Variable	Bedeutung
α_μ, α_k	geplante Ankunftszeit in einem Ort μ bzw. im Ort der Zustandsvariablen auf Stufe k des Single Vehicle Routing-Modells
$\gamma(x)$	die Zugehörigkeitsfunktion der Multimenge Γ
Γ	eine Multimenge
$\delta(x)$	beliebige reellwertige Funktion im Beweis der Anzahl Knoten im Statusvektorbaum
$\epsilon(i)$	die Einsparung bei Entfernen von Auftrag i aus der Route
$\theta_\kappa(\upsilon_\kappa)$	Zielfunktion im Gesamt-Modell
κ	Entscheidungsstufe im Gesamt-Modell
λ_μ^f	geladene Menge des Fahrzeugs f nach dem Be- oder Entladen in Ort μ
μ, ν, μ_k	beliebige Orte (zur Zustandsvariablen auf Stufe k des Single Vehicle Routing-Modells gehörend)
τ	ein Zeitpunkt
τ_κ	Zeitpunkt des Eingangs von Auftrag Nr. κ auf Stufe κ im Gesamt-Modell
υ_κ	Entscheidungsvariable im Gesamt-Modell auf Stufe κ
$\varphi(i), \varphi^f(i), \varphi_k(i)$	der Status des Auftrags i (in Fahrzeug f / auf Stufe k des Single Vehicle Routing-Modells)
$\Phi, \Phi^f, \Phi_k,$	ein Statusvektor (des Fahrzeugs f / auf Stufe k des Single Vehicle Routing-Modells)
$\phi_\kappa(\chi_{\kappa-1}, \upsilon_\kappa, \tau_\kappa)$	Übergangsfunktion des Gesamt-Modells
χ_κ	Zustand des Gesamt-Modells auf Stufe κ
$\mathcal{X}_\kappa$	Menge aller möglichen Zustände auf Stufe κ im Gesamt-Modell
ω_μ	die Wartezeit bis zum Beginn des Zeitfensters von Ort μ
a_f	Startposition von Fahrzeug f
A	die Menge aller Startpositionen der Fahrzeuge
b	ein Element der Menge B
B	eine beliebige Menge
c	die Kostenfunktion eines Graphen
$c_{n,k^*}^{B/L/Lq}$	die Kostenfunktion des Zustandsgraphen mit Ladebedingung Backhauls, LIFO bzw. LIFO-q für n Aufträge mit Quelle auf Stufe k^*
c_{n,k^*}^S	die Kostenfunktion des Statusvektorbaums für n Aufträge mit Quelle auf Stufe k^*
c_{n,k^*}^Z	die Kostenfunktion des Zustandsgraphen für n Aufträge mit Quelle auf Stufe k^*
$c^{Orte}, c_{\Phi,\mu}^{Orte}$	die Kostenfunktion des Graphen G^{Orte} bzw. $G_{\Phi,\mu}^{Orte}$
$c_{\mu,\nu}$	für die Strecke von Ort μ zu Ort ν benötigte Fahrzeit

