FLOTTENPLANUNG VON FLUGZEUGEN

Dem Fachbereich Mathematik
an der Technischen Universität Darmstadt
zur Erlangung des Grades eines

Bachelor of Science

eingereichte

B a c h e l o r a r b e i t

vorgelegt von

Sebastian Erik Vock

aus Heppenheim

Erstkorrektor: Dr. habil. Marco Lübbecke
Zweitkorrektor: Prof. Dr. Stefan Ulbrich

September 2010

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig angefertigt habe. Ich habe alle Stellen, die ich aus Quellen wörtlich oder inhaltlich übernommen habe, als solche gekennzeichnet. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 15.09.2010

Sebastian Erik Vock

Inhaltsverzeichnis

1. Einleitung

Der Flugverkehr nimmt in unserem heutigen globalen Transportsystem eine wichtige Rolle ein. Durch die, in den letzten Jahren verstärkt auf den Markt drängenden, „Billig-Flieger“ entsteht für die etablierten Fluggesellschaften neuer Konkurrenzdruck. Um hier Wettbewerbsvorteile zu erlangen, sind eine noch exaktere Planung des Flugbetriebes und höhere Auslastungen der Flugzeuge nötig. Im Vordergrund steht dabei vor allem eine Senkung der Kosten.

Die weltweit größte Fluggesellschaft, gemessen an der Anzahl Flugzeuge, ist mit mehr als 2.000 Flügen täglich und einer Flotte von ca. 1.200 Flugzeugen Delta Air Lines Inc.. Die Planung des Flugbetriebes in dieser Größenordnung ist ein komplizierter und mehrstufiger Prozess. Das Flottenplanungsproblem / Fleet-Assignment-Problem (FAP) ist ein elementarer und entscheidender Teil dieses Planungsprozesses, da der Einsatz von Flugzeugen die höchsten Kosten für Fluggesellschaften verursacht. Eine intelligente Planung kann den Ertrag und die Wettbewerbsfähigkeit der Fluggesellschaften steigern.
Weitere Elemente des Planungsprozesses sind unter anderem die Preisplanung, die Einsatzplanung der Crew oder die Flugnetzwerkplanung.

Als Fleet-Assignment Problem bezeichnet man die Aufgabe, einem gegebenen Flugplan eine Menge von Flugzeugen verschiedener Typen kostenminimal zuzuordnen. Einfluss auf die Zuordnung haben die Eigenschaften der Flugzeugtypen und die zu bedienenden Flugstrecken. Die Menge aller Flugzeuge eines Flugzeugtyps bezeichnet man auch als Flotte.
Die mathematische Forschung beschäftigt sich seit der Mitte des letzten Jahrhunderts mit der Problemstellung der FAPs. Ziel ist es, ein universell einsetzbares Modell zu entwickeln, das in angemessener Zeit mit Hilfe von rechnergestützen Optimierungsverfahren lösbar ist. Erste Ergebnisse zur Entwicklung eines solchen Modells stammen von Ferguson und Dantzig aus dem Jahr 1956 [6].
Man kann die bisherigen Arbeiten zu diesem Thema und die daraus entstandenen Verfahren grob in zwei Bereiche unterteilen [8]. Die exakten Verfahren können theoretisch ein FAP exakt lösen. Manchmal wird dabei aber der Ablauf von Teilschritten durch den Einsatz von Heuristiken beschleunigt. Die heuristischen Verfahren können ein FAP im Allgemeinen nicht exakt lösen, sondern beschränken sich auf lokale Such-Ansätze.

Ziel dieser Arbeit ist es, Ansätze herauszuarbeiten, wie man den Lösungsvorgang eines FAPs beschleunigen kann. Als Grundlage dienen Ergebnisse aus früheren Ver-

suchen zum Lauzeitverhalten eines FAPs. Diese Ergebnisse wurden mit realen Flugplandaten für Nord-Amerika von Hane et al. im Jahr 1995 berechnet. Bereitgestellt wurden die Daten von Delta Airlines Inc..
Nach der Einführung in die Thematik und der Vorstellung eines Basismodells betrachten wir zunächst verschiedene Presolving-Ansätze. Ziel dieser Verfahren ist es das Problem zu vereinfachen, bevor man mit dem Lösungsprozess beginnt. Dabei betrachten wir vor allem Besonderheiten in der Modellstruktur, die zu Vereinfachungen führen.
Danach gehen wir auf den eigentlichen Lösungsprozess ein, der sich in zwei große Abschnitte unterteilt. Das Lösen des relaxierten Problems und das darauf aufbauende Branch & Bound-Verfahren. Bei beiden Abschnitten versuchen wir, durch Modifikation der Standard-Entscheidungsregeln, eine Beschleunigung zu erzielen. Ziel ist es ohne große Veränderungen an der Implementierung eines Standard-Solvers (CPLEX, OSL, gurobi, ...) auszukommen und dennoch die Laufzeit zu verkürzen.

2. Die Problemstellung

Ein FAP können wir als lineares Optimierungsproblem formulieren. Genauer gehört es zur Familie der Integer-Probleme. Gegeben sind eine Menge an nutzbaren Flugzeugtypen und ein Flugplan. Die Elemente des Flugplanes bezeichnet man als Flug-Legs. Definiert wird ein Flug-Leg durch Start- und Zielflughafen und einer dazugehörenden Abflugzeit.

Die Qualität der gesuchten Zuordnung von Flugzeugtypen zu Flügen wird durch eine große Anzahl von Faktoren beeinflusst, die wir in die Modellierung mit einbeziehen müssen:

- Die Eigenschaften des Flugzeugtyps (Sitzkapazität, Reichweite, Wartungskosten, Treibstoffverbrauch, ...).
- Die Anzahl der verfügbaren Typen.
- Die erwartete Nachfrage für das Flugleg. Diese setzt sich aus der Point-to-Point[1] Nachfrage, sowie der Transfer-Nachfrage[2] zusammen.
- Die Treibstoffkosten.
- Die Wartungsmöglichkeiten, verfügbare Infrastruktur an Start- und Zielort.
- Die Verfügbarkeit von Gateplätzen (teilweise unterschiedlich für verschiedene Flugzeugtypen).
- Die Lärmschutzvorschriften an einzelnen Flughäfen.
- ...

Gesucht ist nun eine kostenminimale Zuordnung der Flugzeugtypen zu den Flug-Legs, unter der Bedingung, dass die Nachfrage komplett befriedigt wird.

Grundlegend unterscheidet man in der bisherigen Forschung zwei Arten von FAPs, zyklische und azyklische.
In der mittelfristigen Planung werden hauptsächlich zyklische FAPs verwendet. Hier schließt das Ende einer Planungsperiode direkt an den Beginn der nächsten identischen Planungsperiode an. Es kann vorkommen, dass sich ein Flugleg über zwei Planungsperioden erstreckt. Wartungsarbeiten an den Flugzeugen müssen in festen

[1] Der Ankunftsort ist der Zielort.
[2] Der Ankunftsort ist nur ein Zwischenstop.

Intervallen - und damit in jeder Periode - in die Planung mit einbezogen werden.
Im Gegensatz dazu können wir den Planungshorizont eines azyklischen FAPs als unbeschränkt ansehen. Diese Variante wird hauptsächlich in der kurzfristigen Einsatzplanung verwendet. Generell gilt hier für ein Flug-Leg, dass die Ankunftszeit zeitlich immer nach der Abflugzeit liegt.
Seit 1994 ist bekannt, dass ein Fleet-Assignment-Problem ab 3 Flugzeugtypen ein NP-vollständiges Problem ist [9]. Ein solches NP-vollständiges Problem lässt sich theoretisch nicht innerhalb einer polynomiell beschränkten Rechenzeit lösen. Alle für diese Probleme bekannten Algorithmen erfordern exponentiellen Rechenaufwand.

Durch das enorme Wachstum im Bereich des Luftverkehrs wurden spezielle Netzwerkstrukturen entwickelt oder angepasst um das Transportvolumen effizient zu bewältigen. Eine Struktur, die vor allem für ein Many-to-Many Distributionsproblem geeignet ist, ist das Hub & Spoke-Netzwerk [19].
Diese Struktur wird bei fast allen Fluglinien weltweit zur Strukturierung der internationalen Flugnetzwerke eingesetzt. Seit 1987 finden wir diese Struktur auch für das Flugnetzwerk der Inlandsflüge in Nordamerika[3].
Bei diesem Verfahren werden die Flughäfen in zwei Kategorien eingeteilt. Hubs sind zentrale Umschlagplätze im Netzwerk, die mit den restlichen Knoten, genannt Spokes, über Direktflüge verbunden werden.
Interkontinentalflüge oder andere Langstreckenflüge haben ihre Ausgangs- oder Endpunkte vorrangig an Hubs. Hier müssen die Passagiere dann umsteigen um mit Anschlussflügen zu den Spokes zu gelangen. Flüge von Spoke zu Spoke findet man in dieser Anordnung sehr selten.
Der große Vorteil dieser Struktur besteht darin, dass die benötigte Verbindungszahl stark reduziert wird, ohne Einbußen bei der Anzahl der angebundenen Städte zu haben. Um n Knoten je paarweise miteinander zu verbinden benötigt man beim Hub & Spoke-Verfahren $n-1$ Verbindungen, beim Point-to-Point Verfahren sind es $\frac{(n\cdot(n-1))}{2} = \frac{n^2\cdot n}{2}$ Verbindungen. Nachteilig für das Hub & Spoke-Netzwerk ist, dass manche Ziele nur durch mehrmaliges Umsteigen zu erreichen sind.
Diese Struktur spielt eine große Rolle bei der Modellierung des Problemes. Sie bietet uns die Möglichkeit auf Grund von Abhängigkeiten Nebenbedingungen zu entfernen oder zusammenzufassen und so die Problemgröße zu reduzieren.

[3] Am 28. Oktober 1978 wurde der Airline Deregulation Act verabschiedet, der den kommerziellen Luftverkehr in den USA deregulierte.

3. Vorstellung des mathematischen Modells

Das Modell mit Hilfe dessen wir unsere Überlegungen durchführen wollen gehört zu einem zyklischen FAP mit einem täglichen Planungshorizont. Es wurde entwickelt um den täglichen nationalen Flugplan der USA von Delta Airlines beschreiben und lösen zu können [10]. Als Zielsetzung soll eine kostenminimale Zuordnung berechnet werden. Kostenminimalität kann erreicht werden, in dem weniger Flugzeuge oder vorrangig Flugzeuge, die geringere Kosten verursachen, eingesetzt werden.

Wie im vorigen Kapitel dargestellt, schließen beim zyklischen FAP zwei identische Planungsperioden direkt aneinander an. Betrachten wir die Bewegungen eines Flugzeugtyps an einem Flughafen im Netzwerk, so ist klar, dass um 0.01 Uhr der gleiche Zustand vorliegen muss wie am Periodenende um 23.59 Uhr.
Sei $A_{f,o}(t)$ die Anzahl Flugzeuge des Flugzeugtyps f an Flughafen o zum Zeitpunkt t. Ausserdem bezeichne $A^S_{f,o}(t^-, t^+)$ die Anzahl Starts von Flugzeugtyp f an Flughafen o innerhalb des Zeitraumes t^- bis t^+ und analog $A^L_{f,o}(t^-, t^+)$ die Anzahl Landungen. Dann folgt als Bedingung:

$$A_{i,o}(0.01) = A_{i,o}(23.59),\ \forall f, o \tag{3.1}$$

Woraus sofort folgt:

$$A^S_{i,o}(0.01, 23.59) = A^L_{i,o}(0.01, 23.59),\ \forall f, o \tag{3.2}$$

Zur graphischen Veranschaulichung kann man einen solchen Tagesablauf als Kreis modellieren. Die Anordnung der Ereignisse erfolgt nach dem zeitlichen Ablauf. Die Knoten im Graph repräsentieren Abflüge und Ankünfte, die Kanten die Standzeiten eines oder mehrerer Flugzeuge eines Typs an einem Flughafen. Einen solchen Kreis für einen Flugzeugtypen an einem Flughafen nennen wir **Timeline**.

