Fakultät für Wirtschaftswissenschaften
Wismar Business School

Harald Mumm

Ermittlung der minimalen Touranzahl für das Fahrzeugroutenproblem mit Zeitfenstern bei kleiner Fahrzeugkapazität und großer Ortsanzahl

Heft 05/2020

Die Fakultät für Wirtschaftswissenschaften der Hochschule Wismar, University of Applied Sciences – Technology, Business and Design bietet die Präsenzstudiengänge Betriebswirtschaft, Wirtschaftsinformatik und Wirtschaftsrecht sowie die Fernstudiengänge Betriebswirtschaft, Business Consulting, Business Systems, Facility Management, Quality Management, Sales and Marketing und Wirtschaftsinformatik an. Gegenstand der Ausbildung sind die verschiedenen Aspekte des Wirtschaftens in der Unternehmung, der modernen Verwaltungstätigkeit, der Verbindung von angewandter Informatik und Wirtschaftswissenschaften sowie des Rechts im Bereich der Wirtschaft.
Nähere Informationen zu Studienangebot, Forschung und Ansprechpartnern finden Sie auf unserer Homepage im World Wide Web (WWW): https://www.fww.hs-wismar.de/.

Herausgeber:
Prof. Dr. Hans-Eggert Reimers
Fakultät für Wirtschaftswissenschaften
Hochschule Wismar
University of Applied Sciences – Technology, Business and Design
Philipp-Müller-Straße
Postfach 12 10
D – 23966 Wismar
Telefon: ++49/(0)3841/753 7601
Fax: ++49/(0)3841/753 7131
E-Mail: hans-eggert.reimers@hs-wismar.de

Vertrieb:
Fakultät für Wirtschaftswissenschaften
Hochschule Wismar
Postfach 12 10
23952 Wismar
Telefon: ++49/(0)3841/753-7468
Fax: ++49/(0) 3841/753-7131
E-Mail: Silvia.Kaetelhoen@hs-wismar.de
Homepage: https://www.fww.hs-wismar.de/

ISSN 1612-0884
ISBN 978-3-948862-02-2
JEL- Klassifikation: C61

Printed in Germany

Inhaltsverzeichnis

1 Einleitung

Dem Ansatz in [Mu2020] zur exakten Lösung des sogenannten Vehicle-Routing-Problem-with-Time-Windows (Abk. VRPTW) mit den Probleminstanzen aus [Gehring/Homberger] war kein Erfolg beschieden, weil es zu viele effektive Bellman-Moore-Pfade für Ortsanzahlen oberhalb von 150 Orten gibt. Der Ansatz ging in drei Schritten vor:

Schritt 1: Ermittlung aller Zeitfenster- und kapazitiv-möglichen effektiven Pfade vom Depot zu den Kunden (Orten) und zurück,

Schritt 2: Bestimmung der minimalen Touranzahl mit einer Anfangslösung (Abk. ZF0-Problemtyp) sowie

Schritt 3: Bestimmung des kostengünstigsten Tourenplans (bzgl. der Länge des zurückgelegten Weges) für die Touranzahl aus Schritt 1 (Abk. ZF1-Problemtyp) .

In dieser Arbeit wird nur der Problemtyp 'ZF0' behandelt, jedoch mit einer anderen Methode als in [Mu2020]. In Schritt 2 wurden ausgehend von einer teuren Stich-Anfangslösung im BaP-Prozess effektive Pfade aus Schritt 0 mit negativ reduzierten Kosten gesucht, bis ein fraktionales Optimum gefunden wurde. Dazu wurden diejenigen Spalten aus Schritt 0 in den Master eingefügt, die die kleinsten negativ reduzierten Kosten versprachen. Es mussten also sämtliche Spalten mittels Dualpreisen bewertet werden. Ein Tutorium zu diesem Prozedere findet man in [Feillet2010] oder als Zitat in [Mu2018]. Das erfordert sehr viel Rechenzeit, weil die Anzahl der eff. Pfade bzgl. der Ortsanzahl exponentiell ansteigt. In dieser Arbeit wird nun genau umgekehrt vorgegangen. Die Dualpreise (je Ort einer) nach jeder Masterberechnung (ein Beispiel für ein Masterprogramm für den Problemtyp 'ZF1' findet man in [Mu2020] S. 6) werden absteigend sortiert, und die drei Orte mit den höchsten Dualpreisen bilden ein Ergebnistripel, sofern sie tripelfähig sind, also kapazitiv und fensterzulässig. Sind sie es nicht, wird noch untersucht, ob im Tripel ein Paar paarfähig ist. Diese neuartige Methode wird **Dualpreisabstieg** genannt. Da wir nur Tripel und Paare zulassen, liefert der Dualpreisabstieg zur Zeit nur optimale Ergebnisse, wenn die Bedarfe für alle Orte größer gleich 13 sind (die Belieferung von vier Orten ist dann nicht mehr möglich, weil 4*13=52) bei einer Fahrzeugkapazität von 50. Wenn die ermittelte Touranzahl aber den aufgerundeten unteren Ladequotienten erreicht, ist sie natürlich auch optimal, wenn einige Bedarfe kleiner als 13 sind. In dieser Arbeit soll untersucht werden, bis zu welcher Ortsanzahl die neue Methode auf einem handelsüblichen Desktop PC funktioniert und welche Hauptspeicherausstattung (RAM) ausreicht. Die Laufzeit stand nicht im Vordergrund der Betrachtung. Die Pragmatik dieser Arbeit liegt in der Bestimmung einer ganzzahligen Anfangslösung für den Problemtyp 'ZF1'. Im Gehring-Homberger Benchmark gibt es drei Typen von Beispielen:

