RWTH AACHEN UNIVERSITY

Lehrstuhl für Operations Research

Masterarbeit

Preprocessing-gestütztes Lösen des Vehicle Routing Problems mit Zeitfenstern mittels verschiedener IP-Formulierungen

Tackling the vehicle routing problem with time windows by using different, preprocessing supported IP-formulations

Vorgelegt dem Lehrstuhl für:

Operations Research

Vorgelegt von:

Alena Gridchina

Wirichsbongardstr. 65

52062 Aachen

Erstgutachter:

Prof. Dr. Marco Lübbecke

Zweitgutachter:

Prof. Dr. Hans-Jürgen Sebastian

Datum der Abgabe

30/09/2013

Inhaltsverzeichnis

1. Einleitung

Seit Jahrhunderten müssen Menschen sich mit Nahrung, Gütern und diversen weiteren lebensnotwendigen Dingen versorgen. Diese Versorgungsprozesse konnten seit der Erfindung des Automobils und des Flugzeugs in vielerlei Hinsicht vereinfacht werden: Güter werden weltweit transportiert, Flugzeuge befördern Menschen von einem Kontinent zum anderen, die öffentlichen Transportmittel befördern täglich Millionen von Menschen.[1] Allerdings ist die Aufgabe der Logistik dadurch komplexer geworden: Planung, Organisation und Koordination der logistischen Leistungen erfordert die Einhaltung vieler Anforderungen, die mittlerweile nur mit Hilfe der Mathematik bewältigt werden können.[2]

Laut eines Berichtes der Europäischen Union[3] ließ das Wachstum der Logistik sie in den letzten Jahrzehnten zu einem der bedeutendsten Wirtschaftsfaktoren werden. So belief sich in Deutschland (im Jahr 2007) die Zahl der Beschäftigten im Transportbereich (sowohl Luft-, See- als auch Straßentransport) auf ca. 1.4 Millionen. Im selben Jahr waren 87.500 Unternehmen im Transportsektor verzeichnet, deren Gesamtumsatz mehr als 218 Milliarden Euro betrug. In 2011 lag das Marktvolumen bereits bei 223 Mrd. Euro.[4] In den Jahren 2000 bis 2008 ist das Frachttransport-Volumen, gemessen in Tonnekilometer,[5] jährlich um 2% gestiegen. Dieses Wachstum resultiert teilweise aus den steigenden Kundenanforderungen, dem wachsenden Online-Versandgeschäft und der sich stetig verbessernden Produktionstechnik.

Basierend auf den immer stärker verwobenen Logistikpfaden, und deren wachsenden Verknüpfungen, stehen viele Unternehmen vor der Herausforderung, ihre Geschäftsprozesse möglichst gut aufeinander abzustimmen.[6] Die Logistik nimmt dabei einen wichtigen Stellenwert ein, da sie für die notwendige Mobilität von Personen und

[1] Vgl. [2]
[2] Vgl. [51]
[3] Der komplette Absatz bezieht sich auf den Bericht [2]. Andere Quellen werden entsprechend kenntlich gemacht.
[4] Vgl. Studie des Fraunhofer Instituts: Supply Chain Services [28].
[5] Maßeinheit in der Logistik für die Beförderungsleistung von Gütern, welche dem transportierten Gewicht von einer Tonne über ein Kilometer entspricht (*kurz*: tkm).
[6] Eine optimal abgestimmte Logistik bildet die Grundlage für eine effiziente Umsetzung einer kompletten Supply Chain [50].

Gütern sorgt. Die mit Optimierung von Logistikprozessen verbundene Einsparung an Ressourcen verursacht nicht nur eine Senkung der Logistikkosten,[7] sondern auch eine Senkung der Umweltbelastung.[8] Zur Optimierung der logistischen Leistung werden Optimierungsmethoden aus dem Operations Research[9] genutzt. Das Operations Research kennt diverse Standardprobleme, welche sich auf logistische Herausforderungen der Realität subsumieren lassen. Eines der gängigsten dieser Standardprobleme ist das Vehicle Routing Problem. Ist Selbiges erweitert um die Notwendigkeit des Einhaltens von Zeitfenstern, so spricht man vom Vehicle Routing Problem mit Zeitfenstern (*engl.*: Vehicle Routing Problem with Time Windows, *kurz*: VRPTW). Jedoch ist VRPTW aus komplexitätstheoretischer Sicht sehr anspruchsvoll, weswegen moderne Lösungsverfahren schnell an ihre Grenzen stoßen.

In dieser Masterarbeit werden zunächst ausgewählte, (Mixed-)Integer-Programming-basierte Modellformulierungen[10] des VRPTW vorgestellt und erläutert. Anschließend wird insbesondere auf das Problem der so genannten Symmetrie eingegangen, das im Rahmen des Branch-and-Bound-Verfahrens, welches zur Lösung der (M)IP-Modellformulierungen verwendet wird, auftreten kann. Hiernach werden Preprocessing-Techniken als Bausteine der Umsetzung eines Lösungsverfahrens für das VRPTW eingeführt und veranschaulicht. Konkret handelt es sich hierbei um diverse Maßnahmen zur Reduzierung der Modellgröße. Schließlich werden, im Rahmen eines Experimentalteils, die beschriebenen Modellformulierungen in Verbindung mit diversen Preprocessing-Techniken implementiert und sodann zum Lösen diverser VRPTW-Instanzen von M. Solomon verwendet. Insbesondere sollen die Experimente einen tieferen Einblick in die Bedeutung der verschiedenen Preprocessing-Elemente bei der Unterstützung des Lösungsprozesses gewähren.

Abgeschlossen wird die Arbeit mit einer Zusammenfassung der wichtigsten Erkenntnisse, sowie einer kritischen Würdigung der verwendeten Ansätze und einem Ausblick.

[7] Vgl. [5]
[8] Vgl. [48]
[9] Eine Einführung in Operations Research bietet [25].
[10] Eine Einführung in (Mixed-)Integer-Programming bietet [61].

2. VRPTW

Aufgrund seiner Vielseitigkeit wird VRPTW in vielen Branchen angewendet und unterstützt dadurch diverse logistische Abläufe des alltäglichen Lebens.[1] Vor dem Hintergrund seiner Komplexität und der Möglichkeit einer Unterstützung des Lösungsvorgangs mittels diverser Preprocessing-Techniken, besteht die Motivation dieser Masterarbeit darin, diese Techniken im Zusammenhang mit verschiedenen Modellen der (Gemischt-)Ganzzahligen-Programmierung (*engl.*: (Mixed-)Intger-Programming, *kurz*: (M)IP) für das VRPTW zu untersuchen.

In den folgenden Abschnitten werden das VRPTW definiert und die Nomenklatur, die im weiteren Verlauf dieser Arbeit verwendet wird, eingeführt. Im Abschnitt „Praxisrelevanz“ wird der Bezug zu realen Anwendungsmöglichkeiten des VRPTW anhand von Praxisbeispielen hergestellt. Daraufhin werden ein Überblick über ausgewählte Veröffentlichungen zum Thema VRPTW gegeben und die entsprechenden Lösungsverfahren genannt. Abschließend werden, auf Basis des zuvor definierten Problems, drei (M)IP-basierte Modellformulierungen vorgestellt und miteinander verglichen.

2.1. Definition

In dieser Arbeit wird das Vehicle Routing Problem mit Zeitfenstern, erweitert um Kapazitätsrestriktionen, betrachtet.[2] Dieser Abschnitt orientiert sich bei der verbalen und formalen Definition an den Ausführungen und Notationen von Toth, Vigo [62] und Kohl [41]. Teilweise wurde auch eigene, von den Vorlagen abweichende Nomenklatur verwendet.

Verbal kann das Vehicle Routing Problem mit Zeitfenstern folgendermaßen beschrieben werden: Von einem Depot sollen Kunden mit einer Flotte von (identischen) Fahrzeugen begrenzter Kapazität beliefert werden. Jeder Kunde hat eine Nachfrage

[1] Einen Überblick über den VRPTW-Einsatz in der Praxis ist [26] zu entnehmen.

[2] Da das VRP als Grundproblem Kapazitätsrestriktionen enthält, werden in einem Großteil der VRPTW-Literatur (insbesondere jener, welcher dieser Arbeit zugrunde liegt, siehe [22], [43]) ebenfalls Kapazitäten impliziert, ohne dass dabei ausdrücklich vom Capacitated Vehicle Routing Problem (CVRPTW) die Rede ist. Diese Lesart soll im Folgenden übernommen werden.

nach Gütern, die von genau einem Fahrzeug befriedigt werden muss. Dabei darf die Gesamtnachfrage entlang einer Tour die Kapazität des entsprechenden Fahrzeugs (wobei alle Fahrzeuge gleiche Kapazität haben) nicht überschreiten. Die Befriedigung der Kundennachfragen benötigt jeweils eine Servicezeit. Der Beginn dieser Servicezeiten darf nur innerhalb begrenzter Zeitfenster der Kunden liegen (prinzipiell erreicht werden kann ein Kunde jedoch beliebig früh, dann müssen die Fahrzeuge allerdings warten, bis das entsprechende Zeitfenster geöffnet ist). Ein Zeitfenster ist dabei durch seine Start- und Endzeit definiert. Alle eingesetzten Fahrzeuge müssen im Depot starten und nachdem sie die Kunden auf ihrer Tour bedient haben, wieder ins Depot zurückkehren. Das Ziel des VRPTW besteht in der Regel darin, eine, bezogen auf die Gesamtreisezeit, kostenminimale Tourenzuweisung für die Fahrzeugflotte zu bestimmen.[3]

Formal kann das beschriebene Problem wie folgt definiert werden: Gegeben sei ein gerichteter Graph $G = (V,A)$ mit der Knotenmenge $V = \{0,\ldots,n\}$ und der Kantenmenge A. Das Depot ist durch den ersten Knoten $0 \in V$ repräsentiert.[4] Alle anderen Knoten $i \in V_c$, wobei $V_c = V \setminus \{0\}$, stellen die Kundenstandorte dar. Jeder Kunde i hat eine diskrete Nachfrage q_i, die die Gesamtkapazität C eines einzelnen Fahrzeugs nicht überschreiten darf. Gegeben sei ferner die Distanzmatrix $D_{i,j}$, welche die Distanzen der Städte untereinander enthält. Überdies existiert ein Parameter $t_{i,j}$, welcher als "virtuelle" Reisezeit, die für das Überqueren einer Kante $(i,j) \in A$ inklusive der Servicezeit s_i am Knoten i benötigt wird, interpretiert ist.[5] Vom Depot werden die Kunden mit einer Menge identischer Fahrzeuge $K = \{1,\ldots,m\}$ beliefert, dabei ist die Fahrzeugflotte auf m Fahrzeuge beschränkt, wobei alle Fahrzeuge über die gleiche Kapazität C verfügen. Zusätzlich müssen die Zeitfenster $[R_i, D_i]$, beschrieben durch den frühestmöglichen Zeitpunkt R_i (*engl.*: Release Time) des Lieferbeginns und den spätestmöglichen Zeitpunkt D_i (*engl.*: Deadline Time) des Lieferbeginns eingehalten werden. Das Zeitfenster W_i eines Knotens i ist genau dann eingehalten, wenn $R_i \leq t_i \leq D_i$, wobei t_i den Beginn der Lieferzeit bei Knoten i darstellt. Das Ziel ist, in der Regel, die Minimierung der Gesamtreisekosten $\sum_{(i,j)\in T}$, mit T als Menge aller Kanten, welche im endgültigen Tourenplan verwendet werden.

[3] Für eine Übersicht über verschiedene VRPTW-Zielfunktionen siehe beispielsweise [26].

[4] In vielen Publikationen wie beispielsweise [16] und [38] findet man eine Erweiterung der Knotenmenge von $V = \{0,\ldots,n\}$ auf $V = \{0,\ldots,n,n+1\}$. Der Knoten 0 ist aus Gründen der günstigeren Handhabung in Start-($\{0\}$) und Enddepot ($\{n+1\}$) aufgeteilt.

[5] Die „Nettoreisezeit" (ohne Servicezeit s_i) ergibt sich aus der Distanzmatrix $D_{i,j}$, da angenommen wird, dass sich die Distanz mit dem Faktor 1 in die benötigte Reisezeit ohne Servicezeit umrechnen lässt.

2.2. Praxisrelevanz

Die Anzahl der VRP-Varianten und die Menge an wissenschaftlichen Publikationen[6] zu selbigen sind Indikatoren für die Bedeutung, die dieser Problemstruktur in der Praxis zukommt. VRP-Variationen, und hierunter insbesondere das VRPTW, werden in den verschiedensten Branchen angewendet und unterstützen viele Prozesse im alltäglichen Leben. Im Folgenden seien einige Beispiele für den erfolgreichen Einsatz der VRPTW-basierten Verfahren in der Praxis genannt.

Créput et al. [17] untersuchten eine Einsatzmöglichkeit des VRPTW im medizinischen Notfallmanagement. Das VRPTW wird im Rahmen dessen auf das dynamische VRPTW (DVRPTW) erweitert, um den Aspekt der Wechselhaftigkeit in auftretenden medizinischen Notrufen und die Flexibilität der Einsatzfahrzeuge zu berücksichtigen. Mithilfe dieses Ansatzes können, nach einem Anruf eines Patienten mit bestimmten Beschwerden, die Ärzte mit der richtigen Ausrüstung geortet und zeitgerecht zum Patienten geschickt werden.

In einem weiteren Beispiel konnte die Firma Waste Management Inc., ein Entsorgungsunternehmen in Nordamerika, in 2003 ein von Kim et al. [39] entwickeltes Verfahren zur Routenplanung der Müllabfuhr-Fahrzeuge einsetzen. Mit dem VRPTW-basierten Ansatz konnte das Unternehmen seine Produktivität verbessern und in 2004 die Kosten um geschätzte 44 Millionen US-Dollar senken. Ebenso konnte ein Lebensmittelhändler in Portugal die Anzahl der gefahrenen Kilometer um 150.000 Kilometer pro Jahr reduzieren und gleichzeitig die Kapazitätsauslastung um 16% erhöhen.[7]

Auch in der Luftfahrtindustrie wird das VRPTW angewendet. Beispielsweise konnten Azadian et al. [7] die Verspätungen von Zeit-empfindlichen Cargos mit Berücksichtigung von Dynamiken in Flugplänen deutlich verringern. Neben einer Verbesserung der Lieferfähigkeit konnten auch die Tourenkosten erheblich gesenkt werden. Ebenfalls in der Schifffahrtsindustrie ließen sich mit VRP-basierten Lösungsansätzen beachtliche Resultate erzielen. Br∅nmo et al. [14] stellen einen VRPTW-basierten Lösungsansatz für flexible und kurzfristige Cargos vor, der auf zwei verschiedene Fälle aus der Schifffahrtsindustrie angewendet wurde. Die Untersuchung ergab, dass besonders in Zeitperioden mit volatiler Nachfrage flexible Cargos die Schiffsladungen deutlich besser auslasten und den Unternehmensprofit steigern können.

[6] Eine Suche nach „Vehicle routing with time windows“ bei www.siencedirect.com ergibt eine siebenfache Zunahme an Publikationen zu dieser Thematik seit dem Jahr 2000 bis heute.
[7] Vgl. [4]

Seit dem progressiven Wachstum des Online-Handels, mit einem jährlichen Wachstum von ca. 8%[8] ist die Tourenplanung überdies besonders für Brief- und Paketzusteller, wie die Deutsche Post AG, immer wichtiger geworden. Mit über 1,7 Millionen Sendungen in 2011 allein von der Deutschen Post ist der Versandhandel seit dem Jahr 2000 um fast 50% gewachsen, Tendenz steigend.[9] Durch seine Berücksichtigung zeitlicher Aspekte eignet sich das VRPTW für diese Art von Problemen besonders gut.[10] Eine ähnliche Problematik stellt der E+1 Service von DHL dar, welcher die Pakete für Strecken kürzer als 500 km spätestens einen Tag nach dem Eingang des Pakets in der Eingangs-Produktionsniederlassung zu liefern versucht.[11]

Die Fahrzeuge im Problem des VRPTW müssen nicht zwingend als Fahrzeuge interpretiert werden. Betrachtet man eine Maschine und mehrere Aufträge, die auf dieser Maschine erledigt werden müssen, so kann eine „Tour“ bestehend aus “Kunden“, die von einem „Fahrzeug“ bedient werden, auch als die Zuordnung von Aufträgen auf eine Maschine interpretiert werden. Die „Distanzen“ zwischen den „Kunden“ können als Umrüstkosten von einem Auftrag auf der Maschine zum nächsten aufgefasst werden.[12] Auf diese Weise kann VRPTW auch auf andere Problematiken subsumiert werden.

Generell gewinnt eine effiziente Tourenplanung nicht nur im Hinblick auf vielversprechende Kosteneinsparpotentiale, sondern auch durch sich verändernde politische und ökonomische Anforderungen, mehr an Relevanz denn je: Laut BP p.l.c., einem Energiekonzern, sind die Preise für Rohöl und Kraftstoff allein seit dem Jahr 2000 um das Vierfache gestiegen.[13] Zusätzlich stellen verschärfte Gesetze zum CO_2-Ausstoß viele produzierende Unternehmen vor eine schwierige Aufgabe, diese Gesetze einzuhalten.[14] In Anbetracht dieser Entwicklungen nehmen VRP-basierte Ansätze in der Unternehmenslogistik einen hohen Stellenwert ein und bilden eine Grundlage für diverse moderne Tourenplanungs-Software-Produkte wie z.B. Map & Guide von der Planung Transport Verkehr AG.[15]

[8] Vgl. hierzu die Studie von PricewaterhouseCoopers [52].
[9] Nach den eigenen Angaben der Deutschen Post [33].
[10] Meistens werden nicht nur „reine“ VRPTW-Varianten, sondern auch problemspezifische Erweiterungskomponenten der selbigen eingesetzt, wie beispielsweise Pickup-and-Delivery [10], [27] oder das Location Routing Problem [59].
[11] Laut eigenen Angaben von DHL [24].
[12] Für die Routing-Scheduling-Kombination sei auf [61] verwiesen.
[13] Historische Dokumentation der Ölpreise, entnommen von [11].
[14] Vgl. [1]
[15] Vgl. [53]

Auch „reverse“ und grüne Logistik werden zunehmend wichtig. Unter „reverse“ Logistik werden rückläufige Prozesse wie zum Beispiel Entsorgung und Retouren in einer Lieferkette verstanden. Beispielsweise versuchen Autohersteller aufgrund der steigenden Rohstoffpreise aus verschrotteten Autos wertvolle Rohstoffe wieder zu gewinnen und in den Produktionsprozessen einzusetzen.[16] Grüne Logistik stellt ihrerseits vielmehr ökologische Ziele wie Minimierung der Emissionen in den Vordergrund.[17]

2.3. Literaturüberblick

Das Vehicle Routing Problem mit Zeitfenstern gehört zu einem der, in der Wissenschaft und Praxis, meist untersuchten Problemen aus dem Bereich des Operations Research. Aus dem von Danzig et al. erstmals eingeführten Vehicle Routing Problem entwickelten sich zahlreiche Varianten,[18] u.a. das VRPTW, als eine Erweiterung des VRP. Selbiges wird seitdem in Theorie und Praxis intensiv untersucht. Es zählt zu der Klasse der $\mathcal{NP}$-schweren Probleme. Ist ein Problem $\mathcal{NP}$-schwer, so bedeutet dies unter anderem, dass nach dem heutigen Wissensstand kein Algorithmus existiert, der das Problem mit polynomiellen Zeitaufwand lösen kann.[19] Savelsbergh [55] beweist, dass das Travelling Salesman Problem mit Zeitfenstern (*engl.:* Travelling Salesman Problem with Time Windows, *kurz:* TSPTW) $\mathcal{NP}$-schwer ist. VRPTW, als eine Erweiterung des TSPTW,[20] ist demnach auch $\mathcal{NP}$-schwer. Sogar das Finden einer zulässigen Lösung bei einer gegebenen Anzahl an Fahrzeugen erweist sich als $\mathcal{NP}$-schwer, wie Kohl [40] zeigen konnte.

Preprocessing, als eine Technik zur Reduzierung der Instanzkomplexität, ist ein wichtiger Bestandteil der VRPTW-Modellierung. In der allgemeinen Literatur zu VRPTW ist es jedoch nicht stark repräsentiert. Preprocessing-Techniken für das VRPTW wurden zum ersten Mal von Desrochers [21] in 1992 vorgestellt. In der darauf folgenden Zeit setzten sich unter anderen Larsen [45] und Kallehauge et al. [38] ausführlicher mit dem Thema Preprocessing im Kontext des VRPTW auseinander. In den meisten Fällen wird Preprocessing jedoch nur in äußerster Kürze behandelt.[21]

[16] Vgl. [34]
[17] Eine tiefgehende Studie zu dieser Thematik wurde von Lin et al. [48] vorgestellt.
[18] Einen umfassenden Überblick über das VRP und seine Varianten bietet Drexl et al. [26].
[19] Für mehr Information zur Komplexität und $\mathcal{NP}$-Schwere sei auf [15], [46] und [60] verwiesen.
[20] Tatsächlich stellt das TSPTW einen Sonderfall des VRPTW dar, in welchem lediglich ein Fahrzeug zur Verfügung steht, das unbegrenzte Kapazität aufweist.
[21] Vlg. [41] und [47]

Zur Lösung des VRPTW stehen zahlreiche Verfahren und Methoden zur Verfügung. Abbildung 2.1 verschafft einen Überblick über eine Reihe der gängigsten Verfahren und ist lediglich eine grobe Klassifikation der verschiedenen Ansätze.

