Mumm, Harald

Working Paper

Hybrider Ansatz zur Lösung des Fahrzeugroutenproblems mit Zeitfenstern bei großen Ortsanzahlen

Wismarer Diskussionspapiere, No. 02/2020

Provided in Cooperation with:
Hochschule Wismar, Wismar Business School

Suggested Citation: Mumm, Harald (2020) : Hybrider Ansatz zur Lösung des Fahrzeugroutenproblems mit Zeitfenstern bei großen Ortsanzahlen, Wismarer Diskussionspapiere, No. 02/2020, ISBN 978-3-942100-69-4, Hochschule Wismar, Fakultät für Wirtschaftswissenschaften, Wismar

This Version is available at:
https://hdl.handle.net/10419/214848

Fakultät für Wirtschaftswissenschaften
Wismar Business School

Harald Mumm

Hybrider Ansatz zur Lösung des Fahrzeugroutenproblems mit Zeitfenstern bei großen Ortsanzahlen

Heft 02/2020

W D P
Wismarer Diskussionspapiere / Wismar Discussion Papers

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 978-3-942100-69-4
JEL- Klassifikation: C61

Printed in Germany

Inhaltsverzeichnis

1 Einleitung

Zur Lösung ganzzahliger linearer Optimierungsaufgaben ist die Methode 'Branch-and-Bound' wohl bekannt. Sie wurde zur Methode 'Branch-and-Price' (Abk. BaP) für sehr große Optimierungsaufgaben verallgemeinert. Dabei werden bei nicht ganzzahligen Lösungen solange zwei neue ganzzahlige Optimierungsaufgaben formuliert, die zusätzliche, sich gegenseitig ausschliessende Nebenbedingungen erhalten, bis das ganzzahlige Optimum erreicht wird. Diese Methode soll in dieser Arbeit dahingehend hinterfragt werden, ob sie wirklich für große Probleme geeignet ist? Ein Teil dieser Methode ist die Spaltengenerierung, die dafür sorgt, dass ausgehend von einer Startlösung weitere Spalten hinzugefügt werden, sofern sie sich lohnen. Das Ziel dieser Arbeit ist es herauszufinden, bis zu welcher Ortsanzahl die Lösung von Fahrzeug-Routen-Problemen mit Zeitfenstern (engl. Vehicle-Routing-Problem with Time Windows, oder kurz VRPTW) aus dem Gehring/Homberger- Benchmark (siehe [Gehring/Homberger]) mit einem Spaltengenerierungsverfahren unter Nutzung der in WDP8/2018 beschriebenen Standardsoftware JORLIB (siehe [JORLIB2015]) in vertretbarem Zeitaufwand möglich ist?

Angestrebt werden schnelle exakte Lösungen für bis zu 1000 Orte. Dafür wird ein hybrider Ansatz gewählt, d. h. a priori Routenenumeration gekoppelt mit Minimumsuche im Subprogramm sowie Branch-and-Price für das Masterproblem. In bisherigen Artikeln mit ähnlichen Zielen wird meistens nur die Mathematik dazu beschrieben und in einem Kapitel 'Computational Results' werden dann Rechenzeiten für diese und jene Beispielinstanz angegeben, ohne die Implementierung im Detail zu beschreiben. Die Rechenzeiten fallen wie vom Himmel. Deshalb soll hier eine detaillierte Beschreibung der gewählten Implementierung erfolgen, um auch den Entstehungsweg der Rechenzeiten offen zu legen. Dabei soll jedoch die LKW-Kapazität anstelle von 200, wie im Gehring/Homber-Benchmark (siehe [Gehring/Homberger]), nur 50 betragen.

Diese Lücke soll in diesem Artikel am Beispiel unseres Zugangs zur Problemlösung geschlossen werden. Erläuterungen zur verwendeten Software 'JORLIB' (siehe [JORLIB2015]) und zum mathematischen Modell werden an dieser Stelle nicht noch einmal vorgenommen, weil sie bereits in [Mu2018] enthalten sind.

Das Grundprinzip bei jedem Spaltengenerierungsalgorithmus für ein gemischt ganzzahliges Optimierungsproblem (MILP) besteht in der Erzeugung einer ganzzahligen Anfangslösung für ein relaxiertes Masterprogramm, beim VRPTW z. B. durch Pendeltouren, und der schrittweisen Hinzunahme weiterer geeigneter Spalten, die den Zielfunktionswert der Anfangslösung soweit verbessert, bis es keine neuen Spalten mehr gibt. Damit hat man dann eine untere Grenze(lower bound) des gemischt ganzzahligen Optimierungsproblems (Minimierungsproblem) bestimmt.

Das Finden der Spalte mit den kleinsten negativen Kosten ist ebenfalls ein gemischt ganzzahliges Optimierungsproblem, beim VRPTW kann man es nach Feillet (siehe [Feillet2004]) als ein sogenanntes 'Elementary Shortest Path Problem with Resource Constraints' (Abk. ESPPRC) betrachten. Bei den Ressourcen handelt es sich beim VRPTW um die Ladekapazitäten des LKWs und um Zeitfenster, die einzuhalten sind. Einen didaktischen Einstieg in die Welt des kürzesten Wege-Problems findet man sogar interaktiv auf einer Website der TU-München (siehe [Graphalgorithmen]).