Der Dateiname beginnt mit einem 'C': Die geografischen Daten sind geclustert,

Der Dateiname beginnt mit einem 'R': Die geografischen Daten sind zufällig generiert und

Der Dateiname beginnt mit 'RC': Die geografischen Daten sind sowohl geclustert als auch zufällig generiert worden.

Deren unterschiedlicher Computer-Ressourenbedarf soll untersucht werden. Die Implementierungssprache ist Java in der Version 8, die Entwicklungsumgebung ist Netbeans 8.2, und der LP-Solver ist CPLEX in Version 12.10 (siehe [CPLEX1210]). Die Programmierarbeiten wurden auf einem Desktop TVA-5288 PC mit Intel(R) Core(TM) i9-9900K CPU 3,6 GHZ Taktfrequenz und 64 GB RAM unter Windows 10 durchgeführt. Die CPU-Auslastung lag im Mittel nur bei 10 %.

2 Grundbegriffe und der Algorithmus des Dualpreisabstiegs

Die mathematische Formulierung des VRPTW und eine Einführung in die BaP-Methode entnehme man [Feillet2010] oder als Zitat [Mu2018]. Hier werden nur wichtige Begriffe der Implementierung definiert, die darüber hinaus gehen. Die Summe aller Bedarfe wird mit *Bedarfe abgekürzt. Die Fahrzeugkapazität q betrage bis auf Widerruf 50. Als Ladequotient LQ wird der Quotient aus *Bedarfe und q definiert. Als untere Touranzahlschranke UTAS wird die kleinste ganze Zahl angesehen, die größer ist als der Ladequotient LQ. Dazu ein Beispiel: In der G/H-Beispieldatei c1_10_1.txt gilt *Bedarfe=17940. Deshalb ist hier LQ gleich 358,8 und UTAS=359. Wenn man für ein G/H-Beispiel die minimale Touranzahl gleich der UTAS ermittelt hat, ist dieser Wert optimal (minimal), egal wie man ihn ermittelt hat. Wir machen uns diesen Sachverhalt immer dann zu Nutzen, wenn wir z. B. eine große Beispieldatei mit 1000 Orten in zwei mittelgrosse Dateien mit 500 Orten aufspalten und die minimale Touranzahl für die 1000 Orte durch die Summe der minimalen Touranzahlen für die beiden Dateien mit 500 Orten ermitteln, sofern der UTAS-Wert für die 1000 Orte nicht überschritten wird. Für das Problem der minimalen Touranzahl betragen die Tourkosten stets Eins. Wenn man Touren für den BaP-Prozess mit negativ reduzierten Kosten braucht, müssen die Dualpreise größer als Eins sein. Die Nebenbedingungen im Master-Programm sorgen dafür, dass alle Orte angefahren werden. Die Zielfunktion ist lediglich eine Summe der Tourvariablen. Im BaP-Prozess wird nach bestimmten Ortspaaren verzweigt (siehe [Mu2020]), deren Items entweder nur gemeinsam in Touren auftreten dürfen (Zweig Pair) oder nur einzeln (Zweig Single). Vorab sei nochmals darauf hingewiesen, dass der Dualpreisabstieg, so wie er hier vorgestellt wird, nur dann optimale Ergebnisse liefert, wenn bei einer LKW-Kapazität von 50 alle Orts-Bedarfe größer oder gleich 13 sind. Mit anderen Worten: Touren mit maximal drei Ortsanfahrten reichen stets aus. (Der Algorithmus des Dualpreisabstiegs wurde von meinem hochgeschätzten Lehrer Hans Röck entwickelt.) Bei der Problematik 'ZF0' gehen alle Variablen mit den Kosten von Eins in die Master-Zielfunktion ein. Damit neue Spalten den Wert der Zielfunktion verbessern, muss die Summe der Dualpreise der darin enthaltenen Orte größer als Eins und bestenfalls gleich Drei sein. Im Gegensatz zu [Mu2020], wo alle möglichen effektiven Pfade a priori berechnet vorliegen und diejenigen anhand der Dualpreise ausgewählt werden, deren neg. reduzierte Kosten am kleinsten sind, werden beim Dualpreisabstieg erst die Pfade anhand der Dualpreise bestimmt. Die ermittelten Pfade sind noch nicht effektiv. In einem Nachbereitungsschritt muss aus den maximal sechs möchlichen Reihenfolgen für Tripel noch diejenige ermittelt werden, deren Kosten für den Problemtyp 'ZF1' am kleinsten sind, analog für Paare. Wir demonstrieren den Algorithmus am G/H-Beispiel c1_2_3.txt für die ersten zehn Orte. Für die Lösung des ersten Masterprogramms mit der Anfangslösung (Stichlösung) ergeben sich folgende absteigend sortierte Dualwerte, die in der Methode 'setObjective' der Klasse 'ExactPricingProblemSolver' protoliiert wurden:

```
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=1
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=2
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=3
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=4
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=5
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=6
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=7
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=8
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=9
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=10
```