Sollte es laut Flugplan vorkommen, dass ein Abflug und eine Ankunft desselben Flugzeugtyps an einem Flughafen gleichzeitig stattfinden, legen wir fest, dass der Ankunftsknoten immer vor dem Abflugknoten in der Timeline angeordnet ist.
Um ein fehlerfreies Modell zu erhalten muss zu jedem Abflugknoten, in der Timeline eines anderen Flughafens, für den gleichen Flugzeugtyp ein Ankunftsknoten existieren. Verbunden sind die beiden Knoten über ein Flug-Leg. Dieses Flug-Leg

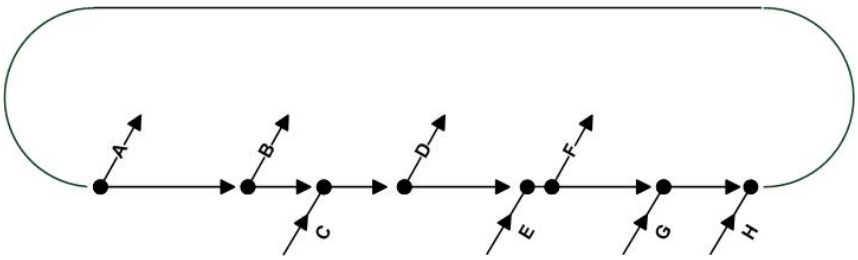

Abbildung 3.1.: Timeline (nach [10])

modellieren wir als Menge von Entscheidungsvariablen, die später angeben welcher Flugzeugtyp das Flug-Leg fliegt. Diese Netzwerkstruktur eines FAP nennt man **Time Space Network**.
Es gibt in der Literatur noch eine zweite Netzwerkstruktur, das **Connection Network**, auf das wir hier aber nicht näher eingehen wollen [1].

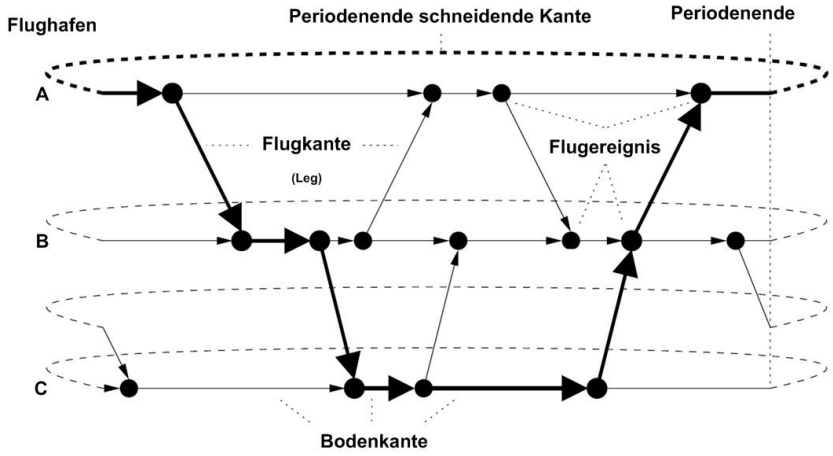

Abbildung 3.2.: Schematische Darstellung des Time-Space-Network (nach [8])

In unserem Modell (Time-Space-Network) gibt es keine Quellen und Senken, die Anzahl der verwendeten Flugzeuge ist konstant. Das vorliegende Modell beschreibt also zwangsläufig einen Fluss durch das Netzwerk (Zirkulation).

Wir führen nun die verwendeten Variablen, Parameter und Mengendefinitionen ein. Die vorgestellten Definitionen und Schreibweisen sind der Arbeit von Hane et al. entnommen [10].

Ein **Flugleg** $i \in L$ ist eindeutig durch einen Startflughafen $o \in C$ (Origin), einen Zielflughafen $d \in C$ (Destination) und einen Startzeitpunkt t definiert.
Unter **required-throughs** versteht man Paare von Legs, bei denen eine deutliche Ertragssteigerung erzielt werden kann, wenn man beide Flüge zu einem einzigen Flug mit Zwischenstopp zusammenfasst.

Der erste Knoten in einer Timeline ist definiert als $\{fot_1\}$, der letzte Knoten als $\{fot_n\}$. Die inzidente Kante zu $\{fot_n\}$ und $\{fot_1\}$ enthält die Uhrzeit 3a.m. EST. Unter der **ready time** versteht man die Zeit, zu der ein Flugzeug in einer Stadt nach der Landung wieder einsatzbereit ist. Größere Flugzeuge und Flugzeuge an größeren Flughäfen benötigen mehr Zeit bis zu ihrer ready time.

t	Zeitpunkt von Start oder Landung
t^-	Bezeichnet die Zeit vor Zeitpunkt t
t^+	Bezeichnet die Zeit nach Zeitpunkt t
C	Menge der Städte
F	Menge der verfügbaren Flugzeugtypen / Flotten
$S(f)$	Anzahl Flugzeuge in der Flotte f, $f \in F$
L	Menge der Flüge im Flugplan, auf deren Elemente entweder über die fortlaufene Nummerierung $\{i\}$ oder $\{o, d, t\}$, mit $o, d \in C$ und Startzeitpunkt t, zugegriffen werden kann.
$O(f)$	Menge der Kanten für einen Flugzeugtyp die 3a.m. EST enthalten, $f \in F$
H	Menge der required throughs, mit Elementen (i, j); $i, j \in L$
N	Menge der Knoten im Netzwerk, bezeichnet mit $\{fot\}$ mit $f \in F, o \in C, t$ Zeit von Start/Landung in o
c_{fi}	Kosten für einen Flug $i \in L, f \in F$
$X_{fodt} = X_{fi} \in \mathbb{R}$	Entscheidungsvariable, ob Flugzeugtyp f das Leg (o, d, t) fliegt oder nicht
$Y_{fott^+}, Y_{fot^-t} \in \mathbb{R}$	ground-arc Variablen, bestimmen für jeden Zeitpunkt, an jedem Flughafen und für jeden Flugzeugtyp, die Zahl der am Boden befindlichen Flugzeuge. $f \in F, o \in C$ und $[t, t^+]$ oder $[t^-, t]$

Zum Zeitpunkt 3a.m. EST ist die minimale Anzahl Flugzeuge in der Luft, was die Kardinalität der Menge $O(f)$ bedeutend verringert. Die Berechnung eines Schnitts durchs Netzwerk zu diesem Zeitpunkt, um die Anzahl der vorhandenen Flugzeuge zu bestimmen, wird erleichert.

Für die Entscheidungsvariablen X_{fodt}, auch geschrieben als X_{fi}, gilt

$$X_{fodt} = X_{fi} = \begin{cases} 1, & \text{falls Flotte } f \text{ den Flug von } o \text{ nach } d \text{ zur Zeit } t \text{ fliegt} \\ 0, & \text{sonst} \end{cases}$$

Optimieren möchte man nun die Gesamtsumme der Kosten über alle Flugzeugtypen und alle Flüge, bei Betrachtung eines täglichen Planungshorizontes. Als Gleichung dargestellt ergibt sich folgende Zielfunktion

$$\min \sum_{i \in L} \sum_{f \in F} c_{fi} X_{fi}$$

unter den Nebenbedingungen:

$$\sum_{f} X_{fi} = 1, \qquad \forall i \in L \tag{3.3}$$

$$\sum_{d} X_{fdot} + Y_{fot^-t} - \sum_{d} X_{fodt} - Y_{fott^+} = 0, \qquad \forall \{fot\} \in N \tag{3.4}$$

$$X_{fi} - X_{fj} = 0, \qquad \forall (i,j) \in H \text{ und } \forall f \in F \tag{3.5}$$

$$\sum_{i \in O(f)} X_{fi} + \sum_{o \in C} Y_{fot_n t_1} \leq S(f), \qquad \forall f \in F \tag{3.6}$$

$$Y_{fott^+} \geq 0, \qquad \forall \left\{fott^+\right\} \in N \tag{3.7}$$

$$X_{fi} \in \{0,1\}, \qquad \forall i \in L \text{ und } f \in F \tag{3.8}$$

Die Menge von Bedingungen, die aus Gleichung (3.3) entstehen, stellen sicher, dass jeder Flug $i \in L$ von genau einem Flugzeugtyp $f \in F$ geflogen wird. Diese Nebenbedingungen nennen wir **Cover-Bedingungen**. Gleichung (3.4) garantiert, dass nie mehr Flugzeuge von einem Flughafen $o \in C$ abfliegen oder gleichzeitig dort stationiert werden können, als vorher dort stationiert waren und angekommen sind.
Durch (3.6) wird gewährleistet, dass nicht mehr Flugzeuge vom Modell eingesetzt werden können, wie vorhanden sind. Dazu addieren wir zum Zeitpunkt 3a.m. EST alle Flugzeuge die sich in der Luft befinden, sowie alle Flugzeuge die momentan an Flughäfen stationiert sind.
Nebenbedingung (3.5) garantiert, dass die required throughs von einem Flugzeug gleichen Typs geflogen werden. Solange gewährleistet ist, dass die ready time des ersten Fluges mit der Abflugzeit des zweiten Fluges übereinstimmt, reicht diese Formulierung aus um die Bedingung zu garantieren. Ansonsten wären weitere Nebenbedingungen nötig, um den Einsatz weiterer Flugzeuge zu verhindern. Die Ungleichungen (3.4) - (3.6) fassen wir im Folgenden unter dem Begriff **Balance-Bedingungen** zusammen.

Der Kostenkoeffizient c_{fi} setzt sich aus mehreren Kosten zusammen. Er beinhaltet die anfallenden variablen Kosten für jeden Flug und anteilig die auf diesen Flug entfallenden Fixkosten der Airline. Zu den variablen Kosten zählen auch Kosten, die bei einer möglichen Überbuchung anfallen. Diese hängen von der Nachfrage, Überbuchungsrate, Preisstruktur und den noch verfügbaren Plätzen bei diesem Flug ab. Trotz ihrer komplexen Zusammensetzung sind diese Kosten noch einfach zu ermitteln. Die anteiligen Fixkosten zu ermitteln ist nicht einfach, da diese auch vom späteren Ergebnis des FAPs abhängen. Dazu zählen zum Beispiel die Kosten, die anfallen wenn ein Flugzeugtyp einen Flughafen zum ersten Mal anfliegt. Dabei entstehen Kosten zur Errichtung der benötigten Infrastruktur und Schulungskosten für das Personal.

Die Definition der ground arc Variablen Y_{fot} im Bereich der reellen Werte ist möglich, da durch die Ganzzahligkeit aller anderen Variablen, die die Anzahl der Flugzeuge

betreffen, sichergestellt ist, dass die Y_{fot} nur ganzzahlige Werte annehmen. Im Worst-Case muss in der Branch & Bound-Phase aber auch nach diesen Variablen verzweigt werden, was die Laufzeit erheblich verlängern würde. In unserem Fall reduzieren wir aber durch die Definition der Y_{fot} als reelle Variablen die Dimension des ganzzahligen Optimierungsproblemes, was in der Branch & Bound-Phase Vorteile hat.

Um die später vorgestellten Modifikationen an den Datensätzen und den Algorithmen auf ihre Wirksamkeit zu überprüfen benötigt man reale Testdatensätze. Wir nutzen zur Veranschaulichung und Begründung Ergebnisse, die Hane et al. berechnet haben. Diesen Ergebnissen liegen Realdaten zu Grunde, die damals von Delta Airlines Inc. bereitgestellt wurden. Sie sind aus 6 Flugplänen aus dem Zeitraum zwischen September 1991 und Juni 1992 entnommen. Die Größe eines kompletten Datensatzes umfasst ca. 2.500 Flüge, die 150 Städte verbinden. Es sind maximal 11 verschiedene Flugzeugtypen vorhanden. Tabelle 3.1 gibt einen Überblick über die Größe der Datensätze.
1A - 2E sind kleinere Datensätze, 1-6 bezeichnen die kompletten Datensätze. Mit den Zahlen wird der Flugplan kenntlich gemacht, aus dem die Daten entnommen wurden. Die Buchstaben definieren die in diesem Teilproblem zur Verfügung stehenden Flugzeugtypen. Problem 1E und 2E sind also aus verschiedenen Flugplänen, benutzen jedoch die gleichen Flugzeugtypen.

Name	Flotten	Flüge	Variablen	Zeilen	Spalten	Einträge $\neq 0$
1A	2	158	312	914	544	1.504
1B	2	1.201	2.126	3.211	3.598	10.038
1C	4	239	869	1.911	1.485	4.075
1D	4	1.605	5.861	13.103	16.299	40.216
1E	7	2.320	12.504	29.629	37.993	92.018
2A	2	161	301	1.299	909	2.205
2B	2	1.260	2.235	5.998	6.441	15.853
2C	4	261	943	3.221	2.843	6.833
2D	4	1.709	6.236	13.689	17.148	42.371
2E	7	2.376	12.851	30.018	38.638	93.769
1	11	2.559	22.679	47.994	65.254	159.064
2	11	2.637	23.512	48.982	66.942	163.472
3	11	2.627	23.118	48.674	66.429	161.751
4	11	2.588	22.737	48.159	65.202	159.282
5	11	2.590	22.745	48.204	65.213	159.357
6	11	2.589	22.746	48.109	65.164	163.472

Tabelle 3.1.: Ursprüngliche Problemgrößen

Solche Teilprobleme sind bei der Entwicklung eines Modells und Algorithmus zwingend notwendig um schnelle, aufschlussreiche Ergebnisse zum Laufzeitverhalten zu

bekommen. Voraussetzung ist aber, dass die Teilprobleme auf Teildatensätzen beruhen, die in Struktur und Beschaffenheit dem ursprünglichen Problem entsprechen. Sind diese Bedingungen erfüllt nennt man das Teilproblem **ausbalanciert**. Für die hier betrachteten Datensätze bedeutet dies, dass dieselbe Anzahl Starts und Landungen an jedem Flughafen vorhanden sein müssen und dass nicht mehr Flugzeuge benötigt werden dürfen wie vorhanden sind.