Die neu zu erzeugenden Spalten müssen negative, durch Dualwerte reduzierte, Kosten aufweisen, damit sie sich lohnen. Damit ist das Subproblem ein kürzester-Weg-Problem, wobei die Distanzen zwischen den Knoten negativ sein können und der zugrunde liegende Ortsgraph Zyklen enthalten kann. Dadurch sind der Dijkstra-Algorithmus oder auch der einfache Bellman-Ford-Algorithmus nicht mehr anwendbar. Der Zugang in dieser Arbeit geht von einem 'Shortest Path Problem with Resource Constraints' für das Subproblem aus, d. h. es werden a priori alle möglichen und zulässigen Wege vom Depot zu den Kunden und zurück ermittelt. Im Subproblem wird dann von dieser Menge derjenige Weg ausgesucht, der die kleinsten negativen Kosten aufweist (siehe [Feillet2010]).

In der Nebensache geht es um die Sammlung von Erfahrungen bei der Implementierung der Methode 'Branch-and-Price' für das VRPTW mit komplizierten Softwarepaketen wie JORLIB (ca. 500 Java-Klassen) und dem LP-Solver 'CPLEX'.

Die Implementierungssprache in dieser Arbeit ist Java-Version 8.2 und der LP-Solver ist CPLEX in Version 12.8 (siehe [CPLEX128]). Die Programmierarbeiten wurden auf einem Notebook Thinkpad T530 mit Intel(R) Core(TM) i5-3320M CPU 2,6 GHZ Taktfrequenz und 16 GB RAM durchgeführt.

Wir wollen die Einleitung mit einem Beispiel beenden:

Gegeben sind ein Depot und sechs Orte. Ihre Lage wird durch zweidimensionale Koordinaten in der Ebene beschrieben. Die sechs Orte haben einen Bedarf an Gütern in gewissen Mengeneinheiten. Die Lieferung der Bedarfe kann nur innerhalb von Zeitfenstern, gegeben durch das Intervall [ZFU,ZFO] erfolgen. Die Belieferung erfordert nach Ankunft der Fahrzeuge eine gewisse Servicezeit. Sie kann spätestens beim ZFO-Wert beginnen. Die Zeitfensterangaben und Servicezeiten sind hier keine Angaben in Minuten sondern Weglängen mit einem euklidischen Abstandsbegriff. Die Zeitfenstergrenzen 750 und 809 sowie die Serviczeit 90 stehen z. B. für eine Zeit, die das Fahrzeug für den euklidischen Abstand von 750, 809 bzw. 90 mit konstanter Geschwindigkeit bentigt.

Hier nun die Beispielwerte:

Lfd	xKoord	yKoord	Bedarf	ZFU	ZFO	Servicezeit
0	70	70	0	0	1351	0
1	33	78	20	750	809	90
2	59	52	20	0	1240	90
3	10	137	30	147	219	90
4	4	28	10	0	1183	90
5	25	26	20	128	179	90
6	86	37	10	478	531	90

Tabelle 1: Eingabedaten für das Depot und die der ersten sechs Orte

Gesucht ist die minimal notwendige Touranzahl und dazu die minimal zurückzulegende Entfernung vom Depot mit der laufenden Nummer (Lfd) 0 zu den Orten mit der laufenden Nummer 1 bis 6 und zurück zum Depot, wenn die LKW-Kapazität konstant 50 beträgt.

Dieses Beispiel `C1_2_3` und auch alle weiteren stammen aus dem Gehring/Homberger-Benchmark, siehe [Gehring/Homberger] für die ersten 6 Orte.

Zuerst wird die minimale Touranzahl ermittelt:

Lfd	Länge	Orte in Tour	Ort1	Ort2	Ort3	Ladung
1	191.12	2	3	1	0	50
2	126.84	2	2	5	0	40
3	197.4	2	4	6	0	20

Tabelle 2: Minimale Touranzahl für das Problem aus Tabelle 1

Die minimale Touranzahl beträgt 3, die Gesamtlänge noch 515,36. (Es soll noch einmal darauf hingewiesen werden, dass es hier keine Maßeinheit für die Länge gibt, weil mit euklidischen Abständen in der Ebene gerechnet wird.)

Danach werden die Touren mit minimaler Gesamtlänge zu vorgegebener Touranzahl von 3 berechnet, wobei die Lösung aus Tabelle 2 als Anfangslösung verwendet wird.

Lfd	Länge	Orte in Tour	Ort1	Ort2	Ort3	Ladung
1	73.35	1	6	0	0	10
2	191.12	2	3	1	0	50
3	163.23	3	4	5	2	50

Tabelle 3: Minimale Tourlängen für das Problem aus Tabelle 1

Die minimale Gesamtlänge beträgt mit diesen Touren nur noch 427,7.

2 Spaltengenerierung und Branch-And-Price

2.1 Grundlagen und Master

Die Grundlagen zum Verfahren der Spaltengenerierung findet man u.a. in [GRUIRN2005] und [Feillet2010] und als Zitat in [Mu2018]. Es gibt ein **Masterprogramm** (MP), welches in der Zielfunktion minimale Kosten fordert und in den Nebenbedingungen die Kundenabdeckung garantiert, und ein Subprogramm, welches kürzeste Wege, die zeitfenster- und kapazitätsgültig sind, ermittelt. Diese Wege, auch Touren oder Spalten genannt, werden ausgehend von einer Initiallösung schrittweise hinzugefügt, falls man erwarten kann, dass sich das Masterergebnis verbessert. Sofern noch nicht alle Wege gefunden sind, spricht man auch vom 'restricted Master Programm', kurz RMP.