Variable	Bedeutung
c^*_ν	die in der optimalen Route benötigte Fahrzeit von einem Ort ν' zum Ort ν
$C(x_{k-1}, y_k)$	Kostenfunktion im Single Vehicle Routing-Modell
$\overleftarrow{C}((\Phi,\mu)^T)$	Kosten des bisher bekannten kürzesten Wegs von der Quelle des Graphen bis zum Knoten $(\Phi,\mu)^T$
$\overrightarrow{C}_i((\Phi,\mu)^T,(\Phi',\mu')^T)$	geschätzte Kosten des kürzesten Wegs zur Abarbeitung des Auftrags i für den Übergang von Zustand $(\Phi,\mu)^T$ zu $(\Phi',\mu')^T$
$\overrightarrow{C}((\Phi,\mu)^T, V^*)$	Schätzfunktion: minimale Fahrzeit von Zustandsvektor $(\Phi,\mu)^T$ zu einem Zielknoten aus der Menge V^*
d_i	Delivery-Ort von Auftrag i
D, D_f	Menge der Delivery-Orte (des Fahrzeugs f)
$\tilde{e}_\Phi(x)$	die Zugehörigkeitsfunktion der Multimenge der Kanten $\tilde{E}(Phi)$
$E(\Phi), \tilde{E}(Phi)$	die Menge der ausgehenden Kanten eines Knotens Φ, als einfache Menge und als Multimenge
E	die Kantenmenge eines Graphen
$E^{B/L/Lq}_{n,k^*}, {}_kE^{B/L/Lq}_{n,k^*}$	die Kantenmenge des Zustandsgraphen mit Ladebedingung Backhauls, LIFO bzw. LIFO-q für n Aufträge mit Quelle auf Stufe k^* (auf Stufe k)
$E^S_{n,k^*}, {}_kE^S_{n,k^*}$	die Kantenmenge des Statusvektorbaums für n Aufträge mit Quelle auf Stufe k^* (auf Stufe k)
$E^Z_{n,k^*}, {}_kE^Z_{n,k^*}$	die Kantenmenge des Zustandsgraphen für n Aufträge mit Quelle auf Stufe k^* (auf Stufe k)
$E^{Orte}, E^{Orte}_{\Phi,\mu}$	die Kantenmenge des Graphen G^{Orte} bzw. $G^{Orte}_{\Phi,\mu}$
$\mathcal{E}(i)$	die heuristische Ersparnis eines Auftrags i bei Streichung der zugehörigen Orte aus einer Route
f	ein Fahrzeug aus F
F	Menge der Fahrzeuge
g_i	im Fahrzeug für den Transport des Auftrags i benötigte Kapazität
G	ein Graph
$G^{B/L/Lq}_{n,k^*}, {}_kG^{B/L/Lq}_{n,k^*}$	der Zustandsgraph mit Ladebedingung Backhauls, LIFO bzw. LIFO-q für n Aufträge mit Quelle auf Stufe k^* (auf Stufe k)
G^S_{n,k^*}	Statusvektorbaum für n Aufträge mit Quelle auf Stufe k^*
G^Z_{n,k^*}	Zustandsgraph für n Aufträge mit Quelle auf Stufe k^*
$G^{Orte}, G^{Orte}_{\Phi,\mu}$	Graph der geographischen Verteilung aller (für die zukünftige Route eines Fahrzeugs mit Statusvektor Phi und Position μ noch anzufahrenden) Orte
$h(x_{k-1}, y_k)$	Übergangsfunktion im Single Vehicle Routing-Modell
i	ein Auftrag aus der Menge N
k	eine Stufe im Single Vehicle Routing-Modell

Variable	Bedeutung
k^*	die Start-Stufe im Single Vehicle Routing Modell, die Anzahl der zum Zeitpunkt τ geladenen Aufträge
l_f	Kapazität des Fahrzeugs f
$\mathcal{L}$	eine Lösung des dynamischen Pickup and Delivery Vehicle Routing Problems mit Zeitfenstern
m	Anzahl der Fahrzeuge
M	eine „große Zahl“ als Konstante für die Modellierung
n, n_f	Anzahl der Aufträge (des Fahrzeugs f) zu einem Zeitpunkt
$N, N_f^\tau, \tilde{N}_f$	Menge der Aufträge (des Fahrzeugs f zum Zeitpunkt τ / die in Fahrzeug f geladen sind)
p_i	Pickup-Ort von Auftrag i
P, P_f	Menge aller Pickup-Orte (des Fahrzeugs f)
q	eine ganze Zahl
$\tilde{q}_\Phi(x)$	die Zugehörigkeitsfunktion der Multimenge der Nachfolger eines Knotens $\tilde{Q}(Phi)$
$Q(\Phi), \tilde{Q}(Phi)$	die Menge der Nachfolger eines Knotens Φ, als einfache Menge und als Multimenge
$R_{N_f^\tau}^\tau$	eine optimale Route für Fahrzeug f zur Bedienung aller Aufträge $i \in N_f\tau$
s_μ	Servicezeit: benötigte Zeit zum Be- bzw. Entladen im Ort μ
S	ein Sortierkriterium für die Reihenfolge der Fahrzeuge
t	Zeitpunkt
$\underline{t}_\mu, \overline{t}_\mu$	untere und obere Zeitfenstergrenze für die Ankunft im Ort μ
$\underline{T}_f, \overline{T}_f$	untere und obere Zeitfenstergrenze für die zukünftige Route von Fahrzeug f
$U(\mathcal{L})$	die Umgebung einer Lösung $\mathcal{L}$ im Verfahren der Tabu Suche
V	die Knotenmenge eines Graphen
$V_{n,k^*}^{B/L/Lq}, {}_kV_{n,k^*}^{B/L/Lq}$	die Knotenmenge des Zustandsgraphen mit Ladebedingung Backhauls, LIFO bzw. LIFO-q für n Aufträge mit Quelle auf Stufe k^* (auf Stufe k)
$V_{n,k^*}^S, {}_kV_{n,k^*}^S$	die Knotenmenge des Statusvektorbaums für n Aufträge mit Quelle auf Stufe k^* (auf Stufe k)
$V_{n,k^*}^Z, {}_kV_{n,k^*}^Z$	die Knotenmenge des Zustandsgraphen für n Aufträge mit Quelle auf Stufe k^* (auf Stufe k)
${}_kW_{n,k^*}$	Menge der möglichen Statusvektoren auf Stufe k
$V^{Orte}, V_{\Phi,\mu}^{Orte}$	die Knotenmenge des Graphen G^{Orte} bzw. $G_{\Phi,\mu}^{Orte}$
$x_{\mu,\nu}^f$	Binärvariable, die angibt, ob Fahrzeug f die Strecke zwischen den Orten μ und ν benutzt
x_k	Zustandsvariable auf Stufe k im Single Vehicle Routing-Modell