Normalerweise ist es sehr aufwendig aus einem kompletten Datensatz einen ausbalanciertes Teildatensatz zu erstellen. Eine bekannte, zulässige Lösung des Problemes, bedeutet eine enorme Zeitersparnis. Wir können einzelne Flugzeugtypen und die dazugehörigen Flughäfen und Flüge isolieren und erhalten so ein nach Voraussetzung ausbalancierten Teildatensatz. Gleichzeitig stellen wir damit sicher, dass jeder vorhandene Flug auch von einem vorhandenen Flugzeugtyp geflogen werden kann.

4. Presolving

Presolving bedeutet mit verschiedenen Methoden die Problemgröße zu verringern bevor man versucht den Lösungsalgorithmus darauf anzuwenden. Dadurch verkleinert man den Raum der zulässigen Lösungen für das Problem, was direkten Einfluss auf die Laufzeit zur Berechnung einer Lösung hat.
Möglichkeiten beim Presolving sind, bestimmte Ungleichungen, die immer oder nie erfüllt sind oder bestimmte Variablenbelegungen, die generell zu Unzulässigkeiten führen, zu entfernen. Mit Hilfe von algebraischen Methoden können wir diese Situationen erkennen und beheben. Über diese algebraischen Vereinfachungen hinaus wollen wir uns mit der Struktur des Modelles und den Datensätzen beschäftigen. Hier besteht weiteres Potential zur Reduktion. Bei einem Großteil von Modellen hilft eine gute Kenntnis der dem Modell zu Grunde liegenden Strukturen und Eigenschaften, um später eine kompakte mathematische Formulierung zu entwickeln.

4.1. Algebraisch

Die Möglichkeit mit Hilfe algebraischer Methoden das Problem zu vereinfachen ist heutzutage in jedem bekannten Solver (gurobi, CPLEX, ...) integriert. Deswegen wollen wir hier nur kurz ein Beispiel geben, wie eine solche Vereinfachung aussehen kann. Unser Ziel ist es, dass das Problem später möglichst wenige Variablen und linear unabhängige Nebenbedingungen enthält sowie eine relativ dicht besetzte Matrix vorliegt [2].

Beispiel:

$$\left.\begin{aligned} x_1 &\geq 8 \\ x_2 &\geq 7 \\ x_1 + x_2 &\leq 15 \end{aligned}\right\} \Rightarrow x_1 = 8 \text{ und } x_2 = 7$$

Die dritte NB können wir sofort entfernen und weitere Nebenbedingungen der Art $x_1 + x_3 \geq 25$ können wir zu $x_3 \geq 17$ reduzieren.

4.2. Zusammenfassen mehrerer Knoten

Wir wollen jetzt die topologische Struktur des Modells näher betrachten. Dabei interessieren wir uns für die graphentheoretische Seite der Modellierung und die daraus abgeleiteten Nebenbedingungen. Die Darstellung eines Tagesablaufes für einen Flugzeugtyp an einem Flughafen als kreisförmige Timeline soll gewährleisten, dass alle Flüge korrekt durchgeführt werden. Dazu ist es notwendig, dass ein Flugzeug zuerst landet, bevor es wieder startet um einen darauf folgenden Flug durchzuführen. In der Timeline wird diese Abfolge durch zwei Knoten dargestellt, die durch eine Kante verbunden sind. Für die Zuordnung ist nicht relevant, wie lange diese inzidente Kante ist, solange sie mindestens die ready-time des Flugzeugtyps umfasst. Es führt zu keinem Informationsverlust im Modell, wenn wir den Ankunftsknoten und den zugehörigen Abflugknoten zu einem einzigen Knoten zusammenfassen und die inzidente Kante aus dem Modell entfernen. Für die Problemgröße bedeutet dies jedoch, dass im besten Fall die Anzahl Knoten halbiert werden kann. Eine gefundene optimale Lösung des so verkleinerten Problems lässt sich, durch Aufspalten der Knoten, später wieder in den ursprünglichen Zustand zurückversetzen.

4.3. Einführung der Inselstruktur

Im Flugverkehr wird, wie schon beschrieben, vorwiegend die Hub & Spoke-Struktur zur Erstellung eines Netzplanes verwendet. Darauf aufbauend begannen die Airlines den täglichen Flugplan für bestimmte Flughäfen so umzustrukturieren, dass den Passagieren umfangreichere Umsteigemöglichkeiten geboten werden können. Der komplette Tag wurde in Komplexe von Ereignissen eingeteilt. Diesen Vorgang nennen wir **Clusterung**. Man bildet Perioden in denen nur Ankünfte stattfinden, auf die Perioden von Abflügen folgen, getrennt durch Zeiten mit wenig bis gar keiner Flugaktivität. Durch diese Bündelung werden die umfangreichere Umsteigemöglichkeiten ermöglicht, ohne dass große Wartezeiten in Kauf genommen werden müssen. Abb. 4.1 stellt eine typische Timeline für einen Hub dar, an welchem eine solche Einteilung in Komplexe durchgeführt wurde.

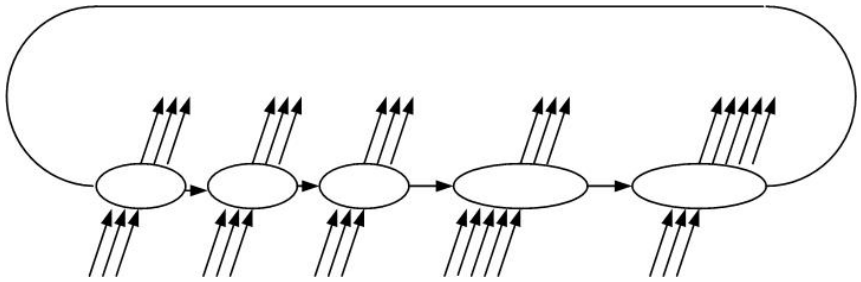

Abbildung 4.1.: Timeline für einen Hub nach der Clusterung (nach [10])

Eine typische Timeline für einen Spoke mit Einteilungen in Komplexen ist in Abb. 4.3 zu sehen. Morgens starten vorwiegend Verbindungsflüge zu den angebundenen Hubs. Im Laufe des Tages gibt es vereinzelte Starts und Landungen und Abends gibt es nur noch ankommende Flüge. Dadurch wird die Anzahl Flugzeuge, die tagsüber an einem Spoke stationiert ist, reduziert. Gleichzeitig wird damit versucht auch die Anzahl über Nacht stationierter Flugzeuge zu minimieren. An manchen Spokes ist es sogar möglich keinerlei Flugzeuge über Nacht dort zu stationieren, wenn die letzten Aktivitäten noch einmal Verbindungsflüge zu einem Hub sind. Das bringt für die Airline eine Kostenersparnis, die sich vor allem durch die an diesen Flughäfen nicht benötigte Wartungs-Infrastruktur ergibt. Auf Grund des Flugplanes ist es jedoch nicht oft möglich an einem Spoke keinerlei Flugzeuge über Nacht zu stationieren.

Die am Boden befindlichen Flugzeuge werden durch die Kanten in der Timeline repräsentiert. Jeder dieser Kanten besitzt eine Gewichtung[1], welche die Anzahl am Boden befindlicher Flugzeuge angibt. Betrachten wir die Tagesabläufe an allen Flughäfen, nach der Clusterung, können wir an manchen Orten tagsüber Zeitspannen identifizieren, an denen keine Flugzeuge am Boden sind. Anstatt den zugehörigen Kanten den Wert 0 zuzuweisen, entfernen wir diese komplett aus Timeline und Modell. Dadurch reduzieren wir die Variablenanzahl. Abbildung 4.2 zeigt eine Timeline für einen Hub nach dieser Modifikation. Es haben sich isolierte „Inseln“ gebildet, die keine adjazenten Kanten mehr besitzen. Formal definieren wir eine **Insel** als ein Zeitintervall, währenddessen ein oder mehrere Flugzeuge am Boden sind und davor und danach keine Flugzeuge am Boden sind. Daraus folgt sofort, dass die während einer Insel durchgeführte Anzahl Landungen und Starts übereinstimmen muss (vgl. Gleichung 3.1 und 3.2).
Eine Insel kann nicht nur durch die gerade beschriebene Strukturierung des Flugplanes entstehen, sondern auch durch die im Abschnitt davor dargestellte Zusammenfassung eines oder mehrerer Abflug- und Ankunftsknoten.

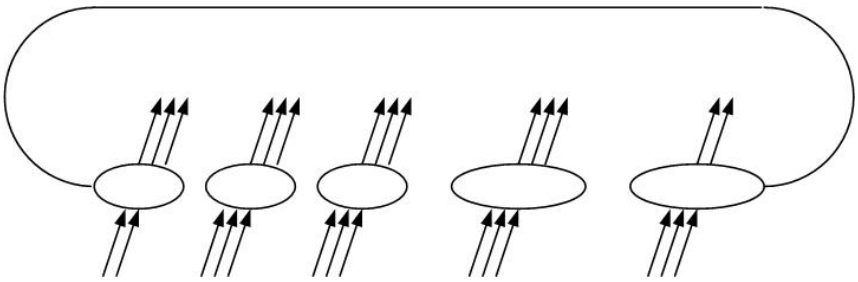

Abbildung 4.2.: Timeline für einen Hub mit Inseln (nach [10])

Durch die Einführung der Clusterung können wir weitere Situationen identifizieren, die es uns ermöglichen Variablen aus dem Modell zu entfernen. Besonders interessant für weiteren Betrachtungen sind Inseln, die aus einer Landung und einem Start bestehen. Die beiden zugehörigen Flug-Legs müssen vom gleichen Flugzeug

[1] Die ground-arc Variablen.

durchgeführt werden. In diesem Fall können wir beide Entscheidungsvariablen zu einer einzigen zusammenfassen. Bei der Entscheidung welcher Flugzeugtyp eingesetzt werden soll müssen wir weiterhin beide Flug-Legs und deren Eigenschaften beachten.
Im Spezialfall, dass am Zielflughafen des zweiten Flug-Legs die gerade beschriebene Situation erneut eintritt, dass dieses Flugzeug den nächsten Flug übernehmen muss, können wir eine weitere Variable aus dem Modell entfernen.
Bei der Analyse der realen Flugplandaten können wir eine solche Kette von zusammenhängenden Flügen oft vorfinden. So sind wir mit Hilfe dieser Methode in der Lage etliche Entscheidungsvariablen zusammenzufassen. Formal bilden wir eine Menge E aus mehreren einzelnen Entscheidungsvariablen, die im Modell nur noch durch eine einzige Variable X_{fE} repräsentiert werden. Bei der Zusammenfassung mehrerer Variablen werden $|E| - 1$ Variablen aus dem Modell entfernt.

Eine weitere Möglichkeit besteht darin, Flugzeugtypen, die den Flugplan wegen ihrer längeren ready time nicht einhalten können, die Durchführung von konsekutiven Flügen zu verbieten. Abbildung 4.4 zeigt eine Timeline in der es, auf Grund einer solchen Situation, zum Einsatz eines zusätzlichen Flugzeuges kommt. Um zu gewährleisten, dass nur die minimale Anzahl Flugzeuge eingesetzt wird, müssen wir diese Situation verhindern. Das können wir erreichen, indem wir die Kombinationen von Flugzeugtypen und zusammenhängenden Flügen isolieren die fehlerhaft sind und diesen Typen den Einsatz verbieten ($X_{fi} = 0$).