Es folgt ein Beispiel für ein Master-Programm mit den Touren aus Tabelle 2 für den Aufgabentyp 'ZF1' (minimale Distanz bei vorgegebener Touranzahl).

```
\ENCODING=ISO-8859-1
\Problem name: ilog.cplex

Minimize
 obj: 191.12 x1 + 126.84 x2 + 197.4 x3
Subject To
erfuelltBedarfInOrt1: x1   = 1
erfuelltBedarfInOrt2: x2   = 1
erfuelltBedarfInOrt3: x1   = 1
erfuelltBedarfInOrt4: x3   = 1
erfuelltBedarfInOrt5: x2   = 1
erfuelltBedarfInOrt6: x3   = 1
erfuelltTouranzahl:   x1 +x2 + x3    = 3
End
```

In der Zielfunktion taucht jede der 3 verwendeten Touren mit ihrer Länge als Gewicht auf. Die ersten sechs Nebenbedingungen der Art 'erfuelltBedarfInOrt' garantieren, dass alle sechs Orte beliefert werden. Die Nebenbedingung 'erfuelltTouranzahl' garantiert, dass entweder nur 3 von 6 Touren mit dem Wert 1 in der Lösung verwendet werden (ganzzahlige Lösung) oder die Werte auch kleiner als 1 sein müssen (gebrochene Lösung) . Der Ort Nr. 1 ist z. B. in der Tour mit Lfd 1 enthalten (siehe Nebenbedingung 'erfuelltBedarfInOrt1') und der Ort Nr. 2 in der Touren mit Lfd 2 . Für den Aufgabentyp 'ZF0' (minimale Touranzahl) sieht das Masterproblem sehr ähnlich aus. Lediglich die Gewichte der Touren fallen in der Zielfunktion weg, sind also gleich '1'.

2.1.1 Minimierung der Touranzahl

Bevor das eigentliche VRPTW gelöst wird, soll zuerst die minimal notwendige Touranzahl ermittelt werden. (Dieses Problem wird im folgenden mit ZF0 kodiert.) Die Koeffizienten (Kosten) der Variablen in der Zielfunktion des VRPTW, so wie es in [Mu2018] beschrieben wurde, sind dann gleich Eins. Eine Anfangslösung für das RMP ist die Stichlösung, d. h. jeder Kunde wird durch eine separate Tour bedient (Pendeltour). Wenn man eine Lösung für dieses Problem bei maximal Tripeltouren (je Tour können nur maximal drei Orte angefangen werden) hätte, könnte man Beispiele mit 1000 Orten z. B. in drei Aufgaben zerlegen, und zwar für die Orte 1-333, Orte 334-666, Orte 667-1000. Wenn nun die Summe ZF0(1-333)+ZF0(334-666)+ZF0(667-1000)=ULS (untere Ladeschranke) ist, hätte man auch ZF0(1-1000) als ULS ermittelt. ULS ist die kleinste positive ganze Zahl, die größer ist als die Summe der Bedarfe aller Orte dividiert durch die Ladekapazität der Fahrzeuge. Auf diese Art und Weise bekommt man z. B. heraus, dass die minimale Touranzahl für das GH-Beispiel `r1_10_1.txt` aus [Gehring/Homberger] 363 beträgt, wenn man von einer Fahrzeugkapazität von 50 Mengeneinheiten ausgeht.

2.1.2 Minimierung der Entfernung bei fester Touranzahl

Das VRPTW, so wie es in [Mu2018] beschrieben wurde, erhält eine zusätzliche Nebenbedingung über die Anzahl der Touren gemäß des Ergebnisses nach 2.1.1 (Minimierung der Touranzahl). Die Anfangslösung für das RMP zur Minimierung der Entfernung (ZF1) bei minimaler Touranzahl ergibt sich dann z. B. aus der Lösung aus 2.1 (Minimierung der Touranzahl).

Bei der Bestimmung der minimalen Entfernung bei vorgegebener minimaler Routenanzahl funktioniert das eben beschriebene Verfahren mit der Zerlegung in mehrere Dateien bei nur Tripeln in den Ergebnisrouten nicht mehr. Aber immerhin kann man Näherungswerte angeben, wie z. B. für das obere Beispiel `r1_10_1.txt` eine Entfernung von ca. 168656 Entfernungseinheiten.

2.2 Das Subproblem

Das Besondere an unserem Zugang zum Subproblem liegt darin, dass wir nicht in jedem Iterationsschritt eine Optimierungsaufgabe zur Bestimmung des kürzesten Weges vom Depot und zurück lösen, so wie noch in [Mu2018] beschrieben, sondern nur eine einfache Minimierungssuche in einer Reihe von Touren vornehmen, weil wir a priori alle möglichen Pfade (Wege, Touren) vom Depot zu den Kunden und zurück zum Depot ermitteln, die kapazitäts- und fensterzulässig sind. Das heißt, wir lösen a priori ein sogenanntes 'Shortest Path Problem with Resource Constraints' (SPPRC), auf Deutsch: Kürzester Weg Problem mit Ressourcen-Einschränkungen. Wir versprechen uns von diesem Vorgehen eine Zeitersparnis bei der Lösung des Subproblems. Die Ressourcen sind in unserem Fall die Kosten (Länge des Weges) ,die Ankunftszeit sowie die Ladekapazität.

Es wird in jedem Iterationsschritt des BaP-Prozesses (BaP-Abkürzung für Branch-and-Price) nicht nur die Tour mit den kleinsten Kosten, sondern auch die mit den zweitkleinsten Kosten ermittelt und hinzugefügt. Man könnte auch die ersten drei Touren mit negativen Kosten verwenden usw. Von den Tourkosten der a priori ermittelten Touren werden aber erst im Subproblem die Dualwerte der Orte (ergeben sich aus dem RMP) abgezogen und das Minimum der Kosten bestimmt.