Gesucht ist ein Ortstripel, für das die Summe der Dualwerte maximal ist. Man kann dazu die ersten drei Orte (1,2,3) verwenden. Leider ist das aus diesen drei Orten gebildete Tripel nicht ressourenzulässig (die Ressourcen sind Kapazität und Zeitfenster), wie man der Datei c1_2_3.txt in [Gehring/Homberger] entnehmen kann. Die ersten beiden Orte ergeben aber ein ressourenzulässiges Paar. Es wird nun ein neues Tripel aus den Orten (1,2,4) gebildet und damit gelingt ein ressourenzulässiges neues Tripel. Die beiden neuen Variablen für die Spaltengenerierung sind also mit folgendem Tripel und Paar verbunden:

```
Master-Value: 0.0 Verlauf: [1, 2, 4, 0, 0, 0, 0, 0, 0, 0]
 creator: exactPricing,1.0

Master-Value: 0.0 Verlauf: [1, 2, 0, 0, 0, 0, 0, 0, 0, 0]
 creator: exactPricing,1.0
```

Eine neue Berechnung des Masterprogramms mit der Anfangslösung, wofür in dieser Arbeit die Stichlösung verwendet wird, zuzüglich dieser beiden neuen Variablen bringt folgende absteigend sortierte Dualwerte hervor:

```
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=2
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=3
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=4
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=5
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=6
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=7
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=8
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=9
Trace5:ExactPricingProblemSolver.setObjective, Dual=1.0,Kunde=10
Trace5:ExactPricingProblemSolver.setObjective, Dual=-1.0,Kunde=1
```

Alle Ortstripel (2,3,4) bis (2,3,10) sind nicht ressourenzulässig, wie man wieder der Datei c1_2_3.txt in [Gehring/Homberger] entnehmen kann, aber das Ortstripel (2,4,5). Damit ergeben sich folgende neue Spalten:

```
Master-Value: 0.0 Verlauf: [2, 4, 5, 0, 0, 0, 0, 0, 0, 0]
creator: exactPricing,1.0

Master-Value: 0.0 Verlauf: [2, 3, 0, 0, 0, 0, 0, 0, 0, 0]
 creator: exactPricing,1.0
```

Die Java-Klasse 'ExactPricingProblemSolver' ist eine Klasse aus dem Programmpaket 'JORLIB' und die Methode 'setObjective' ist eine Methode in dieser Klasse, die für die Bereitstellung der Dualpreise verwendet wird.

3 Suchraumerkundungsstrategien

Bei jedem Verzweigungsereignis werden zwei Knoten (Single, gerade ID und Pair, ungerade ID) angelegt, die in einer Warteschlange abgelegt werden. Welcher Knoten als Nächster verarbeitet wird, hängt von der Suchraumerkundungsstrategie ab. Der JORLIB-Standard ist die Tiefensuche (engl. Depth-First), was bedeutet, dass stets nur Single-Knoten gelöst werden, und zwar solange, bis das Ergebnis zu schlecht wird oder die optimale ganzzahlige Lösung gefunden wurde. Bei der Breitensuche (engl. Breadth-First) werden zuerst alle Knoten der ersten Hierarchiestufe gelöst, anschließend der zweiten usw. Bei der Strategie 'Bestes Ergebnis' (engl. Best-Objective) wird die Knotenwarteschlange nach den Zielfunktionswerten der fraktionalen Optima absteigend sortiert und bei der Strategie 'Beste Abschätzung' (engl. Best-Bound) nach den Abschätzungen für die ganzzahligen Lösungen.