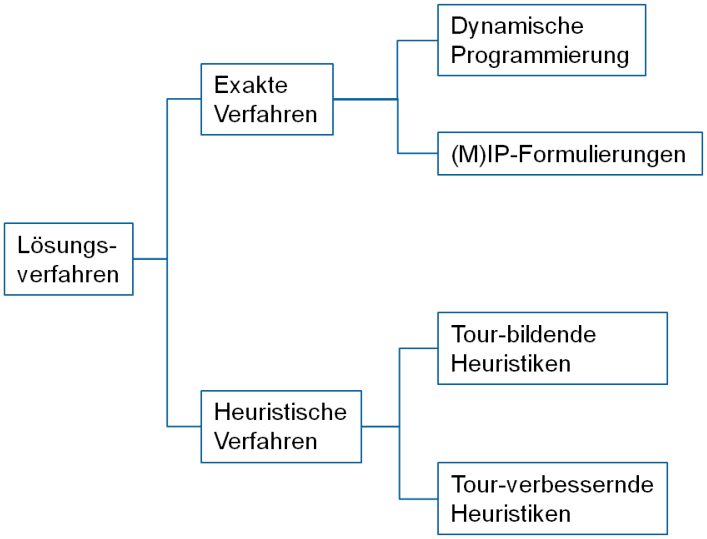


Abbildung 2.1.: Klassifizierung von Lösungsverfahren für das VRPTW, angelehnt an [44] und [45]

Demnach können Lösungsverfahren in zwei Hauptkategorien unterteilt werden: In heuristische und exakte Lösungsverfahren. Heuristische Verfahren versprechen nicht zwingend eine optimale, dafür aber eine vergleichsweise schnelle Lösung. Eine systematische Unterteilung der Heuristischen stellt eine nicht eindeutig lösbare Aufgabe dar. Larsen [45] teilt beispielsweise Heuristiken in zwei Kategorien ein: Tour-bildende und Tour-verbessernde Heuristiken. Zu der ersten Kategorie gehören Heuristiken, die Touren schrittweise aufbauen. Hierzu zählen Heuristiken wie wie zum Beispiel die “Savings“- und und “Insertion“-Heuristiken.[22] Zu der zweiten Kategorie gehören Heuristiken, die schon gegebene Touren schrittweise verbessern, wie “Local-Search“-, “Simmulated-Annealing“- und “Genetic-Algorithm“-Heuristiken.[23] Die Letzteren gehören zu den sogenannten Meta-Heuristiken, die lokal eine schlechtere Lösung erlauben, um global eine bessere Lösung zu finden.[24] Es ist schwierig zu beurteilen, welche aus der enormen Auswahl an Heuristiken die beste ist, da sie anhand verschiedener Kriterien bewertet werden können: Effektivität, Schnelligkeit der Lösung oder Flexibilität.[25] Zunehmend versucht man Heuristiken und exakte Verfahren zu

[22] Ausführliche Beschreibungen dieser Heuristiken sind in [45] zu finden.
[23] Eine umfassende Studie zu diesen Heuristiken ist in [13] zu finden.
[24] Eine Abgrenzung zwischen Heuristiken und Meta-Heuristiken bietet [61].
[25] Ein Vergleich verschiedener Heuristiken ist [31] zu entnehmen.

kombinieren, um die Vorteile beider Ansätze auszunutzen.[26]

Im Vergleich zu den heuristischen Verfahren haben exakte Verfahren zum Ziel, das besagte Problem optimal zu lösen, falls eine optimale Lösung vorhanden ist. Diese Verfahren können aufgrund der $\mathcal{NP}$-Schwere des VRPTW lediglich Instanzen in sehr beschränkter Größenordnung in praktikabler Rechenzeit optimal lösen. Die Anzahl der exakt gelösten Instanzen steigt jedoch jährlich sowohl aufgrund der fortschreitenden technischen Möglichkeiten, als auch aufgrund verbesserter Algorithmen. So konnte „Quintiq“, ein Unternehmen für Optimierungssoftware, einige der Instanzen von Homberger mit 1000 Städten optimal lösen und damit einen Weltrekord auf diesen Instanzen aufstellen.[27]

Exkate Verfahren können unter anderem mittels “Column Generation“ oder “Dynamischer Programmierung“ gelöst werden.

Viele der exakten Verfahren basieren auf der Set-Partitioning-Formulierung.[28] Bramel und Simchi-Levi [12] haben gezeigt, dass die Set-Partitioning-Formulierung eine „starke“ Formulierung des VRPTW darstellt, die „gute“ duale Schranken liefert. In den meisten Fällen wird diese Art von Formulierungen mit dem Column-Generation-Verfahren gelöst. Der Grundgedanke von Column Generation besteht darin, dass bei Optimierungsproblemen mit „vielen“ Variablen das Problem zunächst mit „wenigen“ Variablen gelöst wird und erst dann weitere nötige Variablen erzeugt und dem Problem hinzugefügt werden.[29] In der Regel wird Column Generation in Verbund mit anderen Techniken angewendet. So entsteht beispielsweise aus Column Generation und dem Branch-and-Bound-Verfahren das Branch-and-Price- oder Branch-Cut-and-Price-Verfahren.[30] Laut Baldacci et al. [8] zählt Column Generation zu den erfolgreichsten Methoden für das VRPTW. Mit ihrem Algorithmus erzielen sie bessere Ergebnisse als Desaulniers et al. [20] und Jepsen et al. [36], die bisher einige der besten Lösungsergebnisse aufweisen konnten, und lösen noch nie vorher gelöste Instanzen von M. Solomon.[31]

In der Dynamischen Programmierung, ähnlich wie bei Column Generation, wird das zu lösende Optimierungsproblem ebenfalls in Subprobleme zerlegt. Man versucht die Subprobleme optimal zu lösen, sodass diese später zu einem ganzen, gelösten

[26] Hierzu sei auf [37] verwiesen. Ein Vergleich zwischen Heuristiken komplexerer Natur ist in [31] zu finden.
[27] Laut eigenen Angaben der Webseite von Quintiq Inc. [54].
[28] Vgl. hierzu beispielsweise [36] oder [45]
[29] Column Genearation wird in [23] und [58] ausführlich behandelt.
[30] Vlg. [45]
[31] Vlg. hierzu [8]

Problem zusammengesetzt werden können.[32]

Viele exakte Lösungsverfahren zum VRPTW basieren auf der (Gemischt-)Ganzzahligen-Programmierung. (M)IPs können mittels Branch-and-Bound (*engl.*: Verzweigung und Beschränkung, *kurz:* B&B) gelöst werden. Im folgenden Kapitel werden ausgewählte (M)IP-Formulierungen vorgestellt.

2.4. (M)IP-Formulierungen des VRPTW

(M)IP-Formulierungen stellen eine Erweiterung von IP-Formulierungen dar, und zeichnen sich dadurch aus, dass einige der verwendeten Entscheidungsvariablen stetig, und einige ganzzahlig sind.[33] Für das VRPTW existiert eine Vielzahl an Formulierungsmöglichkeiten. Drei der gängigsten (M)IP-basierten Formulierungen zum optimalen Lösen des VRPTW und Erläuterungen derselbigen bietet dieser Abschnitt. Zusätzlich wird auf die Unterschiede zwischen den verschiedenen Formulierungen eingegangen. Es werden Grundnotationen aus Abschnitt 2.1 verwendet. Überdies notiere für eine Knotenmenge $S \subset V$ der Operator $\delta^+(S)$ die Menge aller Kanten, welche aus S herausführen. Es ist also $\delta^+(S) = \{(i,j) \in A : i \in S, j \in V \setminus S\}$. Besteht $S = \{i\}$ aus lediglich einem Knoten $i \in V$, so sei vereinfachend statt $\delta^+(\{i\})$ auch $\delta^+(i)$ zu schreiben. Entsprechend notiere $\delta^-(S)$ die Menge aller Kanten, welche in S hineinführen. Es ist also $\delta^-(S) = \{(j,i) \in A : j \in V \setminus S, i \in V\}$. Auch hier gelte eine vereinfachende Notationsmöglichkeit von $\delta^-(\{i\})$ als $\delta^-(i)$.

Überdies wird das Depot in ein virtuelles Startdepot $\{0\}$ und virtuelles Enddepot $\{n+1\}$ aufgeteilt, wobei aus $\{0\}$ sämtliche Kanten herausführen, welche aus dem Ursprungsdepot herausführten und in $\{n+1\}$ sämtliche Kanten hineinführen, welche in das Ursprungsdepot hineinführten. Zudem existiert die direkte Verbindungskante $(0, n+1)$ mit den Verbindungskosten $c_{0,n+1} = 0$. M ist eine hinreichend groß gewählte Konstante.

2.4.1. 3-Index-Formulierung

Wie aus Abschnitt 2.1 bekannt, muss jeder Kunde von einem Fahrzeug bedient werden. Die 3-Index-Formulierung impliziert eine Unterscheidung der Fahrzeuge, sodass in der Endlösung genau festgestellt werden kann, welches Fahrzeug zu welchen

[32] Vgl. [42]
[33] Für mehr Informationen zu (M)IP sei auf [25] verwiesen.

Kunden fährt.[34] Hierzu wird ein ausdrücklicher Laufindex $k \in K$[35] zur Modellierung der verschiedenen Fahrzeuge verwendet. Im Rahmen dessen legt die Binärvariable $x_{i,j}^k$ fest, ob das Fahrzeug $k \in K$ Knoten $i \in V$ bedient und direkt von Knoten $i \in V$ nach Knoten $j \in V$ fährt. Variable t_i^k gibt den Beginn der Lieferzeit von Fahrzeug $k \in K$ bei Knoten i an. Gegeben diese Variablen, wird die 3-Index-Formulierung, angelehnt an Kohl et al. [41], wie folgt formuliert:

$$\text{minimiere} \sum_{(i,j)\in A} \sum_{k\in K} c_{i,j} x_{i,j}^k \tag{2.1}$$

s.t.

$$\sum_{j\in V:(i,j)\in A} \sum_{k\in K} x_{i,j}^k = 1 \qquad \forall i \in V_c \tag{2.2}$$

$$\sum_{j\in V} x_{0,j}^k = 1 \qquad \forall k \in K \tag{2.3}$$

$$\sum_{i\in V} x_{i,n+1}^k = 1 \qquad \forall k \in K \tag{2.4}$$

$$\sum_{i\in V_c} q_i \sum_{j\in V:(i,j)\in A} x_{i,j}^k \leq C \qquad \forall k \in K \tag{2.5}$$

$$R_i \leq t_i^k \leq D_i \qquad \forall i \in V;\ k \in K \tag{2.6}$$

$$\sum_{i\in V:(i,h)\in A} x_{i,h}^k - \sum_{j\in V:(h,j)\in A} x_{h,j}^k = 0 \qquad \forall h \in V_c;\ k \in K \tag{2.7}$$

$$t_i^k + t_{i,j} \leq t_j^k + (1 - x_{i,j}^k)M \qquad \forall (i,j) \in A;\ k \in K \tag{2.8}$$

$$x_{i,j}^k \in \{0,1\} \qquad \forall (i,j) \in A;\ k \in K \tag{2.9}$$

$$t_i^k \in \mathbb{Z}^{\geq 0} \qquad \forall i \in V;\ k \in K \tag{2.10}$$

Die Zielfunktion (2.1) ist selbsterklärend. Gleichung (2.2) fordert, dass jeder Kunde $i \in V_c$ von genau einem Fahrzeug $k \in K$ genau ein mal in Richtung eines anderen Kunden $j \in V_c$ oder des Zieldepots $\{n+1\}$ verlassen wird. Die Gleichungen (2.3) und (2.4) stellen sicher, dass jedes Fahrzeug $k \in K$ genau ein mal das Startdepot $\{0\}$ verlässt und genau ein mal das Zieldepot $\{n+1\}$ erreicht. Fahrzeuge, die in der endgültigen Lösung keine Verwendung finden, fahren hierbei eine virtuelle „Dummy-Tour“ vom Startdepot direkt ins Zieldepot, welche Zielfunktions-neutral ist. Die Kapazitäten der Fahrzeuge werden in Gleichung (2.5) eingehalten, indem

[34] Im Gegensatz zum Fall der 2-Index-Formulierung, welche in Abschnitt 2.4.2 vorgestellt werden wird.

[35] Die Verwendung des selbigen führt zu dem Problem der so genannten Symmetrie, welches im Abschnitt 3.2 erläutert wird.

für jedes Fahrzeug $k \in K$ die kumulierte Nachfrage aller Kunden entlang seiner Tour durch C nach oben beschränkt wird. Gleichung (2.6) stellt sicher, dass der Beginn der Bedienzeit bei Kunden $i \in V_c$ durch jedes etwaige Fahrzeug $k \in K$ im Zeitfenster des Knotens i liegt. Der Flusserhalt wird in Gleichung (2.7) gefordert, indem sie verlangt, dass jedes Fahrzeug $k \in K$, welches einen Kunden $h \in V_c$ erreicht, selbigen auch wieder verlässt. Gleichung (2.8) indes stellt sicher, dass für jedes potentielle Bedienfahrzeug $k \in K$ der Beginn der Bedienzeit beim Folgeknoten j vom Vorgängerknoten i nicht früher stattfinden kann, als der Beginn der Bedienzeit bei i, zuzüglich ihrer Dauer, sowie der benötigten Reisezeit von i nach j. Die Gleichung für die Kante $(i,j) \in A$ und das Fahrzeug $k \in K$ greift genau dann, wenn die Variable $x_{i,j}^k$ den Wert 1 annimmt. Ansonsten wird sie mittels der „Big M"-Konstruktion redundant. Insbesondere impliziert die beschriebene Konstellation auch die Möglichkeit des Entstehens von Wartezeiten, falls ein Knoten vor dem Beginn seines Zeitfensters erreicht wird. Die Gleichungen (2.9) und (2.10) definieren den Wertebereich der entsprechenden Variablen.

Zudem sind sämtliche Lösungen, welche den obigen Restriktionen genügen, zwingend frei von unzulässigen Subtouren, also solchen, welche nicht den Depotknoten enthalten: Aufgrund von Ungleichung (2.8) sind die Bedienzeitpunkte entlang einer Fahrzeugroute streng monoton steigend (da $t_{i,j} > 0 \ \forall (i,j) \in A$), somit vermag sich eine gültige Tour nur im Depot zu schließen, da der Depotknoten von Gleichung (2.8) ausgenommen ist und hier daher keine „Bedienzeitenkollision" resultieren kann. Für jede potentielle Subtour, welche nicht den Depotknoten enthält, würde hingegen ein Widerspruch folgen, da hier ein Knoten in der besagten Subtour existieren müsste, der zugleich den frühesten und den spätesten Bedienzeitpunkt aufweist.

Im Rahmen jeder Summation und jeder Restriktion wird, betreffend etwaige Tuppel $\{i,j\}$, zusätzlich die Existenz der entsprechenden Kante $(i,j) \in A$ sichergestellt,[36] da sich hierdurch, bei einer vorab erfolgten Verwendung von Preprocessing, die Modellgröße im Hinblick auf Spalten- und Zeilenanzahl verringern lässt.

2.4.2. 2-Index-Formulierungen

Gegeben die Kapazität der Fahrzeuge ist homogen, besteht kein Grund für die Unterscheidung der Fahrzeuge, und eine damit verbundene Problematik der sogenannten "Symmetrie", wie sie in Abschnitt 3.2 noch vorgestellt wird. Diese Erkenntnis wird in der im Folgenden beschriebenen 2-Index-Formulierungen ausgenutzt. Durch das Weglassen des Indexes k reduziert sich die Anzahl benötigter Variablen zunächst um den Faktor $|K|$, jedoch werden hierdurch auch neue, zusätzliche Entscheidungsva-

[36] Hierbei handelt es sich um eigene Ergänzung der Modell-Formulierung.

riablen notwendig. Die folgende 2-Index-Formulierung basiert auf der Formulierung von Letchford,[37] ist jedoch im Rahmen dieser Arbeit, mittels zwei zusätzlicher Restriktionen, erweitert um das Element der Kapazitätsberücksichtigung.

$$\text{minimiere} \sum_{(i,j)\in A} c_{i,j}x_{i,j} \tag{2.11}$$

s.t.

$$\sum_{(j,i)\in\delta^-(i)} x_{j,i} = 1 \qquad \forall i \in V_c \tag{2.12}$$

$$\sum_{(i,j)\in\delta^+(i)} x_{i,j} = 1 \qquad \forall i \in V_c \tag{2.13}$$

$$\sum_{(i,j)\in\delta^+(0)} x_{i,j} = \sum_{(j,i)\in\delta^-(n+1)} x_{j,i} \qquad \leq m \tag{2.14}$$

$$\sum_{(i,j)\in\delta^+(i)} u_{i,j} = \sum_{(j,i)\in\delta^-(i)} v_{j,i} \qquad \forall i \in V_c \tag{2.15}$$

$$u_{i,j} + t_{i,j}x_{i,j} \leq v_{i,j} \qquad \forall i \in V_c,\ j \in V_c,\ i \neq j \tag{2.16}$$

$$R_i x_{i,j} \leq u_{i,j} \qquad \forall i \in V_c;\ j \in V;\ i \neq j \tag{2.17}$$

$$D_j x_{i,j} \geq v_{i,j} \qquad \forall i \in V;\ j \in V_c;\ i \neq j \tag{2.18}$$

$$u_{i,j} \leq min\{D_i, D_j - t_{ij}\}x_{i,j} \qquad \forall i \in V_c;\ j \in V;\ i \neq j \tag{2.19}$$

$$v_{i,j} \geq max\{R_j, R_i + t_{i,j}\}x_{i,j} \qquad \forall i \in V;\ j \in V;\ i \neq j \tag{2.20}$$

$$\sum_{j\in V_c} f_{j,i} - q_i = \sum_{j\in V_c} f_{i,j} \qquad \forall i \in V_c \tag{2.21}$$

$$f_{i,j} \leq x_{i,j}C \qquad \forall (i,j) \in A \tag{2.22}$$

$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.23}$$

$$u_{i,j} \in \mathbb{Z}^{\geq 0} \qquad \forall i \in V;\ k \in K \tag{2.24}$$

$$v_{i,j} \in \mathbb{Z}^{\geq 0} \qquad \forall i \in V;\ k \in K \tag{2.25}$$

Die Variable $x_{i,j}$ hat die gleiche Bedeutung wie die Variable $x^k_{i,j}$ aus der 3-Index-Formulierung, mit dem Unterschied, dass der Index k nicht berücksichtigt wird. Hierbei werden neue Variablen eingeführt. So beschreibt die Variable $u_{i,j}$ den Bedienstartzeitpunkt bei Knoten i, gegeben der Knoten j ist der unmittelbare Nachfolger. Entsprechend beschreibt $v_{i,j}$ den Bedienstartzeitpunkt bei j, geben der Knoten i ist der von Knoten j. Variable $f_{i,j}$ wird als Flussmenge auf der Kante $(i,j) \in A$ interpretiert. Die Entscheidungsvariable $x_{i,j}$ nimmt den Wert 1 an, wenn die Kante

[37] Vgl. [47]

$(i,j) \in A$ benutzt wird, und sonst den Wert 0.

Die Gleichungen (2.12) und (2.13) sorgen dafür, dass jeder Knoten $i \in V_c$ genau einmal besucht und genau einmal verlassen wird. Das Depot darf hingegen mehrmals besucht und verlassen werden, mit der Einschränkung, dass es genau so oft besucht wie verlassen wird und insgesamt höchstens so oft besucht und verlassen wird wie die Flotte Fahrzeuge aufweist. Eben dies regelt Gleichung (2.14). In Gleichung (2.15) werden die Servicezeiten-Variablen $u_{i,j}$ und $v_{i,j}$ gekoppelt. Gleichung (2.16) stellt sicher, dass der Beginn der Bedienzeit bei j, gegeben der Vorgängerknoten ist i, nicht früher stattfinden kann, als der Beginn der Bedienzeit bei i zuzüglich der Reisezeit von i nach j, gegeben die Kante (i,j) ist in einer Tour enthalten. Die Gleichungen (2.17) bis (2.20) stellen sowohl bezüglich der $u_{i,j}$ als auch der $v_{i,j}$ sicher, dass die Zeitfenster eingehalten sind, wobei die Gleichungen (2.19) und (2.20) auf „möglichst starke“ Koeffizienten der $x_{i,j}$-Variablen zurückgreifen, welche Elemente des Preprocessings abdecken.

Zusätzlich eingefügte Gleichungen (2.21) und (2.22) stellen mittels eines Hilfsflussnetzes die Einhaltung der Kapazitäten sicher. Gleichung (2.21) regelt hierbei den Flusserhalt, wobei jeder Kunde $i \in V_c$ eine Senke mit einem Bedarf im Ausmaß seiner Nachfrage q_i darstellt. Gleichung (2.22) koppelt zum einen die Möglichkeit eines Güterflusses entlang einer Kante an die Befahrung dieser Kante. Zum anderen beschränkt sie den maximalen Güterfluss entlang jeder Kante nach oben durch die Kapazitätskonstante C, womit die Einhaltung der Fahrzeugkapazitäten sicher gestellt ist, da jede initiale (und somit maximale) Fahrzeugbeladung der Flussmenge $f_{i,j}$ entlang der ersten Kante (i,j) der Tour dieses Fahrzeugs entspricht. In Gleichungen (2.23) bis (2.25) wird der Wertebereich der Variablen definiert. Der Vorteil dieser und der folgenden Formulierung besteht darin, dass keine Subtour-Eliminations-Bedingungen gefordert sind.[38]

Zum Vergleich wird eine weitere 2-Index-Formulierung herangezogen, entnommen der Arbeit von Desrochers [22] und um die Einschränkung einiger Gleichungen auf die Kundenmenge V_c angepasst. Hierbei wird eine weitere Variable y_i definiert, welche die seit der Abfahrt vom Depot, bis zur Ankunft bei Knoten i, bereits abgeflossene Ladungsmenge des entsprechenden Lieferfahrzeuges angibt.

In (2.27) wird sicher gestellt, dass jeder Kunde genau ein mal verlassen wird. Hierbei sei zu beachten, dass diese Gleichung nur für die Kundenmenge V_c gültig ist.

[38] Subtour-Eliminations-Bedingungen sind in der Regel für alle mögliche Teilmengen von Kunden erforderlich, vlg. hierzu [49].