Variable	Bedeutung
X_k	Menge der Zustandsvariablen auf Stufe k im Single Vehicle Routing-Modell
y_k	Entscheidungsvariable auf Stufe k im Single Vehicle Routing-Modell
Y_k	Menge der Entscheidungsvariablen auf Stufe k im Single Vehicle Routing-Modell
z	ein künstlicher Endort der Fahrzeuge im linearen Modell
$Z(\Phi)$	die eineindeutige Repräsentation des Statusvektors im Zehnersystem

Index

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] A. S. Alfa, S. S. Heragu, and M. Chen. A 3-opt based simulated annealing algorithm for vehicle routing problems. *Computers & Industrial Engineering*, 21(4):635–639, 1991.

[2] K. Altinkemer and B. Gavish. Heuristics for unequal weight dekivery problems with a fixed error guarantee. *Operations Research Letters*, 6:149–158, September 1987.

[3] K. Altinkemer and B. Gavish. Parallel savings based heuristic for the delivery problem. *Operations Research*, 39:456–469, 1991.

[4] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387, March 2004.

[5] E. Baker and J. Schaffer. Computational experience with branch exchange heuristics for vehicle routing problems with time window constraints. *American Journal of Mathematical and Management Sciences*, 6:261–300, 1986.

[6] M. L. Balinski and R. E. Quandt. On an integer program for a delivery problem. *Operations Research*, 12:300–304, 1964.

[7] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 6 edition, 1957.

[8] D. Bertsimas and G. VanRyzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *OperationsResearch*, 39:601–615, 1991.

[9] J. Bramel and D. Simchi-Levi. A location based heuristic for general routing problems. *Operations Research*, 43(4):649–660, 1995.

[10] G. Brown and G. Graves. Real-time dispatch of petroleum tank trucks. *Management Science*, 27:19–32, 1981.

[11] M. Caramia, G. F. Italiano, G. Oriolo, A. Pacifici, and A. Perugia. Routing a fleet of vehicles for dynamic combined pickup and delivery services. In *Proceedings of the Symposium on Operation Research*, pages 3–8, Berlin/Heidelberg, 2001. GOR, Springer-Verlag.

[12] N. Christofides. Vehicle routing. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The Travelling Salesman Problem*, chapter 12, pages 431–448. John Wiley & Sons Ltd., 1985.

[13] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operations Research Quarterly*, 20(3):309–318, 1969.

[14] N. Christofides, A. Mingozzi, and P. Toth. exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1979.

[15] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, chapter 11, pages 315–338. Wiley, Chichester, 1979.

[16] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[17] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In F. Varela and V. Bourgine, editors, *Proceedings of the European Conference on Artificial Life*, Amsterdam, 1991. Elsevier.

[18] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. Vrp with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 7, pages 157–193. siam, 2002.

[19] G. B. Dantzig and J. Ramser. The truck dispatching problem. *Management science*, 6:80–91, 1959.

[20] U. Derigs and A. Metz. A matching-based approach for solving a delivery/pick-up vehicle routing problem with time constraints. *OR Spectrum*, 14(2):91–106, June 1992.