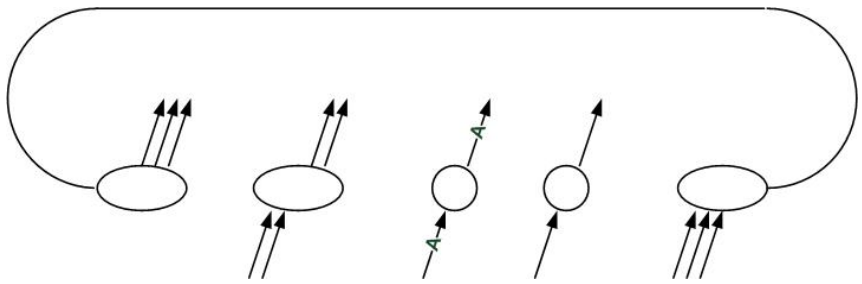

Abbildung 4.3.: Timeline für einen Spoke nach der Clusterung (nach [10])

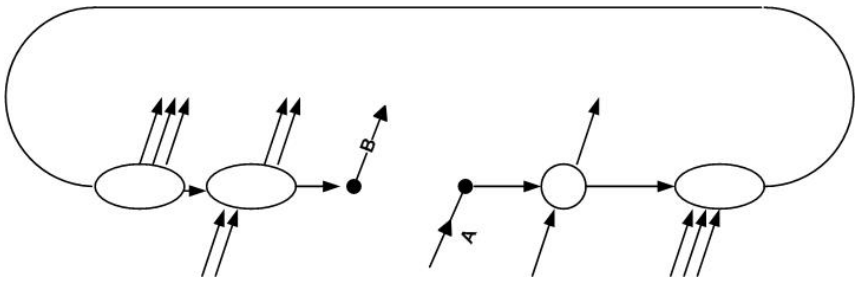

Abbildung 4.4.: Timeline für einen Spoke mit fehlerhafter Verbindung (nach [10])

Auf Grund des Flugplanes ist es in der Praxis fast nicht möglich die Inselstruktur an allen Flughäfen zu erzwingen. Das bedeutet für unsere spätere Lösung, dass wir

nicht in der Lage sind die untere Schranke für die Anzahl der eingesetzten Flugzeuge zu erreichen. Die entscheidende Frage ist, an welchen Flughäfen wir auf diese Bedingung verzichten, ohne dass wir das gesamte Problem unlösbar machen.
Wir wollen hier nur kurz ein von Hane et al. entwickeltes Vorgehen vorstellen, mit dem man diese Flughäfen identifizieren kann. Sie entdeckten einen Zusammenhang zwischen der Summe der Dauer der ground-arcs an einem Flughafen und dem Einsatz der Inselstruktur. Je größer diese Summe, desto eher sollte man an diesem Flughafen keine Insel-Struktur einsetzen. An Hubs sollte man sowieso darauf verzichten, da man hier nie mit der minimalen Anzahl Flugzeuge auskommt, hier kommt es fast nie vor das sich kein Flugzeug mehr auf dem Boden befindet [10].

Zum Abschluss wollen wir die Ergebnisse darstellen, die alle vorgestellten Presolving-Maßnahmen auf die Problemgröße haben:

Problem-name	Flüge	Fehlerhafte Verbindungen	ground-arcs entfernt	0/1-Variablen entfernt
1	2.559	1.022	8.466	7.162
2	2.637	961	8.524	7.355
3	2.627	888	8.464	7.016
4	2.588	868	8.434	7.246
5	2.590	881	8.451	7.339

Tabelle 4.1.: Anzahl entfernter ground-arc Variablen auf Grund der Inselstruktur, fehlerhafter Verbindungen und Ketten von Flug-Legs

Problem-name	Vereinfachungen	Anzahl Flotten	Anzahl Zeilen	Anzahl Spalten	Iterationen Simplex	Laufzeit in Sekunden
1A	Ohne	2	914	544	786	7,2
	Mit	2	212	357	407	1,6
1B	Ohne	2	3.211	3.598	4.519	165,5
	Mit	2	987	1.860	1.966	32,6
1C	Ohne	4	1.911	1.485	2.359	59,1
	Mit	4	734	1.319	1.807	24,4
1D	Ohne	4	13.103	16.299	35.467	8.250,2
	Mit	4	5.297	8.969	16.742	1.879,6

Tabelle 4.2.: Vergleich der Problemgrößen und Kennzahlen des Lösungsvorgangs ohne und mit Vereinfachungen

5. Lösen des linearen Problems

Nachdem wir das Presolving abgeschlossen haben, wollen wir uns nun mit dem Lösungsvorgang beschäftigen.
Bekannte Verfahren um ein ganzzahliges Optimierungsproblem zu lösen sind Schnittebenenverfahren, auf die wir hier nicht näher eingehen möchten, und Branch & Bound. Beim Branch & Bound-Verfahren ist es notwendig eine Relaxation[1] des ursprünglichen Problems zu lösen, bevor man mit dem Branching beginnen kann. Als bekanntester Algorithmus für die Lösung eines linearen Optimierungsproblems ist das Simplex-Verfahren zu nennen, das aber nicht zur Klasse der polynomialen Algorithmen gehört (Bsp. Klee-Minty-Würfel) [15]. Alternativ dazu kann man Innere-Punkte-Verfahren verwenden, die auf der Ellipsoid-Methode beruhen und auch im schlechtesten Fall maximal polynomiale Laufzeit gewährleisten.
Jedoch besitzt das Simplex-Verfahren den Vorteil bei einer geringfügigen Änderungen am Problem effizient eine neue optimale Lösung berechnen kann. Wurde an einem Problem eine Bedingung/Variable hinzugefügt oder geändert und wurde das ursprüngliche Problem schon erfolgreich gelöst, kann das neue Problem mit dem Simplex-Verfahren sehr schnell reoptimiert werden. Diese Eigenschaft ist insbesondere für das Branch & Bound-Verfahren von Bedeutung, da hier in jedem Schritt ein nur geringfügig abgewandeltes lineares Problem gelöst werden muss [15].
Möglichkeiten den Lösungsvorgang zu beschleunigen bestehen in der Verwendung eines alternativen Lösungsalgorithmus oder in der Modifikation einzelner Teilschritte der Algorithmen.

5.1. Simplexalgorithmus

Mit Hilfe von Tabelle 4.2 bekommen wir erste Richtwerte für die Laufzeit des Problems und mögliche Zusammenhänge zwischen Problemgröße und Laufzeit. Die Probleme wurden mit dem primalen Simplexverfahren gelöst und die ursprünglichen Eingabegrößen gegen die Iterationenzahl und Laufzeit aufgetragen. An Hand dieser Zahlen können wir einen nicht linearen Zusammenhang zwischen der Anzahl der Flugzeugtypen und der Iterationszahl erkennen. Das deutet darauf hin, dass wir für die Datensätze mit 11 Flugzeugtypen mit einer immensen Laufzeit rechnen müssen.

[1] Relaxation bedeutet bei einem ganzzahligen Optimierungsproblem auf die Forderung der Ganzzahligkeit zu verzichten.

5.1.1. Startbasis

Um ein lineares Problem mit dem Simplex-Algorithmus lösen zu können, benötigt man eine zulässige Startbasis - den Startpunkt des Simplexverfahren. Um eine solche Startbasis zu bestimmen gibt es mehrere Möglichkeiten. Die einfachste besteht darin, eine schon bekannte zulässige Lösung des Problems zu verwenden.
Um eine solche Startbasis zu berechnen, gibt es verschiedene Verfahren. Das bekannteste ist die **Phase I** des Simplexalgorithmus. Mit diesem modifizierten Simplex-Verfahren kann man zuverlässig und relativ einfach eine zulässige Basis berechnen. Dieses Verfahren gehört aber ebenfalls nicht zur Klasse der polynomialen Algorithmen. Alternativ dazu, bietet die von Maros & Mitra entwickelte Crash-Technik die Möglichkeit zur Berechnung einer Startbasis [14]. Hierbei versucht man möglichst früh mehreren Variablen einen Lösungswert zuzuordnen, anstatt sie sukzessive in einem Pricing-Verfahren als Basisvariable zu bestimmen und dabei den Wert zuzuordnen. Nach Ergebnissen von Gould & Reid kann die Lösung der LP-Probleme dadurch beschleunigt werden [13].

Problem-name	Phase I und Volles Pricing	Crash-Basis und Devex-Pricing
1A	1,6	1,4
1B	32,6	29,2
1C	24,4	12,6
1D	1.879,6	762,8

Tabelle 5.1.: Vergleich der Laufzeiten in Sekunden mit unterschiedlichen Pricing-Varianten und Methoden zur Bestimmung der Startbasis

In Tabelle 5.1 sind die Laufzeiten für einige Testdatensätze dargestellt. Bei der ersten Spalte wurde der primale Simplex-Algorithmus verwendet und die Startbasis mit Hilfe der Phase I des Simplex bestimmt. Bei den Werten in der zweiten Spalte wurde die Startbasis mit Hilfe der Crash-Technik bestimmt und es kam während des primalen Simplex eine andere Pricing-Variante zum Einsatz (Devex-Pricing). Wir können erkennen, dass die Crash-Technik und die Verwendung einer alternativen Pricing-Variante den Lösungsvorgang noch einmal beschleunigen. Mit den Möglichkeiten durch den Einsatz alternativer Pricing-Varianten wollen wir uns im folgenden Abschnitt genauer beschäftigen.

5.1.2. Pricing

Die einfachste Modifikation, die man am Simplex-Verfahren vornehmen kann, ist eine andere Pricing-Variante einzubinden. Pricing ist der Schritt in dem bestimmt wird, welche Nichtbasisvariable im nächsten Schritt in die Basis eintreten soll. Ziel

ist es eine Variable zu wählen, die eine möglichst große Verbesserung des Zielfunktionswertes ermöglicht.
In den meisten Lehrbüchern wird bei der Vorstellung des Simplex-Verfahren als Pricing Dantzigs-Regel in Verbindung mit vollem Pricing verwendet. Hier werden für alle Nichtbasisvariablen die reduzierten Kosten berechnet (volles Pricing) und man wählt denjenigen Indize, mit den kleinsten reduzierten Kosten (Dantzigs Regel) als neue Basisvariable[2]. In jedem Schritt wird die größtmögliche Verbesserung der Variable x_j auf sich selbst bestimmt und diejenige mit dem größten Potential gewählt. Der Nachteil dieses Verfahrens besteht darin, dass es sehr aufwendig sein kann, wenn die Anzahl der Variablen groß ist [15]. Da unser Problem mit ca. 22.000 Variablen heutzutage keine Herausforderung mehr für die vorhandene Rechnertechnik darstellt, sind wir auf einen Wechsel nicht zwingend angewiesen. Jedoch bieten alternative Pricing-Varianten auch schon bei diesen Größen erhebliches Verbesserungspotential.

Eine alternative Methode ist Steepest-Edge Pricing. Hier wählt man $x_j \in \mathbb{N}$, so dass die Richtung Δx den Ausdruck $\frac{c^T \Delta x}{\|\Delta x\|}$ minimiert.[3] Der Vorteil dieser Methode liegt darin, dass weniger Iterationen notwendig sind. Nachteilig ist, dass in jedem Schritt ein zusätzliches Gleichungssystem zur Lösung der Δx gelöst werden muss [12]. Im schlechtesten Fall hebt das den Geschwindigkeitszuwachs wieder auf.
Eine abgewandelte Variante des Steepest-Edge (SE) Verfahrens ist Devex-Pricing. Bei dem von Paula Harris entwickelten Verfahren werden die Spalten der Matrix und die reduzierten Kosten vor der Auswahl auf eine einheitliche Norm skaliert. Dieses Verfahren ist heute in den meisten Solvern integriert und gilt als sehr robuste und effiziente Variante [11].
Hane et al. führten mehrere Versuche durch um die verschiedenen Pricing Varianten miteinander zu vergleichen. In Tabelle 5.2 sind die Iterationen und Laufzeiten gegenübergestellt. Wir können erkennen, dass der duale Simplex mit Steepest-Edge am schnellsten ist. Bei den Zahlen des primalen Simplex kann man erkennen, dass hier beide Pricing-Varianten etwa gleichschnell sind.

Problem-	Primal + Devex		Primal + SE		Dual + SE	
name	Iterationen	Laufzeit	Iterationen	Laufzeit	Iterationen	Laufzeit
1E	13.519	571,2	12.573	557,1	9.623	536,8
2E	13.833	646,8	13.858	660,8	8.385	469,9
1	33.101	3.257.5	32.097	3.194,5	15.408	1.431,8
2	29.031	2.785.3	32.934	3.256,3	14.302	1.554,8

Tabelle 5.2.: Werte zur Laufzeit in Sek. mit unterschiedlichen Pricing-Varianten

[2]Reduzierte Kosten geben an, wie stark der Zielfunktionswert abfällt, wenn diese Variable mit Wert 1 in die Basis aufgenommen wird.
[3]Zur Erklärung der Variablen siehe Anhang A.2

5.2. Innere-Punkte-Verfahren

Interessant für unsere Fragestellung ist eine Klasse von Algorithmen zur Lösung von LP-Problemen, die auf der Ellipsoidmethode von Khachiyan aufbauen. Die Innere-Punkte-Verfahren (I-P) sind nach Schade, auf Grund ihrer Leistungsfähigkeit, auch für sehr große Probleme als momentaner Maßstab innerhalb der bekannten Verfahren anzusehen [18]. Diese Einschätzung beruht auf der maximal polynomialen Laufzeit und auf der relativ schnellen Konvergenz, besonders bei dünn besetzten oder hochgradig entarteten Problemen [13]. Ausserdem sind sie noch relativ jung und bieten voraussichtlich noch großes Verbesserungspotential. Wir betrachten hier ein Primal-Duales I-P Verfahren. Primal-Dual bedeutet, dass in die Iterationen nicht nur die primalen Variablen einfließen, sondern gleichzeitig auch Variablen des dualen Problemes verwendet werden.
Im Gegensatz zum Simplexverfahren laufen I-P Verfahren nicht die Ecken des Polyeders ab. Sie bewegen sich im Inneren des Polyeders und versuchen sich iterativ einem Optimum zu nähern. Die Punkte, die sie dabei absuchen, können nie auf dem Rand liegen, sondern sind immer innere Punkte.