Würde man das Depot mit einbeziehen, so müsste auch das Enddepot $n+1$ einmal verlassen werden, was per Konstruktion nicht möglich ist.[39] Die gleiche Regelung gilt auch für die nächste Gleichung (2.28), die sicherstellt, dass alle Kunden, die verlassen wurden, vorher genau ein mal besucht worden sind. Sodann stellt Gleichung (2.29) sicher, dass, der Beginn der Bedienzeit bei Knoten $i \in V_c$ in dessen Zeitfenster liegt. Das Depot kann auch hier vernachlässigt werden, da es die längste Deadline D_i aufweist und das Zeitfenster „automatisch“ eingehalten wird.

$$\text{minimiere} \sum_{(i,j)\in A} c_{i,j} x_{i,j} \tag{2.26}$$

s.t.

$$\sum_{j\in V} x_{i,j} = 1 \qquad \forall i \in V_c \tag{2.27}$$

$$\sum_{(i,j)\in\delta^+(i)\ i\in V_c} x_{i,j} - \sum_{(j,i)\in\delta^-(i)\ i\in V_c} x_{j,i} = 0 \qquad \forall i \in V_c \tag{2.28}$$

$$R_i \le t_i \le D_i \qquad \forall i \in V_c \tag{2.29}$$

$$t_i + t_{i,j} \le t_j + (1 - x_{i,j})M \qquad \forall (i,j) \in A \tag{2.30}$$

$$y_i + q_i \le y_j + (1 - x_{i,j})M \qquad \forall (i,j) \in A \tag{2.31}$$

$$0 \le y_i \le C \qquad \forall i \in V_c \tag{2.32}$$

$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.33}$$

$$t_i \in \mathbb{Z}^{\ge 0} \qquad \forall i \in V;\ k \in K \tag{2.34}$$

Gleichung (2.30) fußt auf dem selben Prinzip wie (2.8). Basierend auf dem gleichen Mechanismus wie in Gleichung (2.30) und Gleichung (2.8), stellt Gleichung (2.31) sicher, dass von der Ladung des entsprechenden Lieferfahrzeuges, welches von $i \in V_c$ nach $j \in V_c$ reist, zur Ankunft bei j mindestens so viel Ladung abgeflossen ist, wie zur Ankunft bei i abgeflossen war, zuzüglich der Nachfrage welche in i befriedigt wurde. Gleichung (2.32) beschränkt sodann die (maximale) abgeflossene Ladungsmenge, und somit de facto auch die (maximale) Ladung jedes Fahrzeuges (da sie für jeden Knoten $i \in V_c$ gefordert wird) auf die Kapazitätsschranke C. Schließlich wird in Gleichungen (2.33) und (2.34) der Wertebereich der Variablen angegeben.

Aufgrund einer sehr ähnlichen Argumentation zu der im Fall des Modells von Letchford [47], ist eine Lösung, welche den aufgeführten Restriktionen genügt, zwingend frei von Subtouren, welche nicht den Depotknoten enthalten.

[39] Hierbei handelt es sich um eine eigene Ergänzung der Modell-Formulierung.

3. Branch-and-Bound und Symmetrie

In diesem Abschnitt wird zunächst der Branch-and-Bound Algorithmus zur Lösung der (M)IP-Formulierungen grob schematisch erläutert, um basierend darauf sodann die Problematik der Symmetrie, die in der 3-Index-Formulierung vorhanden ist, zu veranschaulichen.

3.1. Grundstruktur des B&B Algorithmus'

Das Ziel der B&B-Methode besteht darin, durch gezieltes Suchen und Ausschließen, das Auffinden (und Nachweisen) einer optimalen Lösung zu beschleunigen. Dabei wird der Lösungsraum in Teilräume zerlegt (Branch) und mittels Schranken (Bound) werden ungünstige Teilräume „abgeschnitten". Ohne B&B müssten theoretisch alle möglichen Belegungen der Entscheidungsvariablen durchprobiert werden,[1] um zu der (nachweislich) optimalen Lösung des Problems zu gelangen.[2]

Die Zerlegung erfolgt, angefangen in einem Wurzelknoten,[3] iterativ in jedem weiteren Teilknoten (Teilprobleme), wodurch eine Baumstruktur entsteht, wie in Abbildung 3.1 illustriert wird. Hierbei wird in jedem Knoten eine LP-Relaxation[4] des Ursprungsproblems gelöst. Die Teilräume werden erzeugt, indem in jedem Knoten nach dem Lösen des LPs, einer ganzzahligen Entscheidungsvariablen mit fraktionaler Wertebelegung w^* einerseits die nächstgelegene kleinere ganze Zahl $\lfloor w^* \rfloor$ als „obere"-, sowie andererseits die nächstgelegene größere ganze Zahl $\lceil w^* \rceil$ als „untere"Schranke zugewiesen werden.

[1] Das Durchprobieren aller möglichen Kombinationen wird auch vollständige Enumeration genannt.

[2] Für eine vertiefende Beschreibung des B&B-Algorithmus' sei auf [18] und [25] verwiesen.

[3] Ein Wurzelknoten hat hier die Bedeutung einer Startlösung eines Problems und nicht eines Knotens in einem Graph.

[4] LP steht für Lineares Programm. LP-Relaxation eines Problems bedeutet das Vernachlässigen der Ganzzahligkeits-Bedingungen, um (im Fall eines Minimierungsproblems) zu einer unteren Schranken der Lösung zu gelangen. „Gute" Schranken bei der LP-Relaxation sind insofern wichtig, als dass sie Anzahl der möglichen Nachfolgeknoten reduzieren.

Die verschiedenen Fälle, welche sich für einen Knoten im Verlauf des B&B-Algorithmus' einstellen können, sollen im Folgenden mithilfe der Abbildung 3.1 exemplarisch erläutert werden. Grundsätzlich ist zu jedem Zeitpunkt des B&B-Algorithmus' eine beste bis jetzt bekannte ganzzahlige Lösung in der Variablen ZF^*_{Inc} gespeichert. Initial wird ihr Wert (im Fall eines Minimierungsproblems) auf $ZF^*_{Inc} = +\infty$ gesetzt.

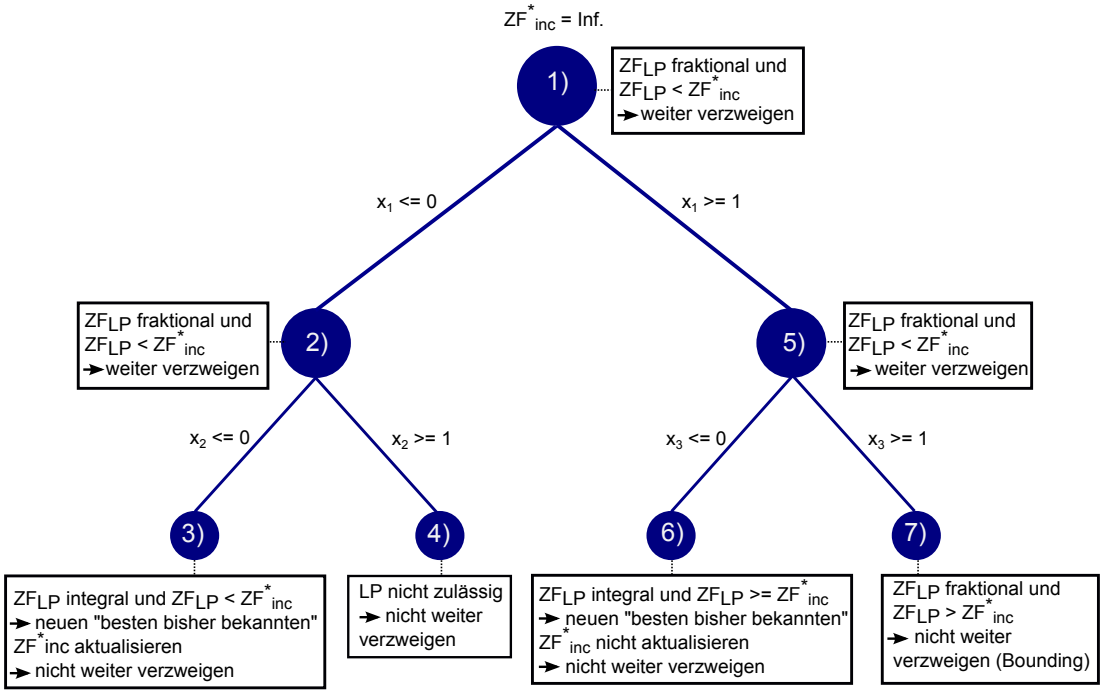


Abbildung 3.1.: Branch-and-Bound-Baum, angelehnt an [25]

Es sei angemerkt, dass die Branching-Variablen x_1, x_2 und x_3 exemplarisch ausgewählt worden sind. Insbesondere beeinflusst der Index einer Variablen nicht (beziehungsweise nur bedingt - je nach Branching-Regel) die Branching-Reihenfolge.

Findet sich in einem Knoten eine ganzzahlige Lösung deren Zielfunktionswert ZF^* besser als der Zielfunktionswert der besten bisher bekannten ganzzahligen Lösung ZF^*_{Inc} ist (im Fall eines Minimierungpsroblems also $ZF^* < ZF^*_{Inc}$ gilt), so wird die neue Lösung (inklusive einer Variablenbelegung) als bester bisher bekannter ganzzahliger Wert gespeichert, und ZF^*_{Inc} entsprechend aktualisiert (dies entspricht dem Fall in Knoten 3). Dieser Knoten wird nicht weiter verfolgt. Der Knoten wird ebenfalls nicht weiter verfolgt, wenn $ZF^* \geq ZF^*_{Inc}$ gilt. In diesen Fall wird die gefundene Lösung jedoch ignoriert, da diese nicht besser ist, als die beste bisher bekannte Lösung (dies entspricht dem Fall in Knoten 6).

Eine weitere Möglichkeit ist, dass das LP durch den letzten Branching-Schritt unzulässig wird. Auch hier muss der entsprechende Knoten nicht weiter verfolgt werden (dies entspricht dem Fall in Knoten 4).

Ist die Lösung ZF^* des aktuellen Knotens fraktional, so ist sie ebenfalls mit dem

Zielfunktionswert ZF^*_{Inc} der besten bis jetzt bekannten ganzzahligen Lösung zu vergleichen. Gilt hierbei $ZF^* \geq ZF^*_{Inc}$, so muss der Knoten nicht weiter verfolgt werden, da die fraktionale Lösung nicht besser als die aktuelle beste bisher bekannte Lösung ist, und sich ihr Zielfunktionswert mit jedem darauf folgenden Branching-Schritt höchstens verschlechtern wird. In diesem Fall spricht man auch vom so genannten „Bounding“ (dies entspricht dem Fall in Knoten 7). Gilt hingegen $ZF^* < ZF^*_{Inc}$, so ist noch nicht auszuschließen, dass sich in den Nachfolgeknoten des aktuellen Knotens eine neue beste ganzzahlige Lösung finden wird. Folglich ist hier weiter zu Branchen (dies entspricht den Fällen in den Knoten 1, 2 und 5). Die letzte beschriebene Situation ist die in der praktischen Umsetzung bei Weitem am häufigsten auftretende. Der B&B-Algorithmus endet, sofern keine weiteren Knoten, die noch nicht verzweigt wurden, vorhanden sind.

Den obigen Ausführungen folgend, ist es ersichtlich, dass sich bei ein und demselben zugrunde liegenden Modell, je nach Selektionsstrategien bezüglich der als nächstes zu bearbeitenden B&B-Knoten und der als nächstes zu verzweigenden Variablen, völlig unterschiedliche Lösungsverläufe ergeben können. Wie im Folgenden diskutiert werden wird, kann sogar bereits die Wahl eines “ungünstigen“ Modells ungewünschte Verläufe des B&B-Algorithmus’ bedingen.

Eine Erweiterung der B&B-Methode stellt die sogenannte Branch(-and-Bound)-and-Cut-Methode (*kurz*: B&C-Methode) dar, bei welcher zusätzlich in jedem Knoten versucht wird “möglichst effektive“ Nebenbedingungen (Cuts) zu ergänzen, um die Ganzzahligkeit der aktuellen LP-Lösung zu verbessern. In dieser Arbeit sollen die Begriffe Branch-and-Bound und Branch-and-Cut weitestgehend synonym behandelt werden.

3.2. Symmetrie

In diesem Abschnitt wird das Problem der Symmetrie und der Einfluss, den sie auf den B&B-Algorithmus haben kann, einführend erläutert.

Die Symmetrie kann in einer bestimmten Variablendefinition eines Problems liegen.[5] Im Fall der 3-Index-Formulierung impliziert Variable $x^k_{i,j}$ eine Unterscheidung zwischen den Fahrzeugen $k \in K$. Sind die Fahrzeuge jedoch identisch, so ist es für eine optimale Lösung des Problems irrelevant, ob beispielsweise Fahrzeug $k = 1$ oder Fahrzeug $k = 3$ eine bestimmte Tour fährt. Für den B&B-Algorithmus sind jedoch

[5] Symmetrie hat hier die Bedeutung von Permutationen und nicht von Symmetrie in Graphen, vgl. [35].

die beiden Touren, welche sich durch diese verschiedene Fahrzeugzuweisung ergeben würden, zwei verschiedene Lösungen, welche im Zweifelsfall beide abgearbeitet werden, und somit unnötig Zeit konsumieren.

Anhand der Abbildung 3.2 sollen zwei verschiedene Fälle von Symmetrie veranschaulicht werden. Dargestellt sind drei Lösungen für eine vereinfachte Beispielinstanz, die sich nur durch verschiedene Fahrzeug-Tour-Zuweisungen unterscheiden. Alle drei Lösungen sind optimal. Jeder Tour wird, entsprechend dem verwendeten Fahrzeug, eine bestimmte Farbe zugeordnet.

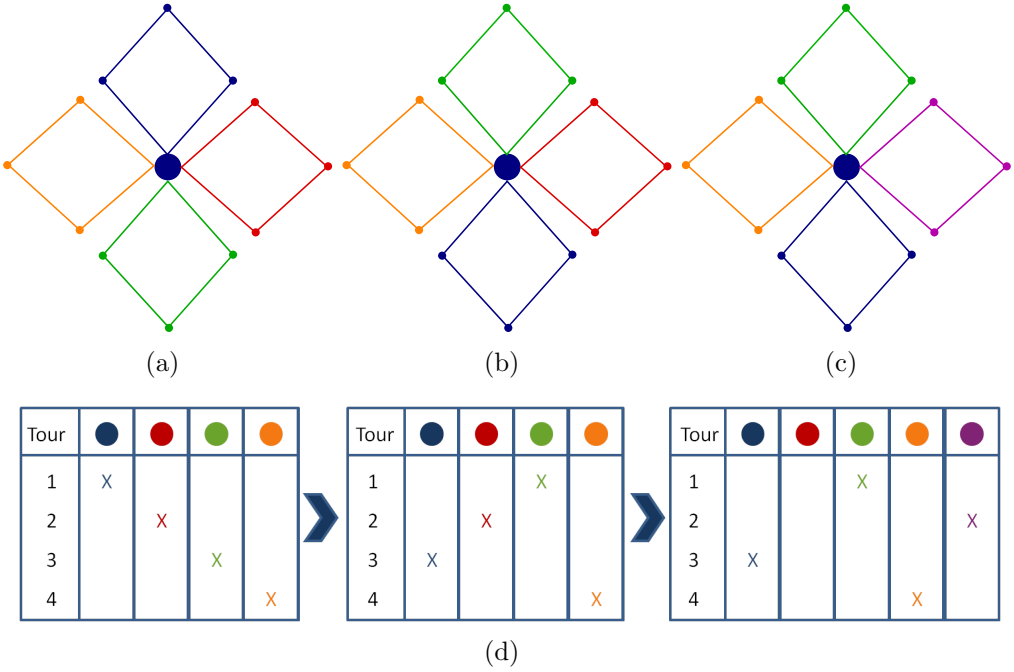


Abbildung 3.2.: Symmetrie-Fälle und zugehörige Matrixdarstellungen, angelehnt an [25]

Der erste Fall der Symmetrie (im Folgenden auch "Symmetrie-Fall 1" genannt) wird durch die Gegenüberstellung von Bild a) und Bild b) in Abbildung 3.2 illustriert: In Bild a) wird die „obere" Tour durch ein blaues Fahrzeug und die „untere" Tour durch ein grünes Fahrzeug gefahren. Bild b) unterscheidet sich lediglich dadurch, dass die beiden Fahrzeuge getauscht wurden und die „obere" Tour nun von dem grünen Fahrzeug und die „untere" Tour von dem blauen Fahrzeug gefahren wird. Ebenso könnten auch das rote und das gelbe oder das rote und das blaue Fahrzeug getauscht werden. Die Anzahl möglicher Tauschkombinationen, welche dabei entstehen, steigt mit einer erhöhten Fahrzeuganzahl rapide an: Konkret, gegeben n Touren mit n zugewiesenen Fahrzeugen, beträgt die Anzahl insgesamt möglicher Belegungsvariationen im Sinne des "Symmetrie-Falls 1" $n!$.

Der zweite Fall der Symmetrie (im Folgenden auch “Symmetrie-Fall 2“ bezeichnet) wird durch die Gegenüberstellung von Bild b) und Bild c) in Abbildung 3.2 illustriert: In Bild b) wird die “rechte“ Tour durch ein rotes Fahrzeug gefahren, in Bild c) wird die selbe Tour hingegen durch ein violettes Fahrzeug gefahren, welches in Bild a) und b) gar nicht in der Menge der ausgewählten Fahrzeuge enthalten war. Diese Art der Symmetrie birgt je nach Anzahl der insgesamt zur Verfügung stehenden Grundmenge an Fahrzeugen ebenfalls eine große Anzahl an möglichen Tauschkombinationen: Konkret, gegeben n Touren und $|K|$ in der Grundmenge zur Verfügung stehende Fahrzeuge, so beträgt die Anzahl möglicher Belegungsvariationen rein im Sinne des “Symmetrie-Falls 2“ $\binom{|K|}{n}$.

Betrachtet man die illustrierten Fahrzeug-Zuweisungen, jeweils kodiert in einer Matrix, wobei jede Zeile einer der Touren, und jede Spalte einem der Fahrzeuge entspricht, wird deutlich, wie viele grundverschiedene Belegungsmuster der korrespondierenden Entscheidungsvariablen “praktisch gesehen“ ein und die selbe Tour beschreiben. Es sei angemerkt, dass keine ausdrücklichen Entscheidungsvariablen für die Zuweisung eines Fahrzeuges zu einer bestimmten Tour existieren, sondern die hier diskutierten Zuordnungen viel mehr implizit in den Belegungen der $x_{i,j}^k$ enthalten sind.

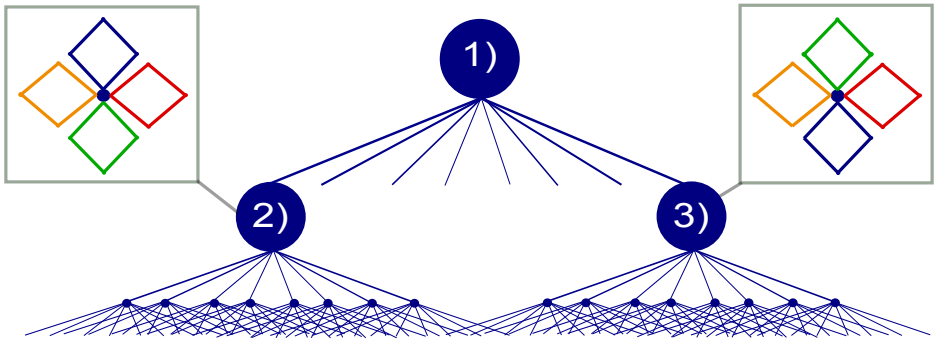


Abbildung 3.3.: Symmetrie im B&B-Baum, angelehnt an [25]

Mit dem Hintergrundwissen aus dem vorherigen Abschnitt können die Folgen der Symmetrie im B&B-Algorithmus deutlich gemacht werden: Jede Symmetriepermutation des Optimums entspricht einer eigenen Variablen-Belegung und somit potentiell einem eigenen Teilbaum im B&B-Verfahren. Von diesen Teilbäumen kann offenbar keiner im Rahmen des Boundings ausgeschlossen werden. Ohne ausdrückliche Unterbindung würden selbige Teilbäume zur Gänze abgearbeitet werden, obwohl sie, bis auf einen, völlig irrelevant sind. In Abbildung 3.3 wird die Problematik, die die Symmetrie in einem B&B-Baum erzeugt, veranschaulicht.

Die Knoten 2) und 3) repräsentieren (sowie sämtliche Knoten in dieser Abbildung) keine B&B-Knoten, sondern zwei Teilbäume, die je zwei gleichwertige (in manchen Variablen noch fraktionale) Lösungen enthalten, welche von dem B&B-Algorithmus nicht ausgeschlossen werden können, und somit zu diversen weiteren, obsoleten Branching-Verzweigungen führen.

Es gibt viele Wege den Effekt der Symmetrie zu verringern, sowie zum Beispiel das Ergänzen zusätzlicher, sogenannter "Symmetrie-brechender" Ungleichungen, oder das Umformulieren des Modells in einer Art und Weise, welche das Entstehen von Symmetrien von vornherein vermeidet. Ein Beispiel für die zweite Maßnahme bildet das Verwenden der 2-Index-Formulierung von Letchford et al. [47] statt der 3-Index-Formulierung von Kohl et al. [41].[6]

[6]Für vertiefende Literatur zum Thema Symmetrie sei auf [56] verwiesen.

4. Preprocessing

Das in dieser Arbeit verwendete Preprocessing hat die Vereinfachung der VRPTW-Instanzen zum Ziel. Das Preprocessing wird vor dem eigentlichen Lösen des Problems, also der LP-Relaxation und dem B&B-Verfahren, eingesetzt und setzt sich aus vier iterativen Schritten zusammen:

- Ermittlung der frühesten und spätesten Abfahrtszeiten
- Ermittlung der Präzedenzen
- Zeitfensterstraffung
- Kantenelimination.

Im Folgenden werden die einzelnen Komponenten ausführlicher beschrieben. Dabei zielen alle Schritte des Preprocessings insbesondere darauf ab, möglichst viele überflüssige Kanten aus dem Ursprungsgraph zu löschen, um somit den Graphen zu vereinfachen.[1] Der komplette Abschnitt 4 stützt sich auf die Arbeit von Ascheuer [6] und Kallehauge [38].[2]

4.1. Ermittlung der frühesten und spätesten Abfahrtszeiten

Die Abfahrtszeitpunkte sind durch die Zeitfenster eingeschränkt. Aus diesem Grund müssen die Grenzen dieser Abfahrtszeitpunkte ermittelt werden. Zunächst werden die so genannten frühestmögliche Abfahrtszeiten (*engl.*: Earliest Departure Time, *kurz*: EDT) und spätestmögliche Abfahrtszeiten (*engl.*: Latest Departure Time *kurz*: LDT) für jeden Knoten $i \in V_c$ ermittelt, wobei für die in dieser Arbeit vorgestellten Folgeschritte des Preprocessings strenggenommen lediglich die frühestmöglichen Abfahrtszeiten notwendig sind, wobei für die in dieser Arbeit vorgestellten Folgeschritte

[1] Als Motivation kann Abbildung A.2 aus dem Anhang hinzugezogen werden, welche einen Graph vor und nach dem Preprocessing darstellt.