Der zugrunde liegende Orts-Graph weist negative Kosten und Zyklen auf, weshalb der einfache Bellman-Ford-Algorithmus, so wie in [Graphalgorithmen] beschrieben, nicht verwendet werden kann. Anstelle davon wird hier der Bellman-Moore-Algorithmus verwendet, der in jedem Knoten mit mehrdimensionalen Marken arbeitet. Die Marken verweisen auf den Vorgänger im Weg vom Depot zum aktuellen Knoten.

Um Zyklen zu vermeiden, müssen auch alle bereits auf dem Weg besuchten Knoten registriert werden. Die im Bellman-Moore-Algorithmus ermittelten Marken im Depot weisen zurück auf den letzten Ort der Tour und die Marke im letzten Ort auf den vorletzten Ort der Tour usw. Das Depot erscheint im Ortsgraph zweimal, einmal mit der Nummer 0 und einmal mit der Nummer n+1 als Knoten. Sämtliche Marken im Depot (Knoten n+1) weisen also auf zulässige Wege vom Depot über einen oder mehrere Orte zum Knoten n+1. Diese Wege werden in dieser Arbeit auch effektive Pfade genannt. Die Marken wurden vorher auf Dominanz geprüft, d. h. offensichtlich schlechtere Wege, die später ankommen und dabei noch länger sind und die gleichen Orte aufsuchen, wurden vorher verworfen.

Einen Markenalgorithmus (englisch: Label-Setting-Algorithm) dieser Art findet man für positive Distanzen in [GRUIRN2005] S. 310 und ein ausführliches Beispiel ebenda S. 312.

Für das Beispiel aus der Einleitung ergeben sich die folgenden kapazitäts- und zeitfernster-zulässigen Touren:

Lfd	Kosten	Gran	Ort1	Ort2	Ort3	Ort4	Ort5	Ladung
1	73.35	1	6					10
2	125.88	1	5					20
3	156.47	1	4					10
4	179.88	1	3					30
5	42.20	1	2					20
6	75.71	1	1					20
7	173.89	2	1	4				30
8	95.72	2	1	2				40
9	126.84	2	2	5				40
10	159.34	2	2	4				30
11	209.15	2	2	3				50
12	252.22	2	3	6				40
13	277.34	2	3	4				40
14	191.12	2	3	1				50
15	197.40	2	4	6				20
16	162.27	2	4	5				30
17	161.60	2	5	6				30
18	153.41	2	5	1				40
19	88.66	2	6	2				30
20	141.54	2	6	1				30
21	142.19	3	6	2	1			50
22	200.28	3	6	4	2			40
23	214.83	3	6	4	1			40
24	179.69	3	5	4	1			50
25	229.79	3	5	6	1			50
26	197.99	3	4	5	6			40
27	163.23	3	4	5	2			50
28	318.28	3	3	4	6			50
29	176.76	3	2	4	1			50
30	162.56	3	2	5	6			50

Tabelle 4: Alle möglichen Touren vom Depot und zurück bei den ersten 6 Orten des G/H-Beispiels C123

Ortsanzahl(ohne Depot)	Anzahl Pfade	davon gesättigt	Rechenzeit(ohne Ausgabe)
5	17	5	weniger als 1 Sekunde
10	101	44	weniger als 1 Sekunde
50	32680	25320	2 Sekunden
60	74546	60018	4 Sekunden
70	123.454	101931	6 Sekunden
80	207.507	174.709	11 Sekunden
90	498.832	431.644	31 Sekunden
100	932.291	818.327	55 Sekunden
110	1.366.719	1.212.582	1 Minuten, 38 Sekunden
120	2.059.142	1.884.145	3 Minuten, 2 Sekunden
125	2.783.848	2.505.833	22 Minuten, 10 Sekunden

Tabelle 5: Rechenzeiten für die Ermittlung aller zulässigen Pfade

Die gesättigten Touren (Ladung 50) sind insofern interessant, als dass sie evt. für die Spaltengenerierung ausreichen, und zwar immer dann, wenn man in einem ersten Schritt erkannt hat, dass ULS viele Touren reichen.

Bei n=130 reichte der Speicherplatz (RAM) des verwendeten Notebooks von 8 GB für die Berechnung aller möglichen Pfade schon nicht mehr aus. Daran erkennt man, dass das anfängliche Ziel, auf diese Art, alle Pfade für n=1000 zu ermitteln, nicht durchzuhalten sein wird. Beispielhafte Rechenzeiten findet man in Tabelle 5.

Jeder Ortsknoten enthält sämtliche Pfadeinträge, wie er kapazitäts- und zeitfensterkonform erreicht werden kann. Das Depot mit der Nr.0 kommt zweimal vor, beim zweiten Mal trägt es die Nummer n+1. Am Ende des SPPRC-Algorithmusses liegen im Knoten Nr. n+1 sämtliche für die Spaltengenerierung relevanten Pfade vor. Dieser Algorithmus heisst Bellman-Moore-Algorithmus.

Für die Implementierung des Bellman-Moore-Algorithmus wurden folgende Klassen verwendet.

```
public class Pfadeintrag {

    Stack<Integer>  pfad=new Stack();
    double ab;  //Ankunftszeit
    int nochfrei; //Ladung
    double  kosten;  //Tourlaenge}
public class Knoten{
    int nummer;
    int x;  //x-Koordinate
    int y;  //y-Koordinate
    int b;  //Bedarf
    int sz; //Serviczeit
    int [] zf;//Zeitfenster

ArrayList<Pfadeintrag> pfadeintraegeZu= new ArrayList() ;
.....
}
```

Die Instanzvariable 'pfad' in der Klasse 'Pfadeintrag' ist notwendig, um Zyklen in den Pfaden zu erkennen. Hier werden die Nummern der bereits besuchten Kunden abgelegt.