5.2.1. Primal-Duales-Prädiktor-Korrektor Verfahren

Es gibt unterschiedliche Varianten des I-P Algorithmus, die ihre Vorteile in verschiedenen Situationen besitzen. Es gibt die Kurzschrittverfahren, Langschrittverfahren und die Prädiktor-Korrektor Verfahren. In der Praxis ist das Prädiktor-Korrektor Verfahren den beiden anderen Varianten in Bezug auf die Laufzeit, die zur Lösung eines Problems benötigt wird, überlegen. Theoretisch ist für dieses Variante jedoch keine Komplexitätsoberschranke bekannt [13].
In jeder Iteration eines I-P Algorithmus muss die Cholesky-Zerlegung der Koeffizientenmatrix berechnet werden. Dieser Schritt ist der aufwändigste Teil einer Iteration. Das Prädiktor-Korrektor Verfahren, das von Sanjay Mehrotra entwickelt wurde, benutzt in mehreren Iterationen die gleiche Cholesky-Zerlegung um zwei verschiedene Richtungen zu berechnen: Einen Prädiktor und einen Korrektor. Der Prädiktor stellt sicher, dass in der nächsten Iteration eine Verbesserung stattfindet. Der Korrektor ist abhängig von der möglichen Schrittweite der Prädiktor-Richtung und korrigiert die neue Lösung in Richtung Zentrum des Polyeders. Obwohl pro Iteration zwei Schritte ausgeführt werden, konvergiert diese I-P Variante vor allem in Nähe der Optimallösung, sehr schnell. Dadurch wird die Iterationszahl drastisch verringert [17]. Da das Primal-Duale-Prädiktor-Korrektor (PD-PC) Verfahren sehr effizient zu implementieren ist und auch in kommerziellen wie frei verfügbaren Solvern schon integriert, setzte Hane et al. es zur Lösung des LPs ein.
Nachteilig an den I-P Verfahren ist, dass das Resultat zwar eine optimale Lösung des Problemes ist, man aber noch die zugehörige Basis finden muss. Diese benötigt man zwingend um mit Branch & Bound beginnen zu können. Das schränkt die Eignung

für eine Verwendung in unserem Fall etwas ein. Die Problemstellung eine Basis aus einer optimalen Lösung zu finden nennt man Crossover-Problem.

Hat man ein LP mit dem Innere-Punkte-Verfahren gelöst, so wissen wir, dass die gefundene Optimallösung auf einer Seitenfläche des Polyeders liegt. Klar ist, dass die zugehörige Basislösung auch auf dieser Seitenfläche liegen muss. Bei einem so dünn besetzten Problem[4] wie dem FAP ist es jedoch sehr problematisch die Nichtbasisvariablen mit $X_{fi} = 0$ und die Basisvariablen mit $X_{fi} = 1$ zu identifizieren.

Problem-name		I-P Verfahren		Simplex		Total
		Iterationen	Laufzeit	Iterationen	Laufzeit	Laufzeit
1A	Primal	-	-	260	1,4	1,4
	PD-PC	11	1,3	234	< 1	<2,3
1B	Primal	-	-	1.409	29,2	29,2
	PD-PC	14	8,5	1.195	8,8	17,3
1C	Primal	-	-	753	12,6	12,6
	PD-PC	15	6,1	1.012	10,3	16,4
1D	Primal	-	-	8.031	762,8	762,8
	PD-PC	28	201,2	6.695	380,2	581,4

Tabelle 5.3.: Vergleich der Iterationszahl und Laufzeit in Sekunden zwischen primalem Simplex und primal-dualem-Prädiktor-Korrektor Verfahren

In der vorhergehenden Tabelle erkennt man, dass das Finden einer optimalen Basis aus einer I-P-Lösung etwa gleich viel Zeit beansprucht, wie das eigentliche Finden der Lösung. Deswegen wollen wir im nächsten Abschnitt versuchen diesen Vorgang zu optimieren.

5.2.2. Crossover-Problem

Um nach der Lösung eines LPs mit dem Inneren-Punkte-Verfahren eine zugehörige Basislösung zu erhalten, muss ein Crossover-Algorithmus angewandt werden. Diese Algorithmen berechnen aus einem inneren Punkt, der dem Abbruchkriterium des IP-Verfahren genügt, eine Basislösung, deren Zielfunktionswert mindestens genauso gut wie der des Inneren Punktes ist [13]. Eine Möglichkeit ist die Verwendung von Meggidos Algorithmus zum Finden einer optimalen Basis bezüglich einer I-P-Lösung. Hierbei ist die Iterationszahl durch die Anzahl Spalten und Zeilen der Eingabematrix beschränkt und der Aufwand einer Iteration entspricht ungefähr dem Aufwand eines Simplexschrittes [16].
Hane et al. konnten diesen Algorithmus auf Grund der fehlenden Integration in IBM OSL nicht verwenden. Der integrierte Algorithmus verwendete das Simplex-Verfahren um eine zugehörige Basislösung zu bestimmen [10].

[4]Trotz Aggregation sind weniger als 20% der Matrixeinträge ungleich Null

5.3. Störung der Kostenkoeffizienten

Dag Wedelin stellte in seiner Arbeit zum Crew Scheduling Problem fest, dass durch eine Störung der Kostenvektoren die Konvergenz des verwendeten Algorithmus beschleunigt wurde [21].
Beim FAP kann man zusätzlich zu den Kostenkoeffizienten c_{fi} auch die Kosten der Bodenstandzeiten stören. Die Störung kann vom Löser ohne weiteres Eingreifen selbstständig durchgeführt werden. Dabei werden in jedem Iterationsschritt die Kostenvektoren durch einen kleinen aufaddierten Betrag „gestört“. Den optimalen Zielfunktionswert ändern wir durch solche Störungen nicht, da die Werte der Kosteneinträge in einem Bereich zwischen 1.000 und 50.000 liegen [10].

Durch die Störung wird die Dimension der optimalen Seitenfläche verkleinert [10]. Dadurch wurde die Laufzeit im Durchschnitt um 19% bei Anwendung des dualen Simplexalgorithmus beschleunigt. Führt man vor dem I-P Verfahren eine solche Störung durch, vergrößert man die Anzahl Iterationen, was einen möglichen Geschwindigkeitsvorteil pro Iteration wieder ausgleicht. Jedoch zeigt sich im Zusammenspiel mit der Fixierung von Variablen auf ganzzahlige Werte eine unerwartete Verbesserung der Laufzeit. Auf die Einzelheiten der Variablenfixierung wird im nächsten Abschnitt eingegangen.

5.4. Variablenfixierung

In der Branch & Bound-Phase wird die relaxierte Lösung des Problems nach den nicht-ganzzahligen Variablen verzweigt. Die Tiefe des Suchbaumes hängt von der Anzahl der nicht-ganzzahligen Variablen ab und den Möglichkeiten, wie viele Werte sie annehmen können. Ist die relaxierte Lösung zufällig schon komplett ganzzahlig, ist kein Branch & Bound mehr notwendig. Tritt die Situation auf, dass der relaxierte Zielfunktionswert höchstens 1% vom zugehörigen diskreten Optimalwert entfernt ist, kann es vorteilhaft sein Variablen aus der kontinuierlichen Lösung auf ganzzahlige Werte zu fixieren bevor man mit dem Verzweigen beginnt. Für die Auswahl der zu fixierenden Variablen gibt es mehrere Möglichkeiten.

Die naheliegendste ist, dass man alle Variablen $X_{fi} \geq 0,99$ auf 1 fixiert. Dieses Vorgehen ist unabhängig von der zu Grunde liegenden Netzwerkstruktur und kann immer angewendet werden. Die zweite Möglichkeit begründet auf speziellen, der Modellstruktur des FAP zu Grunde liegenden, Eigenschaften. Dazu sortiert man die Flugzeugtypen vor der Lösung des relaxierten Problems aufsteigend nach der Sitzkapazität. Ergibt sich in der relaxierten Lösung $X_{3i} = X_{5i} = 0,5$, so bedeutet das, dass nur die Typen 3,4 und 5 den Flug i fliegen können. Für alle restlichen Typen kann man die zu diesem Flug gehörenden Entscheidungsvariablen auf 0 fixieren. Diese Variante bietet im Gegensatz zur allgemeinen Fixierung bedeutend mehr Verringerungspotential an nicht ganzzahligen Variablen.

Die dritte Möglichkeit beruht auf einer allgemeinen Eigenschaft von linearen Problemen. Man berechnet die Differenz zwischen kontinuierlicher und diskreter Optimallösung (GAP) und die reduzierten Kosten für alle Nichtbasisvariablen. Für alle Nichtbasisvariablen, deren reduzierte Kosten größer sind als die GAP gilt, dass diese nicht in die Basislösung aufgenommen werden können. Diese Methode ist bei allen linearen Problemen anwendbar, jedoch kann die Situation auftreten, dass alle reduzierten Kosten kleiner sind als die GAP. In diesem Fall würde mit dieser Vorgehensweise keine Variable fixiert werden. Dieser Fall tritt beim FAP ein [10]. Somit besitzt diese Variante für unser Problem lediglich theoretisches Verbesserungspotential.

Durch die Fixierung von Variablen haben wir die Struktur des LPs verändert, die Matrix wurde weiter ausgedünnt. Um dadurch keine Nachteile beim folgenden Lösungsvorgang zu bekommen, ist es sinnvoll das Problem vor dem Start der Branch & Bound-Phase noch einmal durch einen algebraischen Presolver optimieren zu lassen. Dieser entfernt alle Zeilen und Spalten aus dem Problem, in denen schon eine ganzzahlige Entscheidungsvariable vorhanden ist. Für das Branch & Bound-Verfahren sind nur noch die nicht-ganzzahligen Variablen von Bedeutung. Das so entstandene Problem nennt man **Crushed-Modell**. Dieses ist wieder ein lineares Problem, was am effektivsten mit dem dualen Simplex mit Steepest-Edge gelöst wird [10].
Wie effektiv der Vorgang der Fixierung ist und wie stark die Größe des Problems reduziert werden kann hängt von der Größe der GAP ab. Je kleiner die GAP, desto effektiver sind Variablenfixierungen. Im Fall des FAP zeigt sich, dass die Lösungen des relaxierten LPs schon sehr nahe an der ganzzahligen Lösung liegen und es sich bezüglich der Laufzeit lohnt Variablenfixierungen durchzuführen [10].

Problem-	Original-Modell		# X fixiert		Crushed-Modell	
name	Zeile	Spalte	zu 1	zu 0	Zeilen	Spalten
1C	819	1.399	193	306	125	191
1C-R			230	282	31	80
1D	5.521	9.146	1.538	2.118	213	415
1D-R			1.536	2.082	241	687
1E	10.382	19.434	1.774	6.684	1.740	3.047
1E-R			2.033	6.481	1.063	2.182

Tabelle 5.4.: Ergebnisse der Fixierung

Tabelle 5.4 gibt eine Übersicht über die Kennzahlen für die fixierten und unfixierten Datensätze. Bei den mit einem zusätzlichen R gekennzeichneten Problemen wurden vor der Durchführung des I-P Verfahrens Störungen der Kosten der Bodenstandzeiten durchgeführt.
Wir sollten festhalten, dass eine Fixierung erst sinnvoll ist, wenn mehr als zwei Flugzeugtypen vorhanden sind. Im Fall $X_{1i} + X_{2i} = 1$ wird immer die Substitution

$X_{1i} = 1 - X_{2i}$ durchgeführt, wodurch alle Spalten entfernt werden.
In Tabelle 5.5 werden die Laufzeiten für die Daten mit Störungen und Fixierung dargestellt. Vergleichen wir Tabelle 5.3 und 5.5 sehen wir, dass durch diese Modifikationen erhebliche Leistungssteigerungen möglich sind. Beispielsweise bei Datensatz 1D werden nur noch 125, 4 Sekunden zur Lösung benötigt, anstatt vorher 581, 4 Sekunden. Nicht zu erwarten war, dass die Störung der Eingabedaten im Zusammenspiel mit der Variablenfixierung zu einer Verbesserung der I-P Verfahren Laufzeit führt. Das zeigt, dass auch auf den ersten Blick sinnlos erscheinende Überlegungen in Kombination mit anderen durchaus zum gewünschten Effekt führen können.