[2] Ascheuer behandelt in seiner Arbeit [6] die Preprocessing-Maßnahmen im Kontext von TSPTW. Kallehauge wendet diese Maßnahmen auf VRPTW an [38].

des Preprocessings strenggenommen lediglich die frühestmöglichen Abfahrtszeiten notwendig sind. Die spätestmöglichen Abfahrtszeiten seien dennoch der Vollständigkeit halber vorgestellt.

4.1.1. Früheste Abfahrtszeiten

Frühestmögliche Abfahrtszeiten geben den frühesten Zeitpunkt an, zu dem ein Kunde verlassen werden kann. Betrachtet man den Weg von einem Knoten $i \in V_c$ zum Knoten $j \in V_c$, so kann der Knoten $j \in V_c$ frühestens zum Zeitpunkt $R_i + t_{i,j}$ erreicht werden, da Knoten $i \in V_c$ nicht vor R_i verlassen werden kann und $t_{i,j}$ die kürzest mögliche Reisezeit von i nach j darstellt (gegeben die Dreiecksungleichung[3] gilt). Liegt diese frühestmögliche Ankunftszeit im Zeitfenster von j, so ist sie zugleich die frühestmögliche Abreisezeit bei j, gegeben der Vorgängerknoten war i. Liegt die frühestmögliche Ankunftszeit hingegen vor dem Beginn des Zeitfensters von j, so stellt sich R_j als effektive, frühestmögliche Abfahrtszeit ein. Die frühestmögliche Abfahrtszeit bei Knoten j, gegeben der Vorgängerknoten war i, im Folgenden auch als $EDT(i,j)$ bezeichnet, ermittelt sich somit zu

$$EDT(i,j) = max\{R_j, R_i + t_{i,j}\} \quad \forall i \in V_c \; . \tag{4.1}$$

Diese Servicezeiten-ignorierende Argumentation ist deshalb möglich, weil selbige bereits „virtuell“ in der Distanzmatrix enthalten sind (da $t_{i,j} = s_i + d_{i,j}$). Dies impliziert, dass ein Fahrzeug nach seiner Ankunft bei Knoten j sofort zum nächsten Kunden weiterfahren kann.

Sollte die Kante $(i,j) \in A$ nicht existieren, ist $EDT(i,j) = +\infty$ zu setzen.

4.1.2. Späteste Abfahrtszeiten

Spätestmögliche Abfahrtszeiten geben den spätesten Zeitpunkt an, zu dem ein Kunde in Richtung eines bestimmten anderen Kunden verlassen werden kann. Betrachtet man den Weg von einem Knoten $i \in V_c$ zu einem Knoten $j \in V_c$, so muss der Knoten $i \in V_c$ spätestens zum Zeitpunkt $D_j - t_{i,j}$ verlassen werden, damit Knoten $j \in V_c$ noch innerhalb seines Zeitfensters erreicht werden kann, gegeben diese Abreisezeit liegt vor der Deadline D_i von $i \in V_c$. Ansonsten ist die spätestmögliche Abreisezeit nach $j \in V_c$ gerade D_i. Die spätestmögliche Abreisezeit bei i, gegeben der Nachfolgeknoten ist j, im Folgenden auch als $LDT(i,j)$ bezeichnet, berechnet sich somit

[3] Die Dreiecksungleichung $c_{ij} \leq c_{i,h} + c_{h,j}$ stellt sicher, dass der Umweg von i nach j über h niemals kürzer ist, als der direkte Weg von i nach j.

zu

$$LDT(i,j) = min\{D_i, D_j - t_{,j}\} \quad \forall i \in V_c. \tag{4.2}$$

Sollte die Kante $(i,j) \in A$ nicht existent sein, so ist $LDT(i,j) = +\infty$ zu setzen.

4.2. Ermittlung der Präzedenzen

Präzedenzen sind Reihenfolgen, in denen Kunden besucht werden können. Das Vorhandensein der Zeitfenster führt dazu, dass durch diese Präzedenzen zwischen den Knoten impliziert sein können.

Zunächst kann ein Präzedenzgraph $G_p = (V, A_p)$ definiert werden. Hierbei ist eine Kante (j,i) genau dann in A_p enthalten, wenn j vor i besucht werden muss. Dies lässt sich auch schreiben als $j \prec i$.

Nun können mit Hilfe des zuvor berechneten Parameters $EDT(i,j)$ Präzedenzen bestimmt werden. Eine Präzedenz besteht, wenn ein Knoten j vor einem anderen Knoten i besucht werden muss. Dies ist der Fall, wenn die frühestmögliche Abreisezeit bei j, gegeben i wäre der Vorgängerknoten, die Deadline von j übersteigt. Also gilt:

$$EDT(i,j) > D_j \Rightarrow j \prec i. \tag{4.3}$$

Analog muss j nach i besucht werden, wenn die frühestmögliche Ankunftszeit von j bei i die Deadline von i übersteigt:

$$EDT(j,i) > D_i \Rightarrow i \prec j. \tag{4.4}$$

Im Kontext des TSPTW gilt überdies die intuitive Schlussfolgerung:[4]

$$k \prec i \wedge i \prec j \Rightarrow k \prec j, \tag{4.5}$$

womit sich die Präzedenzkenntnisse entscheidend erweitern lassen. Für den VRPTW-Kontext gilt diese logische "Konklusions-Kette" jedoch nicht, da der "Mittelknoten" i von einem anderen Fahrzeug besucht werden könnte. Dies hat ebenfalls zur Folge, dass der VRPTW-Präzedenzgraph (im Gegensatz zum Präzedenzgraph des TSPTW) nicht notwendigerweise frei von Zyklen sein muss. So verheißt ein etwaiger

[4] Vgl. hierzu [6]

Zyklus $i - j - i$ in G_p lediglich, dass die Knoten i und j nicht von ein und demselben Fahrzeug angefahren werden können, wogegen das TSPTW in diesem Fall nicht lösbar wäre. Technisch gesehen ist der Präzedenzgraph des VRPTW, im Gegensatz zu dem des TSPTW, weder zwingend azyklisch noch transitiv geschlossen.[5]

4.3. Zeitfensterstraffung

Falls ein Zeitfenster nachweislich von keiner zulässigen Lösung in seinem vollen Umfang genutzt wird, kann dieses entsprechend durch Anhebung der Release Zeit und/oder Senkung der Deadline verkürzt werden, womit in weiteren Schritten überflüssige Kanten aus dem Graph eliminiert werden können.

4.3.1. Anhebung der Ankunftszeiten

Werden alle möglichen Vorgänger i eines Knotens j, also jene für die gilt $(i,j) \in A$, betrachtet und ist die niedrigste, frühestmögliche Ankunftszeit $R_i + t_{ij}$ größer als die Release Zeit R_j, so kann diese offenbar entsprechend angehoben werden:

$$R_j = max\{R_j, min_{(i,j)\in A}\{R_i + t_{i,j}\}\} \qquad \forall j \in V s.t. \delta^-(j) \neq \emptyset. \tag{4.6}$$

Abbildung 4.1 soll den Mechanismus der Anhebungsregel (4.6) verbildlichen.

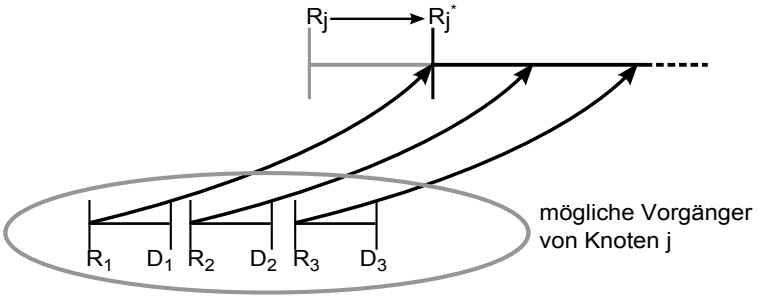


Abbildung 4.1.: Visualisierung einer Anhebung der Release Zeit R_j gemäß Regel (4.6), angelehnt an [21]

Nun werden alle in Frage kommenden direkten Nachfolger k von Knoten j betrachtet, also jene für die gilt $(j,k) \in A$. Die zweite Anhebungsregel resultiert aus der folgenden Überlegung: Wann muss der Knoten j frühestens verlassen werden, um bei mindestens einem der Nachfolgerknoten k Wartezeit zu vermeiden. Hierzu wird die kleinste Abfahrtszeit bei j, $R_k - t_{j,k}$ ermittelt, gegeben der jeweilige Knoten k wird zu seiner Release Zeit erreicht, betrachtet über sämtliche möglichen Nachfolger

[5] Vgl. hierzu [38]

von j. Liegt selbige über der Release Zeit von j (und zugleich unter der Deadline, womit Ankunfts- und Startzeit jeweils zusammenfallen), so darf R_j entsprechend angehoben werden, da die Zulässigkeit einer Lösung vom eventuellen Entfallen einer Wartezeit unberührt bleibt:

$$R_j = max\{R_j, min\{D_j, min_{(j,k)\in A}\{R_k - t_{j,k}\}\}\} \qquad \forall j \in V s.t. \delta^+(j) \neq \emptyset. \tag{4.7}$$

Abbildung 4.2 soll den Mechanismus der Anhebungsregel (4.7) verbildlichen.

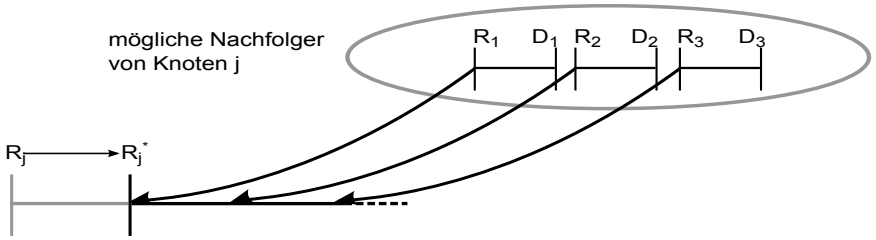


Abbildung 4.2.: Visualisierung einer Anhebung der Release Zeiten R_j gemäß Regel (4.7), angelehnt an [21]

Das Hauptelement in Abbildung 4.2 ist die Release Zeit R_j vom Knoten j, die auf R_j^* angehoben wird. Somit kann die überflüssige Zeitspanne des Zeitfensters „eliminiert“ werden.

4.3.2. Senkung der Deadlines

Entsprechend der Release-Zeiten-Anhebung wird die Senkung der Deadlines vorgenommen. Werden alle möglichen Nachfolger k eines Knotens $j \in V$, also jene für die gilt $(j,k) \in A$, betrachtet und ist jede, bezogen auf die Deadline von k spätestmögliche Abfahrtszeit $D_k - t_{j,k}$ bei j kleiner als die Deadline D_j (und zugleich größer als R_j, womit Ankunfts- und Startzeit jeweils zusammenfallen), so kann D_j entsprechend auf die größte dieser spätestmöglichen Abfahrtszeiten gesenkt werden, da j in keiner zulässigen Tour später verlassen werden wird:

$$D_j = min\{D_j, max_{(j,k)\in A}\{D_k - t_{j,k}\}\} \qquad \forall j \in V s.t. \delta^+(j) \neq \emptyset. \tag{4.8}$$

Abbildung 4.3 soll den Mechanismus der Senkungsregel (4.8) verbildlichen. Hier wird die Deadline D_j auf D_j^* abgesenkt.

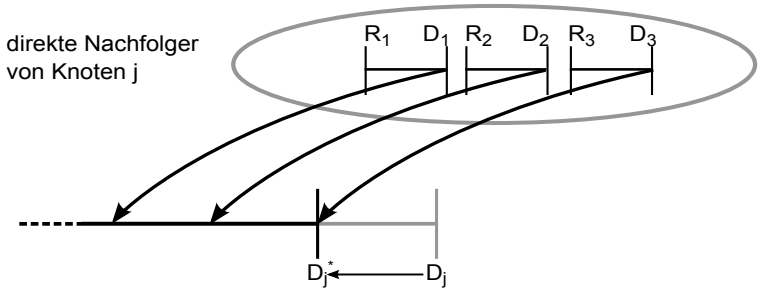


Abbildung 4.3.: Visualisierung einer Senkung der Deadline gemäß Regel (4.8), angelehnt an [21]

Eine zusätzliche Senkung der Deadline D_j wird wie folgt vorgenommen: Werden alle möglichen Vorgänger i eines Knotens j, also jene für die gilt $(i,j) \in A$, betrachtet und ist die höchste, spätestmögliche Ankunftszeit bei j, $D_i + t_{i,j}$, kleiner als die Deadline D_j (und zugleich größer als R_j, womit Ankunfts- und Startzeit jeweils zusammenfallen), so kann D_j offenbar entsprechend gesenkt werden:

$$D_j = min\{D_j, max\{R_j, max_{(i,j)\in A}\{D_i + t_{i,j}\}\}\} \qquad \forall j \in V s.t. \delta^-(j) \neq \emptyset. \tag{4.9}$$

Abbildung 4.4 soll den Mechanismus der Senkungsregel (4.9) verbildlichen. Hierbei wird D_j auf D_j^* herabgesenkt.

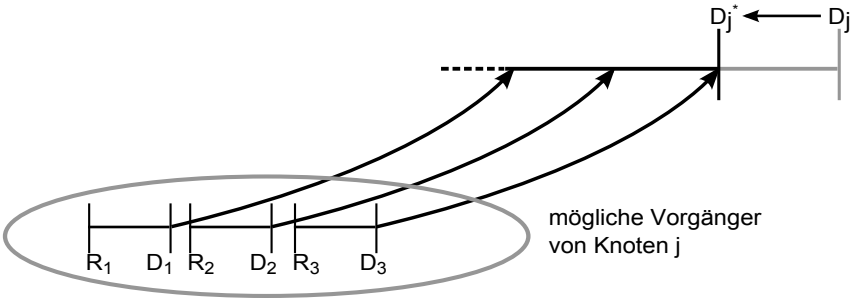


Abbildung 4.4.: Visualisierung einer Senkung der Deadline gemäß Regel (4.9), angelehnt an [21]

4.4. Kantenelimination

Elimination von Kanten ist einer der Hauptgründe, aus denen Preprocessing durchgeführt wird. In diesem Schritt werden mittels der vorab ermittelten Präzedenzen überflüssige Kanten eliminiert, welche nachweislich in keiner zulässigen Lösung vorkommen können.

Gilt die Präzedenzordnung $(i,j) \in A_p$, sprich $i \prec j$, so kann in keiner zulässigen Lösung die Kante $(j,i) \in A$ auftreten. Also kann die folgende Kantenlöschungsregel formuliert werden:

$$(i,j) \in A_p \Rightarrow \ lösche\ Kante\ (j,i)\ aus\ A.$$

Für drei Knoten $i,j,k \in V$ gilt im Kontext des TSPTW ferner die Kantenlöschungsregel

$$(i,j) \in A_p \wedge (j,k) \in A_p \Rightarrow \ lösche\ Kante\ (k,i)\ aus\ A.$$

Aus analoger Gegenargumentation zu jener im Fall der Präzedenz-Implikations-Kette (4.5) ist diese Kantenlöschungsregel für den Fall des VRPTW jedoch nicht gültig. Selbiges gilt für Kantenlöschungsregeln komplexerer Natur.[6]

Generell bieten sich im Fall des VRPTW vorab grundsätzlich die beiden "trivialen" Kantenlöschungsregeln

$$R_i + t_{i,j} > D_j \Rightarrow \ lösche\ Kante\ (i,j)\ aus\ A \tag{4.10}$$

$$q_i + q_j > C \Rightarrow \ lösche\ Kante\ (i,j)\ aus\ A \tag{4.11}$$

an. (4.10) ist selbsterklärend und impliziert von (4.3). (4.11) bedient sich der Tatsache, dass zwei Kunden, deren Gesamtnachfrage die Fahrzeugkapazität C übersteigt, niemals auf ein und der selben Fahrzeugtour liegen können.

4.5. Preprocessing Algorithmus

Die Preprocessing-Schritte hängen eng miteinander zusammen. Gegebenfalls können neue Präzedenzen gefunden werden, nachdem die Zeitfensterstraffung vorgenommen wurde. Durch neu entdeckte Präzedenzen können wiederum mehr Kanten gelöscht werden. Weniger Kanten begünstigen eine effektivere Zeitfensterstraffung, da ein beliebiger Knoten $i \in V$ dadurch weniger mögliche Vorgänger und Nachfolger hat.

[6] Zur Thematik erweiterter Kantenlöschungsregeln sei beispielsweise auf Ascheuer [6], Dash et al. [19] oder Beckhäuser-Hoffmann [9] verwiesen.

Algorithm 1 Preprocessing

1: Berechne $EDT(i,j)$ *wie in* (4.1) *beschrieben*
2: Ermittele die Präzedenzen *wie in* (4.2) *beschrieben*
3: Straffe die Zeitfenster *wie in* (4.3) *beschrieben*
4: Eliminiere Kanten *wie in* (4.4) *beschrieben*
5: **while** Veränderung in den Zeitfenstern aufgetreten **do**
6: Berechne $EDT(i,j)$ *wie in* (4.1) *beschrieben*
7: Ermittele die Präzedenzen *wie in* (4.2) *beschrieben*
8: Straffe die Zeitfenster *wie in* (4.3) *beschrieben*
9: Eliminiere Kanten *wie in* (4.4) *beschrieben*
10: **end while**

Aus diesem Grund sollten alle Schritte mehrmals durchgeführt werden, bis zum Beispiel keine weitere Zeitfensterstraffung mehr erzielt werden kann. Das Vorgehen wird im obigen Algorithmus "Algorithm 1" veranschaulicht.[7]

[7] Angelehnt an [32].

5. Experimente und Ergebnisse

In den folgenden Abschnitten werden der experimenteller Aufbau und die resultierenden Ergebnisse vorgestellt und diskutiert. Umfassende Tests mit Preprocessing-Techniken zeigen deren Auswirkungen auf das Lösungsverhalten der unterschiedlichen (M)IP-Formulierungen.

5.1. Verwendete Instanzen

Für das VRPTW existiert eine Vielzahl von Beispielinstanzen,[1] die für einen Vergleich der Effektivität verschiedener Lösungsansätze herangezogen werden können. Im Folgenden werden sowohl die Solomon VRPTW-Instanzen, als auch eigene Modifikationen derselbigen vorgestellt.

5.1.1. Solomon Instanzen

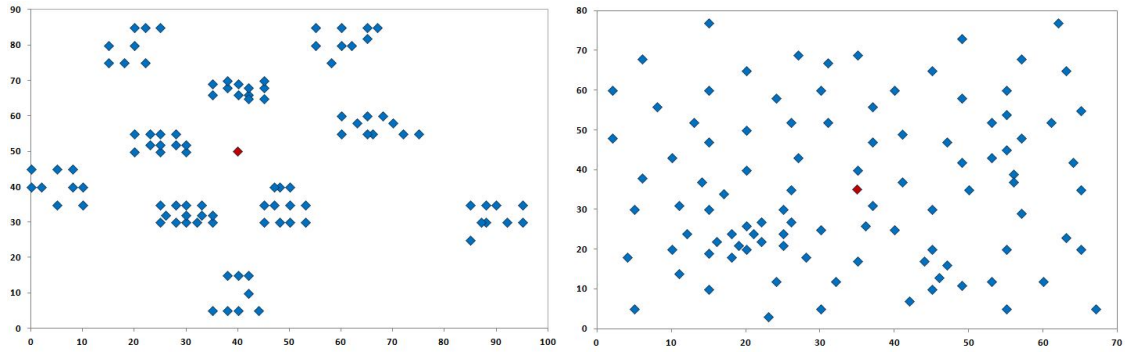
Eine Reihe der populärsten Instanzen für VRPTW wurden von Marius M. Solomon vorgestellt. Die Instanzen sind symmetrisch[2] und unterscheiden sich in zwei wichtigen Kriterien: Dem Umfang der Zeitfenster und dem Muster der geographischen Kundenverteilung. Basierend auf der durchschnittlichen Zeitfensterweite sollen zunächst zwei "Instanz-Klassen" unterschieden werden: "Klasse 1" steht für Instanzen mit „relativ kleinen"-, "Klasse 2" für Instanzen mit "relativ großen" durchschnittlichen Zeitfensterweiten. Jede dieser Klassen kann, basierend auf der Kundenverteilung, je in die 3 topologischen Kategorien "R", "C" und "RC" unterteilt werden. R steht dabei für „Random", also gänzlich zufällig verteilte Kundenstandorte. C steht für „Cluster", also eine Verteilung, bei der die Kundenstandorte in mehreren Punktewolken um das Depot verteilt sind. Entsprechend handelt es sich bei den RC-Instanzen um „Random-Cluster", also eine Mischung aus der Cluster- und Random-Verteilung der Standorte.

[1] u.a. von Gehring und Homberger, [30] sowie von Solomon [57], frei verfügbar auf http://neo.lcc.uma.es/vrp/vrp-instances.

[2] Symmetrische Instanzen implizieren die Deckungsgleichheit eines Wegs (i,j) mit dem Rückweg (j,i).

Alle Instanzen können überdies nach drei Größen unterschieden werden: 25, 50 und 100 Städte.