Für die Implementierung der Best-Objective-Strategie in JORLIB benötigt man die folgende Klasse:

```
import java.util.Comparator;
import org.jorlib.frameworks.columnGeneration.branchAndPrice.BAPNode;
public class BOFbapNodeComparator implements Comparator<BAPNode>{
    @Override
    public int compare(BAPNode o1, BAPNode o2) {

      return (int)Double.compare(o1.getObjective(), o2.getObjective());
    }
}
```

Für die Implementierung der Best-Bound-Strategie in JORLIB benötigt man diese Klasse:

```
import java.util.Comparator;
import org.jorlib.frameworks.columnGeneration.branchAndPrice.BAPNode;
public class BBFbapNodeComparator implements Comparator<BAPNode>{
    @Override
    public int compare(BAPNode o1, BAPNode o2) {
              return (int)Double.compare(o1.getBound(), o2.getBound());
    }
}
```

Für die Implementierung der Breitensuche in JORLIB ist diese Klasse nützlich:

```
import org.jorlib.frameworks.columnGeneration.branchAndPrice.BAPNode;
import java.util.Comparator;
public class BFSbapNodeComparator implements Comparator<BAPNode>{
    @Override
    public int compare(BAPNode o1, BAPNode o2) {
                return Integer.compare(o1.nodeID, o2.nodeID);
           }
}
```

Man sieht hieran, dass JORLIB sehr professionell implementiert wurde und man als Anwendungsprogrammierer sehr leicht in die Implementierung der Standardsoftware eingreifen kann.

4 Berechnungsergebnisse und Datenstrukturen für die Implementierung

Auf die inhaltlichen Unterschiede der G/H-Beispiele in den Klassen 'r', 'rc' und 'c' wurde bereits in der Einleitung hingewiesen. Die rasante Erhöhung der Rechenzeiten mit wachsender Ortsanzahl(n) zeigt die folgende Tabelle: In allen vier Fällen wurde hier die UTAS

r1_4_1.txt			
n	Zielfunktionswert	Rechenzeit	Knotenanzahl
100	39	6 Sekunden	169
200	72	311 Sekunden	343
300	109	2988 Sekunden	565
400	143	16380 Sekunden	843

Tabelle 1: Rechenzeiten beim G/H-Beispiel r1_4_1.txt mit 100, 200, 300 und 400 Orten

erreicht. Die Rechenzeiten variieren bei verschiedenen Beispieltypen sehr stark, wie das nächtse Beispiel zeigt. In allen vier Fällen wurde hier die UTAS erreicht.

c1_4_1.txt			
n	Zielfunktionswert	Rechenzeit	Knotenanzahl
100	35	2 Sekunden	175
200	71	82 Sekunden	487
300	108	499 Sekunden	649
400	144	3522 Sekunden	759

Tabelle 2: Rechenzeiten beim G/H-Beispiel c1_4_1.txt mit 100, 200, 300 und 400 Orten

Eine dritte Vergleichsreihe dient dem G/H-Beispiel rc1_4_1.txt :

rc1_4_1.txt			
n	Zielfunktionswert	Rechenzeit	Knotenanzahl
100	35	5 Sekunden	137
200	70	335 Sekunden	385
300	106	5079 Sekunden	1197
400	143	26220 Sekunden	1813

Tabelle 3: Rechenzeiten beim G/H-Beispiel rc1_4_1.txt mit 100, 200, 300 und 400 Orten

In allen vier Fällen wurde hier ebenfalls die UTAS erreicht. Um einen Eindruck zu vermitteln, wie viele Spalten generiert werden mussten, sei die Anzahl für 400 Orte hier exemplarisch angegeben: 139129. Die Leserin oder der Leser werden sich vielleicht fragen, wie sich die Rechenzeit auf Masterprogramm und Subprogramm verteilt hat? Hierzu führt JORLIB Protokoll und liefert z.B. für 400 Orte im rc-Beispiel die Nachricht:

```
Total Time spent on master problems: 54112 Total time spent on
 pricing problems:  26171550  (ms)
```

Man erkennt an diesen Zahlen, dass bei der Methode des Dualpreisabstiegs der überwiegende Zeitanteil beim Subprogramm anfällt. Eine Schlussfolgerung daraus könnte sein, den teuren CPLEX-Solver durch ein Open-Source-Produkt, wie GLPK
(siehe [Andrew Makhorin]), auszutauschen.

Für 200 Orte wurden die G/H-Beispiele r1_2_1 bis r1_2_5, rc1_2_1 bis rc1_2_5 sowie c1_2_1 bis c1_2_5 gerechnet. Bei den r-Beispielen betrug der Zielfunktionswert durchweg 71, bei den rc-Beispielen 72 und bei den c-Beispielen wieder 71. Die Rechenzeit betrug bei den r- und rc-Beispielen mit der Best-Objective-Suchstrategie durchschnittlich ca. 10 Minuten und bei den c-Beispielen durchschnittlich 2,5 Minuten.

Es folgt eine tabellarische Übersicht zu einigen Berechnungsergebnissen, bei der der Unterschied zwischen den G/H-Beispieltypen 'rc', 'r' und 'c' im Vordergrund steht.