Problem-	Iterationen		Laufzeit		
name	I-P	Simplex	I-P	Simplex	Total
1C	17	121	3,9	0,1	4,0
1D	30	214	124,7	0,7	125,4
1E	36	3.412	859,9	9,7	869,6

Tabelle 5.5.: Laufzeiten in Sekunden nach Variablenfixierung und Störung der ground-arc Kostenvariablen

5.5. Vergleich und Zusammenfassung

Abschließend wollen wir die Ergebnisse der vorhergehenden Abschnitte beurteilen und Entscheidungen treffen welche Algorithmen und Modifikationen eine Verbesserung im Laufzeitverhalten für welche Datensätze ermöglichen.
Wie wir im Abschnitt über Innere-Punkte Verfahren feststellen konnten, spielt die Cholesky-Zerlegung und deren Eigenschaften für die benötigte Laufzeit der I-P Verfahren eine entscheidende Rolle. In Abbildung 5.1 stellen wir die Laufzeiten von Dualem Simplex mit Steepest Edge und I-P Verfahren gegenüber. Man kann erkennen, dass der Simplex bei den 7-Flotten Problemen schlechtere Laufzeiten erzielt als das I-P Verfahren. Bei den 11-Flotten Problemen zeigt sich genau ein umgekehrtes Bild.
Diese unterschiedlichen Ergebnisse sind auf die Anzahl an Matrixeinträgen ungleich Null in der Cholesky-Zerlegung zurückzuführen. Berechnet man den Quotienten aus A und L[5] und vergleicht diese mit den Laufzeiten, erkennt man, dass mit steigendem $\frac{A}{L}$ die Vorteilhaftigkeit des Dualen Simplex steigt. Bei kleinem $\frac{A}{L}$ Faktor bietet das I-P Verfahren eine kürzere Laufzeit.

Wir können festhalten, dass es keinen universellen Algorithmus zur Lösung des relaxierten Problems in einem FAP oder allgemein in einem ganzzahligen Optimierungsproblem gibt. Wir haben im letzten Kapitel zwei Varianten entwickelt, die für unterschiedliche Datensätze unterschiedlich gut geeignet sind. Interessant ist, dass

[5]Wobei A und L jeweils die Anzahl Variablen pro Matrix ungleich Null in der Zerlegung angeben

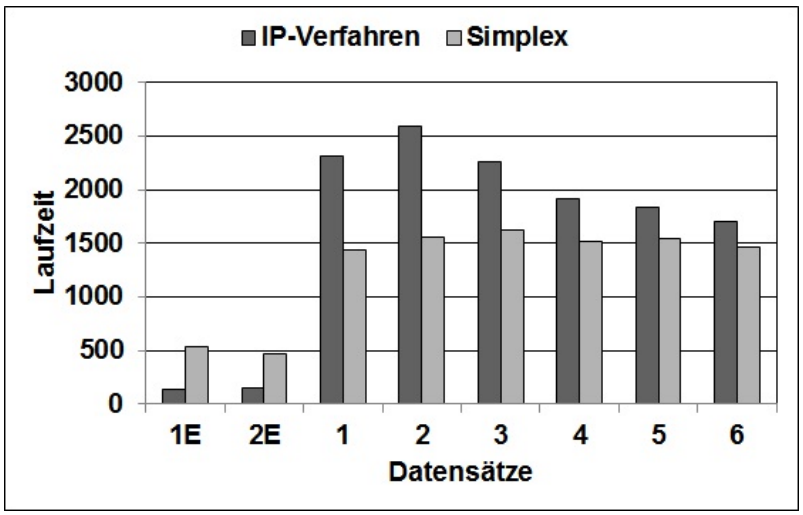

Abbildung 5.1.: Darstellung der Laufzeiten von IP-Verfahren und Simplex

nicht die Größe des Datensatzes entscheidend ist, sondern der Faktor $\frac{A}{L}$ in Bezug auf die Cholesky-Zerlegung.

Zum Abschluss des Kapitels wollen wir nun die Abläufe der beiden Algorithmen tabellarisch darstellen. In Tabelle 5.6 ist der Algorithmus zur Lösung der Datensätze mit 11-Flugzeugtypen und für die restlichen Datensätze dargestellt. Der eigentliche Unterschied besteht in der Verwendung unterschiedlicher LP-Lösungsalgorithmen.

	Teilschritt	Algorithmus 1 für 11-Typen	Algorithmus 2 für 7-Typen
1.	Vereinfachung durch den Löser	X	X
2.	Störung der Kostenkoeffizienten	X	X
3.	Dualer Simplex mit Steepest-Edge Pricing	X	
4.	Innere-Punkte Verfahren (Primal-Dual)		X
5.	Rückgängigmachen der Störungen	X	X
6.	Reoptimieren	X	
7.	Rückgängigmachen der durchgeführten Vereinfachungen	X	X
8.	Fixieren der $X \geq 0.99$ zu 1	X	X
9.	Vereinfachen ohne die Cover-Rows zu verletzen	X	X
10.	Dualer Simplex mit Steepest-Edge Pricing	X	X
11.	Branch & Bound	X	X

Tabelle 5.6.: Vergleich der LP-Lösungsalgorithmen (nach [10])

6. Branch & Bound-Phase

Im folgenden Kapitel werden wir uns mit dem Branch & Bound-Verfahren beschäftigen. Unsere Zielsetzung ist auch hier, die Laufzeit durch Modifikationen des Algorithmus zu beschleunigen.

Das Branch & Bound-Verfahren lässt sich in zwei grundlegende Prinzipen aufteilen:

1. **Branching:** Wir verzweigen das ursprüngliche Problem P_0 in k Teilprobleme, so dass deren Lösungsmengen untereinander disjunkt sind und die Vereinigung genau die Lösungsmenge von P_0 ergibt. Wie Domschke/Drexl festhalten, hängt die Vorgehensweise zur Zerlegung eines Problems P_0 in Teilprobleme wesentlich von der Art und Weise des zu lösenden ganzzahligen oder kombinatorischen Optimierungsproblems ab [5]. Durch die Verzweigung entstehen in den Teilproblemen neue Nebenbedingungen oder schon vorhandene werden modifiziert.

2. **Bounding:** Für jedes neu entstandene Teilproblem berechnen wir Schranken für den Zielfunktionswert, an Hand derer wir dann entscheiden können, ob Teilprobleme weiter verzweigt werden oder nicht.
   Die (globale - für alle Teilprobleme) untere Schranke nimmt den Wert der bisher gefundenen optimalen ganzzahligen Lösung eines Teilproblems an. Um die (lokale - für das spezielle und alle daraus abgeleiteten Teilprobleme) obere Schranke zu berechnen, bilden wir für jedes Teilproblem eine Relaxation und lösen diese mit Hilfe des in Kapitel 5 angegebenen Algorithmus.

Ein Flussdiagramm zum Ablauf des Branch & Bound-Verfahrens ist im Anhang A.3 zu finden und in Abbildung 6.1 ist ein möglicher Verzweigungsbaum zu sehen. Für weiterführende Informationen oder Erklärungen empfehlen wir [5] oder [20].

Die Verzweigungsstrategie beim Branching hat wesentlichen Einfluss auf die Laufzeit. Je mehr Verzweigungen, desto tiefer wird der entstehende Baum, desto mehr Relaxationen sind zu lösen und desto länger wird die Laufzeit. Unser Ziel muss es also sein, möglichst geschickt Teilprobleme zu bilden und so einen flachen ausgeglichenen Baum zu ermöglichen.
Eine sehr universell einsetzbare Verzweigungsregel wählt eine Entscheidungsvariable X_{fi} mit nicht-ganzzahligem Wert aus und fixiert diese in den neuen Teilbäumen auf den Wert 0 oder 1. Die Auswahl der Entscheidungsvariable erfolgt über eine Prioritätsfunktion, die *jeder* Variable eine Priorität zuordnet. Im allgemeinen Fall

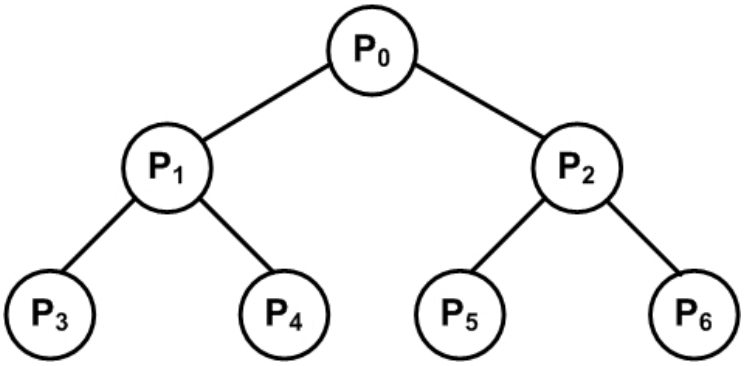

Abbildung 6.1.: Verzweigungsbaum Branch & Bound

betrachtet man den nicht-ganzzahligen Anteil der Variablen und wählt diejenige mit dem kleinsten (oder größten) Anteil [5]. Wir werden nun eine andere Verzweigungsvariante einführen, die auf Grund der Problemstruktur des FAPs Vorteile besitzt.

Wir betrachten eine Cover-Bedingung (Gleichung 3.3):

$$\sum_{f} X_{fi} = 1, \quad \forall i \in L$$

Eine solche Gleichung, die gleichzeitig eine Teilmenge an Entscheidungsvariablen definiert, nennt man **Special-Ordered-Set Typ 3 (SOS3)**. Die Variablen einer Gleichung / Menge sind, bezüglich der Lösung und Zulässigkeit des Problemes, direkt voneinander abhängig. Diese Abhängigkeit nutzen wir aus und verzweigen anstatt nach einzelnen Variablen nach Teilmengen von Variablen. Dadurch erhalten wir eine ausgeglichenere Baumstruktur [7].
Dazu partitionieren wir die Menge der Entscheidungsvariablen einer Cover-Zeile in zwei Mengen $\{I_l, I_u\}$ und verzweigen nach folgenden Bedingungen:

$$\sum_{f \in I_l} X_{fi} = 0, \quad \text{sowie nach} \quad \sum_{f \in I_u} X_{fi} = 0, \tag{6.1}$$

Die Prioritätsfunktion, auf Grund der wir nun entscheiden nach welcher Cover-Zeile wir verzweigen, wird auf die *Mengen von Entscheidungsvariablen* angewendet, anstatt auf jede einzelne Variable. Die explizite Wahl der Prioritätsfunktion und damit die Bestimmung der Prioritäten, kann dem Solver überlassen werden oder durch Modifikationen beeinflusst werden.

6.1. Modifizierte Prioritätsfunktionen

Eine modifizierte Prioritätsfunktion zu verwenden ist eine sehr einfache Art den Lösungsvorgang zu optimieren. Dadurch sind keinerlei Änderungen in der eigentli-

chen Implementierung des Algorithmus notwendig. Es werden lediglich alternative Hilfsfunktionen angewendet.

Sinnvoll ist es, die Prioritätsberechnung an die Modelldaten und die gewünschte Zielsetzung anzupassen. Beim FAP geht es darum eine möglichst kostenminimale Zuordnung zu bestimmen. Jedem Flugleg X_{fi} ist in der Zielfunktion ein Kostenkoeffizient c_{fi} zugeordnet. Daher wird es für die Verzweigung von Vorteil sein, die Prioritäten in Abhängigkeit der Verteilung der Kostenkoeffizienten in den einzelnen Cover-Bedingungen festzulegen.

Im Folgenden werden zwei mögliche Varianten dargestellt, die auf unterschiedliche Art und Weise die Streuung der c_{fi} (Varianz) pro Nebenbedingung in die Prioritätsfunktion mit einbeziehen.

Die erste Variante **„Spreizung“** definiert die Priorität als:

$$\sum_{f} |c_{f^-,i} - c_{f,i}| = Priority(i) \tag{6.2}$$

Dazu müssen wir die Flugzeugtypen aufsteigend nach der Sitzkapazität sortieren. f^- ist der Flugzeugtyp mit der nächstkleineren Sitzkapazität zu Flugzeugtyp f, der Flug-Leg i fliegen kann. Nach Definition darf diese Summe erst mit der Flotte, die die zweitgrößte Sitzkapazität besitzt, beginnen. Gemessen wird dann die Spreizung der möglichen Zielfunktionswerte über die Kostenkoeffizienten c_{fi} (Gleichung 6.2). Der Nachteil dieser Variante ist, dass der Algorithmus doppelte Schleifen über alle Spalten der Matrix durchläuft, was zu einer sehr langen Berechnungszeit führt.

Die zweite Variante **„Absolute-Differenz“** betrachtet lediglich das Minimum und das Maximum der Kostenkoeffizienten für eine Menge von Entscheidungsvariablen. Diese Methode hat den Vorteil, dass sie eine sehr effiziente Berechnungsdauer hat.

$$\max_{f} c_{fi} - \min_{f} c_{fi} = Priority(i) \tag{6.3}$$

Die dritte Variante **„Standard“** ist bei IBM OSL standardmäßig eingesetzte Prioritätsfunktion. Für jede mögliche neue Nebenbedingung, die durch Verzweigung einer Cover-Zeile entstehen könnte, wird eine Abschätzung der zu erwartenden Kostenänderung berechnet. Diesen Wert nennen wir Degradation. Für jede im Gleichungssystem vorkommende Cover-Zeile sind nun mehrere Degradations-Werte bekannt. Jeder Cover-Zeile wird als Priorität der maximal vorkommende Degradationswert aller möglichen Verzweigungen dieser Cover-Zeile zugewiesen. Ausgewählt wird die Nebenbedingung mit der minimalen Priorität.