Jede Instanz beinhaltet folgende Informationen: Die Nummerierung der Kundenstandorte, X- und Y-Koordinaten der Kundenstandorte, Kundennachfragen, Zeitfensteranfang und -ende sowie die Servicezeiten, Fahrzeuganzahl und -kapazitäten.[3] Die X-Y-Koordinaten sind innerhalb einer Kategorie identisch. Die Kundennachfragen sind je nach Instanz variabel. Klasse 1 hat zwar generell kurze Zeitfenster, diese variieren in ihrer Größe jedoch teilweise innerhalb der jeweiligen Kategorie und Klasse. Das Zeitfenster des Depots beginnt grundsätzlich zum Zeitpunkt Null und endet mit einem Zeitpunkt, der so beschaffen ist, dass grundsätzlich alle Fahrzeuge in das Depot zurückkehren können. Die Servicezeiten betragen konstant 10 (R-Kategorie), 90 (C-Kategorie) und 10 (RC-Kategorie) Zeiteinheiten. Die Anzahl der Fahrzeuge liegt konstant bei 25 Stück für alle Klassen und Kategorien. Die Kapazitäten der Fahrzeuge unterscheiden sich jedoch je nach Instanz-Klasse. In Klasse 1 beträgt die Kapazität der Fahrzeuge in allen Kategorien 200 Einheiten. Fahrzeuge aus Instanzen der Klasse 2 und Kategorie C haben sämtlich eine Kapazität in Höhe von 700 Einheiten, Fahrzeuge aus allen anderen Kategorien der Klasse 2 haben 1000 Einheiten Kapazität.



(a) Cluster Kategorie (b) Random Kategorie

Abbildung 5.1.: Geographische Verteilung der Kategorie C und R, Instanzen aus [57]

Für die Experimente wurden die Kategorien C und R ausgewählt, da diese, im Hinblick auf die geographische Beschaffenheit, zwei gegensätzliche Extrembeispiele darstellen und daher aus experimenteller Sicht besonders interessant erscheinen. In Abbildung 5.1 werden die zwei im Experimentalteil verwendeten Klassen exemplarisch anhand je einer Instanz mit 100 Städten graphisch veranschaulicht. Die Lage des Depots wird dabei jeweils durch einen roten Punkt, die Lage der Kundenstandorte durch blaue Punkte gekennzeichnet. Die 100 Städte Instanzen sind deshalb

[3] Abbildung einer beispielhaften Instanz wird im Anhang A.1 präsentiert.

zur Illustration ausgewählt, weil sie die Struktur der Kundenverteilung besonders anschaulich zur Geltung bringen. So gut wie alle Instanzen mit dieser Städtezahl beziffern jedoch im Rahmen des Branch-and-Cut-Verfahrens benötigte CPU-Zeiten, welche die Toleranzschwelle der in dieser Arbeit durchgeführten Experimente, namentlich 18000 Sekunden = 5 Stunden, überschreiten. Außerdem gilt Selbiges, ungeachtet der Städteanzahl, für die meisten Instanzen der Klasse 2. Daher werden in den folgenden Experimenten lediglich Instanzen der Klasse 1 mit 25 und 50 Städten betrachtet.

Für die Experimente werden die in Tabelle 5.1 aufgeführten Instanzen verwendet. Der Buchstabe zu Beginn jedes Dateinamens, kodiert die Kategorie (C = Cluster, R = Random) der entsprechenden Instanz. Die durch einen Punkt abgetrennte Zahl am Ende des Dateinamens beziffert die Städteanzahl. Tabelle 5.1 gibt überdies einen stichwortartigen Überblick über die Zeitfenstereigenschaften der verwendeten Instanzen.

Tabelle 5.1.: Zeitfenstereigenschaften der verwendeten Instanzen

Instanz	Zeitfenstereigenschaften
C101.25/C101.50	enge, variable Zeitfenster
C103.25/C103.50	ca. 50% sehr enge und 50% sehr weite Zeitfenster
C109.25/C109.50	konstante Zeitfenstergröße von 360 Zeiteinheiten
R101.25/R101.50	konstante Zeitfenstergröße von 10 Zeiteinheiten
R102.25/R102.50	ca. 50% Zeitfenster der Größe 30 und 50% sehr weite Zeitfenster
R106.25/R106.50	enge, variable Zeitfenster

5.1.2. Modifikation der Instanzen

Eine Modifikation der *C*- und *R*-Instanzen soll die Aussagekraft der Ergebnisse unterstützen. Vor allem soll das Preprocessing unter dem Einfluss vergrößerter Zeitfensterweiten genauer untersucht werden. Dazu werden die Zeitfensterweiten der ausgewählten Originalinstanzen um den Faktor zwei vergrößert. Die X- und Y-Koordinaten bleiben unverändert. Auf diese Weise soll untersucht werden, wie sich größere Zeitfensterweiten ceteris paribus auf das Lösungsverhalten mit und ohne Preprocessing auswirken.

Es ist überdies denkbar, dass auch das Muster der Kundenverteilung alleine entscheidenden Einfluss auf das Lösungsverhalten einer gegebenen Instanz nimmt. Aus

diesem Grund werden zusätzliche Instanzen vorbereitet, welche das Vergleichen der Kategorien Cluster und Random ceteris paribus ermöglichen sollen. Hierzu werden alle Parameter, namentlich Zeitfenster, Servicezeiten und Kundennachfrage, auf den beiden Instanztypen jeweils gleich gesetzt. Als einziger Unterschied verbleibt die topologische Verteilung der Kundenstandorte: Cluster und Random.

Die Ergebnisse zu den beschriebenen Experimenten werden im nächsten Kapitel 5.3.2, Tabelle 5.11 vorgestellt.

5.2. GAMS und CPLEX

Alle Experimente wurden auf einem 64-Bit Betriebssystem mit Intel Core Duo© 1.3 GHz Prozessor und 4 GB RAM durchgeführt. Als Modellierungssprache wird das gängige System GAMS (General Algebraic Modelling System), Version 24.0.1, eingesetzt.[4] Als Solver wird IBM ILOG CPLEX Version 12.5.0.0 verwendet.

CPLEX verfügt über eine Reihe von Symmetrie-brechenden Ungleichungen. In der Standarteinstellung entscheidet CPLEX automatisch, ob und wie viele Ungleichungen dieser Art dem Modell im Rahmen des Branch-and-Cut-Verfahrens hinzugefügt werden. Der Benutzer hat allerdings die Möglichkeit, den Einsatz der Symmetrie-brechenden Ungleichungen zu kontrollieren. So können diese Ungleichungen gar nicht, bis sehr aggressiv eingesetzt werden.[5] Alle Experimente sind mit der Standardeinstellung durchgeführt worden.

5.3. Experimente und Diskussion der Ergebnisse

Die Auswirkungen des Preprocessings auf das Lösungsverhalten der Instanzen werden in den folgenden Abschnitten ausführlich untersucht. Zunächst werden Zahlen zu Zeitfensterstraffungen und Kantenelimination vorgestellt und diskutiert. Sodann werden die Ergebnisse zu den Instanzen mit 25 Städten auf der 3-Index-Formulierung von Kohl et al. [41] und der 2-Index-Formulierung von Letchford et al. [47] präsentiert und analysiert.

Analog erfolgt die Vorstellung der Ergebnisse zu den Instanzen mit 50 Städten. Zum Vergleich werden die Ergebnisse zu den modifizierten Instanzen, die in Abschnitt 5.1.2 beschrieben worden sind, herangezogen.

[4] Mehr Informationen zu GAMS ist unter [29] zu finden.
[5] Mehr Informationen zu CPLEX ist unter [3] zu finden.

Für die Experimente werden folgende Annahmen getroffen: Da keine einheitliche Regelung für die Reisekostenberechnung existiert,[6] wird die Berechnungsregel von Kohl et al. [41] verwendet: $c_{i,j} = \lfloor 10 * \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor / 10$, die die euklidischen Distanzen auf eine Nachkommastelle abrundet. Zusätzlich wird ein Skalar 0,1 zu jeder Kante, welche nicht aus dem Depot herausführt, addiert, sodass die Dreiecksungleichung $c_{ij} \leq c_{i,h} + c_{h,j}$ erfüllt und die optimale Lösung dadurch nicht beeinträchtigt ist. Diese Regelung erhöht den Zielfunktionswert um den Wert $((n+1) - 1/10)$, der anschließend vom Zielfunktionswert wieder subtrahiert werden kann. Die "virtuelle" Reisezeit $t_{i,j}$, welche letztendlich die effektiv ins Modell einfließende Kostenmatrix darstellt, setzt sich aus der tatsächlichen Reisezeit $d_{i,j}$ zuzüglich der Servicezeiten s_i zusammen (was die Gültigkeit der Dreckeisungleichung für sämtliche Knotentupel offenbar ebenfalls nicht berührt, da die Servicezeiten für alle Knoten identisch sind). Hierdurch kann es, ohne jedwede Unterbindung seitens der Restriktionen, im ungünstigsten Fall (zum Beispiel bei langen Servicezeiten) dazu kommen, dass ein Knoten erst nach Abschluss seiner Deadline verlassen wird. Da allerdings keine Forderung besteht, dass die Bedienzeit innerhalb des Zeitfensters abgeschlossen sein muss, sondern lediglich, dass der Ankunftszeitpunkt nicht hinter dem Ende des Zeitfensters liegen darf, ist diese Tatsache unproblematisch. Eine Konstellation, in welcher die Bedienzeit eines Knotens, aber eben nicht die Ankunftszeit, dessen Deadline überschreitet ist in Abbildung 5.2 illustriert.

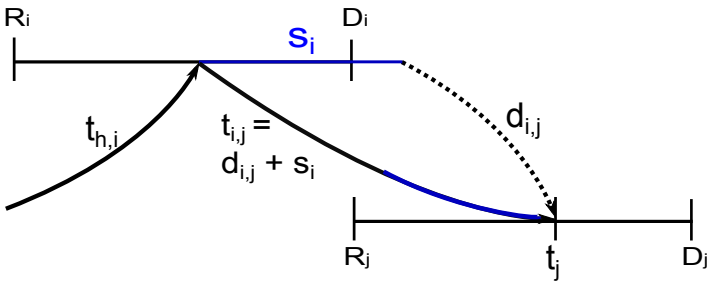


Abbildung 5.2.: Zeitfensterüberschreitung durch zu lange Servicezeiten

Diverse Instanzen wären ohne die Tolerierung von Überschreitungen der Knotendeadlines durch die Bedienzeiten gar a priori unlösbar, da die Weite vieler ihrer Zeitfenster geringer als die Länge der Bedienzeiten ausfällt. So hat beispielsweise die Instanz C101.25 mit 25 Standorten konstante Servicezeiten von 90 Zeiteinheiten bei allen Standorten außer dem Depot. Die Zeitfensterweiten sind jedoch im Durchschnitt 60 Zeiteinheiten weit und erreichen einen Maximalwert von 89 Zeiteinheiten.

[6] Vlg. hierzu [45].

5.3.1. Zahlen zu Zeitfensterstraffung und Kantenelimination

Das Ziel der Experimente in diesem Abschnitt besteht darin, den Effekt des Preprocessings auf die Zeitfensterweiten und die Kantenmenge zu untersuchen. Zu diesem Zweck werden die Preprocessing-Ergebnisse der Originalinstanzen mit denen der modifizierten Instanzen verglichen.

Tabelle 5.2 gibt einen Überblick über die betrachteten Parameter.

Tabelle 5.2.: Bezeichner der Ergebnis-Tabelle 5.3

Kürzel	Bezeichnung
Instanz	Instanz-Bezeichnung
TW_v	Summe der Zeitfensterweiten vor dem Preprocessing
TW_n	Summe der Zeitfensterweiten nach dem Preprocessing
Δ_{TW}	prozentuale Differenz zwischen TW_v und TW_n
Adj_v	Anzahl der Adjazenzen vor dem Preprocessing
Adj_n	Anzahl der Adjazenzen nach dem Preprocessing
Δ_{Adj}	prozentuale Differenz zwischen Adj_v und Adj_n
Adj_z	Anzahl der Adjazenzen nach (4.10) und (4.11)
Δ_{AdjK}	Prozent der Kanten gelöscht durch (4.10) und (4.11)
Δ_{AdjP}	Prozent der Kanten gelöscht durch "ambitioniertes" Preprocessing

Zur Beurteilung der Ergebnisse werden die Zeitfensterweiten vor (TW_v) und nach (TW_n) dem Preprocessing dokumentiert und einander gegenübergestellt (Δ_{TW}). Δ_{TW} berechnet sich wie folgt: $\Delta_{TW} = TW_v/TW_n - 1$. Analog wird die Anzahl der Adjazenzen vor (Adj_v) und nach (Adj_n) dem Preprocessing in Δ_{Adj}, das sich als $\Delta_{Adj} = Adj_v/Adj_n - 1$ berechnen lässt, verglichen. Zusätzlich werden die Adjazenzen nach den "trivialen" Kantenlöschungsregeln (4.10) und (4.11) angegeben (Adj_z) und der entsprechender prozentualer Unterschied zu der initialen Anzahl der Adjazenzen als Δ_{AdjK} angegeben. Dabei berechnet sich Δ_{AdjK} nach dem gleichen Prinzip wie Δ_{TW} und Δ_{Adj}. Interessant ist, wie viele Kanten alleine durch die ambitionierteren Preprocessing-Maßnahmen (Δ_{AdjP}) gelöscht werden konnten. Dieser Unterschied Δ_{AdjP} ergibt sich als die Differenz aus Δ_{Adj} und Δ_{AdjK} und ist in Prozentpunkten gerechnet.

Einen Überblick über die Auswirkung der Zeitfensterstraffung und Kantenelimination gibt Tabelle 5.3, die nach den oben erläuterten Angaben aufgebaut ist. Die Tabelle betrachtet ausschließlich Instanzen mit 50 Städten, da das Preprocessing im

Fall von 25 Städten keine nennenswerten Resultate zu erzielen vermag. Die, wie in Abschnitt 5.1.2 beschrieben, modifizierten Instanzen sind durch ein „m“ hinter der Instanz-Bezeichnung gekennzeichnet.

Tabelle 5.3.: Zeitfensterstraffung und Kantenelimination auf Original- und modifizierten Instanzen

Instanz	TW_v	TW_n	Δ_{TW}	Adj_v	Adj_n	Δ_{Adj}	Adj_z	Δ_{AdjK}	Δ_{AdjP}
C101.50	5479	5477	0,03%	2652	1216	54,15	1318	50,30%	3,85
C101.50m	32299	32297	0,006%	2652	1848	30,32	1949	26,51%	3,80
C103.50	31994	31388	1,93%	2652	2163	18,44	2288	13,73%	4,71
C103.50m	63988	63383	0,95%	2652	2351	15,15	2351	11,35%	3,80
C109.50	20472	20468	0,02%	2652	1895	28,54	1996	24,74%	3,80
C109.50m	40944	40944	0,00%	2652	2039	23,11	2140	19,31%	3,80
R101.50	960	959,6	0,04%	2652	808	69,53	909	65,72%	3,81
R101.50m	1920	1920	0,00%	2652	1115	57,96	1216	54,15%	3,81
R102.50	3422	3160	8,29%	2652	1429	46,12	1597	39,78%	6,34
R102.50m	13581	13320	1,96%	2652	2085	21,38	2186	17,57%	3,81
R106.50	4162	3900	6,71%	2652	1632	36,30	1793	30,02%	6,28
R106.50m	8324	8063	3,24%	2652	1930	27,22	2031	23,42%	3,80

Auffällig ist, dass die Zeitfensterstraffung generell sehr geringe Erfolge hat. Überdies nimmt ihre Effektivität im Fall der größeren Zeitfensterweiten zusätzlich ab. Dies mag sich wie folgt begründen: Sind die Zeitfenster zu lang, kann der Ansatz der Zeitfensterstraffung, welcher stark auf eine Einschränkung der potentiellen Vorgänger und Nachfolger eines Knotens angewiesen ist, sein Potential nicht entfalten, da lange Zeitfenster tendenziell weniger Präzedenzen, und somit weniger eindeutige Vorgänger- und Nachfolger-Konstellationen zwischen den Knoten erzwingen.

Teilweise sind einige Zeitfenster unberührt von der Straffung geblieben. Dies kommt nur auf den modifizierten Instanzen, also jenen mit größeren Zeitfenstern, vor. Beispielsweise konnten auf der C109.50 Instanz 0,019% der Zeitfensterweiten verkleinert werden, während auf der zugehörigen C109.50m Instanz keine Zeitfensterverringerungen erzielt werden konnten. Diese Beobachtung korrespondiert mit den Ausführungen im vorigem Absatz. Generell ist ein eindeutiger Trend zu sehen: Alle Instanzen mit künstlich vergrößerten Zeitfenstern weisen in den meisten Fällen doppelt so kleine Δ_{TW}-Werte als die Δ_{TW}-Werte der Originalinstanzen auf.

Aus den Ergebnissen lassen sich keine Verbindungen zwischen der Zeitfensterstraffung und Kantenelimination feststellen: Wenn vergleichsweise viel an Zeitfensterweite gestrafft werden konnte, so bedeutet es nicht zwangsläufig, dass auch mehr Kanten gelöscht werden konnten.

Im Vergleich zur Zeitfensterstraffung konnten bei der Löschung von Adjazenzen mehr Erfolge erzielt werden. Teilweise konnten bis zu 70% der ursprünglichen Kanten gelöscht werden, wie Abbildung 5.3 veranschaulicht.[7] Wenn man „Δ_{Adj}“ betrachtet, so beträgt die Differenz an vorhandenen Kanten vor und nach Preprocessing im Durchschnitt ca. 35% und in manchen Fällen sogar über 50%. Betrachtet man allerdings „Δ_{AdjK}“, so wird deutlich, dass die meisten Kanten schon allein mithilfe der “trivialen“ Kantenlöschungsregeln (4.10) und (4.11)[8] gelöscht werden konnten. Damit sind in etwa nur 5% der gelöschten Kanten Adj_z auf die ambitionierteren Preprocessing-Regeln zurück zu führen.

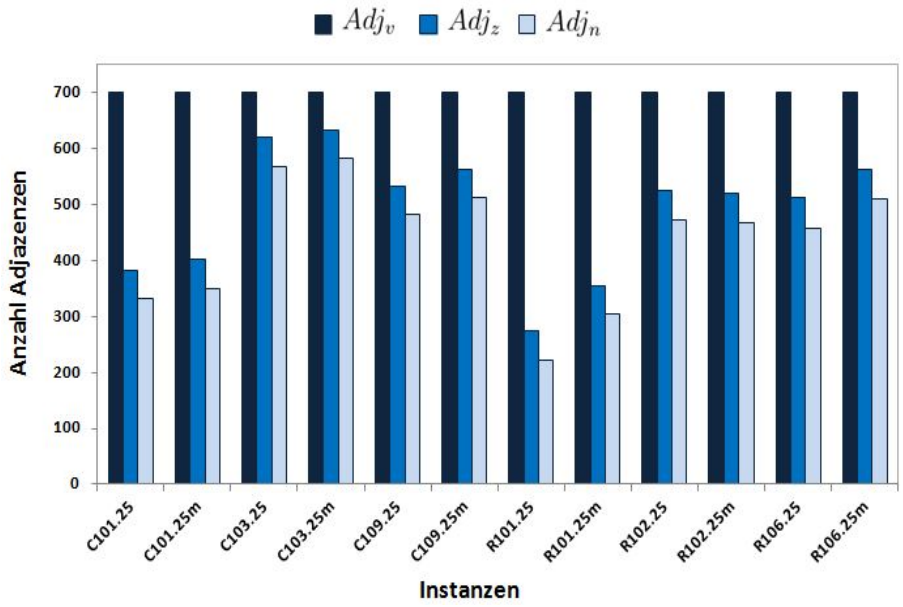


Abbildung 5.3.: Vergleich der Adjazenzenanzahl aus Tabelle 5.3

Interessanterweise werden auf den R-Instanzen im Durchschnitt mehr Kanten als auf den C-Instanzen gelöscht. Intuitiv gesehen wäre jedoch das Gegenteil plausibel, da die C-Instanzen so aufgebaut sind, dass eine günstige Tour tendenziell durch eine Punktewolke definiert ist. Das bedeutet, dass die Mehrzahl der Kanten von einer Punktewolke zu jeder anderen Punktewolke gelöscht werden können. Die ausgewählten R-Instanzen haben jedoch stellenweise deutlich engere Zeitfenster als die

[7] Die Benennung der Balken in Abbildung 5.3 gleicht der in Tabelle 5.3.
[8] Zur Erinnerung: (4.10) definiert sich als $R_i + t_{i,j} > D_j$, (4.11) definiert sich als $q_i + q_j > C$.

C-Instanzen, was wie bereits im obigen Abschnitt angedeutet wurde, eine effektivere Kantenlöschung begünstigt, und hier offenbar den Effekt der Standortverteilung auf die Kantenlöschung dominiert.

5.3.2. Experimente auf verschiedenen Modell-Formulierungen

Im Folgenden werden die Ergebnisse der 3-Index-Formulierung von Kohl et al. [41] und Letchford et al. [47] auf Instanzen mit 25 und 50 Städten vorgestellt. Die Laufzeit wurde auf maximal fünf Stunden begrenzt. Alle nachfolgenden Tabellen greifen auf die in Tabelle 5.4 präsentierten Spaltenbezeichnungen zurück.

Tabelle 5.4.: Bezeichner der Ergebnis-Tabellen 5.5, 5.6, 5.7, 5.8 und 5.9

Kürzel	Bezeichnung
Instanz	Instanz-Bezeichnung
$Opt*$	optimaler Zielfunktionswert
LB_r	untere Schranke im Wurzelknoten
$Iter_n$	durchschnittliche Anzahl B&B-Iterationen pro Knoten
Inc_T	Zeit in Sekunden, in der die erste ganzzahlige Lösung gefunden wurde
Opt_T	Zeit in Sekunden, in der die optimale Lösung gefunden wurde
Sol_T	für Wurzelknoten und Branch&Cut benötigte Zeit in Sekunden
$Total$	insgesamt benötigte Rechenzeit in Sekunden

Mit Preprocessing gerechnete Instanzen unterscheiden sich durch das kleine „p“ hinter der Instanzbezeichnung. Die Werte in Klammern geben die jeweilige B&B-Knotenanzahl, die bearbeitet wurde, bis entweder eine ganzzahlige Lösung in “Inc_T“ oder der optimale Wert in „Opt_T“ gefunden wurde. In der Spalte „Sol_T“ in Klammern findet sich die insgesamt bearbeitete B&B-Knotenanzahl. Zur Übersicht wird die Gesamtlösungszeit in der letzten Spalte „$Total$“ abgetragen. Der Tabelle 5.5 können die Ergebnisse entnommen werden.