LFD	Dateiname	Zielfunktionswert	Rechenzeit	Knotenanzahl
1	rc1_2_1.txt	72	382 Sekunden	355
2	r1_2_1.txt	71	287 Sekunden	283
3	c1_2_1.txt	71	49 Sekunden	477

Tabelle 4: Gegenüberstellung der Rechenzeiten bei drei Beispielen mit 200 Orten

Es wurde auch ein optimaler Wert für ein Beispiel mit 1000 Orten errechnet, und zwar durch Aufteilung in zwei Gruppen zu je 500 Orten. Die folgende Tabelle beinhaltet dazu die Details:

LFD	Dateiname	Ortsnummern	Zielfunktionswert	Rechenzeit	Knotenanzahl
1	c1_10_1.txt	1 bis 500	177	184 Minuten	1439
2	c1_10_1.txt	501 bis 1000	182	166 Minuten	1335

Tabelle 5: Aufteilung des Beispiels c1_10_1.txt mit 1000 Orten in zwei Beispiele mit je 500 Orten

Wenn man die Laufzeiten verbessern will, kommt man nicht umhin, sich zumindest exemplarisch mit den verwendeten Algorithmen und Datenstrukturen zu beschäftigen. Eine zentrale Datenstruktur der hier beschriebenen Implementierung ist eine Java-ArrayList, die die aus den Dualpreisen abgeleitete Spaltenkandidaten enthält. Für das G/H-Beispiel C1_2_1.txt mit 200 Orten (RZ= 3 Min., obj=71), bei dem insgesamt 3039 neue Spalten für 485 Knoten bis zum Optimum generiert werden mussten, gab es folgende Aufrufhäufigkeiten:

1. Anzahl Aufrufe der Methode 'expandiereRangTripel': 12.774.396
2. Anzahl Aufrufe der Methode 'findeMaxPreisTripelInListe': 12.774.396

Besonders das mehrfache (12.774.396 mal) sequentielle Suchen eines Listenelementes mit maximalem Dualpreis ist Rechenzeit-intensiv, da die betroffene Liste durchschnittlich 695 Elemente enthielt.

Alternativen zur Java-ArrayList, wie Binärbaum oder statische Reihe brachten leider keine wesentlichen Verbesserungen der Laufzeit.

Der Aufwand des Dualpreisabstiegs ist für maximal drei Orte je Tour gleich $O(n^2)$, für maximal vier Orte $O(n^3)$ usw. Hinzu kommt der unvorhersehbare Hauptspeicherplatzbedarf für den Verzweigungsbaum.