2.3 Verzweigungsregeln und deren Implementierung

Im Allgemeinen ist der Master nach der ersten Iterationsrunde der Spaltengenerierung nicht ganzzahlig. Man muss jetzt zwei 'Problem-Söhne' (....) kreieren, die hier nach der Ryan/Foster Regel (siehe [RYAN1981])) gebildet werden. Dazu muss ein Ortspaar in einer im Master gebrochen bewerteten Tour gesucht werden, deren Items in der fraktalen Lösung von Knoten Nr. 0 des BaP-Baumes sowohl als Paar als auch einzeln im Tourverlauf auftreten. Da vorerst maximal fünf Orte je Tour angenommen werden, gibt es diese 10 Auswahlvarianten für zwei Orte, wenn die Ortsnummern bei 0 beginnen:
(0,1),(0,2),(0,3),(0,4),(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)
Hat man ein Orts-Paar mit einem dieser Index-Paare gefunden, werden für den linken Sohn (JORLIBs-Knoten Nr. 2) alle Ergebnistouren des ersten fraktalen Masters (Knoten Nr. 0) gelöscht, in denen die Items des gefundenen Orts-Paares paarweise aufgetreten sind (Zweig Single). Im rechten Sohn (JORLIBs-Knoten Nr. 1) wird umgekehrt verfahren, d. h. es werden alle Ergebnistouren des ersten fraktalen Masters (Knoten Nr. 0) gelöscht, in denen diese Items einzeln aufgetreten sind (Zweig Pair). Durch das Löschen von Spalten sind die Master in Knoten Nr. 2 und Knoten Nr. 1 erstmal nicht mehr lösbar. Deshalb werden künstliche Spalten hinzugefügt, die in der Zielfunktion des Masters sehr hoch, d. h. schlecht, bewertet werden, damit sie niemals in einer späteren Lösung auftauchen. Nun beginnt zuerst für den Knoten Nr. 2 eine neue Iterationsrunde zur Spaltengenerierung mit negativen Kosten, wobei nur noch in einer Teilmenge der ursprünglichen Menge der effektiven Pfade gesucht wird, und zwar in der alle Touren mit gleichzeitigem Erscheinen des ausgewählten Ortspaares gelöscht wurden. Die gelöschten Touren dürfen aber nicht einfach nur gelöscht werden, sondern müssen aufgehoben werden, weil sie später wieder hinzugefügt werden müssen, und zwar wenn im BaP-Baum zurückgegangen wird. In der Klasse 'ExactPricingProblemSolver' wird eine Instanzvariable dafür wie folgt definiert:

```
Stack<HerausPfade> raus_stack= new Stack();  //Stack der rausgefallenen  eff. Pfade
```

Ausserdem enthält sie eine Instanzvariable 'bm', die alle effektiven Pfaden enthält. In der Methode 'branchingDecisionPerformed' dieser Klasse werden beim Abstieg die nicht kompatiblen Pfade gelöscht, hier am Beispiel der Paare (Kodierung 1):

```
HerausPfade hp=   buildModel(1,i,j);  //Loeschen in allen Pfaden bereits geschehen
raus_stack.push(hp);
```

In der Methode 'branchingDecisionReversed' dieser Klasse werden beim Aufstieg die vorher gelöschten Pfade wieder hinzugefügt:

```
HerausPfade hp=raus_stack.pop();
   //Wiedereinfuegen in alleEffPfade
   for (Tour tl:hp.effPfade)
    bm.effPfadeAL.add(tl);
```

Wenn keine Spalten mit negativ reduzierten Kosten mehr existieren, werden von Knoten Nr. 2 wieder zwei Söhne gebildet, und zwar Knoten Nr. 4 (Zweig Single) und Knoten Nr. 3 (Zweig Pair).
Es folgt ein Beispiel für die Lösung des Masters in Knoten Nr. 0 für n=50, aufsteigend sortiert nach der Differenz zur 0,5 zur Suche des Ryan/Foster-Ortspaares.

Master-Wert	Ort1	Ort2	Ort3	Ort4	Ort5
0.5	35	1	0	0	0
0.5	29	50	15	0	0
0.5	48	42	29	0	0
0.5	13	43	39	0	0
0.5	26	40	14	0	0
0.5	9	1	0	0	0
0.5	42	50	0	0	0
0.5	39	2	17	0	0
0.5	48	26	40	0	0
0.5	17	35	9	0	0
0.5	13	43	2	0	0
0.55	45	27	0	0	0
0.55	20	24	0	0	0
0.45	45	27	24	0	0
0.40	10	46	4	0	0
0.60	36	4	34	0	0
...	...	...	...	...	...
0.20	32	47	6	0	0
0.20	32	47	12	0	0
0.20	10	33	0	0	0
0.85	41	31	25	0	0
0.15	20	31	0	0	0
0.15	20	41	0	0	0
0.15	20	25	0	0	0
1.0	21	23	0	0	0
1.0	30	44	0	0	0

Tabelle 6: Sortierte Lösung von Knoten Nr. 0 zur Auswahl eines Ortspaares für die Verzweigungsregel

Schon in der ersten Zeile wird man fündig und findet das Ortspaar 35, 1 als Ryan/Foster-Paar, denn die '1' taucht in der 6. Zeile und die 35 in der 10. Zeile als Single auf.