Was die unteren Schranken betrifft, so fallen diese eher schwach aus und sind nur vereinzelt bei Instanzen mit Preprocessing geringfügig besser als ohne Preprocessing. Da die Effizienz des B&B-Algorithmus' unter anderem von der Güte der unteren Schranke im Wurzelknoten abhängt, beeinflusst dies entsprechend die Rechenzeit.

Allgemein sticht der große Bedarf an Lösungszeiten hervor. Acht aus insgesamt zwölf Instanzen haben die einschränkende Zeit von fünf Stunden voll ausgeschöpft und konnten trotz dessen nicht komplett abgeschlossen werden. Dabei konnte teil-

weise kein Optimum gefunden werden.[9]

Tabelle 5.5.: Ergebnisse des Preprocessing-Vergleichs auf Instanzen mit 25 Städten, Modell-Formulierung von Kohl et al. [41]

Instanz	$Opt*$	LB_r	$Iter_n$	Inc_T	Opt_T	Sol_T	$Total$
C101.25	191,30	191,3	691	0,67	0,70	0,28	4,57
				[0]	[0]	[0]	
C101.25p	191,30	191,3	661	0,59	0,62	0,23	3,70
				[0]	[0]	[0]	
C103.25	190,30	117,9	20,84	14,26	16485,44	17999,55	18056
				[97]	[335172]	[358768]	
C103.25p	190,30	124,3	25,16	5,90	17899,17	17999,74	18054
				[0]	[350212]	[352258]	
C109.25	191,30	75,60	22,87	3,84	15891,11	18000	18010
				[0]	[453529]	[490865]	
C109.25p	191,30	75,60	24,68	10,64	13002,29	18000	18013
				[130]	[295529]	[395236]	
R101.25	617,10	608	751	0,53	0,55	0,22	2,38
				[0]	[0]	[0]	
R101.25p	617,10	608	751	0,50	0,52	0,19	2,98
				[0]	[0]	[0]	
R102.25	547,10	355,70	23,09	8,41	n.e.[1]	9927,56	10289[2]
				[0]	[n.e.]	[249459]	
R102.25p	547,10	366	30,79	1,45	16232,9	17999,50	18010
				[0]	[491738]	[532407]	
R106.25	465,40	317,80	39,62	7,21	n.e.	18134	18142
				[0]	[n.e.]	[2275]	
R106.25p	465,40	317,10	35,59	7,50	n.e.	18135	18146
				[0]	[n.e.]	[950]	

[1] Die Abkürzung „n.e.“ steht für „nicht erreicht“.

[2] Rechnung aufgrund von Speicherkapazitätsmangel abgebrochen.

[9] auffällig ist, dass dies nur auf den R-Instanzen (außer R101.25) der Fall ist, obwohl diese im Durchschnitt sogar kleinere Zeitfensterweiten aufweisen, als die C-Instanzen, und aus dieser Hinsicht tendenziell leichter zu lösen sein sollten. Das Ergebnis legt die Vermutung nahe, dass die Cluster-Struktur der Kunden auf C-Instanzen den Solver ein Stück weit im zügigen Auffinden zulässiger und guter Lösungen unterstützt.

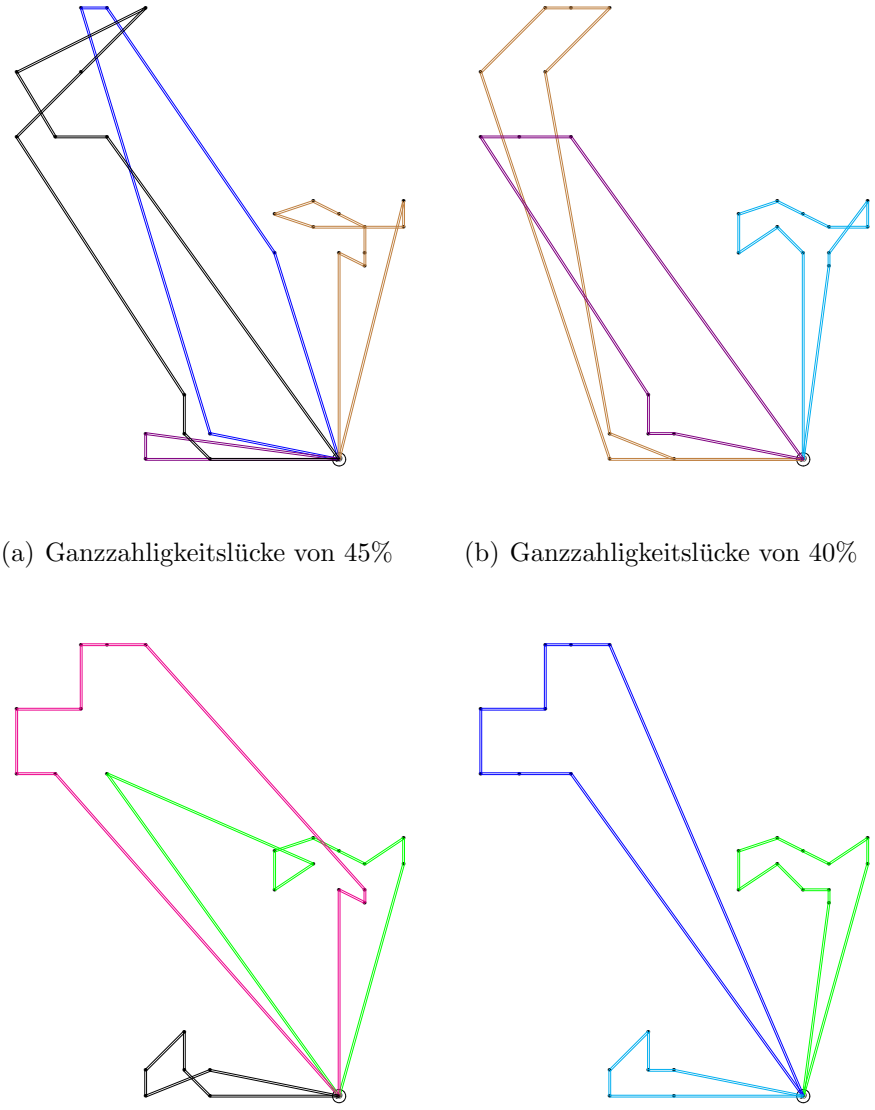
Dagegen konnten beispielsweise R101.25 und C101.25 sowohl mit als auch ohne Preprocessing innerhalb weniger Sekunden optimal gelöst werden. Der Grund hierfür sind die sehr engen Zeitfensterweiten, welche beide Instanzen aufweisen, sodass die Knotenbesuchsreihenfolgen zu einem Großteil bereits vordeterminiert ist. Dies sind auch die einzigen Instanzen, auf denen im Rahmen des Preprocessings mehr als 50% der Kanten gelöscht werden konnten.[10] Der zeitliche Vorsprung der mit Preprocessing gelösten Instanzen (C101.25 und R101.25) sowohl bei der Findung einer ganzzahligen Lösung als auch des Optimums, befindet sich jedoch in einem Bereich von Millisekunden. Das bedeutet, dass die Instanzen so trivial sind, dass mittels Preprocessings kein nennenswerter Lösungsgeschwindigkeit-Unterschied erzielt werden kann.

Auf den restlichen Instanzen zeichnet sich hingegen ein anderer Trend deutlich aus. Bei dem Großteil der Preprocessing-gestützten Instanzen werden sowohl die erste ganzzahlige Lösung als auch das Optimum (falls dieses erreicht wird) schneller gefunden (wenn auch nicht mittels dualer Schranke bestätigt), als im Fall ohne Preprocessing. Besonders drastisch fällt das Resultat im Fall der Instanz R102.25 aus: Hier musste der Branch-and-Cut-Algorithmus im Fall ohne Preprocessing sogar nach weniger als 3 Stunden aus Gründen von Speicherkapazitätsmängeln abgebrochen werden, wogegen im Fall mit Preprocessing im Moment des zeitlich bedingten Abbruchs nach fünf Stunden das Optimum gefunden war.

Bemerkenswert ist, dass (abgesehen von den Instanzen, auf welchen das Optimum schon im Wurzelknoten gefunden wurde) die B&B-Iterationen pro Knoten bei Preprocessing-basierten Instanzen in den meisten Fällen etwas höher liegen, als im Fall ohne Preprocessing. Dies mag darauf zurück zu führen sein, dass die Preprocessing-Maßnahmen der zugrunde liegenden Modell-Matrix eine komplexere Struktur geben, wodurch nach einem Branching-Schritt mehr Simplex-Iterationen benötigt werden, um zum neuen Optimum des jeweils resultierenden Nachfolgeknotens zu gelangen. Der beschriebene Sachverhalt führt dazu, dass beim zeitlich bedingten Abbruch nach jeweils fünf Stunden im Fall ohne Preprocessing teilweise mehr Knoten bearbeitet werden konnten, als im Fall mit Preprocessing, was zunächst nicht intuitiv erscheint. Die Güte der bearbeiteten Knoten, hinsichtlich einer schneller Auffindung des gesuchten Optimums, ist im Fall der Preprocessing-basierten Instanzen jedoch, wie oben bereits angesprochen, höher.

[10] Auf der Instanz C101.25 wurden 54% und auf R101.25 69% der Kanten gelöscht.

An dieser Stelle bietet sich eine Diskussion über die Symmetrie, die im Abschnitt 3.2 vorgestellt wurde, an. Betrachtet man vereinzelte Lösungsergebnisse in unterschiedlichen Stadien des Branch-and-Bound-Algorithmus' bildlich, so wird der Effekt der Symmetrie deutlich. In Abbildung 5.4 sind zu vier verschiedenen Ganzzahligkeitslücken gehörende Lösungen, extrahiert aus dem B&B-Lösungsvorgang der Instanz C101.25, visualisiert.



(a) Ganzzahligkeitslücke von 45%

(b) Ganzzahligkeitslücke von 40%

(c) Ganzzahligkeitslücke von 30%

(d) Ganzzahligkeitslücke von 15%

Abbildung 5.4.: Symmetrie in verschiedenen Lösungsstadien, basierend auf den Ergebnissen aus Tabelle 5.5

Ab Lösung b) kristalliesieren sich drei verschiedene "Tourbezirke" heraus, welche das Muster des finalen Optimums erahnen lassen. Innerhalb dieser Tourbezirke va-

riiert die Fahrzeugbelegung offenbar sowohl im Schritt von b) nach c)[11], als auch im Schritt von c) nach d).[12] Da in beiden Gegenüberstellungen nicht nur die Tourzuordnung, sondern die komplette ausgewählte Fahrzeugflotte variiert wurde, ist bei diesen Beobachtungen vom Symmetrie-Fall 2 im Sinne des Abschnittes 3.2 zu sprechen. Stellt man überdies die Lösungen b) und d) einander gegenüber, so ist eine Mischung aus Symmetrie-Fall 1 und Symmetrie-Fall 2 zu erkennen: das "hellblaue" Fahrzeug übernimmt zwar den Tourbezirk des "braunen" Fahrzeugs, selbiges wechselt jedoch nicht im Gegenzug zum alten Tourbezirk des "hellblauen" Fahrzeugs, sondern verschwindet vielmehr komplett aus der verwendeten Fahrzeugflotte.

Subsumiert auf den B&B-Algorithmus legen die Beobachtungen den Verdacht nahe, dass sämtliche der betrachteten Lösungen aus sehr unterschiedlichen Teilbäumen des B&B-Baumes stammen.

Experimente auf Letchford et al. Modell-Formulierung mit 25 Städten Nun werden die Ergebnisse für die Instanzen mit 25 Städten auf der Letchford-Formulierung in Tabelle 5.6 vorgestellt. Da dieses Modell ohne Fahrzeugindizierung auskommt, und somit zum einen weniger komplex, zum anderen nicht anfällig für die Problematik der Symmetrie ist, sollten geringere Rechenzeiten zu erwarten sein.

Der Tabelle 5.6 können die entsprechenden Ergebnisse entnommen werden. Sie weist dieselben Spalten-Bezeichnungen wie Tabelle 5.5 auf.

In der Tat sind die unteren Schranken im Vergleich zu der 3-Index-Formulierung deutlich besser und zwar bei allen Instanzen, wobei die unteren Schranken mit Preprocessing sich kaum von den unteren Schranken ohne Preprocessing unterscheiden. Teilweise wird mit der unteren Schranke sofort das Optimum gefunden, nämlich auf den Instanzen C101.25 und R101.25. Laut Letchford et al. [47], resultieren bei der Relaxation dieser Formulierung bessere Schranken, falls die Zeitfenster verkürzt werden. Aus diesem Grund soll sich Preprocessing besonders gut als Lösungsunterstützende Maßnahme für diese Formulierung eignen.

Auffällig sind die Rechenzeiten, welche allesamt dramatisch kürzer sind, als im Fall des Modells von Kohl et al. [41]. Abgesehen von einem „Ausreißer" (C103.25p) lassen sich alle Instanzen innerhalb einer Minute lösen. Trotz der geringen Laufzeiten lassen sich kleine Unterschiede innerhalb der Instanzen identifizieren. Das Preproces-

[11] statt des "braunen" Fahrzeugs wird ein "schwarzes", statt des "lillalen" Fahrzeuges ein "pinkes" und statt des "hellblauen" Fahrzeuges ein "grünes" Fahrzeug verwendet.

[12] statt des "schwarzen" Fahrzeugs wird ein "hellblaues" und statt des "pinken" Fahrzeuges ein "dunkelblaues" Fahrzeug verwendet.

sing scheint in den meisten Fällen geringfügig bessere Gesamtrechenzeiten zu liefern, wie Abbildung 5.5 (oben)[13] deutlich macht.

Tabelle 5.6.: Ergebnisse des Preprocessing-Vergleichs auf Instanzen mit 25 Städten, Modell-Formulierung von Letchford [47]

Instanz	$Opt*$	LB_r	$Iter_n$	Inc_T	Opt_T	Sol_T	$Total$
C101.25	191,30	191,30	465	0,06	0,19	0,09	2,42
				[0]	[0]	[0]	
C101.25p	191,30	191,30	460	0,06	0,17	0,08	2,44
				[0]	[0]	[0]	
C103.25	190,30	166,75	154,13	1,59	4,34	51,12	64,5
				[0]	[0]	[268]	
C103.25p	190,30	166,91	111,58	0,45	1,44	175,49	177,21
				[0]	[0]	[4343]	
C109.25	191,30	168,95	168,13	0,33	1,25	21,64	23,48
				[0]	[0]	[374]	
C109.25p	191,30	168,78	141,04	0,27	0,87	19,56	21,32
				[0]	[0]	[419]	
R101.25	617,10	617,10	241	0,11	0,13	0,08	3,85
				[0]	[0]	[0]	
R101.25p	617,10	617,10	241	0,14	0,19	0,09	2,51
				[0]	[0]	[0]	
R102.25	547,10	468,99	63,92	0,48	9,08	33,51	36,69
				[0]	[270]	[1757]	
R102.25p	547,10	468,33	62,22	0,27	7,13	33,50	36,44
				[0]	[309]	[2267]	
R106.25	465,40	417,37	88,82	0,44	1,36	55,46	57,74
				[0]	[0]	[2275]	
R106.25p	465,40	417,56	77,62	0,44	0,83	27,28	29,42
				[0]	[0]	[950]	

Besonders deutlich wird der Mehrwert des Preprocessings jedoch bei der zügigen Findung des Optimums Opt_T, wie Abbildung 5.5 (unten) veranschaulicht. Hierbei kommen Preprocessing-basierte Instanzen in 70% bis 20% der ohne Preprocessing benötigten Zeit zur einer ganzzahligen Lösung beziehungsweise zum Optimum.

[13] Der "Ausreißer" ist durch die graue Farbe gekennzeichnet.

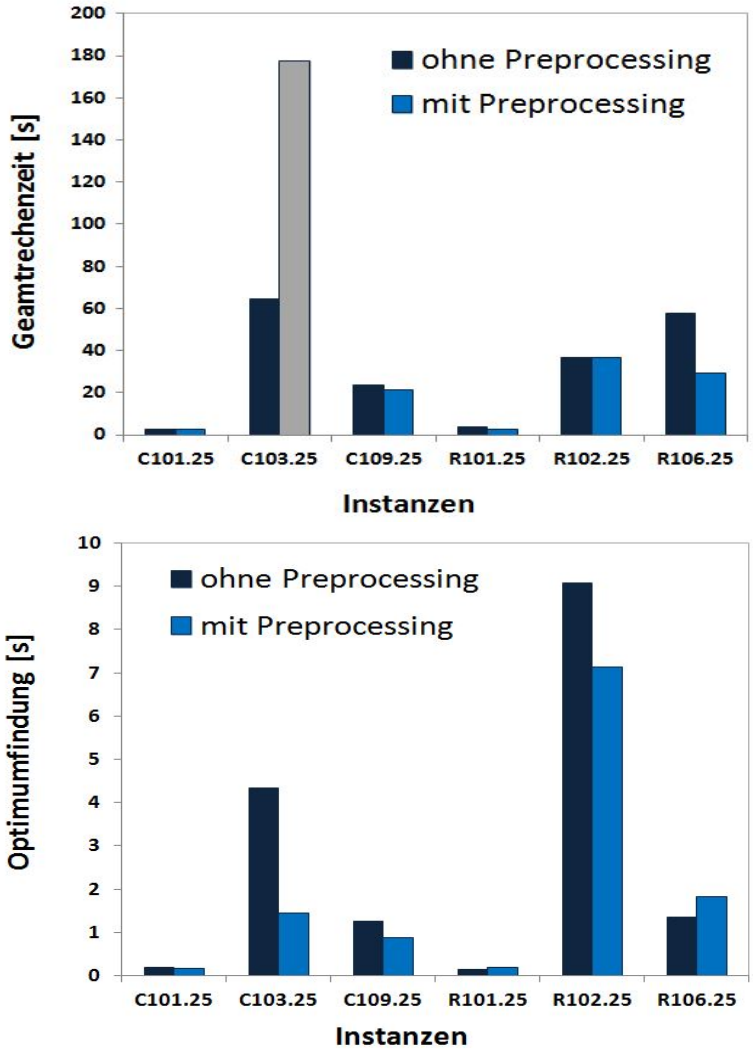


Abbildung 5.5.: Vergleich der Gesamtlösungszeiten und der benötigten Lösungszeit zur Findung des Optimums aus Tabelle 5.6

Dagegen sind die Lösungszeiten, welche für den gesamten B&B-Algorithmus, inklusive Wurzelknoten, verwendet werden, weniger unterschiedlich, womit der zeitliche Vorsprung der Preprocessing-basierten Instanzen beim Auffinden des Optimums, nicht in eine geringere Gesamtlösungszeit umgesetzt werden kann. Der mit Preprocessing verbundene Mehraufwand kann daher bei diesen Instanzgrößen und Formulierung nicht kompensiert werden.

Ein „Ausreißer“, nämlich die Instanz C103.25, weist mit Preprocessing deutlich höhere Laufzeit auf, dafür werden die erste ganzzahlige Lösung und das Optimum deutlich schneller gefunden, wie Abbildung 5.5 verdeutlicht. Im Gegensatz dazu, wurde im Fall ohne Preprocessing viel Zeit benötigt, um eine ganzzahlige und später optimale Lösung zu finden. Sobald diese jedoch gefunden wurde, konnte ihre Op-

timalität schnell bestätigt werden. Die Unterlegenheit der Preprocessing-basierten Instanz kann in diesem Fall auf "unglückliche" Entscheidungen im B&B-Baum zurückzuführen sein.

Bemerkenswert ist zudem der Unterschied zu der 3-Index-Formulierung im Hinblick auf die Anzahl der benötigten Iterationen pro B&B-Knoten. Hier liegt die exakt umgekehrte Situation vor: Die Iterationen pro Knoten sind mit Preprocessing geringer als ohne. Dies mag daran liegen, dass die verhältnismäßig "ungeschickte" 3-Index-Formulierung, im Vergleich zur 2-Index-Formulierung, dem Preprocessing mehr Angriffsfläche in puncto Komplexitätsreduktion bietet, wodurch die Strukturveränderung der zugrundeliegenden Modell-Matrix aufgewogen werden kann.

Einmal mehr besteht in der R-Kategorie die Tendenz dazu, mehr Rechenzeit zu beanspruchen als in der C-Kategorie. Somit scheint die Tatsache, dass auf den C-Instanzen durch die Kunden-Cluster günstige Tourenverteilungen stellenweise zwingend vordeterminiert sind, unabhängig von der gewählten Modell-Formulierung, das Lösungsverhalten des B&B-Algorithmus' positiv zu beeinflussen.

Experimente auf Kohl et al. Modell-Formulierung mit 50 Städten Analog zu den Ergebnissen auf Instanzen mit 25 Städten werden Instanzen mit 50 Städten hinzugezogen und auf beiden Modell-Formulierungen getestet. Mit 25 Städten mehr erhöhen sich die Besuchsreihenfolge- und Tourbildungs-Möglichkeiten über die Städte drastisch, was sich in den Laufzeiten wiederspiegelt. Während auf den Instanzen mit 25 Städten auf 3 Instanzen kein Optimum erreicht werden konnte, sind es auf Instanzen mit 50 Städten 8 Instanzen, auf denen das Optimum nicht erreicht wird. Auf zwei Instanzen (C109.50 und C109.50p) wurde nicht einmal eine ganzzahlige Lösung gefunden, wie Tabelle 5.7, welche in völliger Analogie zur Tabelle 5.5 angelegt ist, entnommen werden kann.

Auf den Preprocessing-gestützten Instanzen ergeben sich bei Abschluss der fünf Stunden Grenze deutlich geringere Ganzzhaligkeits-Lücken als im Fall ohne Preprocessing.[14]

Instanz C103.50p liegt bei der Findung einer ganzzahligen Lösung zeitlich deutlich hinter der Instanz ohne Preprocessing C103.50, diese musste allerdings, im Gegensatz zu der Instanz mit Preprocessing, aufgrund von Speicherkapazitäts-Mangel nach fast drei Stunden abgebrochen werden. In allen anderen Fällen wird mit Preprocessing die ganzzahlige Lösung schneller erreicht als ohne Preprocessing.

[14] Selbiges ist auf den 25-Städte-Instanzen zu beobachten, welche die fünf-Stunden-Grenze komplett ausnutzen.