5 Resultatsermittlung und -bewertung

Der Problemtyp 'ZF0' (Bestimmung der minimalen Touranzahl) konnte mit dem neuen Algorithmus des Dualpreisabstiegs (Abk. DPA) für einige G/H-Beispiele mit bis zu 500 Orten gelöst werden, aber nicht für alle G/H-Beispiele mit 400 Orten. Da in [Mu2020] nur Beispiele mit bis zu 150 Orten gelöst werden konnten, ist die neue Methode trotzdem erfolgreicher als die alte Methode, allerdings nur für den Problemtyp 'ZF0'. Für 1000 Orte konnte ein optimaler Wert durch Aufteilung der 1000 Orte in zweimal 500 Orte erzielt werden. Aber schon für 200 Orte stellte der in dieser Arbeit vorgestellte Algorithmus 'Dualpreisabstieg' seine Leistungsfähigkeit gegenüber dem in [Mu2020] vorgestellten Verfahren mittels Bellman-Moore-Algorithmus unter Beweis, denn das den Bellman-Moore-Algorithmus implementierende Programm stürzte am G/H-Beispiel c1_2_3 mit 200 Orten wegen Hauptspeichermangels ab, wogegen das den DPA implementierende Programm nach 78 Sekunden den optimlalen Wert von 71 nach 78 Sekunden Rechenzeit lieferte. Die Rechenzeiten waren sehr von der Suchraumerkundungsstrategie abhängig. Die besten Rechenzeiten wurde mit der Suchraumerkundungs-Strategie 'Best-Objective' erzielt. Es kam vor, dass andere Strategien, wie die Breitensuche, die doppelte Rechenzeit erforderten. Für r- und rc-Beispiele betrug die Rechenzeit bei 200 Orten durchschnittlich 10 Minuten bei der Suchraumerkundungsstrategie 'Best-Objective'. Für Online-Berechnungen, wenn der Kunde vor dem Monitor sitzt und auf die Berechnungsergebnisse wartet, ist die Methode dieser Arbeit jedoch nicht geeignet sondern nur zur Qualitätsbewertung mittels Heuristik erzielter Lösungen. Die CPU-Auslastung lag im Mittel bei ca. 10%. Eine CPU-Auslastung von 90 % und kürzeren Rechenzeiten ist anzustreben. In [Mu2020] und in dieser Arbeit wurden ein nicht dekomponierter Ansatz für max. drei Orte je Tour sowie zwei dekomponierte Ansätze, davon der eine für maximal fünf Orte je Tour (siehe [Mu2020]) und der andere für maximal drei Orte je Tour untersucht. Zusammengefasst kann festgestellt werden, dass der Ansatz der Dekomposition für das VRPTW bei kleinen Fahrzeugkapazitäten enttäuschend war und nicht das gehalten hat, was seine Erfinder G. Dantzig und P. Wolfe in [DantzigWolfe] vorausgesagt hatten. Unter dem Aspekt der Wissenschaftsgeschichte ist bemerkenswert, dass es 55 Jahre gedauert hat, bis von der erstmaligen Veröffentlichung der Dekompositions-Methode ein für jedermann offenes, kostenloses und leicht anwendbares Softwarewerkzeug in Form von JORLIB (siehe [JORLIB2015]) zur Anwendung dieser Methode in der Praxis entstanden ist. Als Ausblick will der Autor einerseits die Rechenzeit am dekomponierten Modell verbessern und andererseits die Arbeiten am nicht dekomponierten Modell wieder aufnehmen. In diesem Modell werden alle Touren (Variablen), vorerst mit bis zu drei Orten, ähnlich wie im Bellman-Moore-Algorithmus aus [Mu2020] erzeugt und ein gemischt-ganzzahliges Lineares Programm mittels GLPK (siehe [Andrew Makhorin]) erstellt, welches mittels CPLEX-Solver (siehe [CPLEX1210]) gelöst wird. Für das G/H-Modell r1_8_1 mit 800 Orten konnte so das ZF0-Optimum von 285 Touren in 38 Minuten CPLEX-Rechenzeit gelöst werden . (Die 75 Minuten Rechenzeit für die Erstellung der lp-Datei (Eingangsdatei von CPLEX) ist darin aber noch nicht enthalten.) Bzgl. der Vielfalt der G/H-Benchmark-Daten ist anzumerken, dass in den zufällig ausgewählten 20 von insgesamt 150 Dateien stets die untere Touranzahlschranke (UTAS) erreicht wurde. Das kann aber daran liegen, dass die G/H-Benchmarkdaten auf die Ladekapazität von 200 ausgerichtet sind, in dieser Arbeit aber mit der viel kleineren Ladekapazität von 50 davon abgewichen wurde. Neue Benchmarkdaten für diese kleinere Ladekapazität sind also wünschenswert, weil es auch sonst Bedarfe gleich der neuen kleineren Ladekapazität geben kann, die ja nur zu Stichtouren führen. Die Beschränkung auf drei Orte je Tour war bei allen gerechneten Beispielen für den Problemtyp 'ZF0' ausreichend, weil stets der UTAS-Wert erreicht wurde. Das sähe beim Problemtyp 'ZF1' jedoch anders aus.

6 Anhang

Für das G/H-Beispiel rc1_2_1.txt mit 200 Orten soll die Lösung im Detail angegeben werden.

Die unmittelbaren Ergebnistripel und -paare des Dualpreisabstiegs geben noch nicht die kürzeste Reihenfolge der Orte wieder. Sie garantieren nur, dass z. B. bei Tripeln mindestens eine der sechs möglichen Reihenfolgen der Orte ressourcentauglich ist. Damit man die Ergebnisse des Dualpreisabstiegs als Anfangslösung für den Problemtyp 'ZF1' nutzen kann, bestimmt man bei Tripeln die Kosten von allen ressourcentauglichen möglichen Reihenfolgen der beteiligten drei Orte und ermittelt diejenige Reihenfolge, bei der die Kosten am kleinsten sind. Bei Paaren verfährt man analog. Es soll nicht unerwähnt bleiben, dass in die Kosten weder Servicezeiten noch Wartezeiten eingehen.

Mit dieser Nachbereitung hat die Lösung folgende Form, in der sie als Anfangslösung für den Problemtyp 'ZF1' verwendet werden kann:

Lfd	Ort1	Ort2	Ort3	Kosten
1	28	14	22	146.08
2	48	49	43	126.26
3	68	198	83	224.54
4	100	118	105	164.96
5	167	124	121	240.38
6	127	117	193	370.80
7	132	176	135	205.38
8	131	137	157	309.80
9	188	139	146	244.34
10	163	197	171	123.63
11	179	16	187	157.56
12	172	10	186	157.99
13	31	56	81	162.39
14	57	59	189	193.25
15	6	164	25	171.75
16	40	101	19	357.65
17	84	26	77	208.05
18	88	155	96	198.91
19	9	90	174	239.05
20	65	17	60	257.25
21	53	192	8	284.15
22	165	119	114	223.95
23	196	98	175	193.70
24	87	107	185	260.13
25	162	35	75	237.66
26	102	18	86	194.53
27	42	136	150	157.91
28	151	21	145	294.22