Im linken Knoten 'Single' dürfen nur Touren hinzugefügt werden, die nur eines der Items dieses Paares enthalten und im rechten Knoten 'Pair' nur solche Touren, die beide Items enthalten. Alle Ergebnistouren von Knoten Nr. 0, die dagegen verstoßen, müssen im Master gelöscht werden. Damit der Master danach noch lösbar ist, wird eine künstliche Lösung hinzugefügt. Ihre Variablen werden im Master aber von JORLIB so schlecht gewichtet, dass sie niemals im Endergebnis auftreten werden. Als künstliche Lösung kann man die Anfangslösung verwenden. Zur Reduzierung der exponentiell wachsenden Anzahl aller effektiven Pfade wird im nächten Kapitel die Methode der Variablenfixierung hinzugezogen.

3 Variablenfixierung

Die Idee der sogenannten 'Variablenfixierung' (VF) findet man z. B. in [COSTA2019]. Sie besagt, dass man im Subproblem nur diejenigen Touren in die Suche einschliessen muss, die sich lohnen. Das sind Touren, für die gilt: Mittels Dualwerte reduzierte Kosten <= ObereGrenze(OG)-UntereGrenze(UG). Der Wert 'UntereGrenze' ergibt sich aus der nicht ganzzahligen Lösung von Knoten Nr. 0.

Zum VF werden benötigt: Als Lieferant der oberen Grenze eine zulässige ganzzahlige Lösung des gegebenen primalen Tourenplanungsmodells TPM und als Lieferant der unteren Grenze sowie der Dualpreise DP ein fraktionales primales (und somit auch duales) Optimum des relaxierten, reinen LP-Modells RTPM, das aus TPM durch Weglassen der Ganzzahligkeitsforderungen hervorgeht. Die Erfahrungen mit dem VF haben ergeben, dass die Schwachstelle dabei die obere Grenze ist. Werden zuviele Variablen aus dem Suchraum gelöscht, ist sie also zu klein gewählt, bleibt nur die ganzzahlige Anfangslösung übrig, und werden zu wenig Variablen gelöscht, ist sie also zu groß gewählt, bringt das VF nichts.

Wir demonstrieren die Rechenzeiten ohne und mit VF am Beispiel `C1_2_3.txt`
(siehe [Gehring/Homberger]) und den ersten 50 Orten am Beispiel der Problemart 'ZF1' (minimale Distanz). Die Erzeugung der 32680 effektiven Pfaden dauerte zwei Sekunden. Ohne VF dauerte die Berechnung des Optimums 13,5 Minuten und es wurden 12249 Knoten angelegt. Mit VF dauerte die Berechnung des Optimums nur eine Sekunde, und es wurden lediglich 59 Knoten im Branch-Baum e erzeugt. Wenn man noch die 5 Sekunden zum Finden der Dualwerte in Knoten Nr. 0 hinzu addiert, kommt man auf insgesamt 2+5+1=8 Sekunden Rechenzeit.

Mittels Variable-Fixing(VF) wurde die Menge der eff. Pfade auf 4544 (32680) reduziert. Die dazu notwendigen Dualwerte wurden aus der optimalen Lösung von Knoten Nr. 0 gewonnen. Den Wert 'UntereGrenze' erhält man aus dem Zielfunkfionswert im Optimum von Knoten Nr. 0, hier im Beispiel gleich 2282,0. Der Wert für 'ObereGrenze' ist der Wert einer zulässigen Lösung des Ursprungproblems. Hier bedarf es einiger Erfahrungen, um eine geeignete zulässige Lösung für das VF zu finden. Die rechte Seite des VF-Filters ist sehr wichtig für die Reduktion der Variablenanzahl. Nur im Intervall der Differenz OG minus UG von 186 bis 196 für die rechte Seite der VF-Ungleichung erzielte man in diesem Beispiel sehr gute Rechenzeiten.

Hier das Ergebnis nach nur einer Sekunde Rechenzeit, so wie es JORLIB anzeigt:

```
 ========= Loesung ===fuer n=50
lowerBound=2342upperBound=2347
VRP beendet with objective): 2342
Total Number of iterations: 1703
Total Number of processed nodes: 59
Total Time spent on master problems: 550 Total time spent on pricing
problems: 279
Solution is optimal: true
Columns (only non-zero columns are returned):
Master-Value: 1.0 Verlauf: [42, 50, 0, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [21, 23, 0, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [44, 38, 22, 16, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [41, 31, 25, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [35, 7, 49, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [32, 47, 12, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [8, 18, 0, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [9, 1, 0, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [13, 43, 2, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [34, 5, 10, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [3, 37, 0, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [33, 46, 0, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [26, 40, 29, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [14, 15, 19, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [6, 11, 39, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [45, 27, 0, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [36, 4, 17, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [48, 28, 30, 0, 0] creator: exactPricing
Master-Value: 1.0 Verlauf: [20, 24, 0, 0, 0] creator: exactPricing
BUILD SUCCESSFUL (total time: 1 second)
```

Für n=60 entstehen 60018 effektive Pfade. Ohne VF ensteht eine 'out.of.memory Java-exception' beim BaP-Prozess. Mit VF bleiben 5110 Pfade übrig, und die Rechenzeit beträgt für BaP lediglich 33 Sekunden, wobei 1913 Knoten erzeugt wurden. Hinzu kommen 11 Sekunden zur Ermittlung der Dualwerte von Knoten Nr. 0. Wenn man die 4 Sekunden zur Berechnung der effektiven Pfaden hinzuzählt, ergibt das insgesamt eine Rechenzeit von 48 Sekunden für n=60 bis zum Optimum des VRPTW in der Problemklasse 'ZF1'.