Tabelle 5.7.: Ergebnisse des Preprocessing-Vergleichs auf Instanzen mit 50 Städten, Modell-Formulierung von Kohl et al. [41]

Instanz	$Opt*$	LB_r	$Iter_n$	Inc_T	Opt_T	Sol_T	$Total$
C101.50	362,40	357,90	0	2,65	7,32	0,45	11,02
				[0]	[0]	[0]	
C101.50p	362,40	361,20	0	2,26	8,00	0,42	11,50
				[0]	[0]	[0]	
C103.50	361,40	193,02	21,63	114,68	n.e.	9720,75	10299[1]
				[390]	[n.e.]	[64574]	
C103.50p	361,40	193,20	22,30	963,77	n.e.	17999,60	18001
				[2994]	[n.e.]	[119496]	
C109.50	362,40	154,40	0	n.e.	n.e.	17997	18013
				[n.e.]	[n.e.]	[81598]	
C109.50p	362,40	154,40	0	n.e.	n.e.	17997	18011
				[n.e.]	[n.e.]	[67879]	
R101.50	1044	1029,30	0,65	5,35	7,16	18,73	21,77
				[0]	[0]	[1152]	
R101.50p	1044	1029,30	30,06	3,87	4,18	16,62	19,75
				[0]	[0]	[1018]	
R102.50	909	566,30	72,38	1175,92	n.e.	17999,77	18182
				[1407]	[n.e.]	[52320]	
R102.50p	909	641,30	73,01	840,19	n.e.	18000	18167
				[1497]	[n.e.]	[59960]	
R106.50	793	506,60	65,56	1449,09	n.e.	18000	18219
				[1407]	[n.e.]	[57769]	
R106.50p	793	506,70	62,57	1067,70	n.e.	17999,43	18149
				[1587]	[n.e.]	[53929]	

[1] Rechnung aufgrund von Speicherkapazitätsmangel abgebrochen.

Da sowohl auf Instanzen mit Preprocessing als auch ohne Preprocessing die meisten Instanzen kein Optimum erreichen, kann die Optimum-Findung nicht adäquat beurteilt werden. Ähnlich verhält es sich mit der Gesamtrechenzeit, die auf fast allen Instanzen die 18.000 Sekunden-Grenze erreicht.

Experimente auf Letchford et al. Modell-Formulierung mit 50 Städten Analog zu den Ergebnissen auf der Modell-Formulierung von Kohl et al. [41] erfolgt eine

Übersicht der Ergebnisse auf der Modell-Formulierung von Letchford et al. [47], die Tabelle 5.8 zu entnehmen sind.

Ein Vergleich innerhalb der Instanzen auf der Modell-Formulierung von Letchford et al. [47] lässt erkennen, dass auf den R-Instanzen die Optimum-Findung viel mehr Zeit in Anspruch nimmt als auf den C-Instanzen, die Optimum-Bestätigung dauert offensichtlich auf der C-Kategorie deutlich länger, sodass die Gesamtlösungszeiten der beiden Kategorien im Durchschnitt beinahe gleich sind.

Tabelle 5.8.: Ergebnisse des Preprocessing-Vergleichs auf Instanzen mit 50 Städten, Modell-Formulierung von Letchford [47]

Instanz	$Opt*$	LB_r	$Iter_n$	Inc_T	Opt_T	Sol_T	$Total$
C101.50	362,40	361,57	2116	0,62	0,73	0,45	2,47
				[0]	[0]	[0]	
C101.50p	362,40	361,39	2155	0,55	0,63	0,42	2,49
				[0]	[0]	[0]	
C103.50	361,40	326,17	268,35	5,84	191,19	1303,51	1307,25
				[0]	[0]	[268]	
C103.50p	361,40	316,26	291,43	11,48	12,43	762,95	765,68
				[0]	[0]	[4343]	
C109.50	362,40	332,53	339,45	4,27	9,52	5272,37	5275,85
				[0]	[0]	[374]	
C109.50p	362,40	332,06	331,96	4,07	8,53	4120,39	4122,55
				[0]	[0]	[419]	
R101.50	1044	1036,8	1181	0,50	0,73	0,42	1,72
				[0]	[0]	[0]	
R101.50p	1044	1037,77	1316	0,47	0,69	0,41	1,91
				[0]	[0]	[0]	
R102.50	909	806,79	128,97	3,18	278,73	612,99	615,15
				[0]	[2150]	[8447]	
R102.50p	909	807,11	140,75	3,07	1027,05	1075,67	1078,98
				[0]	[11094]	[12042]	
R106.50	793	711,81	217,93	7,97	2295,66	6224,29	6228,21
				[0]	[9661]	[39487]	
R106.50p	793	713,59	177,95	10,48	1155,09	5237,41	5241,74
				[0]	[4008]	[30153]	

Die größte Zeitspanne wird generell bei der Optimumsuche und -bestätigung beobachtet. In den meisten Fällen wird eine ganzzahlige Lösung vergleichsweise schnell gefunden, bis jedoch das Optimum gefunden und tatsächlich als solches identifiziert werden kann, wird viel zusätzliche Zeit in Anspruch genommen. Besonders gut kann dies auf der C109.50 Instanz beobachtet werden. Das Optimum ist nach knapp 10 Sekunden gefunden, bis es als Solches identifiziert, das Lösen also komplett abgeschlossen werden kann, dauert es jedoch über eine Stunde.

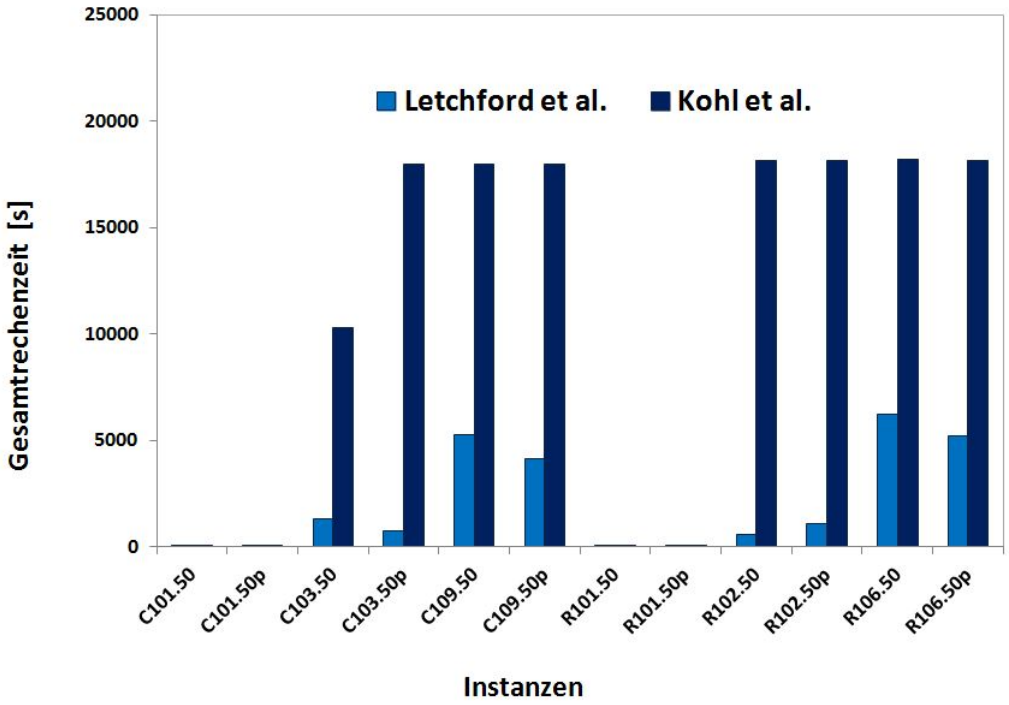


Abbildung 5.6.: Vergleich der Gesamtlösungszeiten aus Tabellen 5.5 und 5.6

Im Vergleich zu den Ergebnissen aus Tabelle 5.7 konnten alle Instanzen unter einer Zeit von zwei Stunden gelöst werden, sodass die zeitliche Einschränkung von fünf Stunden nicht greifen musste. So konnte auf allen Instanzen sowohl eine ganzzahlige Lösung als auch das Optimum erreicht werden, wohingegen in Tabelle 5.7 zwei drittel der Instanzen diesen Status nicht erreichen konnten. Abbildung 5.6 visualisiert den enormen Unterschied in den benötigten Rechenzeiten der beiden Formulierungen. Auch die unteren Schranken auf der Modell-Formulierung von Lethcford et al. [47] sind deutlich besser.

Experimente auf modifizierten Instanzen Nun können die zusätzlichen Modifikationen der Instanzen verglichen werden. Die Spaltenbezeichnungen in Tabelle 5.9 sind die gleichen, wie in den vorherigen Tabellen, außer der letzten Spalte δ_{SolT}, die die prozentuale Veränderung der Gesamtrechenzeit zu der Gesamtrechenzeit mit Ori-

ginalzeitfensterweiten angibt. Positive Werte in dieser Spalte zeigen, wie viel mehr Zeit die modifizierte Instanz zum Lösen benötigt. Negative Werte zeigen, wie viel schneller die modifizierte Instanz gelöst werden konnte.

Tabelle 5.9.: Ergebnisse des Preprocessing-Vergleichs auf modifizierten Instanzen mit 25 Städten, Modell-Formulierung von Letchford [47]

Instanz	$Opt*$	LB_r	$Iter_n$	Inc_T	Opt_T	Sol_T	$Total$	δ_{SolT}
C101.25m	191,30	191,20	466	0,17	0,17	0,11	2,84	17,09%
				[0]	[0]	[0]		
C101.25mp	191,30	191,20	468	0,19	0,19	0,09	1,96	-19,90%
				[0]	[0]	[0]		
C103.25m	190,30	117,48	144,33	0,75	n.e.	17999	18003	27911%
				[0]	[n.e.]	[414687]		
C103.25mp	190,30	161,27	106,73	0,41	2,00	37,53	40,58	-22,89%
				[0]	[0]	[1304]		
C109.25m	187,90	159,40	180,76	0,42	75,29	78,59	80,33	342,15%
				[0]	[1385]	[1461]		
C109.25mp	187,90	159,40	331,96	0,08	17,74	22,25	24,60	15,38%
				[0]	[190]	[344]		
R101.25m	546,40	546,40	382	0,08	0,16	0,13	1,85	-48,05%
				[0]	[0]	[382]		
R101.25mp	546,40	546,40	442	0,11	0,11	0,06	1,58	-62,87%
				[0]	[0]	[442]		
R102.25m	485,10	422,26	78,54	0,34	2,76	50,19	51,74	40,99%
				[0]	[0]	[2592]		
R102.25mp	485,10	421,64	90,35	0,34	44,68	57,69	59,40	62,97%
				[0]	[1980]	[2435]		
R106.25m	417,90	358,27	146,49	0,72	151,56	156,92	169,72	293,93%
				[0]	[2526]	[2745]		
R106.25m	417,90	358,19	120,51	0,37	59,92	110,54	114,39	388,75%
				[0]	[1250]	[2617]		

Das „m“ hinter jeder Instanzbezeichnung steht für „modifiziert“. Für die Experimente mit modifizierten Instanzen werden nur die Instanzen der Größenordnung von 25 Städten ausgewählt, um exemplarisch die Auswirkung einer Verdoppelung der Zeitfensterweiten zu veranschaulichen. Diese Experimente werden nur auf der

2-Index-Modellformulierung von Letchford et al. [47] durchgeführt, da die Kohl et al. Modell-Formulierung bereits ohne vergrößerte Zeitfenster Rechenzeiten aufweist, welche die Ergebnisinterpretationen erschweren, und davon auszugehen ist, dass erweiterte Zeitfenster diesen Effekt erheblich verstärken.

Hierbei ist anzumerken, dass sich durch die Erweiterung der Zeitfensterweiten neue optimale Zielfunktionswerte ergeben können, weil dies die Zeitfensterrestriktionen lockert und sich dadurch neue, zulässige Besuchsreihenfolgen, beziehungsweise Touren, ergeben, welche günstiger als die alten Optima sein können.

Besonders interessant gestaltet sich die Veränderung der Gesamtrechenzeit: In 80% der Fälle wurden die modifizierten Instanzen deutlich langsamer gelöst als mit Original-Zeitfensterweiten. Das bedeutet, dass mit größeren Zeitfenstern die Rechenzeit tendenziell steigt. Das anschaulichste Beispiel bietet die Instanz C103.25m, auf der innerhalb der Rechenzeitbegrenzung von fünf Stunden kein Optimum erreicht werden konnte, wohingegen die Rechnung auf der selben Instanz mit Originalzeitfensterweiten etwas mehr als eine Minute Zeit in Anspruch genommen hat. Die selbe Instanz, gelöst mit Preprocessing benötigt zwei Sekunden, um das Optimum zu finden.

Die Tendenz, dass Preprocessing schnellere Rechenzeiten liefert, bleibt. Zwar sind die Unterschiede nicht mehr so deutlich wie in Tabellen 5.5 oder 5.6 (dies mag wiederum auf die generelle Schwächung zurückzuführen sein, der sich die Elemente des Preprocessings bei erweiterten Zeitfenstern gegenüber sehen), in den meisten Fällen sind Preprocessing-basierte Instanzen trotzdem schneller im Finden des Optimums, im Bestätigen des Optimums, und in der Gesamtlösungszeit.

5.3.3. Zahlen zu den zeitlichen und topologischen Einflüssen einer Instanz

Zum Schluss werden weitere Experimente zu Kennzahlen für Zeitfensterweiten und topologischen Einfluss vorgestellt.

Zahlen zu den zeitlichen Einflüssen einer Instanz Zur Bestimmung der Zeitfensterweiteneigenschaften einer Instanz wäre eine Maßeinheit sinnvoll. Zur Messung der Zeitfensterweiten schlägt Desrochers [22] zwei Kennzahlen vor: Zum Einen

$$TW_1 = \frac{\overline{(D_i - R_i)}}{max_i\{D_i\} - min_i\{R_i\}}, \tag{5.1}$$

das Verhältnis zwischen der durchschnittlichen Zeitfenstergröße

$$\overline{(D_i - R_i)} = \sum_{i \in V_c} D_i - R_i / |V| \tag{5.2}$$

und dem Zeithorizont. Der Wertebereich von TW_1 liegt zwischen 0 und 1, wobei der Wert 1 bei komplett deckungsgleichen Zeitfenstern auftritt. Je kleiner der Wert, desto enger sind tendenziell die Zeitfenster (im Verhältnis zum Zeithorizont), über die Verteilung derselbigen trifft die Kennzahl jedoch keine Aussage. Zum Anderen wird die Kennzahl

$$TW_2 = \frac{\overline{(D_i - R_i)}}{\overline{d_{i,j}}} \tag{5.3}$$

vorgestellt, die das Verhältnis zwischen der durchschnittlichen Zeitfensterweite und der durchschnittlichen Reisestrecke $\overline{d_{i,j}} = \sum_{(i,j) \in A} d_{i,j} / |A|$ angibt. Werte nahe Null können Indikatoren für enge Zeitfensterweiten sein. Alle höheren Werte sprechen für größere Zeitfensterweiten. Bei sehr großen Werten treten die zeitlichen Aspekte der Instanz in den Hintergrund und es handelt sich dadurch vornehmlich um ein einfaches Vehicle Routing Problem. Die Kennzahl 2 trifft ebenfalls keinerlei Aussage über die Verteilung der Zeitfenster.

In Tabelle 5.10 werden die zwei Kennzahlen TW_1 und TW_2 verglichen, um eine Aussage über diese machen zu können.

Betreffend der Zeitfensterweiten-Kennzahlen lassen sich zwar die Zeitfensterweiten der gegebenen Instanzen qualitativ untereinander vergleichen, allerdings bietet insbesondere Kennzahl TW_1 keine eindeutige Prognose für das Lösungsverhalten an, wie man auf Basis der Experimente mit modifizierten Instanzen, die eindeutig zeigen, dass große Zeitfenster längere Laufzeiten verursachen, erkennen kann: Die Kennzahl TW_1 bleibt von der Zeitfensterweiten-Verdopplung unberührt, da sich der Faktor 2 aus Zähler und Nenner herauskürzt, das Lösungsverhalten ist durch die Zeitfenstervergrößerung hingegen entscheidend beeinflusst.

Tabelle 5.10.: Kennzahlen zu den Zeitfensterweiten, Ergebnisse

Instanz	$\overline{(D_i - R_i)}$	$\overline{d_{i,j}}$	$max_i\{D_i\}$	$min_i\{R_i\}$	TW_1	TW_2
C101.25	107,68	18,89	1236	45	0,09	5,70
C101.50	45,21	99,78	1236	37	0,03	0,45
C103.25	635,62	18,89	1236	45	0,53	33,64
C103.50	603,10	99,78	1236	43	0,50	6,04
C109.25	393,69	18,89	1236	360	0,45	20,84
C109.50	377,18	99,78	1236	360	0,43	3,78
R101.25	18,46	33,04	230	10	0,08	0,56
R101.50	14,31	133,80	230	10	0,07	0,11
R102.25	69,85	33,04	230	10	0,32	2,11
R102.50	59,78	133,80	230	10	0,27	0,44
R106.25	83,69	33,04	230	30	0,42	2,53
R106.50	74,04	133,80	230	30	0,37	0,55

Kennzahl TW_2 mag hingegen eine geringe Tauglichkeit als Indikator für das Lösungsverhalten einer Instanz zugesprochen werden (wobei hohe Kennzahlenwerte lange Lösungszeiten, und niedrige Kennzahlenwerte kurze Lösungszeiten andeuten sollten). So hat die Instanz C103.25 beispielsweise einen sehr hohen TW_2-Wert im Vergleich zu anderen Instanzen, was bedeutet, dass diese Instanz sehr weite Zeitfenster, im Verhältnis zu der Skalierung ihrer Distanzmatrix, aufweist, und ist zugleich Instanz mit der längsten Lösungszeit auf den beiden Modellformulierungen von Kohl et al. und von Letchford et al.[15] Andererseits weist die Instanz C109.25 den zweithöchsten TW_2-Wert auf, wird jedoch schneller gelöst, als die R-Instanzen, die einen deutlich geringeren TW_2-Wert haben. Dies wiederum spricht dafür, dass ein Vergleich der Kennzahl TW_2, zwecks Prognosen des Lösungsverhaltens, nur innerhalb von Instanzen Sinn macht, welche die selbe topologische Beschaffenheit (Cluster oder Random) aufweisen.

Eine grafische Veranschaulichung der Kennzahlen sowie der zugehörigen Gesamtlösungszeiten auf der Modell-Formulierung von Letchford et al. [47] werden in Abbildung 5.7 dargestellt. Betrachtet man die Kennzahl TW_1, so ist es ersichtlich, dass Instanzen C103.25-C109.50 demnach die weitesten Zeitfenster aufweisen. Ent-

[15] Vgl. hierzu Tabellen 5.5 und 5.6

sprechend sollten diese Instanzen mehr Zeit zum Lösen aufwenden. Vergleicht man jedoch die Gesamtrechenzeiten[16], so kann keine eindeutige Verbindung zwischen der Kennzahl und der Rechenzeit hergestellt werden: beispielsweise wird die Instanz C109.25 deutlich schneller gelöst, als C109.50, obwohl diese ähnlich weite Zeitfenster aufweisen.

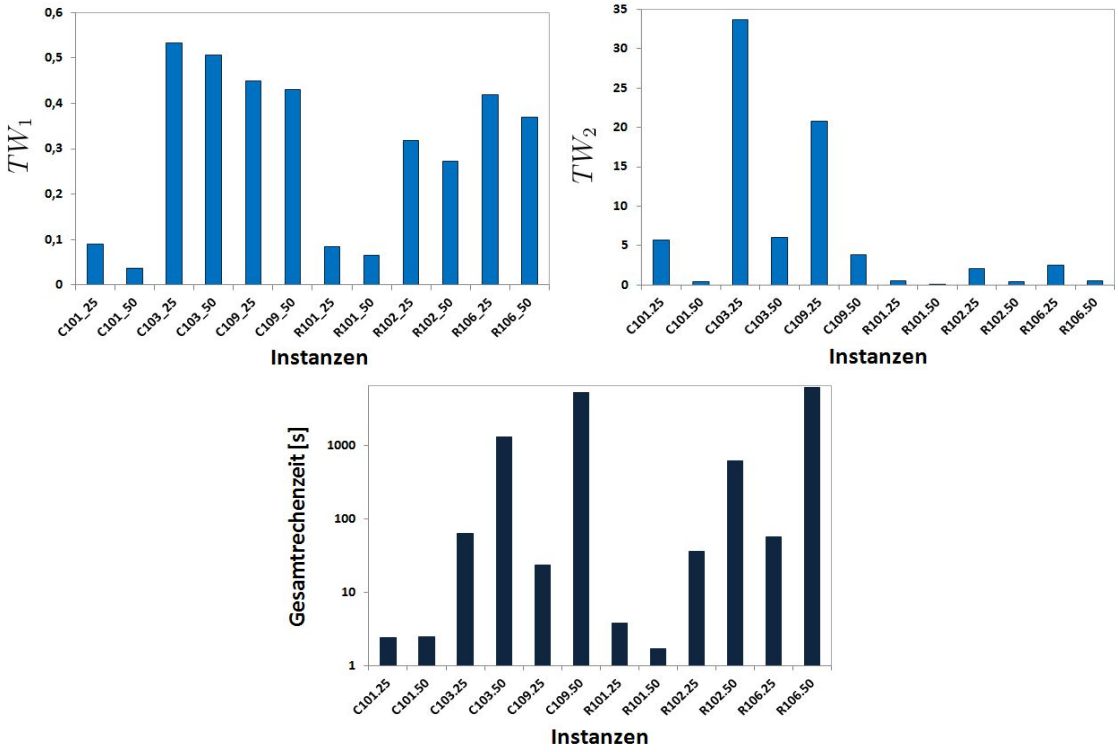


Abbildung 5.7.: Zusammenhang zwischen den Kennzahlen aus Tabelle 5.10 und den Gesamtrechenzeiten aus Tabellen 5.6 und 5.8

Ähnliches Bild ergibt sich bei der Betrachtung der Kennzahl TW_2: hohe Kennzahlen-Werte bedeuten nicht zwangsläufig lange Rechenzeiten und niedrige Kennzahlen-Werte signalisieren nicht unbedingt eine schnellere Lösung. Abbildung 5.7 macht deutlich, dass die Kennzahl TW_2 die Anzahl der Städte nicht berücksichtigt. Diese hat jedoch einen bedeutenden Einfluss auf die Rechenzeit. So weisen Instanzen R106.25 und R106.50 vergleichsweise kleine TW_2-Werte auf, in der Rechenzeit der beiden Instanzen bestehen jedoch große Unterschiede, die auf die Städteanzahl zurückzuführen sind.