[21] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[22] G. Dueck. New optimization heuristics: The great deluge algorithm and the record-torecord -travel. *Journal of Computational Physics*, 104(1):86–92, 1993.

[23] G. Dueck and T. Scheurer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161–175, September 1990.

[24] Y. Dumas, J. Desrosiers, and F. Soumis. *Large scale multi-vehicle dial-a-ride problems*. cole des Hautes Etudes Commerciales, Montral, Canada, 1989.

[25] Y. Dumas, J. Desrosiers, and F. Soumis. The pick-up and delivery problem with time windows. *European Journal of Operational*, 54:7–22, 1991.

[26] A. Fabri. Dynamisches Pickup and Delivery Vehicle Routing mit mehreren Zeitfenstern. Master's thesis, Universität Dortmund, 10 2002.

[27] A. Fabri. Benchmarks für das DPDVRPTW. http://www.wiso.uni-dortmund.de/orwi/de/content/team/mitarbeiter/Benchmarks.html, 10 2005.

[28] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

[29] B. Fleischmann, S. Gnutzmann, and E. Sandvoß. Dynamic vehicle routing based on online traffic information. *Transportation Science*, 38(4):420–433, November 2004.

[30] A. Fraser and D. G. Burnell. *Computer models in genetics*. McGraw-Hill, New York, 1970.

[31] T. Gaskell. Bases for vehicle fleet scheduling. *Operational Research Quarterly*, 18:281–295, 1967.

[32] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Séguin. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390, 1999.

[33] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Séguin. Neighborhood search heuristics for a dynamic. *Transportation Research Part C*, 14:157–174, 2006.

[34] M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the capacitated vrp. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 6, pages 129–154. siam, 2002.

[35] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[36] F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1:190–206, 1989.

[37] F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2:4–32, 1990.

[38] B. Golden, E. Wasil, J. Kelly, and F. Chao. Metaheuristics in vehicle routing. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter pp.33-56. Kluwer, Boston, MA, 1998.

[39] B. L. Golden, T. L. Magnanti, and H. Nguyen. Implementing vehicle routing algorithms. *Networks*, 7:113–148, 1977.

[40] M. Haimovich and A. H. G. Rinnooy Kan. Bounds and heuristics for capacitated vehicle routing problems. *mathematics of operations research*, 10(4):527–542, november 1985.

[41] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions of systems science and cybernetics*, SSC-4 (2):100–107, 1968.

[42] J. Homberger. Neue große benchmarkprobleme für das standardproblem der tourenplanung mit zeitfensterrestriktionen. http://www.uni-duisburg.de/FB5/BWL/WI/or2000/sektion11/homberg.pdf, 2000.

[43] J.-J. Jaw, A. R. Odoni, H. N. Psaraftis, and N. H. M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B*, 20:243–257, 1986.

[44] H. Kawamura, M. Yamamoto, T. Mitamura, K. Suzuki, and A. Ohuchi. Cooperative search on pheromone communication for vehicle routing problems. *IEEE Transactions on Fundamentals*, E81-A:1089–1096, 1998.

[45] S. Kirkpatrick, C. D. J. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[46] K. W. Knight and J. P. Hofer. Vehicle scheduling with timed and connected calls: A case study. *Operational Research Quarterly*, 19(3):299–310, September 1968.

[47] D. E. Knuth. *The Art of Computer Programming*, volume 3 of *Four volumes*. Addison-Wesley, 1973. Seven volumes planned (this is a cross-referenced set of BOOKs).

[48] G. Kontoravdis and J. Bard. A grasp for the vehicle routing problem with time windows. *ORSJ4 Journal on Computing*, 7:10–23, 1995.

[49] A. Larsen. *The Dynamic Vehicle Routing Problem*. PhD thesis, Lingby University, ISSN 0909-3192, 2001.

[50] H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 12(2):173–186, 2003.

[51] S. Lin. Computer solutions of the travelling salesman problem. *Bell System Technical Journal*, 44:2245–2269, December 1965.

[52] S. Mitrovic-Minic, R. Krishnamurti, and G. Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research B*, 38:669–685, 2004.

[53] I. Or. *Traveling Salesman-type Combinatorial Problems and their relation to the Logistics of Blood Banking*. PhD thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 1976.

[54] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. 421-451 1-4, Ann. Oper. Res., 1993 41.