Die Kosten jeder Tour sind beim Problem der Minimierung der Touranzahl (ZF0) gleich Eins. Wenn man das VRPTW dahingehend weiter einschränkt, dass man nur Bedarfswerte von 10, 20, 30 und 40 zulässt, kann man beweisen, dass die Dualwerte der Lösung von Knoten Nr. 0 entweder 0,2 oder 0,4 oder 0,6 oder 0,8 betragen, und zwar für Orte mit Bedarf 10 gleich 0,2; für Orte mit Bedarf 20 gleich 0,4 usw. Diese Erkenntnis hat den Vorteil, dass man a priori die Dualwerte für die VF-Bedingung kennt. Um an die untere Schranke zu kommen, muss man allerdings trotzdem den Spaltengenerierungsprozess bis Knoten Nr. 0 anstoßen.

Ohne VF dauerte die Berechnung der minimalen Touranzahl (45) für n=125 immerhin noch ca. 38 Minuten bei 2783848 effektiven Pfaden, für n=100 (36) bei 932291 effektiven Pfaden noch 4 Minuten und für n=50 (19) bei 32680 effektiven Pfaden nur 6 Sekunden. Mit VF und der Reduzierung der effektiven Pfade auf 197759 (932291) kann die Rechenzeit für n=100 auf 39 Sekunden (4 Minuten) reduziert werden und bei der Verwendung von 101525 (932291) sogar auf 20 Sekunden (4 Minuten) .

Weitere Beispiele für den Einsatz des VF beim Problemtyp 'ZF1' ergeben sich aus der Datei `r1_2_1.txt`, wenn nur die ersten 50, 60, 70, 80 oder 90 Orte verwendet werden. Die minimalen Touranzahlen sind hier gleich 19, 24, 27, 31 bzw. 35. Die Rechenzeit bei n=50 für die minimale Distanz von 2942,87 beträgt 14 Sekunden ohne VF und 6 Sekunden mit VF bei Verwendung von 6215 von 7030 eff. Pfaden. Bei n=60 und n=70 wird durch VF ebenfalls Rechenzeit eingespart.

n	TAZ	Obj	Rechenzeit	Rechenzeit-VF	UG	OG	Pfade	VF-Pfade	Knoten	K-VF
50	19	2943	14 Sek.	6 Sek.	2895	3250	7030	6215	939	329
60	24	3298	1 Min.	6 Sek.	3273	3500	11018	4049	2678	343
70	27	3574	7 Min.	1,5 Min.	3565	3800	29947	10122	4790	2482
70	27	3574	7 Min.	24 Sek.,	3565	3780	29947	7041	4790	872
70	27	3677	7 Min.	23 Sek.,	3565	3750	29947	3402	4790	1248
80	31	4124	—	19 Sek.	4092	4350	40143	18763	—	330
90	35	4615	12Min.	109 sek.	4565	4800	80692	28309	5066	1434

Tabelle 7: Effekte des Variable Fixing an ZF1-Beispielen

Die vierte und fünfte Zeile in dieser Tabelle weisen auf ein Problem hin: Die Bestimmung der oberen Grenze (OG). Man weiss nicht, wie man sie setzen soll? Wird sie zu klein gewählt, ist die Lösung nicht mehr optimal, wie in Zeile 5 ausgewiesen. Wird sie zu groß gewählt, ergeben sich keine Einsparungen bei den Touren. Weniger Touren im Suchraum bedeuten allerdings nicht automatisch eine Verkürzung der Rechenzeit. Für n=80 konnte keine Lösung ohne VF in weniger als 60 Minuten Rechenzeit erzielt werden. Die Spalten 'Pfade' und 'VF-Pfade' beinhalten die Größe des Suchraumes ohne und mit Nutzung des VF. Die Spalten 'Knoten' und 'K-VF' geben die Anzahl der Knoten im BaP-Baum ohne und mit VF-Benutzung an.

4 Resultatsermittlung und -bewertung

Das Hauptziel der Arbeit, einen schnellen exakten Lösungsalgorithmus für das VRPTW bei bis zu 1000 Orten mit der Methode 'Branch-and-Price' zu entwickeln, wurde nicht erreicht. Diese Methode war dafür anscheinend nicht geeignet.

Vertretbare Rechenzeiten von weniger als einer Minute gab es nur in Beispielen mit bis zu 80 Orten. Erreicht werden konnten quantitative Aussagen über die Anzahl von effektiven Pfaden an Beispielen, die so konkret nicht in der Literatur zu finden waren. Bei n=125 gab es z. B. schon fast 3 Mio. derartiger Pfade und in einem anderen Beispiel bei n=200 sogar fast 7 Mio. Diese große Anzahl von effektiven Pfaden ist auch eine Ursache für die sehr schlechten Rechenzeiten bei mehr als 100 Orten, denn schon die Berechnung der fast 3 Mio effektiven Pfade bei 125 Orten dauerte alleine für das Beispiel `c1_2_3.txt` aus [Gehring/Homberger] schon 22 Minuten. Für eine Verbesserung der Rechenzeiten mittels der Methode 'Variable Fixing' sind gute Werte für die oberen und unteren Grenzen sehr wichtig. Die untere Grenze ist relativ leicht aus Knoten Nr. 0 (Wurzel des BaP-Baumes) zu ermitteln, aber eine gute obere Grenze zu finden, ist schwierig.