Zahlen zu den topologischen Einflüssen einer Instanz Aufgrund der zahlreichen unterschiedlichen Parametereigenschaften konnte in den vorherigen Kapiteln keine

[16] Zur besseren Übersicht wird die Gesamtrechenzeit in der Abbildung 5.7 auf einer logarithmischen Skala abgetragen.

eindeutige Verbindung zwischen der geographischen Beschaffenheit der Kundenverteilung (Cluster und Random) und dem Lösungsverlauf einer Instanz festgestellt werden. Hierzu seien nun Experimente auf den modifizierten Instanzen herangezogen, welche bis auf die Topologie der Kunden von ihren Parametereigenschaften her gleichgeschaltet sind. Tabelle 5.11 können die Ergebnisse entnommen werden. Die Angaben zum Optimalen Zielfunktionswert der unteren Schranke werden vernachlässigt, da nicht zu jeder modifizierten Instanz ein optimaler Zielfunktionswert bekannt ist und dies nicht den Schwerpunkt dieser Reihe an Ergebnissen darstellt. Im Mittelpunkt stehen die Rechenzeiten. Alle Experimente wurden auf der Formulierung von Letchford et. al [47] mit 50 Städte-Instanzen durchgeführt. Die verwendeten C-Instanzen behalten ihre herkömmliche Benennung. Die verwendeten R-Instanzen gehen aus der jeweiligen C-Instanz, wie in Abschnitt 5.1.2 beschrieben, hervor. Hierbei kennzeichnet ein "R/" zu Beginn des Dateinamens jeweils die Zugehörigkeit der entsprechenden R-Instanz zu der ursprünglichen C-Instanz.

Tabelle 5.11.: Einfluss der topologischen Beschaffenheit auf den Lösungsverlauf, Ergebnisse

Instanz	$Iter_n$	Inc_T	Opt_T	Sol_T	$Total$
C101.50	2139	0,94	1,17	0,59	5,00
		[0]	[0]	[0]	
R/C101.50	3094	1,14	1,44	0,81	3,71
		[0]	[0]	[0]	
C103.50	294,16	2,57	247,23	3386,55	3388
		[0]	[511]	[11481]	
R/C103.50	685,35	10,50	n.e.	17999	18011
		[0]	[n.e.]	[32506]	
C109.50	218,83	2,70	1711,63	2121,18	2124
		[0]	[6811]	[9595]	
R/C109.50	401,72	3,77	n.e.	17999	18012
		[0]	[n.e.]	[84278]	

Aus den Ergebnissen lässt sich die eindeutige Tendenz feststellen, dass R-Instanzen, bei sonst gleichen Parametern, fast in allen Fällen langsamer gelöst werden als C-Instanzen. Besonders auffällig sind diese Unterschiede auf R/C103.50 und R/C109.50 zu sehen. Während die C-Instanzen innerhalb einer Stunde gelöst werden, können die R-Instanzen selbst im Maximalzeitfenster von fünf Stunden nicht komplett abgeschlossen werden. Ein weiterer auffälliger Unterschied ist die Anzahl der B&B-Interationen pro Knoten. Auf allen R-Instanzen werden deutlich mehr Iterationen durchgeführt. Bei sonst gleichen Bedingungen kann man also tendenziell sagen, dass Random-Kundenverteilungen, im Vergleich zu Cluster-Kundenverteilungen die be-

nötigten Lösungszeiten vergrößern.

Abschließend lässt sich festhalten, dass die Kennzahlen zu Zeitfensterweiten nur bedingt verlässliche Voraussagen zum Lösungsverhalten geben können, da zahlreiche andere Faktoren ebenfalls eine Rolle für den Lösungsverlauf spielen. Insbesondere die topologische Beschaffenheit der Kundenstandorte hat, wie sich herausstellte, einen dramatischen Einfluss auf den Umfang der Rechenzeiten.

6. Zusammenfassung und Ausblick

Die Distribution stellt einen wichtigen Bestandteil vieler verschiedener Industrien dar. Eine möglichst optimal ausgerichtete Lieferkette und reibungslose Distribution bergen viele Kosteneinsparpotenziale in sich. VRPTW ist, aufgrund seines breiten Anwendungsspektrums in der Praxis, ein essenzieller Bestandteil effizienter Logistikprozesse. Bisher können aufgrund seiner Komplexität mit exakten Lösungsverfahren nur begrenzt große Probleme gelöst werden.

Ziel dieser Arbeit war, die Auswirkungen des Preprocessings, welches dazu dient die Modellgröße zu reduzieren und folglich die Rechenzeit zu beschleunigen, auf verschiedenen Modell-Formulierungen und Instanzen zu untersuchen. Ausgehend davon wurde im Abschnitt 1 die zugrundeliegende Definition des Vehicle Routing Problems mit Zeitfenstern vorgestellt, gefolgt von den Ausführungen zur der Praxisrelevanz, Literaturüberblick und drei verschiedenen Modell-Formulierungen des VRPTW.

VRPTW zählt zu einem der komplexesten Optimierungsproblemen. Exakte Lösungsverfahren stoßen dabei schnell an ihre Grenzen. Gründe dafür, unter Anderem die Symmetrie und das Lösungsverhalten des Branch-and-Bound-Verfahrens, wurden im Abschnitt 3.1 und 3.2 behandelt. Sodann wurden im folgenden Abschnitt die Preprocessing-Techniken vorgestellt. Dabei zeigte sich, dass einige Preprocessing-Regeln, die für TSPTW gültig sind, auf VRPTW nicht angewendet werden können.

Im Abschnitt 5.3 wurden auf den Modellen aus dem Abschnitt 2.4 die verschiedenen Techniken des Preprocessings auf den Solomon Instanzen experimentell vorgestellt und getestet. Die Ergebnisse zeigen, dass mit Preprocessing gelöste Instanzen in 2/3 der Fälle im Durchschnitt 30% schnellere Rechenzeiten erreichen, sowohl im Bezug auf die Findung des Optimums als auch auf die Gesamtrechenzeit. Aufschlussreich sind die Vergleiche zwischen den verschiedenen Modell-Formulierungen. So konnten auf der 3-Index-Formulierung über 60% der Instanzen mit 25 und 50 Städten nicht komplett abgeschlossen werden wobei bei 30% der Instanzen das Optimum nicht erreicht werden konnte. Im Gegensatz dazu konnten auf der 2-Index-Formulierung 90% aller Instanzen vollständig abgeschlossen werden.

Zur weitergehenden Analyse wurden Experimente auf modifizierten Instanzen mit erweiterten Zeitfensterweiten durchgeführt. Dabei ist es ersichtlich, dass Zeitfenster, vergrößert um den Faktor 2, im Durchschnitt 60% mehr Rechenzeit in Anspruch genommen haben. Ein Zusammenhang zwischen der Zeitfensterstruktur (konstant oder variabel) und dem Lösungsverhalten konnte jedoch nicht festgestellt werden. Vorgestellte Kennzahlen für eine mögliche Prognose über das Lösungsverhalten ergeben geringe Verwendbarkeit als Vorhersage-Werkzeuge. Zusätzlich konnte in weiteren Experimenten bestätigt werden, dass die Topologie einer Instanz einen deutlichen Einfluss auf das Lösungsverhalten ausübt: R-Instanzen haben deutlich mehr Rechenzeit in Anspruch als die C-Instanzen genommen. Für weitere Untersuchungen würde es sich anbieten, ein Kennzahlensystem zu entwickeln, anhand dessen eine Prognose über das Lösungsverhalten einer Instanz möglich wäre.

Preprocessing ist im Stande, VRPTW-Modelle zu verkleinern und die Rechenzeit zu beschleunigen. Dies ist jedoch stark von der gegeben Instanz und der gegeben Modell-Formulierung abhängig. Zusätzlich ist zu beachten, dass diese Preprocessing-Maßnahmen ausschließlich für das VRPTW gültig sind. Für diverse Erweiterung des VRPTW können sie hingegen unzulässig sein. Diese Erweiterung sind jedoch von Nöten, weil die Industrien und deren Anforderung sich im ständigen Wandel, der zeitgleich neue Themengebiete und mit ihnen neue Herausforderungen mit sich bringt, befinden. Diese bedürfen ihrerseits der Forschung und Entwicklung neuer und effizienten Lösungen, um schnelle und produktive Logistik aufrecht erhalten zu können.

A. Appendix

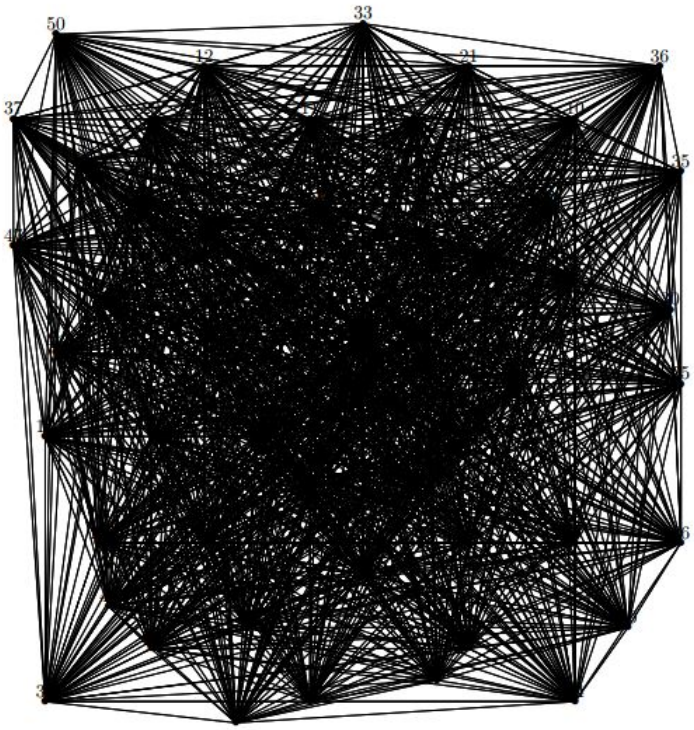
C101

VEHICLE
NUMBER CAPACITY
25 200

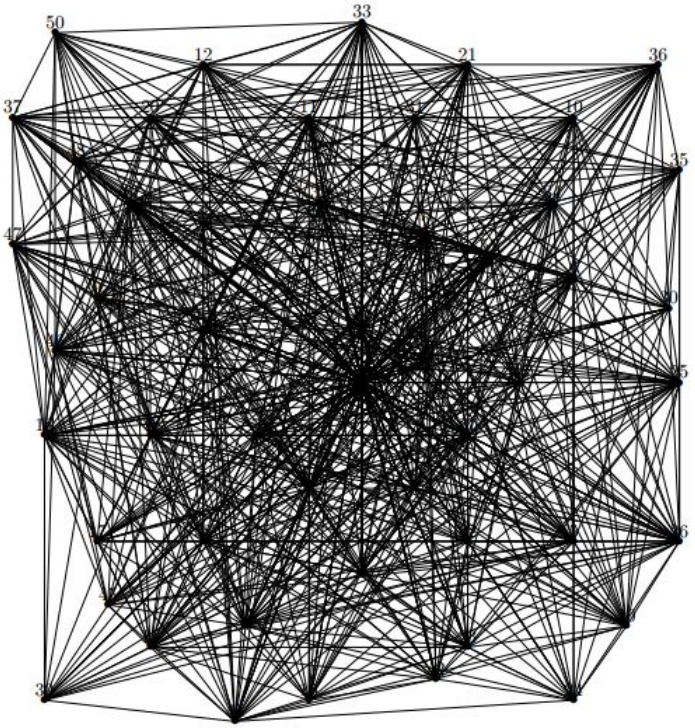
CUSTOMER

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	652	721	90
13	22	75	30	30	92	90
14	22	85	10	567	620	90
15	20	80	40	384	429	90
16	20	85	40	475	528	90
17	18	75	20	99	148	90
18	15	75	20	179	254	90
19	15	80	10	278	345	90
20	30	50	10	10	73	90
21	30	52	20	914	965	90
22	28	52	20	812	883	90
23	28	55	10	732	777	90
24	25	50	10	65	144	90
25	25	52	40	169	224	90

Abbildung A.1.: Solomon Instanz am Beispiel von C101.25



(a)



(b)

Abbildung A.2.: Adjazenzgraph vor a) und nach b) dem Preprocessing auf der Instanz R101.50, angelehnt an [9]

Literaturverzeichnis

[1] Die eU-Verordnung zur Verminderung der CO2 Emissionen von Personenkraftwagen. http://www.bmu.de, 2009.

[2] EU energy and transport in figures - statistical pocketbook 2010, 2010.

[3] IBM. http://www.ibm.com/us/en/, 2013.

[4] Amorim, P., Parragh, S. N., Sperandio, F., and Almada-Lobo, B. A rich vehicle routing problem dealing with perishable food: a case study. *TOP* (2012), pp. 1–20.

[5] Archetti, C., Savelsbergh, M. W. P., and Speranza, M. G. Worst-case analysis for split delivery vehicle routing problems. *Transportation Science Vol. 40*, 2 (2006), pp. 226–234.

[6] Ascheuer, N. Hamiltonian path problems in the on-line optimization of flexible manufacturing systems. Tech. Rep. 03, ZIB, 1996.

[7] Azadian, F., Murat, A. E., and Chinnam, R. B. Dynamic routing of time-sensitive air cargo using real-time information. *Transportation Research Vol. 48*, E (2012), pp. 355–372.

[8] Baldacci, R., Mingozzi, A., and Roberti, R. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research Vol. 218*, 1 (2012), pp. 1 – 6.

[9] Beckhäuser-Hoffmann, S. Die Auswirkungen verschiedener Preprocessing-Routinen und Branch-and-Cut-Frameworks auf das Lösungsverhalten der Time-Bucket-Formulation des Travellung-Salesman-Problems mit Zeitfenstern. Master's thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 2013.

[10] Berbeglia, G., Cordeau, J.-F., and Laporte, G. Dynamic pickup and delivery problems. *European Journal of Operational Research Vol. 202*, 1 (2010), pp. 8–15.

[11] BP. Historical data workbook, 2012.

[12] Bramel, J., and Simchi-Levi, D. On the effectiveness of set covering formulations for the vehicle routing problem with time windows. *Operations Research Vol. 45* (1997), pp. 295–301.

[13] Bräysy, O., and Gendreau, M. Vehicle Routing Problem with Time Windows. Part: II Metaheuristics. *Transportation Science Vol. 39*, 1 (2005), pp. 119–139.

[14] Brønmo, G., Christiansen, M., and Nygreen, B. Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society Vol. 58*, 9 (2007), pp. 1167–1177.

[15] Cook, S. The P versus NP problem. In *Clay Mathematical Institute; The Millennium Prize Problem* (2000).

[16] Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., and Soumis, F. VRP with time windows. In *The vehicle routing problem*, P. Toth and D. Vigo, Eds. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001, pp. 157–193.

[17] Créput, J.-C., Hajjam, A., Koukam, A., and Kuhn, O. Dynamic vehicle routing problem for medical emergency management. In *Self Organizing Maps - Applications and Novel Algorithm Design*. InTech, 2011, ch. 13, pp. 233–250.

[18] Dakin, R. A tree search algorithm for mixed integer programming problems. *The Computer Journal Vol. 8* (1965), pp. 250–255.

[19] Dash, S., Günlük, O., Lodi, A., and Tramontani, A. A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing Vol. 24* (2012), pp. 132–147.

[20] Desaulniers, G., Lessard, F., and Hadjar, A. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science Vol. 42*, 3 (2008), pp. 387–404.

[21] Desrochers, M., Desrosiers, J., and Solomon, M. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research Vol. 40*, 2 (1992), pp. 342–354.

[22] Desrochers, M., Lenstra, J., Savelsbergh, M., and Soumis, F. Vehicle routing with time windows: Optimization and approximation. In *Vehicle routing: Methods and studies*, vol. Vol.16. 1988, pp. 65–84.

[23] Desrosiers, J., and Lübbecke, M. E. A Primer in Column Generation. In *Column Generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds. Springer Berlin Heidelberg, 2005, pp. 1–32.

[24] DHL. Alles über Track&Trace. schwarz auf weiß.

[25] Domschke, W., and Drexl, A. *Einführung in Operations Research*, 6., überarb. und erw. Aufl ed. Springer Berlin Heidelberg, Berlin [u.a.], 2005.

[26] Drexl, M. Rich vehicle routing in theory and practice. *Logistics Research Vol. 5*, 1-2 (2012), pp. 47–63.

[27] Dumas, Y., Desrosiers, J., and Soumis, F. The pickup and delivery problem with time windows. *European Journal of Operational Research Vol. 54*, 1 (1991), pp. 7 – 22.

[28] Fraunhofer, I. Die Top 100 der Logistik, 2012/2013.

[29] GAMS, D. C. GAMS Distribution 24.1. http://www.gams.com/, 2013.

[30] Gehring, H., and Homberger, J. Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of Heuristics Vpl. 8*, 3 (2002), pp. 251–276.

[31] Gendreau, M., and Tarantilis, C. D. Solving large scale vehicle routing problems with time windows. the state of the art, 2010.

[32] Gündüz, H. I. *Optimierung von Distributionsnetzwerken unter Berücksichtigung von Tourenplanungsapekten und zeitlichen Restriktionen.* PhD thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 2011.

[33] Hempsch, C. Optimierung des Distributionsnetzwerkes für Paket in Deutschland, 2012.

[34] Hoyer, C. Netzwerkplanung am Beispiel des Recyclings von Lithium Ionen Batterien aus Elektrofahrzeugen, 2012.

[35] Jans, R., and Desrosiers, J. Efficient symmetry breaking formulations for the job grouping problem. *Computers & OR Vol. 40*, 4 (2013), pp. 1132–1142.

[36] Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research Vol. 56*, 2 (2008), pp. 497–511.

[37] Jourdan, L., Basseur, M., and Talbi, E.-G. Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research Vol. 199*, 3 (2009), pp. 620 – 629.

[38] Kallehauge, B., Boland, N., Madsen, and G., O. B. Path inequalities for the vehicle routing problem with time windows. *Networks Vol. 49*, 4 (2007), pp. 273–293.

[39] Kim, B.-I., Kim, S., and Sahoo, S. Waste collection vehicle routing problem with time windows. *Comput. Oper. Res. Vol. 33*, 12 (2006), pp. 3624–3642.

[40] Kohl, N. *Exact methods for time constrained routing and related scheduling problems*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 1995.

[41] Kohl, N., Desrosiers, J., Madsen, O. B. G., Solomon, M. M., and Soumis, F. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science Vol. 33*, 1 (1999), pp. 101–116.

[42] Kok, A. L., Meyer, C. M., Kopfer, H., and Schutten, J. M. J. A dynamic programming algorithm for the vehicle routing problem with time windows and european community social legislation. *Transportation Science Vol. 44*, 4 (2010), pp. 442–454.

[43] Kolen, A. W. J., Rinnooy Kan, A. H. G., and Trienekens, H. W. J. M. Vehicle routing with time windows. *Operations Research Vol. 35*, 2 (1987), pp. 266–273.

[44] Laporte, G. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research Vol. 59*, 3 (1992), pp. 345–358.

[45] Larsen, J. *Parallelization of the Vehicle Routing Problem with Time Windows*. PhD thesis, DTU Technical University of Denmark, 2001.

[46] Lenstra, J. K., and Rinnooy-Kan, A. H. G. Complexity of vehicle routing problem with time windows. *Networks Vol. 11* (1981), pp. 221–227.

[47] Letchford, A. N., and González, J. J. S. Projection results for vehicle routing. *Math. Program. Vol. 105*, 2-3 (2006), pp. 251–274.

[48] Lin, C., Choy, K. L., Ho, G. T. S., Chung, S. H., and Lam, H. Y. Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, in press (2013).

[49] Lysgaard, J. Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research Vol. 175*, 1 (2006), pp. 210 – 223.

[50] Melo, M. T., Nickel, S., and da Gama, F. S. Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning. *Comput. Oper. Res. Vol. 33*, 1 (2006), pp. 181–208.

[51] Möhring, R. H., and Schenk, M. Mit Mathematik zu Mehr Intelligenz in der Logistik. In *Produktionsfaktor Mathematik*. Springer Berlin Heidelberg, 2009, pp. 157–173.

[52] PricewarehouseCoopers. Customers take control, 2011.

[53] PTV-AG. Map and Guide Software. http://www.mapandguide.com/de/.

[54] Quintiq-Inc. The vehicle routing problem with time windows challenge. http://www.quintiq.com/optimization/world-records.html, 2012.

[55] Savelsbergh, M. Local search in routing problems with time windows. *Annals of Operations Research Vol.4*, 1 (1985), pp. 285–305.

[56] Sherali, H. D., and Smith, J. C. Improving discrete model representations via symmetry considerations. *Management Science Vol. 47*, 10 (2001), pp. 1396–1407.

[57] Solomon, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research Vol. 35* (1987), pp. 254–265.

[58] Spoorendonk, S. *Cut and Column Generation*. PhD thesis, University of Copenhagen, 2008.

[59] Stenger, A., Schneider, M., Schwind, M., and Vigo, D. Location routing for small package shippers with subcontracting options. *International Journal of Production Economics Vol. 140*, 2 (2012), pp. 702 – 712.

[60] Stockmeyer, L. J. Chapter 9 computational complexity. In *Computing*, E. C. Jr., J. Lenstra, and A. R. Kan, Eds., vol. Vol. 3 of *Handbooks in Operations Research and Management Science*. 1992, pp. 455 – 517.

[61] Suhl, L., and Mellouli, T. *Optimierungssysteme - Modelle, Verfahren, Software, Anwendungen*. Springer Berlin Heidelberg, 2006.

[62] Toth, P., and Vigo, D. An overview of vehicle routing problems. In *Transportation*, P. Toth and D. Vigo, Eds. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001, pp. 1–26.

Abbildungsverzeichnis

Tabellenverzeichnis

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften anderer entnommen sind, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form noch nicht als Prüfungsarbeit eingereicht worden.

Aachen, den 11.10.2013

Unterschrift