6.2. Ergebnisse

In Tabelle 6.1 sind die Laufzeiten der verschiedenen Branch & Bound-Varianten dargestellt. Die relaxierten LPs wurden mit dem I-P Algorithmus (siehe Tabelle 5.7) gelöst. Als Startknoten für das Branching wurden optimale Basislösungen genutzt, die mit Hilfe des Crash-Verfahren berechnet wurden.
Die letzte Spalte „Laufzeit“ gibt die benötigte Zeit an, um entweder eine Optimallösung zu finden oder die Knoten-Obergrenze von maximal 2.000 Knoten auszuwerten. Die beiden Spalten „Lösungsposition der Ersten“ bzw. „Lösungsposition der Optimalen“ geben die Knotennummer an, bei der die entsprechende Lösung gefunden wurde. Die angegebene Optimallösung bei den Problemen mit über 2.000 Knoten ist die bis dahin gefundene optimale Lösung. Als besseres Maß für die Qualität des Branching-Verfahrens wird anstatt der Laufzeit auch gerne die Differenz aus Gesamtknotenanzahl und Position der Optimallösung verwendet. Durch die, bis zu diesem Zeitpunkt bekannte, Optimallösung wird die Oberschranke festgelegt. Je früher die optimale Lösung gefunden wird, desto früher besitzt die Oberschranke einen Wert, der verhindert, dass es zu unnötigen Verzweigungen und Auswertungen im weiteren Verlauf des Branch & Bound kommt.

Problem-name	Branching-regel	Knoten-anzahl	Lösungsposition der Ersten	Lösungsposition der Optimalen	# 0/1 Lösungen	Laufzeit in Sek.
1	Standard	>2.000	87	220	5	6.743,3
	Abs-Diff	46	8	8	1	258,9
	Spreizung	96	24	53	3	591,2
2	Standard	74	4	14	2	67,9
	Abs-Diff	68	10	50	5	254,9
	Spreizung	46	8	45	4	178,0
3	Standard	1.763	7	780	4	3.600,0
	Abs-Diff	30	6	11	2	180,6
	Spreizung	42	7	37	5	266,3
5	Standard	>2.000	23	566	6	6.744,0
	Abs-Diff	47	13	44	3	349,0
	Spreizung	167	15	57	3	883,2
6	Standard	499	36	345	5	809,3
	Abs-Diff	141	11	123	4	703,2
	Spreizung	60	11	41	3	249,3

Tabelle 6.1.: Ergebnisse bei Verwendung der unterschiedlichen Prioritätsfunktionen

Wir können in der Tabelle deutlich erkennen, dass die alternativen Prioritätsbe-

rechnungen einen schnellere Lösungsvorgang und einen flacheren Baum ermöglichen. Vor allem bei den Problemen, die im Standard-Fall über 2.000 Knoten besitzen ist eine starke Verbesserung zu erkennen.
Berechnet man die durchschnittliche Knotenanzahl über alle Datensätze ergeben sich:

Standard :	$1.267,2$
Abs-Diff :	$33,4$
Spreizung :	$82,2$

An Hand dieser Zahlen sehen wir die „Absolute-Differenz“ als effektivste Variante an, obwohl die Standard-Variante vereinzelt eine schnellere Laufzeit besitzt. Ihren Vorteil hat die „Absolute-Differenz“ vor allem bei den Problemen, bei denen der Abbruch auf Grund der Knoten-Oberschranke stattgefunden hat.

7. Zusammenfassung

Zu Beginn dieser Arbeit haben wir das grundlegende mathematische Modell eines FAP, basierend auf dem Time-Space-Network, vorgestellt. Außerdem haben wir uns mit den Eigenschaften der zu lösenden Datensätze vertraut gemacht.

Als nächsten Schritt haben wir mehrere Methoden zur Verkleinerung der Problemstruktur vor dem eigentlichen Lösungsvorgang dargestellt: Algebraische Vereinfachungen, Aggregation von Knoten und Bilden von Inseln in einer Timeline. Allein mit diesen Modifikationen war es möglich den Lösungsvorgang des relaxierten FAPs mit dem primalen Simplex-Algorithmus um durchschnittlich ca. 78% zu beschleunigen (siehe Tabelle 4.2).

Danach beschäftigten wir uns mit dem eigentlichen Lösungsvorgang des relaxierten linearen Problems. Wir haben zwei Basisverfahren für diese Aufgabenstellung verglichen: Simplexalgorithmus und Innere-Punkte Verfahren.
Beim Simplexalgorithmus haben wir das primale sowie das duale Verfahren und die Auswirkungen bei Verwendung einer alternativen Pricing-Variante dargestellt. Bei den Innere-Punkte Verfahren haben wir lediglich das Prädiktor-Korrektor-Verfahren betrachtet, da es in Bezug auf das Laufzeitverhalten die beste Variante ist.
Abschließend können wir festhalten, dass sich abweichende Ergebnisse für die unterschiedlichen Datensätze ergeben haben. Bei den Datensätzen mit 11-Flugzeugtypen ist der duale Simplex mit Steepest-Edge Princing schneller und bei den Datensätzen mit 7-Flugzeugtypen das primal-duale-Prädiktor-Korrektor Verfahren.

Im letzten Kapitel haben wir uns mit der Branch & Bound-Phase beschäfigt. Um komplizierte Implementationsarbeit zu vermeiden, haben wir uns nur mit der Verwendung einer alternativen Prioritätsfunktion zur Auswahl der zu verzweigenden Nebenbedingung und Variable beschäftigt. Hierbei konnten wir die durchschnittliche Knotenzahl im Verzweigungsbaum von $1.267,2$, bei Verwendung der Standardfunktion, auf $33,4$, bei Verwendung der „Absoluten Differenz“, reduzieren.

Tabelle 7.1 zeigt nun die endgültigen Resultate für den Algorithmus mit dualem Simplex als LP-Löser, Tabelle 7.2 für den Algorithmus mit Innere-Punkte Verfahren. Dargestellt wird die Gesamtlaufzeit für das Lösen des Problems, inklusive dem Einlesen der Daten sowie dem Ausgeben der Lösung. In Abbildung 7.1 ist eine graphische Gegenüberstellung der Laufzeiten zu sehen.

Problem-	I-P Verfahren		Simplex		B&B		GAP	Laufzeit
name	Iter.	Laufzeit	Iter.	Laufzeit	Knot.	Laufzeit		in Sek.
1	38	2.141,8	3.022	15,33	46	258,93	0,013	2.551,6
2	35	2.346,4	1.777	7,90	68	254,94	0,010	2.747,7
3	38	2.578,9	3.178	32,14	30	180,60	0,221	2.933,3
5	40	1.812,3	3.734	8,17	47	349,03	0,005	2.328,6
6	39	2.205,2	3.739	27,11	141	703,21	0,012	3.069,4

Tabelle 7.1.: Komplette Ergebnisse mit Innere-Punkte-Verfahren

Problem-	Dual-SE Simplex		B&B		GAP	Laufzeit
name	Iterationen	Laufzeit	Knoten	Laufzeit		in Sek.
1	15.351	1.501,8	>500	3.360,8	0,020	5.027,9
2	13.691	1.394,6	10	47,6	0,004	1.633,4
3	15.087	1.529,9	6	103,6	0,002	1.807,2
5	14.753	1.422,0	10	184,5	0,006	1.771,9
6	14.177	1.376,0	103	636,5	0,012	2.176,3

Tabelle 7.2.: Komplette Ergebnisse mit dualem Simplex

Trotz des einen Ausreißers in der Laufzeit bei Datensatz 1, bestätigt sich die Wahl des dualen Simplex als LP-Löser für die kompletten Datensätze mit 11 Flugzeugtypen. Der entwickelte Algorithmus besitzt ein durchschnittlich stabiles Laufzeitverhalten.

Die Ergebnisse machen deutlich, dass auch solche anspruchsvollen Probleme mit Hilfe einer Standardsoftware in angemessener Laufzeit lösbar sind. Notwendig, um ein solches Laufzeitverhalten zu erreichen, ist aber die Kenntnis der verwendeten Algorithmen und mögliche Modifikationen, sowie genaue Informationen über die Eigenheiten des zu Grunde liegenden realen Problems.

Da wir die Verbesserungen in der Branch & Bound-Phase nur durch sehr geringfügige Modifikationen erreicht haben, können wir hoffen, dass hier noch weiteres Verbesserungspotential besteht. Eventuell könnte man durch eine weitere Zusammenfassung von Flügen und der daraus folgenden Variablenfixierung eine flachere Baumstruktur erzielen. Eine weitere Möglichkeit besteht darin, die Flugzeiten als Parameter in das Modell mit einzubeziehen. In diesem Fall könnte man ein Flugzeug, das einen sehr langen Flug durchführt, aus der Menge der verfügbaren Flugzeuge entfernen. Verzweigt man in diese Richtung erhalten wir Teilbäume mit sehr unterschiedlichen Zielfunktionswerten, was direkten Einfluss auf die Baumstruktur hat [10].

Die hier vorgestellte Modellierung ist bis heute das am meisten in der Praxis verwendete Verfahren [8]. Es ermöglicht auch bei Erweiterung der Problemstellung durch Einbeziehung weiterer Einflussfaktoren oder größerer Datensätze, eine gute Darstell- und Lösbarkeit.

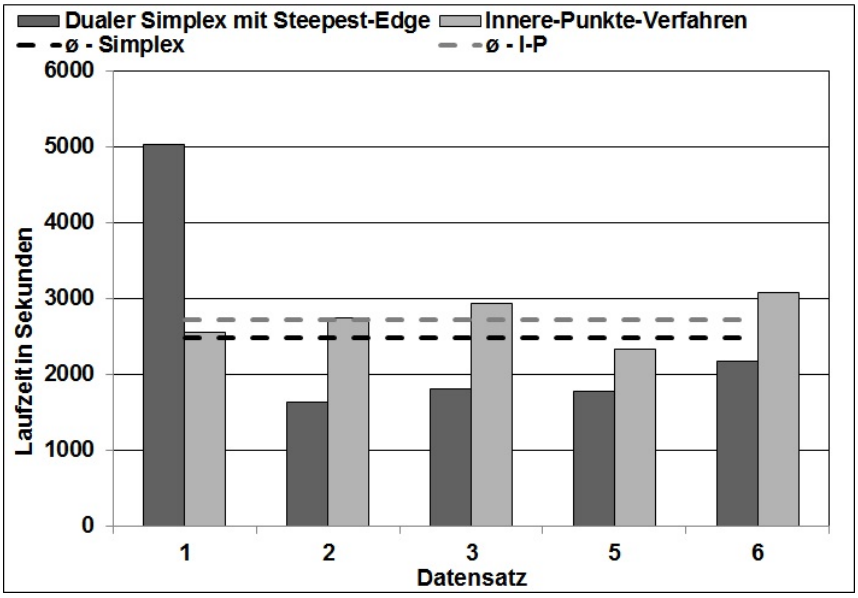

Abbildung 7.1.: Vergleich Laufzeiten

7.1. Stand der Forschung

In der Praxis kann man die, mit Hilfe unseres Modells, berechnete theoretische Lösung nicht direkt umsetzen. Die größten Defizite des Modells liegen darin, dass es den Ablauf für ein einzelnes Flugzeug nicht beachtet und auf unerwartet auftretende Situationen nicht reagieren kann. Durch die Modellierung als zyklisches FAP mit täglichem Planungshorizont bietet es uns keinerlei Anpassungsmöglichkeiten an Unterschiede in der Nachfrage über die Woche hinweg. Ein Ansatz zum Fleet-Assignment auf Basis eines wöchentlichen Planungszyklus, der diese Schwankungen mit einbezieht, wurde 1997 von Desaulniers et al. entwickelt [4].

Auf der Grundlage der Ergebnisse von Hane et al. aus dem Jahr 1995 wurden längst verfeinerte Modelle entwickelt, die teilweise auf eine etwas abweichende Fragestellungen ausgerichtet sind. Alle Forschungen zu diesem Thema wurden in enger Kooperation mit Airlines durchgeführt, die die jeweiligen Ergebnisse auch sehr gut in die Praxis umsetzen konnten. Wenn auch hier die oben erwähnten Einschränkungen gelten.
Zu nennen ist hier unter anderem die amerikanische Firma SA-BRE Inc., die sich mit der Entwicklung von Software-Lösungen für Fluggesellschaften befasst. Sie entwickelte die Software „Fleet Assignment System“. Diese aktualisiert, mit Hilfe von periodisch approximierten Werten für die möglichen erzielbaren Erlöse, die Modelldaten und kann so eine bessere Zuordnung berechnen [8].