Mit der Beschränkung auf Tripel im Tourverlauf (max. drei Orte je Tour) war es möglich, mittels Enumeration aller eff. Pfade und eines ganzzahligen Linearen Programmes (in GMPL) und dem Solver CPLEX das VRPTW für ausgewählte Beispiele aus [Gehring/Homberger] bis n=334 näherungsweise zu lösen, ohne selber das BaP-Verfahren programmieren zu müssen.

Die Rechenzeit zur optimalen Bestimmung der minimalen Touranzahl bei 1000 Orten dauerte mit dieser Methode am Beispiel `r1_10_10.txt` (siehe [Gehring/Homberger]) mittels Zerlegung in drei Ortsmengen und der ermittelten minimalen Touranzahl von 363 schon 1200 Sekunden. Die errechnete Distanz betrug ca. 166117. Dieser Wert kann als obere Grenze für Vergleichsrechnungen verwendet werden.

Mit der Methodik dieser Arbeit können beliebige Bedarfe zwischen 5 und 49 im VRPTW-Problem vorkommen oder Beispiele mit maximal 10 Orten je Tour. Für ein Beispiel mit 200 Orten und einer Bedarfssumme von 3500 konnte eine minimale Touranzahl von 70 Touren in 20 Minuten Rechenzeit ermittelt werden. Hinzu kommt die Rechenzeit für die einmalige Berechnung der ca. 6 Mio. effektiven Pfade, wovon ca. 900000 gesättigt sind, von 3,5 Minuten. Die Berechnung der minimalen Tourlänge bei diesem Beispiel mit 200 Orten gelang leider nicht.

Zusammenfassend kann festgestellt werden, dass für relativ kleine n, wie z. B. n= 90, die reine Enumeration (alle effektiven Pfade stehen für je eine Variable) beim Problemtyp ZF1 wesentlich schneller arbeitet als die Methode BaP (7 Sekunden mit dem Solver CPLEX 12.8 vs. mehrere Minuten mit der BaP-Methode), zumindest wenn sie so implementiert wurde, wie hier beschrieben. Die Rechenzeit für die Bestimmung aller eff. Pfade ist darin noch nicht enthalten. Sie beträgt z. B. für n=90 in der hier gewählten Implementierung zwei Sekunden.

Im Rahmen dieser Arbeit wurden willkürlich die beiden (bzgl. negativ reduzierter Kosten) besten Touren durch das Subprogramm bestimmt und in die Spaltengenerierung übernommen. Es wäre zukünftig zu prüfen, ob eine andere Strategie bessere Laufzeiten ergeben würde. Eine weitere Idee für die Fortentwicklung besteht darin, das VF in jedem Knoten des BaP-Baumes erneut durchzuführen.

Des Weiteren ist zu untersuchen, ob andere Verzweigungsregeln als die hier verwendete, bessere Rechenzeiten ermöglichen und ob eine heuristische Anfangslösung für den Problemtyp 'ZF0' (minimale Touranzahl) besser ist als die hier gewählte Stichlösung.

Positiv zu erwähnen sind die Erfahrungen, die mit dem Programmpaket JORLIB gemacht wurden. Es ist sehr gut handhabbar und funktionierte fast fehlerfrei.

Literatur

[COSTA2019] Luciano Costa, Claudio Contardo, Guy Desaulniers, Exact Branch-and-Cut Algorithms for Vehicle Routing, Transportation Science 53(4):946-985, 2019, ISSN 1526-5447(online)

[CPLEX128] `https://www.ibm.com/de-de/products/ilog-cplex-optimization-studio`

[Feillet2004] Feillet D., Dejax P., Gendreau M., Gueguen, C., An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to some Vehicle Routing Problems, Networks 44(3):216-229, 2004

[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

[GRUIRN2005] Grünert T., Irnich S.: Optimierung im Transport, Touren und Wege, Band 2, Shaker 2005

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

[Andrew Makhorin] 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 .

[Mu2018] Harald Mumm, Logistika 1.0, Android App, Tourenplanung als Strategiespiel, Google Play Store 2018 .

[RYAN1981] D. M. Ryan and B. A. Foster: An Integer Programming Approach to Scheduling, In Computer scheduling of public transport: Urban passenger vehicle and crew scheduling, A. Wren editor, North-Holland 1981, 269-280.

[Graphalgorithmen] https://www-m9.ma.tum.de/Allgemeines/GraphAlgorithmen.

Autorenangaben
Prof. Dr. rer. nat. Harald Mumm
Fachbereich Wirtschaft
Hochschule Wismar
Philipp-Müller-Straße 14
Postfach 12 10
D-23966 Wismar
Telefon: ++49 / (0)3841 / 753 450
Fax: ++49 / (0)3841 / 753 131
E-mail: harald.mumm@wi.hs-wismar.de

WDP - Wismarer Diskussionspapiere / Wismar Discussion Papers

Heft 03/2014: Günther Ringle: Genossenschaftliche Solidarität auf dem Prüfstand

Heft 04/2014: Barbara Bojack: Alkoholmissbrauch, Alkoholabhängigkeit

Heft 01/2015: Dieter Gerdesmeier/ Hans-Eggert Reimers/ Barbara Roffia: Consumer and asset prices: some recent evidence

Heft 02/2015: Katrin Schmallowsky: Unternehmensbewertung mit Monte-Carlo-Simulationen

Heft 03/2015: Jan Bublitz/ Uwe Lämmel: Semantische Wiki und TopicMap-Visualisierung

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

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

ISBN 978-3-942100-69-4