Tabelle 6: Die ersten 28 Ergebnistripel des Dualpreisabstiegs für das G/H-Beispiels rc1_2_1.txt

Lfd	Ort1	Ort2	Ort3	Kosten
29	142	61	33	220.60
30	4	149	66	173.58
31	15	3	138	186.76
32	153	5	134	177.14
33	200	95	36	174.40
34	140	41	34	205.77
35	80	116	141	251.75
36	177	178	2	318.15
37	32	50	154	157.65
38	12	160	11	232.87
39	123	76	1	240.74
40	181	46	93	194.20
41	182	69	113	165.80
42	103	74	64	289.75
43	24	78	71	204.87
44	37	85	191	203.18
45	63	51	79	214.12
46	89	126	158	168.89
47	128	38	23	175.47
48	156	20	44	260.95
49	94	91	67	292.76
50	108	173	166	138.26
51	168	144	55	310.82
52	47	180	19	208.95
53	152	73	97	228.05
54	147	110	13	268.91
55	99	125	39	236.69
56	29	129	70	163.40

Tabelle 7: Weitere 28 Ergebnistripel des Dualpreisabstiegs für das G/H-Beispiels rc1_2_1.txt

Hier die ersten Berechnungsergebnissse noch ohne Nachbereitung:

```
Wert: 1.0 Verlauf: [14, 22, 28, 0, 0, 0, 0, 0, 0, 0]
creator: exactPricing,1.0,q=50

Wert: 1.0 Verlauf: [43, 48, 49, 0, 0, 0, 0, 0, 0, 0]
creator: exactPricing,1.0,q=50

Wert: 1.0 Verlauf: [68, 83, 198, 0, 0, 0, 0, 0, 0, 0]
creator: exactPricing,1.0,q=50

Wert: 1.0 Verlauf: [45, 72, 0, 0, 0, 0, 0, 0, 0, 0]
creator: exactPricing,1.0,q=50

Wert: 1.0 Verlauf: [100, 105, 118, 0, 0, 0, 0, 0, 0, 0]
creator: exactPricing,1.0,q=50
```

Lfd	Ort1	Ort2	Kosten
1	45	72	154.96
2	104	122	166.82
3	183	161	118.91
4	170	148	254.14
5	7	111	188.27
6	195	82	85.96
7	199	58	113.14
8	143	109	155.99
9	133	184	128.93
10	54	115	141.54
11	169	27	118.74
12	62	159	125.83
13	112	194	153.00
14	92	130	65.09
15	52	30	180.79
16	106	120	224.73

Tabelle 8: 16 Ergebnispaare des Dualpreisabstiegs für das G/H-Beispiels rc1_2_1.txt

JORLIB führt eine Protokolldatei zum BaP-Prozess mit folgendem Aufbau:

NodeID	pNodeID	objSol	nodeB	nodeV	cgIter	t_master	t_pricing	nrGC	solStat	nIQ
0	-1	6	4.00	4,00	15	10	54	286	frac	0
2	0	6	4.17	4,17	4	1	9	4	frac	1
1	0	6	4.00	4,00	3	2	1	4	frac	2
3	2	6	4.17	4,00	5	2	4	7	frac	3
5	1	6	4.00	4,00	3	12	1	4	frac	4
7	3	6	4.17	4,00	5	3	5	8	frac	5
9	5	6	4.01	4,00	2	1	0	2	int	6
11	7	4	4.17	-1,00	0	0	0	0	prun	5
12	7	4	4.17	-1,00	0	0	0	0	prun	4
10	5	4	4.00	-1,00	0	0	0	0	prun	3
8	3	4	4.17	-1,00	0	0	0	0	prun	2
6	1	4	4.01	-1,00	0	0	0	0	prun	1
4	2	4	4.17	-1,00	0	0	0	0	prun	0

Tabelle 9: Beispiel für JORLIBs Protokolldatei an einem Modell mit 12 Orten und der Tiefensuche

Legende: Abkürzung-Jorlibs Spaltenüberschrift: Deutsche Erklärung
NodeID-BAPNodeID: Identifikator des Knotens
pNodeID-parentNodeID: Identifikator des Vater-Knotens
objSol-objectiveIncumbentSolution: Oberer Bound
nodeB-nodeBound: Knoten-Schranke
nodeV-nodeValue: Knoten-Wert
cgIter-cgIterations: Anzahl Subprogramme
t_master, t_pricing: Rechenzeit für den Master bzw. Spaltengenerierung in Millisekunden
nrGC-nrGenColumns: Anzahl generierter Spalten
solStat-solutionStatus: Status der Lösung
nIQ-nodesInQueue: Anzahl der Knoten in der warteschlange
frac-Fractional: nicht ganzzahlig
prun-Pruned: aufgegeben, abgeschnitten

Literatur

[CPLEX1210] https://www.ibm.com/de-de/products/ilog-cplex-optimization-studio

[DantzigWolfe] George B. Dantzig; Philip Wolfe (1960). Decomposition Principle for Linear Programs. Operations Research 8, S. 101 bis 111.