Weiter wurde in verschiedenen Arbeiten versucht zusätzliche Komponenten des Planungsprozess in FAP-Modelle mit einzubeziehen. In der Arbeit von Grothklags ist eine kurze Übersicht über einige bis 2006 bekannte Arbeiten enthalten. Zum Beispiel

versuchte Lohatepanont 2002 die Passagierzahlen zu integrieren um bessere Werte für die Erlösstruktur zu erhalten [8].
1998 versuchten Barnhart et al. regelmäßige Wartungsereignisse in die Flottenplanung mit einzubeziehen. Es wurden aber keine akzeptablen Laufzeiten für große Probleminstanzen erzielt, weswegen die Verwendbarkeit dieses Modells eingeschränkt ist.
Interessant sind außerdem noch mehrere Arbeiten, die versuchen die Crew-Planung in die Flotten-Zuweisung mit einzubeziehen, die wir hier aber nicht namentlich aufführen wollen [8].

7.2. Fazit

Die in dieser Arbeit vorgestellten Ergebnisse zeigen, dass es möglich ist umfangreiche Probleme in einer relativ kurzen Zeit erfolgreich zu lösen. Dies ist aber nur möglich, wenn man trotz aller heute möglichen technischen Unterstützung sich intensiv mit der Problemstellung befasst. Die zu Beginn stattfindende Modellierung eines Problems legt die Grundsteine für die spätere erfolgreiche Anwendbarkeit. Nur mit mathematischen Mitteln haben die in dieser Arbeit dargestellten Vereinfachungen und Modifikationen entwickelt und erfolgreich in das Modell integrieren können.

Problem	Anzahl		GAP	Laufzeit
	NB	Variablen	in %	in Sek.
DB	20.642	36.108	0,83	$\approx$ 50.400
3	48.109	22.746	0,002	1.807,2

Tabelle 7.3.: Vergleich von Laufzeiten - heute und damals [3]

Alle Daten bezüglich der Laufzeit beruhen auf den Ergebnissen von Hane et al. aus dem Jahr 1995. Damals war die PC-Technologie lange nicht so weit entwickelt wie heute. Die verwendete Technik war ein IBM RS/6000 320 mit 20 MHz [10]. Heutzutage sind, beispielsweise an der TU Darmstadt im Fachbereich Mathematik, Rechner mit einem AMD Phenom (tm) 9950 Quad-Core Prozessor (4 x 2,6 GHz) mit 8 GB Arbeitsspeicher vorhanden [3].

Tabelle 7.3 verdeutlicht, dass ohne sinnvolle Anpassung der verwendeten Verfahren und vorhergehender Vereinfachungen, heute bedeutend längere Laufzeiten als damals möglich sind. „DB“ bezeichnet ein Modell zur Berechnung einer optimalen[1] Anordnung von Baustellen im Schienennetz der Deutschen Bahn AG. Es ist ein vergleichbares binäres ganzzahliges Optimierungsproblem.

[1] Minimal in Bezug auf die Umwegigkeit.

Abschließend soll festgehalten werden, dass trotz immer besserer Technik nicht auf das mathematische Wissen und die Erfahrung bei der Modellierung verzichtet werden kann. Vor allem bei einem so komplexen Planungsprozess mit einer Vielzahl an Teilproblemen, wie er bei der alltäglichen Planung einer Airline vorkommt, ist die perfekte Integration und Mathematisierung aller Teilprobleme der entscheidende Vorgang.

A. Anhang

A.1. Abkürzungsverzeichnis

Abkürzung	Erklärung
FAP	Fleet-Assignment-Problem
LP	Lineares Optimierungsproblem
SE	Steepest-Edge-Pricing
PD-PC	Primal-Duales-Prädiktor-Korrektor Verfahren
I-P	Innere-Punkte Verfahren
GAP	IP-LP-GAP: Differenz zwischen optimaler Lösung des relaxierten Problemes (LP) und des ganzzahligen Problemes (IP)

A.2. Der Simplex-Algorithmus

$$\textbf{LP (5.1) :} \quad \begin{aligned} &\min c^T x \\ &\text{s.t. } Ax \leq b \\ &\quad x \geq 0 \end{aligned}$$

Input: Eine primal zulässige Basis $B, \overline{x}_B = A_B^{-1} b$
Output: Eine Optimallösung $\overline{x}$ für das LP (5.1) oder die Meldung „Das LP (5.1) ist unbeschränkt".

(1) **BTRAN (Backward Transformation)**
Löse $\overline{y}^T A_B = c_B$.

(2) **Pricing**
Berechne Vektor der reduzierten Kosten: $\overline{z}_N = c_N - A_N^T \overline{y}$.
Falls $\overline{z}_N \geq 0$, dann ist B optimal, **Stop**.
Andernfalls wähle ein $j \in N$ mit $\overline{z}_j < 0$.$x_j$heißt die in die Basis eintretende Variable.

(3) **FTRAN (Forward Transformation**)
Löse $A_B w = A_j$.

(4) **Ratio-Test**
Falls $w \leq 0$, dann ist das LP (5.1) unbeschränkt, **Stop**.
Andernfalls berechne
$\Delta x = \frac{\overline{x}_{Bi}}{w_i} = \min \left\{ \frac{\overline{x}_{Bk}}{w_k} : w_k > 0, k = 1, 2, \ldots, m \right\}$,
wobei $i \in \{1, 2, \ldots, m\}$ und $w_i > 0$. $\overline{x}_{Bi}$ heißt die die Basis verlassende Variable.

(5) **Update**
$\overline{x}_B := \overline{x}_B - \Delta x \cdot w$
$N := (N \setminus \{j\}) \cup \{B_i\}$,
$B_j := j$,
$\overline{x}_j := \Delta x$
Gehe zu (1).

Entnommen aus Skript Optimierung 1 [15].

A.3. Ablaufschema: Branch & Bound

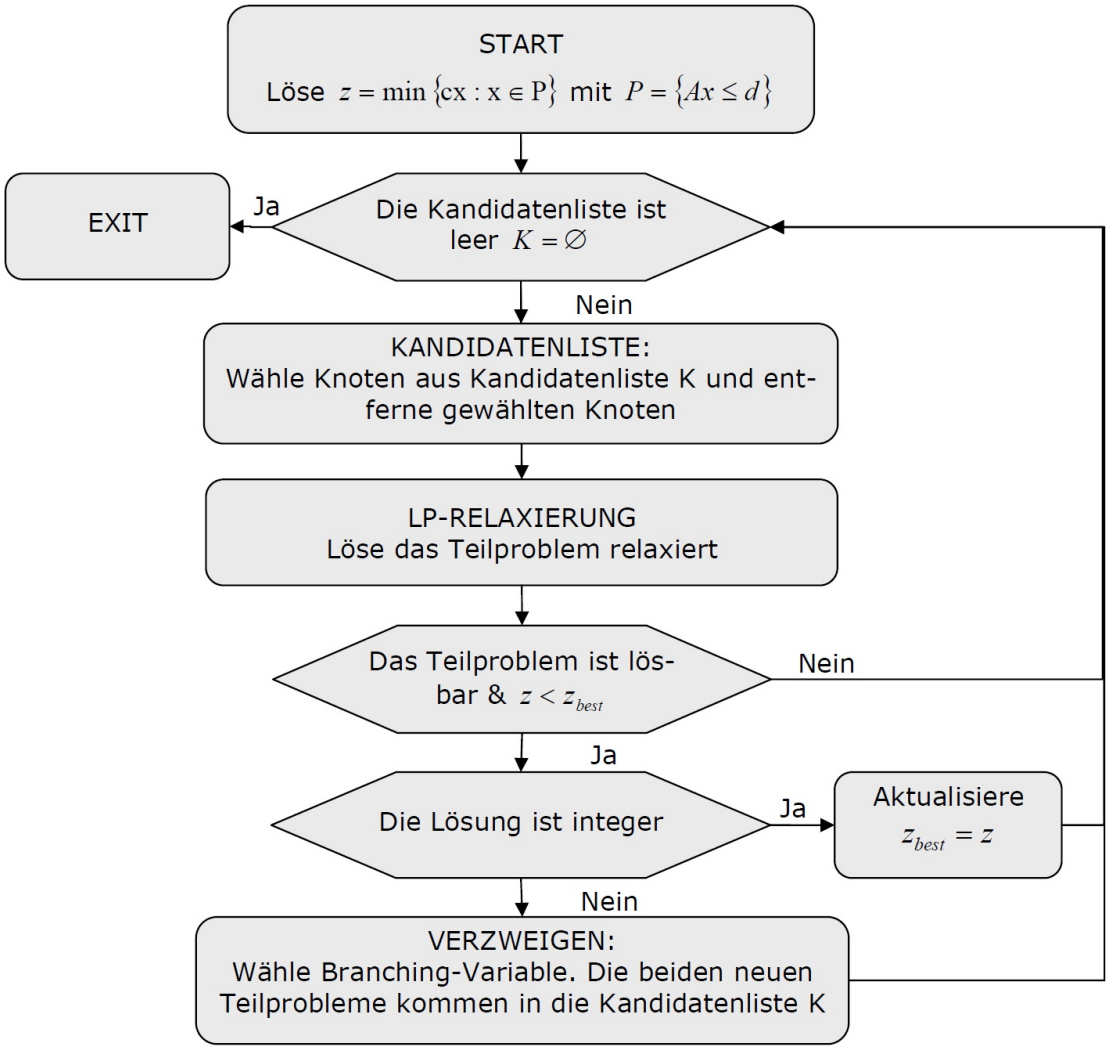

Abbildung A.1.: Ablaufschema Branch & Bound (entnommen aus [20])

Literaturverzeichnis

[1] Jeph Abrara - Applying integer linear programming to the fleet assignment problem. Interfaces, 19:20-28, 1989.

[2] Erling D. Andersen, Knud D. Andersen - Presolving in linear programming. Mathematical Programming 71:221-245, 1995.

[3] Frank Borchert, Ann-Kathrin Heyse, Anne Philipp, Claudia Schermuly, Sebastian Vock - Projektseminar: Optimierung der Baustellenplanung bei der Deutschen Bahn AG, TU Darmstadt, 2010.

[4] Guy Desaulniers, Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, Francois Soumis - Daily Aircraft routing and scheduling. Management Science, Vol. 43, Iss. 6, Seite 841-855, 1997.

[5] Wolfgang Domschke, Andreas Drexl - Einführung in Operations Research, 7.Auflage. Springer-Verlag, Berlin, 2007.

[6] A. R. Ferguson und Georg B. Dantzig - The allocation of aircraft to routes - an example of linear programming under uncertain demand. Management Science, 3, 1956.

[7] Swantje Friedrich - Algorithmische Verbesserungen für die Lösung diskreter Optimierungsmodelle, Kapitel 3. Dissertation, Freien Universität Berlin, 2007.

[8] Sven Grothklags - Flottenzuweisung in der Flugplanung: Modelle, Komplexität und Lösungsverfahren. Dissertation, Universität Paderborn, 2006.

[9] Zonghao Gu, Ellis Johnson, George Nemhauser und Yinhua Wang - Some properties of the fleet assignment Problem. Operations Research Letters 15:59-71, 1994.

[10] Christopher A. Hane, Cynthia Barnhart, Ellis L. Johnson, Roy E. Marsten, George L. Nemhauser, Gabriele Sigismondi - The Fleet Assignment Problem: Solving a Large-Scale Integer Program. Mathematical Programming 70:211-232, 1995.

[11] Paula M.J. Harris - Pivot Selection Methods of the Devex LP Code. Mathematical Programming 5:1-28, 1973.

[12] Christoph Helmberg - Skript Optimierung 1, TU Chemnitz, 2004.

[13] Josef Kallrath - Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis, 1. Auflage. Vieweg Verlag, Braunschweig/Wiesbaden, 2002.

[14] Nalav Gülpinar, Gautam Mitra, Istvan Maros - Creating Advanced Bases For Large Scale Linear Programs Exploiting Embedded Network Structure. Computational Optimization and Applications, Vol.21:1, 71-93, 2002.

[15] Alexander Martin, Mirjam Dür, Stefan Ulbrich - Skript zur Optimierung I: Einführung in die Optimierung, TU Darmstadt, 2009.

[16] Nimrod Megiddo - On Finding Primal- and Dual-Optimal Bases. OSRA Journal on Computing, Vol. 3, Mo. 1, 1991.

[17] Sanjay Mehrotra - On the implementation of a primal-dual interior point method. SIAM Journal on Optimization 2:575-601, 1992.

[18] Philipp Schade - Innere-Punkte-Verfahren mit Redundanzerkennung für die Quadratische Optimierung. Gabler Verlag, Wiesbaden, 2008.

[19] Bernd Wagner - Hub&Spoke-Netzwerke in der Logistik. Gabler Verlag, Wiesbaden, 2006.

[20] Veronika Waue - Entwicklung von Software zur Lösung von gemischt-ganzzahligen Optimierungsmodellen mit einem Branch-and-Cut-Ansatz, Kapitel 5. Dissertation, Freie Universität Berlin, 2007.

[21] Dag Wedelin - An Algorithm für large scale 0-1 integer programming with an application to airline crew scheduling, 1993.