[55] A. Potini and G. Viola. Cabdispatcher: sistema software per l´instradamento di una flotta di veicoli nell´ambito di un servizio di trasporto collettivo. Master's thesis, Università di Roma ''Tor Vergata'', March 2002.

[56] J.-Y. Potvin, T. Kervahut, B. Garcia, and J.-M. Rousseau. The vehicle routing problem with time windows-part 1: Tabu search. *INFORMS Journal on Computing*, 8:158–164, 1996.

[57] J.-Y. Potvin and J.-M. Rousseau. An exchange heuristic for routing problems with time windows. *Journal of Operational Research Society*, 46(12):1433–1446, December 1995.

[58] H. Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14:130–154, 1980.

[59] H. Psaraftis. k-interchange procedures for local search in a precedence-constrained routing problem. *European Journal of Operations Research*, 13:391–402, 1983.

[60] H. N. Psaraftis. Dynamic vehicle routing problems. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing: Methods and Studies*, chapter pp.223-248. Elsevier Science, North-Holland, 1988.

[61] W. Pugh. skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, June 1990.

[62] H. G. M. Pullen and M. H. J. Webb. A computer application to a transport scheduling problem. *The Computer Journal*, 10(1):10–13, 1967.

[63] F. Robuste, C. F. Daganzo, and R. R. Souleyrette. Implementing vehicle routing models. *Transportation Research Part B: Methodological*, 24(4):263–286, August 1990.

[64] S. Roy, J. M. Rousseau, G. Lapalme, and J. A. Ferland. Routing and scheduling for the transportation of disabled persons–the tests. Report tp 5596e, Transport Development Center, Montréal, Canada, 1984.

[65] S. Roy, J. M. Rousseau, G. Lapalme, and J. A. Ferland. Routing and scheduling for the transportation of disabled persons-the algorithm. Report tp 5596e, Transport Development Center, Montréal, Canada, 1984.

[66] R. A. Russell. Am effective heuristic fpr the m-tour traveling salesman problem with some side conditions. *Operations Research*, 25:517–524, May-June 1977.

[67] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, February 1995.

[68] T. Sexton and L. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times. i: Scheduling. *Transportation Science*, 19:378–410, 1985.

[69] T. Sexton and L. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times. ii: Routing. *Transportation Science*, 19:369–398, 1985.

[70] M. Solomon, E. Baker, and J. Schaffer. Vehicle routing and scheduling problems with time window constraints: Efficient implementations of solution improvement procedures. In B. Golden and A. Assad, editors, *Vehicle Routing: Methods and studies*, chapter pp. 85-106. North-Holland, Amsterdam, 1988.

[71] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[72] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 03/04 1987.

[73] M. M. Solomon and J. Desrosier. Time window constrained routing and scheduling problems. *Transportation Science*, 22(1):1–13, February 1988.

[74] M. R. Swihart and J. D. Papastavrou. A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. *European journal of operational research*, 114(3):447–464, 1999.

[75] P. M. Thompson and H. N. Psaraftis. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research*, 41, issue 5:935–946, 1993.

[76] P. Toth and D. Vigo. Heuristic algorithms for the handicapped persons transportation problem. *Transportation science*, 31(1):60–71, 1997.

[77] P. Toth and D. Vigo. Vrp with backhauls. In *The Vehicle Routing Problem*, pages 195–224. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[78] M. S. Tyagi. A practical method for truck dispatching problems. *Journal of the Operational Research Society of Japan*, 10:76–92, 1968.

[79] A. Van Breedam. An analysis of the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem. Ruca working paper 96/14, University of Antwerp, Belgium, 1996.

[80] J. Willard. Vehicle routing using r-optimal tabu search. Master's thesis, The Management School, Imperial College, London, 1989.

[81] N. Wilson and N. Colvin. Computer control of the rochester dial-a-ride system. Technical Report Report R77-31, Dept. of Civil Engineering, MIT, 1977.

[82] N. Wilson, J. Sussman, H. Wang, and B. Higonett. Scheduling algorithms for dial-a-ride systems. Technical Report Urban Systems Laboratory Report USL TR-70-13, MIT, 1971.

[83] N. Wilson and H. Weissberg. Advanced dial-a-ride algorithms research project: Final report. Technical Report Report R76-20, Dept. of Civil Engineering, MIT, 1976.

[84] P. Yellow. A computational modification to the savings method of vehicle scheduling. *Operations Research Quarterly*, 21(2):281–283, 1970.