[Feillet2010] Dominique Feillet, A tutorial on column generation and branch-and-price for vehicle routing problems, Operations Research 2010, S.407 bis 424

[Gehring/Homberger] Benchmarkdaten für das VRPTW bei großen Kundenanzahlen, https://www.sintef.no/projectweb/top/vrptw/homberger-benchmark

[JORLIB2015] Java Operations Research Library,
https://github.com/coin-or/jorlib/releases

[Andrew Makhorin] GNU Linear Programming Kit, https://www.gnu.org/software/glpk/.

[Mu2018] Harald Mumm, Didaktischer Zugang zur Theorie und Praxis moderner Softwarebibliotheken(Frameworks) für die Unternehmensforschung (OR), Wismarer Diskussionspapiere, Heft 8/ 2018 .

[Mu2020] Harald Mumm, Hybrider Ansatz zur Lösung des Fahrzeugrouten-Problems mit Zeitfenstern bei großen Ortsanzahlen, Heft 2/ 2020 .

Autorenangaben
Prof. Dr. rer. nat. Harald Mumm
Wirtschaftswissenschaftliche Fakulät
Hochschule Wismar
Philipp-Müller-Straße 14
Postfach 12 10
D-23966 Wismar
E-mail: harald.mumm@hs-wismar.de

WDP - Wismarer Diskussionspapiere / Wismar Discussion Papers

Heft 04/2015: Herbert Müller: Der II. Hauptsatz der Thermodynamik, die Philosophie und die gesellschaftliche Praxis – eine Neubetrachtung

Heft 05/2015: Friederike Diaby-Pentzlin: Auslandsinvestitionsrecht und Entwicklungspolitik: Derzeitiges bloßes internationales Investitionsschutzrecht vertieft Armut

Heft 02/2016: Günther Ringle: Die soziale Funktion von Genossenschaften im Wandel

Heft 01/2017: Benjamin Reimers: Momentumeffekt: Eine empirische Analyse der DAXsector Indizes des deutschen Prime Standards

Heft 02/2017: Florian Knebel, Uwe Lämmel: Einsatz von Wiki-Systemen im Wissensmanagement

Heft 03/2017: Harald Mumm: Atlas optimaler Touren

Heft 01/2018: Günther Ringle: Verfremdung der Genossenschaften im Nationalsozialismus – Gemeinnutzvorrang und Führerprinzip

Heft 02/2018: Sonderheft: Jürgen Cleve, Erhard Alde, Matthias Wißotzki (Hrsg.) WIWITA 2018. 11. Wismarer Wirtschaftsinformatiktage 7. Juni 2018. Proceedings

Heft 03/2018: Andreas Kneule: Betriebswirtschaftliche Einsatzmöglichkeiten von Cognitive Computing

Heft 04/2018: Claudia Walden-Bergmann: Nutzen und Nutzung von E-Learning-Angeboten im Präsenzstudium Analyse von Daten des Moduls Investition

Heft 05/2018: Sonderheft: Katrin Schmallowsky, Christian Feuerhake, Empirische Studie zum Messeverhalten von kleinen und mittleren Unternehmen in Mecklenburg-Vorpommern

Heft 06/2018: Dieter Gerdesmeier, Barbara Roffia, Hans-Eggert Reimers: Unravelling the secrets of euro area inflation – a frequency decomposition approach

Heft 07/2018: Harald Mumm: Didaktischer Zugang zur Theorie und Praxis moderner Softwarebibliotheken (Frameworks) für die Unternehmensforschung (OR)

Heft 01/2019: Astrid Massow: Deutsche Bank AG und Commerzbank AG – Neubewertung der Unternehmen im Rahmen einer potenziellen Bankenfusion

Heft 02/2019: Günther Ringle: Das genossenschaftliche Identitätsprinzip: Anspruch und Wirklichkeit

Heft 01/2020: Luisa Lore Ahlers: Einführung eines Wissensmanagements in kleinen und mittleren Unternehmen am Beispiel der Stadtwerke Wismar GmbH

Heft 02/2020: Harald Mumm: Hybrider Ansatz zur Lösung des Fahrzeugroutenproblems mit Zeitfenstern bei großen Ortsanzahlen

Heft 03/2020: Martin Seip: Automatisches Validieren von Meldedaten der EU-Bankenaufsicht

Heft 04/2020: Friederike Diaby-Pentzlin: Deficiencies of International Investment Law – What Chances for ‘’Critical Lawyers‘‘ to Civilize Global Value Chains and/or to Transform the Status Quo of the Economic World Order?

ISBN 978-3-948862-02-2